# SHIP DETECTION ON HIGH RESOLUTION REMOTE SENSING IMAGE VIA SCENE USING MASK R-CNN

**A PROJECT REPORT**

***Submitted by***

# DHARSHINI A [REGISTER NO:211417104051] HEMAMALINI R [REGISTER NO:211417104085] JAYASHREE V [REGISTER NO:211417104090]

***in partial fulfillment for the award of the degree of***

# BACHELOR OF ENGINEERING

IN

# COMPUTER SCIENCE AND ENGINEERING PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

**ANNA UNIVERSITY: CHENNAI 600 025**

# AUGUST 2021

**BONAFIDE CERTIFICATE**

Certified that this project report “**SHIP DETECTION ON HIGH RESOLUTION REMOTE SENSING IMAGE VIA SCENE USING MASK R-CNN”** is the bonafide work of **“DHARSHINI A [REGISTER NO:211417104051], HEMAMALINI R [REGISTER NO:211417104085], JAYASHREE V [REGISTER NO: 211417104090]”** who carried out the

project work under my supervision**.**



# SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI,M.E.,Ph.D., M.SANGEETHA,M.Tech HEAD OF THE DEPARTMENT ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on......**5.08.2021**.....................

# INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing his project.

We would like to extend our heartfelt and sincere thanks to our directors **Tmt.C.VIJAYA RAJESWARI, Ph.D**, **Dr.C.SAKTHIKUMAR, M.E.,Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR, B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K. Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S. MURUGAVALLI, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide **M. SANGEETHA, M. Tech** and all the faculty members of Department of CSE for their advice and suggestions for the successful completion of the project.

**NAME OF THE STUDENTS**

DHARSHINI A

HEMAMALINI R

JAYASHREE V

# ABSTRACT

High resolution satellite image processing is one of the most growing fields in research today. There is so much to explore and so many ways to do it that it seems full of endless opportunities and possibilities. There are several features which can be extracted like buildings, roads etc. from land satellite images and ships, boats etc. from satellite images of sea and ocean. In this paper we will be concentrating on detecting ships automatically from the images obtained by various satellites. This is one of the major challenging tasks due to various disturbances and noises in these kinds of images. Ships can be found in different sizes as well as shapes which make it more difficult to find a pattern or some regularity in these images. It is comparatively easier in homogeneous environment consisting of just ships of different types in water. But when it comes to heterogeneous environment consisting of other elements like coasts, harbor, vessel, rocks, islands etc. the challenge increases tenfold.

There are various statistical and image processing approaches which can do this manually then again this won’t be that efficient, changing the parameters again and again with different images and all can be a time consuming and tedious process. That’s why we choose one of the modern approaches which provide us with the opportunity of being automatic or at least semi-automatic, deep learning. Deep learning together with computer vision opens doors to the possibilities which we couldn’t even have thought of. In this paper we will be exploring those possibilities with the help of a certain Convolutional neural network known as Mask R\_CNN and implementing it using transfer learning. This algorithm gives very high accuracy in classification of satellite images without doing any manual extraction and works with complex heterogeneous backgrounds too.

# TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO** | **TITLE** | | **PAGE NO.** |
|  | ABSTRACT | | iv |
|  | LIST OF TABLES | | v |
|  | LIST OF FIGURES | | vii |
| **1.** | **INTRODUCTION** | | 1 |
|  | 1.1 | Overview | 1 |
|  | 1.2 | Problem Definition | 4 |
|  | 1.3 | Scope | 4 |
|  | 1.4 | Challenges | 5 |
| **2.** | **LITERATURE SURVEY** | | 5 |
| **3.** | **SYSTEM ANALYSIS** | | 13 |
|  | 3.1 | Existing System | 13 |
|  | 3.2 | Proposed system | 14 |
|  | 3.3 | Requirement Analysis and Specification | 15 |
|  | 3.3.1 | Input Requirements | 15 |
|  | 3.3.2 | Output Requirements | 17 |
|  | 3.4 | Hardware Environment | 17 |
|  | 3.5 | Software Environment | 17 |
|  | 3.6 | Technologies Used | 17 |
| **4.** | **SYSTEM DESIGN** | | 26 |
|  | 4.1 | Data Flow Diagram | 27 |
|  | 4.2 | UML Diagrams | 28 |
|  | 4.2.1 | Sequence Diagram | 29 |
|  | 4.2.2 | Use-case Diagram | 30 |
|  | 4.2.3 | Activity Diagram | 31 |
|  | 4.2.4 | Collaboration Diagram | 32 |
| **5.** | **SYSTEM ARCHITECTURE** | | 33 |
|  | 5.1 | Architecture Overview | 33 |
|  | 5.2 | Module Design Specification | 33 |
| **6.** | **SYSTEM IMPLEMENTATION** | | 43 |
|  | 6.1 | Ship Detection Code | 43 |
|  | 6.2 | Ship Categorization Code | 57 |
| **7.** | **SYSTEM TESTING** | | 59 |
| **8.** | **CONCLUSION** | | 62 |
|  | Conclusion | | 62 |
|  | **APPENDICES** | | 63 |
|  | A.1 | Sample Screens | 63 |
|  | A.2 | Publications | 70 |
|  | **REFERENCES** | | 71 |

# LIST OF FIGURES

4.1 Dataflow diagram for categorization of ships (Level 0,1,2)

* + 1. Sequence diagram for detecting and categorizing ships
    2. Use case diagram for detecting and categorizing ships
    3. Activity diagram for detecting and categorizing ships
    4. Collaboration diagram for detecting and categorizing ships

5.1.1 Architecture diagram for collecting images, categorizing and detecting of ships

5.3.1 Synthetic Aperture Radar (SAR) Ship image data set

* + 1. Screenshot of image input
    2. Screenshot of ship detected
    3. Screenshot of x-axis and y-axis of detected ship
    4. Screenshot of rectangular box around the detected ship
    5. Screenshot to run the categorization code
    6. Screenshot of Getting the URL for categorizing the ships
    7. Screenshot of web page for categorizing the ships
    8. Screenshot of image uploading page
    9. Screenshot of choosing the image
    10. Screenshot for categorizing the detected ships
    11. Screenshot of no ships detected

# INTRODUCTION

* 1. **OVERVIEW**

Ship detection from remote sensing imagery has been a major application for maritime security. When talking about maritime security, we have to consider many things like traffic surveillance, protection against illegal fisheries, oil discharge control, and sea pollution monitoring. Automated Identification System (AIS) is very effective at monitoring ships which are legally required to install a VHF transponder but fail to detect those which are not and those which disconnect their transponder. So how do you detect these uncooperative ships? SAR images are considered the most suitable sensors for object detection in space technology. It captures a wide surface of the environment, regardless of whether or time of day and flight altitude. Hence it has very high-resolution capabilities and gives high-quality images. SAR has various applications in remote sensing and mapping of different surfaces of the Earth. It can be used in oceanology, glaciology, biomass, volcanoes, forestry, etc. Before deep learning evolved traditional methods of target detection were divided into region selection eg. SIFT-scale invariant feature transform, HOG-histogram of an oriented gradient, and classifiers like SVM (Support Vector Machine) and Adaboost. Unlike the sliding window and regional proposal-based approach, YOLO sees the whole image during the training and testing period and thus encodes contextual information

about classes and their appearance. Further, YOLO makes less number of background errors than Fast R-CNN. But we have used Mask RCNN for instance segmentation. Another most common approach is the CFAR-constant false rate use to detect targets with threshold with pixel’s amplitude hence it is difficult to extract features [10].In addition, these methods are typically dependent on the statistical distribution of sea clutter, leading to poor robustness for new SAR imagery. YOLO predicts the bounding box and object class probability from the complete image in a single estimate. We have to build a program to automatically identify whether a remotely sensed target is a ship or not. We will start by collecting huge data of satellite images of ships from various heights, (Roughly about 30 GB). To make the computations easy, we will mask some of the images, where there are ships present in the picture. While masking, we will make sure that all or almost all possibilities of the ships get covered. We will call this set the “**Ground Truth**”. A suitable model will then be chosen in order to educate our program to identify a ship in the image in the dataset by comparing it with the ground truth. Once a ship is detected we will bound it with the help of a bounding box.

Detecting ships in high resolution satellite images has number of applications in the arena of maritime surveillance, military, safe and effective transportation in the harbors, illegal fishing etc. [2]. Various research work has been done and is still

continuing in this field from the 70s-80s. Several methods have been proposed in this field so that the detection accuracy increases. Even different hybrid methods are proposed but they are not consistent with extreme weather as it may lead to false detection. These state of the art methods were though not optimal in several cases but they were robust up to a certain extent. This is why deep learning was considered. CNN models can be trained with hundreds of examples with all the worst case scenarios and hence are able to give result with great accuracy. The neural network in CNN gives excellent result in object detection which is our prime goal and here the object is a ship. There are two major sections of these layers first for extracting features and the second one for classifying them accordingly. It also contained a hidden and an output layer. Today remote sensing plays a vital role in detection of objects like ships from various images and hence helps in locating the position of ships. Satellite image processing has various applications, some of them being surveillance of traffic on harbors during busy hours, monitoring of coasts and harbors for smuggling activities, rescuing various sea machines like vessels and ships, military operations, pollution control in oceans, monitoring of oil discharge and sea pollution, naval and maritime welfare etc.

For extraction of ship all they need is synthetic aperture radar (SAR) images or panchromatic images as they are of high resolution. Monitoring of ships from satellite

images helps in timely and periodic checking for anonymous movements in their territory region. Though in real time application it is much more complex and computation should be done considering this real time factor while detection and recognition. Remote sensing plays a very important role in monitoring ships as it operates from some distance and has a wide monitoring range. At the same time the sea surface gives better valid information than the appearance of ship.

# PROBLEM DEFINITION

Ship detection and classification based on optical remote sensing images raise considerable attention in the sea surface remote sensing field. Basic feature extraction strategies and algorithms are analyzed associated with their performance and application in ship detection and classification. Based on the analysis, the remaining problems and future development trends are provided for ship detection and classification methods based on optical remote sensing images.

# SCOPE

Ship detection is a crucial application for global monitoring for environment and security. It permits to monitor traffic, fisheries, and to associate ships with oil discharge. Typical discriminate parameters are ship lengths, speed, and radar cross section. A basic classification strategy is proposed.

# CHALLENGES

* + - The high variability of targets caused by the viewpoint variation, imaging sensor parameters, Occlusion, ship wakes, color, speed, and material of ships, etc.
    - High false alarm rate due to islands, heavy clouds, ocean waves, and the various and uncertain. Sea state conditions, like partial cloud cover, fog, wind, and swell.
    - The third issue is the computation burden. Most detection methods have high computational Cost. Hence, reducing computational cost is considered to be a key issue for the large-scale remote Sensing images.

In consideration of these challenges mentioned above, we believe that a practical ship detection Method should meet two requirements: it should be robust to the interference of the high variability of targets and background clutter such as waves, islands, clouds, and so forth. Of equal importance, with the purpose of the engineering applications, it should have lower calculation complexity and satisfy the requirements of real-time processing.

# LITERATURE SURVEY

1. **D. Zeiler, Graham W. Taylor and Rob Fergus presented “ADAPTIVE DECONVOLUTIONAL NETWORKS FOR MID AND HIGH LEVEL**

**FEATURE LEARNING” in 2016, International Conference on Computer Vision.** We present a hierarchical model that learns image decompositions via alternating layers of convolutional sparse coding and max pooling. When trained on natural images,the layers of our model capture image information in a variety of forms:low-level edges,mid-level edge junctions, high-level object parts and complete objects.When combined with a standard classifier, features extracted from these models outperform SIFT, as well as representations from other feature learning methods.

# DRAWBACKS:

While edges only vary in orientation and scale, larger-scale structures are more variable.

1. **Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun presented “SPATIAL PYRAMID POOLING IN DEEP CONVOLUTIONAL NETWORKS FOR VISUAL RECOGNITION” in 2015, IEEE Transactions on Pattern Analysis and Machine Intelligence.** Existing deep convolutional neural networks (CNNs) require a fixed-size (e.g., 224 × 224) input image.In processing test images, our method is 24-102 × faster than the R-CNN method, while achieving better or comparable accuracy on Pascal VOC 2007. In ImageNet Large Scale Visual

Recognition Challenge (ILSVRC) 2014, our methods rank #2 in object detection and #3 in image classification among all 38 teams.

# DRAWBACKS:

Convolutional layers do not require a fixed image size and can generate feature maps of any sizes.

1. **Karen Simonyan\_ & Andrew Zisserman presented “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION” in 2015, computer Vision and Pattern Recognition (cs.CV).** In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3x3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers. **DRAWBACKS:**

Oceanic SAR imagery is affected by different kinds of disturbances, such as speckle or marine discontinuity effects due to random changes in bathymetry and wind currents.

# Olaf Ronneberger, Philipp Fischer, and Thomas Brox presented “U- NET: CONVOLUTIONAL NETWORKS FOR BIOMEDICAL IMAGE

**SEGMENTATION” in 2015, International Conference on Medical Image Computing and Computer-Assisted Intervention.** There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU.

# DRAWBACKS:

Larger patches require more max-pooling layers that reduce the localization accuracy.

1. **Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla presented “SEGNET: A DEEP CONVOLUTIONAL ENCODER-DECODER ARCHITECTURE FOR IMAGE SEGMENTATION” in 2016, IEEE Transactions on Pattern Analysis and Machine Intelligence.** We present a novel and practical deep fully convolutional neural network architecture for semantic pixel- wise segmentation termed Seg Net.It is also significantly smaller in the number of trainable parameters than other competing architectures and can be trained end-to-end using stochastic gradient descent.We also performed a controlled benchmark of Seg

Net and other architectures on both road scenes and SUN RGB-D indoor scene segmentation tasks.

# DRAWBACKS:

The maximum of all the classifier responses in this sub tree becomes the classification score of the query image.

1. **Evan Shelhamer\_, Jonathan Long\_, and Trevor Darrell presented “FULLY CONVOLUTIONAL NETWORKS FOR SEMANTIC SEGMENTATION” in 2016, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).** Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Our fully convolutional network achieves state-of-the-art segmentation of PASCAL VOC (20% relative improvement to 62.2% mean IU on 2012), NYUDv2, and SIFT.

# DRAWBACKS:

Each pixel is labeled with the class of its enclosing object or region, but with shortcomings that this work addresses.

1. **Zhong-Qiu Zhao, Peng Zheng,, Shou-tao Xu, and Xindong Wu presented “OBJECT DETECTION WITH DEEP LEARNING” in 2019, IEEE Transactions on Neural Networks and Learning Systems**. Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years. Their performance easily stagnates by constructing complex ensembles that combine multiple low-level image features with high-level context from object detectors and scene classifiers. Finally, several promising directions and tasks are provided to serve as guidelines for future work in both object detection and relevant neural network-based learning systems.

# DRAWBACKS:

Large variations in viewpoints, poses, occlusions and lighting conditions, it’s difficult to perfectly accomplish object detection.

1. **Shaoqing Ren\_ Kaiming He Ross Girshick Jian Sun presented “FASTER R-CNN: TOWARDS REAL-TIME OBJECT DETECTION WITH REGION PROPOSAL NETWORKS” in 2015, Computer Vision and Pattern Recognition.** In this work, we introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling

nearly cost-free region proposals. For the very deep VGG-16 model, our detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state- of-the-art object detection accuracy on Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

# DRAWBACKS:

A novel dataset that combines many of the properties of both object detection and semantic scene labeling datasets.

1. **Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun presented “DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION” in 2016, IEEE Conference on Computer Vision and Pattern Recognition (CVPR).** Deeper neural networks are more difficult to train. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions1, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

# DRAWBACKS:

Up to now, ship detection in optical satellite images is still challenging with respect to clouds, waves, wake clutters, and the variability of ship sizes.

# Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold

**W. M. Smeulders presented “SELECTIVE SEARCH FOR OBJECT RECOGNITION” in 2011, International Conference on Computer Vision.** For object recognition, the current state-of-the-art is based on exhaustive search. However, to enable the use of more expensive features and classifiers and thereby progress beyond the state-of-the-art, a selective search strategy is needed. Our method is class- independent and is shown to cover 96.7% of all objects in the Pascal VOC 2007 test set using only 1,536 locations per image. Our selective search enables the use of the more expensive bag-of-words method which we use to substantially improve the state- of-the-art by up to 8.5% for 8 out of 20 classes on the Pascal VOC 2010 detection challenge.

# DRAWBACKS:

Existing methods for general object detection cannot be applied to solve this problem effectively. Their relationship and contributions are extensively investigated and evaluated

# SYSTEM ANALYSIS

# EXISTING SYSTEM

Computer-aided ship detection methods greatly free up human resources and typically include two steps: extracting image features, and then using classifiers for classification and localization. These methods can produce stable results under calm sea conditions. However, when disturbances such as waves, clouds, rain, fog, and reflections happen, the extracted low-level features are not robust. Besides, manual selection of features is time-consuming and strongly dependent on the expertise and characteristics of the data itself.

# PROBLEM STATEMENT:

* + - Getting images of ships.
    - Machine learning, testing and training of images, and prediction

# LIMITATIONS:

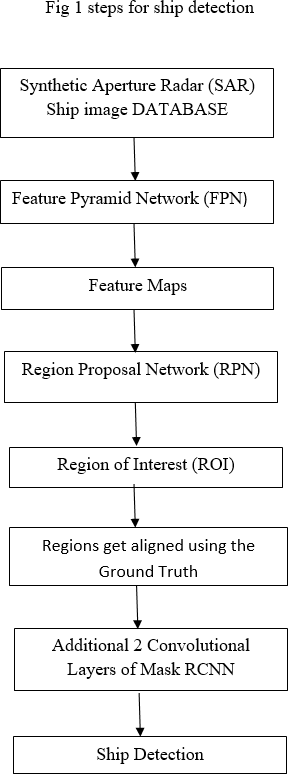
* + - * An important limitation of such approaches is their scalability to large quantities of training images.
      * Watching the ship in seas to affect the eyes of the man and woman.
      * There is no correct identification of the particular ship.

# PROPOSED SYSTEM

This end-to-end system contains four sub-networks with different functions. The feature map of the input image is obtained by the Feature Pyramid Network (FPN) first, and then the scene mask of target and non-target area is extracted by the scene mask extraction network (SMEN). With the feature combination between the output of FPN and the estimated scene mask, the false alarm targets existing in non- target area are eliminated entirely. Then Region Proposal Network (RPN) uses the combined feature map to generate the proposed bounding boxes. After computing the RoI, we have to compute the IoU over all of the predicted regions. IoU stands for Intersection over Union and is calculated with the help of ground truths. This completes the process of Mask RCNN, where we get the masks for the objects in the image. Therefore, we took help from the pretrained weights of the COCO dataset trained on the Mask RCNN model.

# ADVANTAGES:

* + - The false alarms caused by the onshore ship-like objects may decrease the accuracy and feasibility of these DCNN-based detection frameworks.
    - Mask R-CNN, is proposed to reduce the onshore false alarms.
    - This proposed system effectively able to extract all the features of an image.



# REQUIREMENT ANALYSIS AND SPECIFICATION

# INPUT REQUIREMENTS

# JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

# ANACONDA PROMPT

Anaconda command prompt is just like command prompt, but it makes sure that you are able to use anaconda and conda commands from the prompt, without having to change directories or your path. When you start Anaconda command prompt, you'll notice that it adds/("prepends") a bunch of locations to your PATH.

Anaconda command prompt is just like command prompt, but it makes sure that you are able to use anaconda and conda commands from the prompt, without having to change directories or your path.

# ANACONDA

Anaconda helps in simplified package management and deployment.

Anaconda comes with a wide variety of tools to easily collect data from various sources using various machine learning and AI algorithms. It helps in getting an

easily manageable environment setup which can deploy any project with the click of a single button.

# OUTPUT REQUIREMENTS

System with 64bit distribution capable of running 32bit application and 1200\*800 minimum screen resolution with stable internet connection.

# HARDWARE ENVIRONMENT

* + - Hard Disk : 500GB and Above
    - Ram : 4GB and Above
    - Processor : I3 and Above

# SOFTWARE ENVIRONMENT

* + - Operating System : Windows 7, 8, 10 (64 bit)
    - Software : Python 3.7
    - Tools : Anaconda (Jupyter Note Book IDE)

# TECHNOLOGIES USED

* + - Python
    - Scene Mask Region Based Convolution Neural Network & CNN

# THE PYTHON PROGRAMMING LANGUAGE

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-](https://en.wikipedia.org/wiki/Object-oriented_programming) [oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects. Python is [dynamically-typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) object-oriented and [functional programming](https://en.wikipedia.org/wiki/Functional_programming).

# FUTURE OF PYTHON

According to the TIOBE index, Python is the 4th most popular programming language out of 100 With the rise of Ruby on Rails and more recently Node.js, Python's usage as the main prototyping language for backend web development has diminished somewhat, especially since it has a fragmented MVC ecosystem. However, with big data becoming more and more important, Python has become a skill that is more in demand than ever; especially it can be integrated into web applications.

As an opensource project, Python is actively worked on with a moderate update cycle, pushing out new versions every year or so to make sure it remains relevant. In

terms of search volume for anyone interested in learning Python, it has skyrocketed to the 1st place when compared to other languages.

# GOOD TO KNOW

* + - The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
    - Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.
    - Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official [End Of Life date of January](https://pythonclock.org/)

[1, 2020](https://pythonclock.org/) has been established for Python 2, after which time it will no longer be maintained. Python is still maintained by a core development team at the

Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community.

* + - The name Python, by the way, derives not from the snake, but from the British comedy troupe [Monty Python’s Flying Circus](https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus), of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

# CHARACTERISTICS OF PYTHON:

Python is a well-designed language that can be used for real world programming. Python is a very high-level, dynamic, object-oriented, general purpose programming language that uses interpreter and can be used in a vast domain of applications. Python was designed to be easy to understand and use. Python is termed as a very user-friendly and beginner-friendly language in the recent times. Python has gained popularity for being a beginner-friendly language, and it has replaced Java as the most popular introductory language. As a dynamically typed language, Python is really flexible. Furthermore, Python is also more forgiving of errors, so you'll still be

able to compile and run your program until you hit the problematic part. Python is a flexible, simple coding programming language. This language can support different styles of programming including structural and object-oriented. Other styles can be used, too. Python is very flexible, because of its ability to use modular components that were designed in other programming languages. For example, you can write a program in PYTHON and import it to python as a module. Then add something else to it (for example design a GUI for it).

# FRAMEWORKS OF DEEP LEARNING

Deep learning framework is an interface, which allows us to build the deep learning models more easily and quickly, without getting the detailed algorithm. They provide clear and efficient way for defining models using collection of optimized components.

# TENSORFLOW

One of the best deep learning frameworks is TensorFlow. It has been adopted by several giants at scale such as Twitter and IMB essentially due to its highly flexible system architecture. TensorFlow is obtainable on both desktop and mobile and also provisions languages such as Python, PYTHON and R to generate deep learning models beside with packaging libraries. TensorFlow has two main tools. They are:

1. Tensor Board for effective data conception of network modeling and presentation.
2. TensorFlow Serving for express disposition of new algorithms/tests while retentive the same server architecture and APIs. It also delivers combination with other TensorFlow models which is altered from the unadventurous performs and can be prolonged to serve other model and data types.

# PYTORCH

PyTorch has seen a great level of approval within the deep learning framework communal. PyTorch is essentially a port to Torch deep learning framework used for creating deep neural networks and performing tensor computations that are high in terms of complication. PyTorch is a port to the Torch deep learning framework which can be recycled for building deep neural networks and implementing tensor computations. PyTorch is a Python package which delivers Tensor calculations. PyTorch uses vigorous computation graphs.

# KERAS

Keras is written in Python and can run on top of TensorFlow. Keras, on the other pointer, is a high-level API, established with attention to enable firm investigation. So rapid results are needed, Keras will spontaneously take care of the core tasks and produce the output. Both Convolutional Neural Networks and Recurrent Neural Networks are maintained by Keras. It runs faultlessly on CPUs as well as GPUs.

# PANDAS

Pandas is a high-performance, very easy-to-use, data-wrangling package. It has a special data type called Data frames, a type of in-memory data table. Pandas provide diverse data structures and functions to aid in working with structured data in a fast, easy, and expressive way. It is one of the important libraries enabling Python to be a powerful and productive Data Analytics Tool. Pandas combine high performance array-computing features of NumPy with the flexible data handling features of spread sheets and RDBMS (such as SQL). It has a sophisticated indexing functionality to make it easy to reshape, slice and dice, perform aggregations, and also select subsets of data. It contains well defined data structures and manipulation tools to make data analysis fast and easy.

# NUMPY

NumPy, created in 2005 by Travis Oliphant, is the fundamental package for scientific computing with Python. NumPy, which stands for Numerical Python, provides an array object that is up to fifty times faster than traditional lists. NumPy is majorly used for it as an efficient multi-dimensional container of generic data. Allowing NumPy to seamlessly and speedily integrate with a wide variety of databases. It is licensed under the BSD license, enabling reuse with few restrictions.

# MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. 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.

# SCIPY

Scipy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering. The Scipy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the Scipy library is that it should work with NumPy arrays. The combination of NumPy, Matplotlib and Scipy work as a replacement for Mat Lab, which is a popular platform for scientific computing. It easily provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. For simple plotting the pyplot module provides a Mat Lab-like interface. You also have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to Mat Lab users.

# DEEP LEARNING

As the name suggests, this is the whole new way of focusing on how do our brain and a human nervous system works. This Deep Learning is closely observing the neural system of a human being. This helps it to understand the neural system and communication better.

Through this we can get to know how a normal human brain thinks and we can use it to map a new algorithm for it so that we can solve a problem through a machine just as it has been solved by a human brain. Actually deep learning is persuaded from the biological process of nervous system to think better and solve better in a whole new way. It also focuses on how a brain recognizes process on the basis of an image.Deep Learning can also be seen as neural networks which have multi-layer architectures and very huge parameters on which it works.

# FUTURE ASPECTS

As machine learning and deep learning as data scientist are in trend in the market so every company is demanding a data scientist to hold their company at the market level. As machine learning and deep learning has proven themselves for solving problems in an amazing way so they have their future. And there are many more researchers all over the world trying to explore these two learning techniques to

their very core. These learning techniques will be used in future for analyzing the

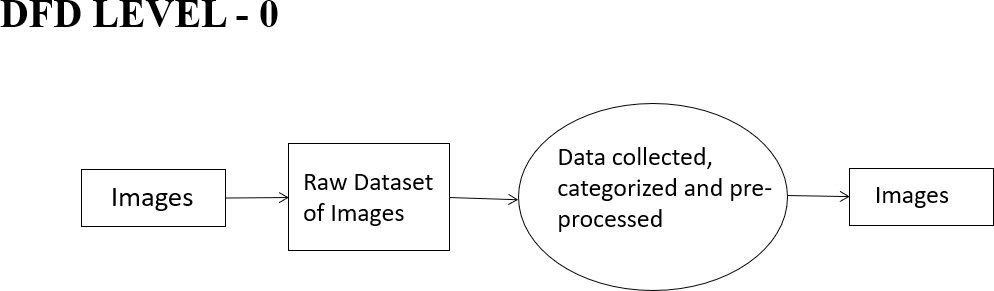
problem correctly and getting the result accordingly. As these techniques will give a bright future to artificial intelligence as well as neuroscience.

We have studied working of machine learning and deep learning and also studied the difference how the two leaning techniques work. So as we conclude we get to know that both of them are equally important in implementing artificial intelligence. So deep learning is a sub-set of machine learning which is further a sub- set of artificial intelligence. This comparison provides us a clear view of machine learning and deep learning

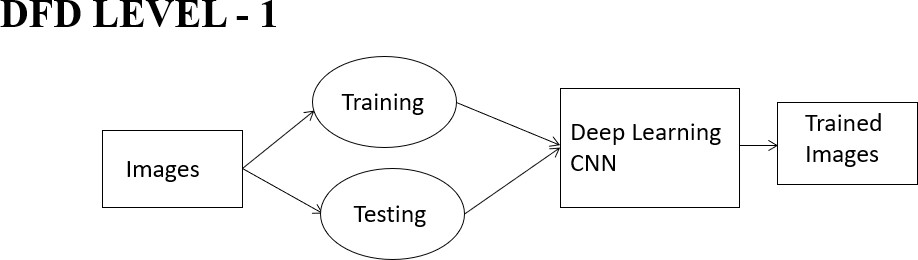
# SYSTEM DESIGN

* 1. **DATA-FLOW DIAGRAM**

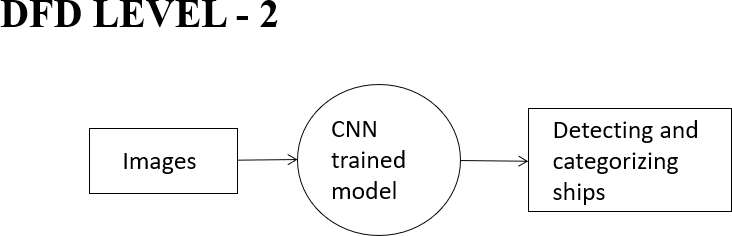
A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.



# 4.1 Dataflow diagram for categorization of ships (Level 0)



**4.1 Dataflow diagram for analyzing the images (Level 1)**



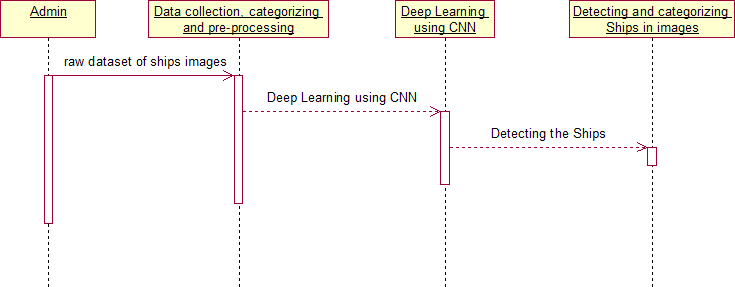
# Dataflow diagram for detecting and categorizing the ship images (Level 2)

* 1. **UML DIAGRAMS**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object- oriented software intensive system under development.

# SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the [logical view](https://en.wikipedia.org/wiki/4%2B1_architectural_view_model) of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



# Sequence diagramfor detecting and categorizing ships

* + 1. **USECASE DIAGRAM**

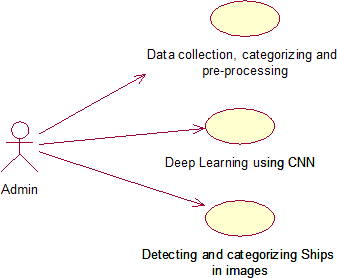
A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

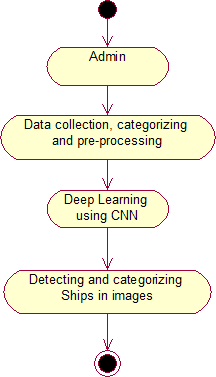
**Communication Link:** The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link. Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using message.



# Use case diagram for detecting and categorizing ships

* + 1. **ACTIVITY DIAGRAM**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

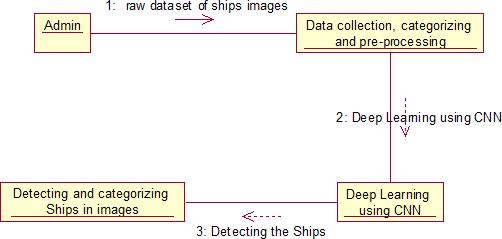


# Activity diagram for detecting and categorizing ships

* + 1. **COLLABORATION DIAGRAM**

A **collaboration diagram**, also known as a communication **diagram**, is an illustration of the relationships and interactions among software objects in the

Unified Modeling Language (UML). These **diagrams** can be used to portray the dynamic behavior of a particular use case and define the role of each object.



# 4.2.4 Collaboration diagram for detecting and categorizing ships

1. **SYSTEM ARCHITECTURE**

# ARCHITECTURE DIAGRAM

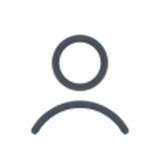
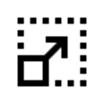
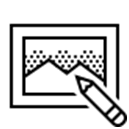
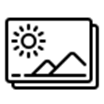


Image Collection

training

Image Pre- processing

Ship Categorization

Feature Extraction

Detecting & Categorizing

Visualization

Neural Network Deep Learning Model

Result

User

Web interface

# Architecture diagram for collecting images, categorizing

**and detecting of ships**

# MODULE DESIGN SPECIFICATION

* Data collection, categorizing and pre-processing
* Deep Learning using CNN
* Detecting and categorizing Ships in images.

# DATA COLLECTION, CATEGORIZING AND PRE-PROCESSING:

A High-Resolution Remote Sensing Image dataset was collected from Kaggle. Many images do not contain ships, and those that do may contain multiple ships. Ships within and across images may differ in size (sometimes significantly) and be located in open sea, at docks, marinas, etc.

A raw dataset was collected with type of ships and images for each type of ships. Since we did not have many number of ships images classified based on the categories, we use python to web crawl images from internet. The web scraped images were stored in different folders. For this project we were using four categories. We have downloaded ship images for the following categories, Accommodation, Container, War and Cruise. Python with vast text processing and networking libraries is the best tool to write one-off web scrapper.

# DEEP LEARNING USING CNN:

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep learning is a class of machine learning algorithms that use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each

successive layer uses the output from the previous layer as input. We use convolutional Neural Network in our project for training and detecting the ships in the given image. We use multilayer neural network and each layer output is given as input to the next layer.

# DETECTING AND CATEGORIZING SHIPS IN IMAGES:

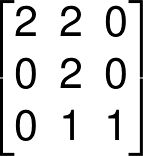
In this work required to locate ships in images, and put an aligned bounding box segment around the ships you locate.

The CNN trained model is used to detect the availability and the category of the ship in the given image. When the user inputs the images to for detection, the prediction method in the trained model is used to detect the presence of the ship, and also the category of the ship. A new image file is created with the category written on top of the actual image.

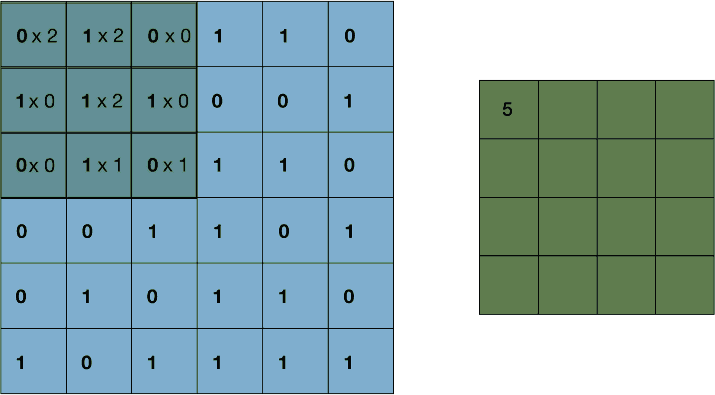
# THE CONVOLUTION ALGORITHM

The convolution is a kind of product operation of a filter also called a kernel with a matrix of image to extract from it some pre-determined characteristics.

Literally-speaking, we use a convolution filter to “filter” the image and display only what really matter to us. The considered image is a matrix, the filters used are also matrices, generally 3x3 or 5x5. Let’s see how convolution works with the following kernel,



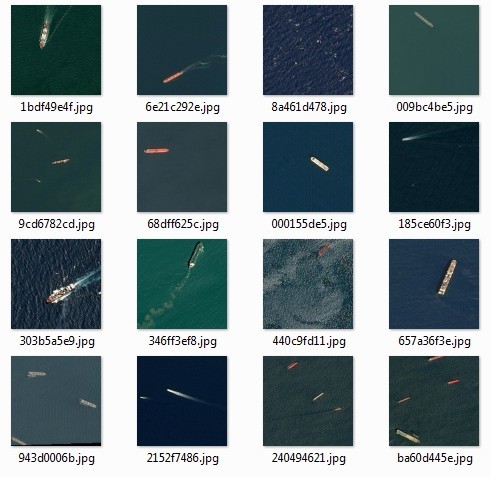
The 6x6px matrix represents an image. At the beginning, the convolution kernel, here the 3x3 matrix is positioned on the top-left corner of the matrix image, the kernel then covers a part of this matrix image, we then make a product element by element (element-wise) of the two overlapping blocks we eventually sum these products and the final result corresponds to a pixel of the output image.



Then, we move the convolution kernel from horizontally to the right by one pixel, we make a new element-wise product then added up to get a new coefficient of the output image. Once at the end of a line, the kernel makes a vertical stride down and starts again from the left, we iterate likewise until the kernel has covered all the matrix image. It is important to note that the kernel always remains on the initial matrix, without overflowing. For sure, we cannot use any filter, the coefficients of our kernel will depend on the features we want the filter to highlight. Let’s see the result of a convolution with some well-known filters.

# METHODOLOGY

The dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. It includes 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. Image chips were derived from Planet Scope full-frame visual scene products, which are orthorectified to a 3-meter pixel size.



# Synthetic Aperture Radar (SAR) Ship image data set

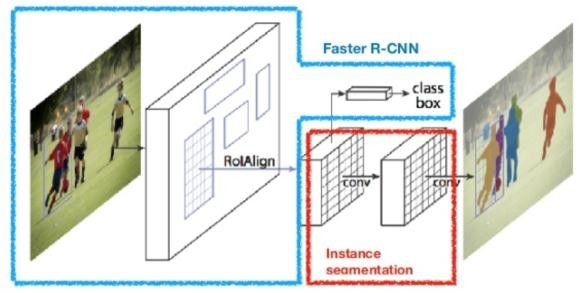
1. **MASK RCNN**

The Mask RCNN framework was created by Face book’s AI Research team or FAIR in 2017. This relatively new Framework is an extension of Faster RCNN. So, just like Fast RCNN and Faster RCNN, Mask RCNN is also a deep neural network. Mask RCNN solves the problems of instance segmentation in machine learning and computer vision

The process of Mask RCNN can be broken down into two steps:

* + Generation of potential regions of interest using RPN and RoI Align and also using the Ground Truths.
  + Prediction of the class of the object, the bounding box and the mask in the pixel

level.

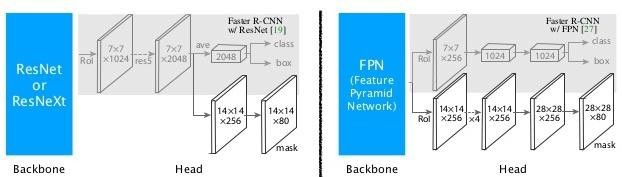


# Mask R-CNN

1. **WORKING OF FAST RCNN IN SHORT**
   * Feature Extraction: Faster RCNN makes use of CovNet for generating feature maps of the image.
   * Propose Potential Regions: The feature maps generated from CovNet are further passed to the RPN (Region Proposal Network) where the bounding boxes over the regions are returned.
   * Making it uniform: These regions and bounding boxes are then passed to RoI (Region of Interest) pooling and RoI Align where they are brought down to the same size. This helps further makes computations easier and faster.
   * FC Layers: Fully Connected layers or FC layers are the final steps in Faster RCNN, where the proposals are passed, and classification takes place. Outputs are bounding boxes over the objects.

# BACKBONE MODEL

The backbone of Mask RCNN is the ResNet 101 architecture. ResNet 101 to Mask RCNN is the same as what CovNet is to Faster RCNN. ResNet 101 extracts features from images and generates feature maps. A FPN or Feature Pyramid Network is formed with the help of these feature maps. These feature maps and the FPN are then passed over to the next layer



# SEGMENTATION MASK:

After computing the RoI, we have to compute the IoU over all of the predicted regions. IoU stands for Intersection over Union and is calculated with the help of ground truths.

IoU = Area of the intersection / Area of the union

When the value of IoU equals 1, it implies that the predicted boxes overlap perfectly with the ground truth boxes. To make things a little less rigid, Non-Max Suppression is applied, where we basically consider all the predicted boxes with IoU > 0.5. The rest of the boxes are removed. In case of same objects, it will choose the box with the highest value of IoU and will discard the rest

This completes the process of Mask RCNN, where we get the masks for the objects in the image. Mask RCNN takes a lot of time to train. On average it takes around a couple of days to be completely trained. Therefore, we took help from the pretrained weights of the COCO dataset trained on the Mask RCNN model.

# CONVOLUTIONAL NEURAL NETWORK:

1. **REGIONAL CNN**
   * In this chapter we will talk about the region-based methods for object detection.
   * In particular, r-cnn (regional cnn), the original application of cnns to this problem, along with its descendants fast r-cnn, and faster r-cnn.
   * In classification, there’s generally an image with a single object as the focus and the task is to say what that image is. But when we look at the world around us, we carry out far more complex tasks. We see complicated sights with multiple overlapping objects, and different backgrounds and we not only classify these different objects but also identify their boundaries, differences, and relations to one another.
   * The goal of a region based cnn is to take an image input to detected and localize the object in the image. All three above mentioned architectures were used for object detection and localization for a driving scenario to detect cars, pedestrians and traffic signs etc. And performed a time study to conclude which of the three architectures can be used as a detector for a real-world driving scenario for detection of objects present on road.
   * We will also discuss their architecture and describe how they perform the task of object detection and localization given the input image.

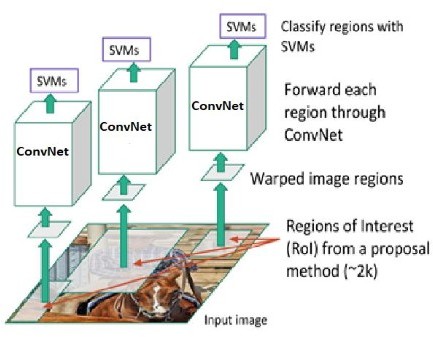
# R-CNN

R-CNN was first introduced by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik in 2014. Region based Convolutional Network(R-CNN) combines two ideas: (1) one can apply high-capacity Convolutional Networks (CNN’S) to bottom-up region proposals in order to localize and segment objects and (2) When

labeling data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific-fine-tuning, boosts performance significantly. They combined region proposals with CNNs and called the resulting model an R-CNN or Region-based Convolutional Network.

# R-CNN Architecture

In R-CNN we input an image and then using the basic feature detection techniques such as edge detection etc. we get Region Proposals Also referred as Regions of interest. This process is also known as selective search.



# SYSTEM IMPLEMENTATION

* 1. **SHIP DETECTION**

import numpy as np

import pandas as pd import warnings

warnings.filterwarnings('ignore') import seaborn as sns

from matplotlib import pyplot as plt import json, sys, random

from PIL import Image, ImageDraw from keras.models import Sequential

from keras.layers import Dense, Flatten, Activation from keras.layers import Dropout

from keras.layers.convolutional import Conv2D, MaxPooling2D from keras.utils import np\_utils

from keras.optimizers import SGD import keras.callbacks

f = open(r'shipsnet.json') dataset = json.load(f) f.close()

input\_data = np.array(dataset['data']).astype('uint8') output\_data = np.array(dataset['labels']).astype('uint8') input\_data.shape

n\_spectrum = 3 # color chanel (RGB) weight = 80

height = 80

X = input\_data.reshape([-1, n\_spectrum, weight, height])

X[0].shape pic = X[0]

rad\_spectrum = pic[0] green\_spectrum = pic[1] blue\_spectum = pic[2] plt.figure(2, figsize = (5\*3, 5\*1)) plt.set\_cmap('jet')

plt.subplot(1, 3, 1) plt.imshow(rad\_spectrum) plt.subplot(1, 3, 2) plt.imshow(green\_spectrum) plt.subplot(1, 3, 3) plt.imshow(blue\_spectum) plt.show()

with open(r'shipsnet.json') as data\_file: dataset = json.load(data\_file)

Shipsnet= pd.DataFrame(dataset) print(Shipsnet.head())

print('')

x = np.array(dataset['data']).astype('uint8') y = np.array(dataset['labels']).astype('uint8') def describeData(a,b):

print('Total number of images: {}'.format(len(a))) print('Number of NoShip Images: {}'.format(np.sum(b==0))) print('Number of Ship Images: {}'.format(np.sum(b==1)))

print('Percentage of positive images: {:.2f}%'.format(100\*np.mean(b))) print('Image shape (Width, Height, Channels): {}'.format(a[0].shape))

describeData(x,y)

from keras.utils import to\_categorical

xReshaped = x.reshape([-1, 3, 80, 80]).transpose([0,2,3,1]) yReshaped = to\_categorical(y, num\_classes=2) print("Data Shape",x.shape)

print('Labels Shape',y.shape)

print('Reshaped Data Shape',xReshaped.shape) print('Reshaped Labels Shape',yReshaped.shape)

def describeDataset(features,labels): print("\n'X' shape: %s."%(features.shape,)) print("\n'y' shape: %s."%(labels.shape,))

print("\nUnique elements in y: %s"%(np.unique(y))) describeDataset(xReshaped,yReshaped)

imgs0 = xReshaped[y==0] imgs1 = xReshaped[y==1]

def plotOne(a,b): """

Plot one numpy array """

plt.subplot(1,2,1) plt.title('Not A Ship') plt.imshow(a[100]) plt.subplot(1,2,2) plt.title('Ship') plt.imshow(b[100])

plotOne(imgs0, imgs1)

def plotTwo(a,b): """

Plot a bunch of numpy arrays sorted by label """

for row in range(3): plt.figure(figsize=(20, 10)) for col in range(3):

plt.subplot(1,8,col+1) plt.title('Not A Ship') plt.imshow(a[row+col]) plt.axis('off') plt.subplot(1,8,col+4) plt.title('Ship') plt.imshow(b[row+col]) plt.axis('off')

plotTwo(imgs0, imgs1)

output\_data.shape output\_data

sns.countplot(output\_data) np.bincount(output\_data)

y = np\_utils.to\_categorical(output\_data, 2)

indexes = np.arange(2800) np.random.shuffle(indexes)

X\_train = X[indexes].transpose([0,2,3,1]) y\_train = y[indexes]

X\_train = X\_train / 255 np.random.seed(42) model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same', input\_shape=(80, 80, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #40x40 model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #20x20

model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #10x10 model.add(Dropout(0.25))

model.add(Conv2D(32, (10, 10), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #5x5 model.add(Dropout(0.25))

model.add(Flatten()) model.add(Dense(512, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(2, activation='softmax')) model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same', input\_shape=(80, 80, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #40x40 model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #20x20 model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #10x10 model.add(Dropout(0.25))

model.add(Conv2D(32, (10, 10), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2))) #5x5 model.add(Dropout(0.25))

model.add(Flatten()) model.add(Dense(512, activation='relu')) model.add(Dropout(0.5))

model.add(Dense(2, activation='softmax'))

sgd = SGD(lr=0.01, momentum=0.9, nesterov=True)

model.compile( loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

model.fit( X\_train, y\_train, batch\_size=32, epochs=2,

validation\_split=0.2, shuffle=True, verbose=2)

image = Image.open(r'Dataset/00e90efc3.jpg') pix = image.load()

n\_spectrum = 3 width = image.size[0]

height = image.size[1]

picture\_vector = []

for chanel in range(n\_spectrum): for y in range(height):

for x in range(width): picture\_vector.append(pix[x, y][chanel])

picture\_vector = np.array(picture\_vector).astype('uint8')

picture\_tensor = picture\_vector.reshape([n\_spectrum, height, width]).transpose(1, 2, 0)

plt.figure(1, figsize = (15, 30))

plt.subplot(3, 1, 1) plt.imshow(picture\_tensor) plt.show()

picture\_tensor = picture\_tensor.transpose(2,0,1)

def cutting(x, y):

area\_study = np.arange(3\*80\*80).reshape(3, 80, 80)

for i in range(80): for j in range(80):

area\_study[0][i][j] = picture\_tensor[0][y+i][x+j] area\_study[1][i][j] = picture\_tensor[1][y+i][x+j] area\_study[2][i][j] = picture\_tensor[2][y+i][x+j]

area\_study = area\_study.reshape([-1, 3, 80, 80]) area\_study = area\_study.transpose([0,2,3,1]) area\_study = area\_study / 255 sys.stdout.write('\rX:{0} Y:{1} '.format(x, y)) return area\_study

def not\_near(x, y, s, coordinates): result = True

for e in coordinates:

if x+s > e[0][0] and x-s < e[0][0] and y+s > e[0][1] and y-s < e[0][1]: result = False

return result

def show\_ship(x, y, acc, thickness=5): for i in range(80):

for ch in range(3):

for th in range(thickness): picture\_tensor[ch][y+i][x-th] = -1

for i in range(80): for ch in range(3):

for th in range(thickness): picture\_tensor[ch][y+i][x+th+80] = -1

for i in range(80): for ch in range(3):

for th in range(thickness): picture\_tensor[ch][y-th][x+i] = -1

for i in range(80): for ch in range(3):

for th in range(thickness): picture\_tensor[ch][y+th+80][x+i] = -1

step = 10; coordinates = []

for y in range(int((height-(80-step))/step)): for x in range(int((width-(80-step))/step) ):

area = cutting(x\*step, y\*step) result = model.predict(area)

if result[0][1] > 0.90 and not\_near(x\*step,y\*step, 88, coordinates): coordinates.append([[x\*step, y\*step], result])

print(result) plt.imshow(area[0]) plt.show()

for e in coordinates:

show\_ship(e[0][0], e[0][1], e[1][0][1])

picture\_tensor = picture\_tensor.transpose(1,2,0) picture\_tensor.shape

plt.figure(1, figsize = (15, 30)) plt.subplot(3,1,1) plt.imshow(picture\_tensor) plt.show()

# 6.3 SHIP CATEGORIZATION

from flask import Flask, render\_template, request,send\_from\_directory,url\_for,redirect,send\_file from werkzeug import secu re\_filename

import os

from predictf import predict

UPLOAD\_FOLDER = 'uploads'

ALLOWED\_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif']) PEOPLE\_FOLDER = os.path.join('uploads')

app = Flask( name ) app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER app.config['UPLOADS\_FOLDER'] = PEOPLE\_FOLDER

def allowed\_file(filename): return '.' in filename and \

filename.rsplit('.', 1)[1] in ALLOWED\_EXTENSIONS

@app.route('/')

def hello\_world():

return render\_template ('index.html') @app.route('/upload')

def upload\_file():

return render\_template('upload.html')

@app.route('/uploader', methods = ['GET', 'POST']) def upload\_ifile():

if request.method == 'POST': file = request.files['file']

if file and allowed\_file(file.filename): filename = secure\_filename(file.filename)

file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename)) return redirect(url\_for('uploaded\_file', filename=filename))

file.save(secure\_filename(file.filename))

return render\_template('up.html') #'file uploaded successfully'

@app.route('/show/<filename>') def uploaded\_file(filename):

result= predict(filename)

print("done")

return render\_template('result.html', fname=filename,result=result)

@app.route('/upload/<filename>') def send\_file1(filename):

return send\_from\_directory(UPLOAD\_FOLDER, filename)

@app.route('/getImage',methods=['GET']) def getImage():

fname =request.args.get('fn')

return send\_file('./uploads/'+fname, attachment\_filename='python.jpg') if name == ' main ':

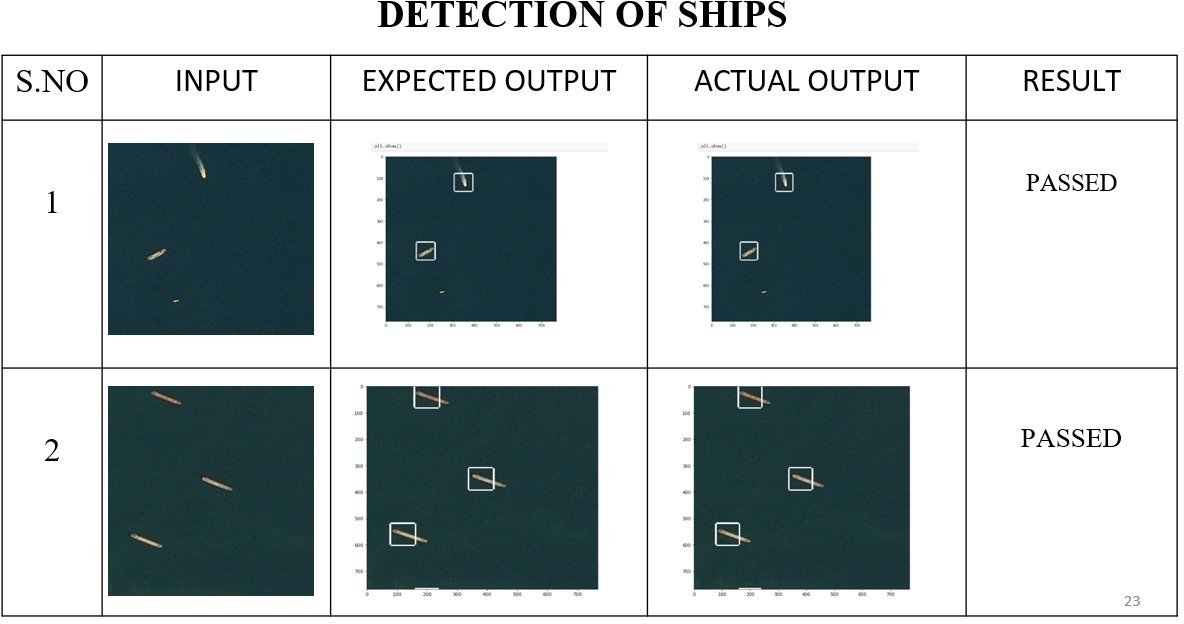
app.run(debug = True)

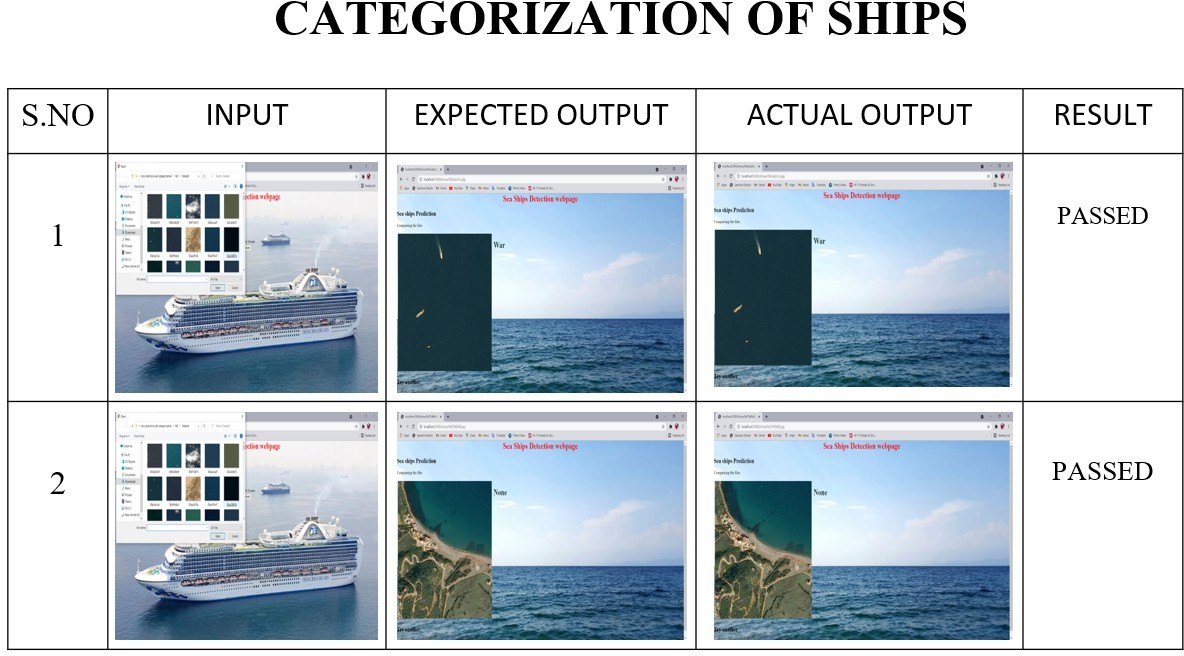
# SYSTEM TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It

verifies that the whole set of programs hang together. System testing requires a test consisting of several key activities and steps for running a program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design be detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough. Testing is one of the important steps in the software development phase. Testing check for the errors, as a whole of the project testing involves the following test cases: Static analysis is used to investigate the structural properties of the Source. Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

* 1. **TEST CASES AND REPORTS**





# 8. CONCLUSION

In this article, we investigate the constraints of thick dissemination and various sizes. Therefore, we propose R- Libra R-CNN, a rotational Libra R-CNN for transport discovery. BFP module and IOU-BS module are presented with the inspiration of separating discriminative highlights to conquer the effect of various sizes. Moreover, the rotational district location with adjusted L1 misfortune is

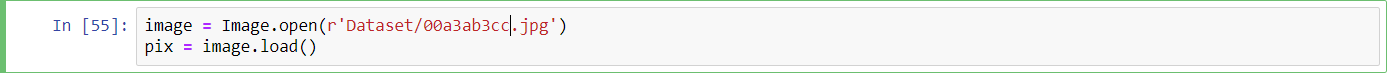
at long last proposed to be hearty against the effect of thick appropriation. Our strategy can acquire reliable upgrades in exactness and perception. What's more, broad tests on the DOTA show that the proposed technique can acquire 3.43% than R2CNN and 4.09% than Libra R-CNN. Alongside transport recognizable proof, we propose

boat characterization framework dependent on the structure and class of boats. The proposed framework would distinguish a boat, yet would likewise characterize it as a warship, load boat and journey transport. It would decrease the quantity of bogus cautions on the coast. We are proposing alongside transport location, a boat order dependent on the kind and classification of the boats. The proposed framework won't just distinguish a transport yet additionally arrange as war transport, holder transport and so on It will decrease the on shore bogus alerts.

# APPENDICES

* 1. **SAMPLE SCREENS**

# STEP 1:



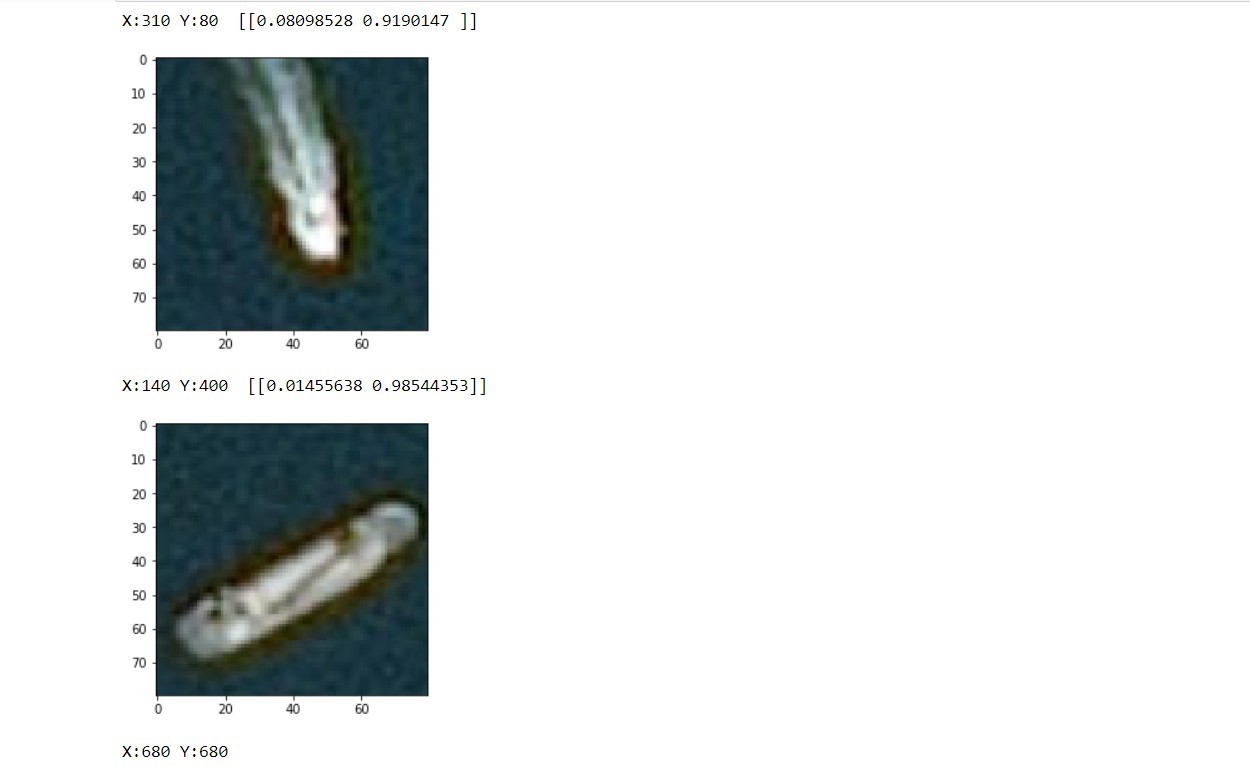
* + 1. **Screenshot of image input**

# STEP 2:



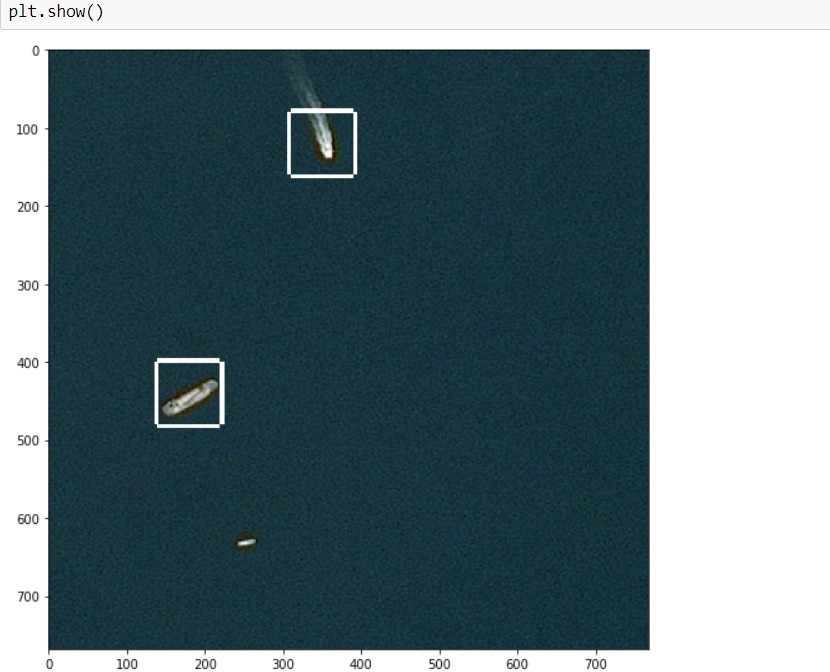
* + 1. **Screenshot of ship detected**

# STEP 3:



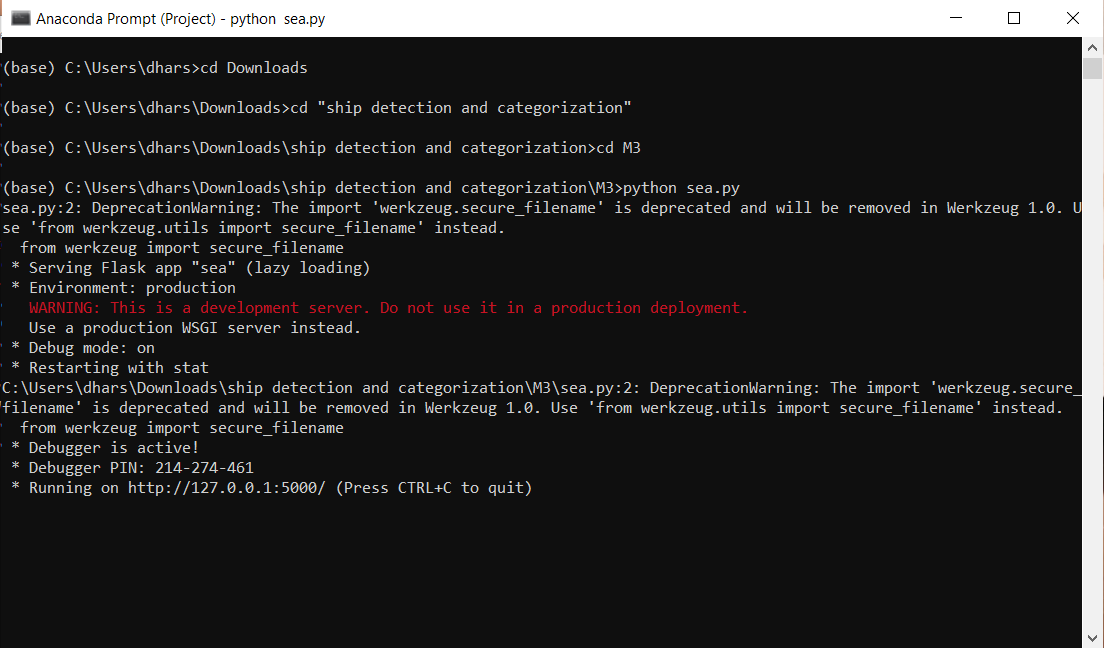
**STEP 4:**

* + 1. **Screenshot of x-axis and y-axis of detected ship**



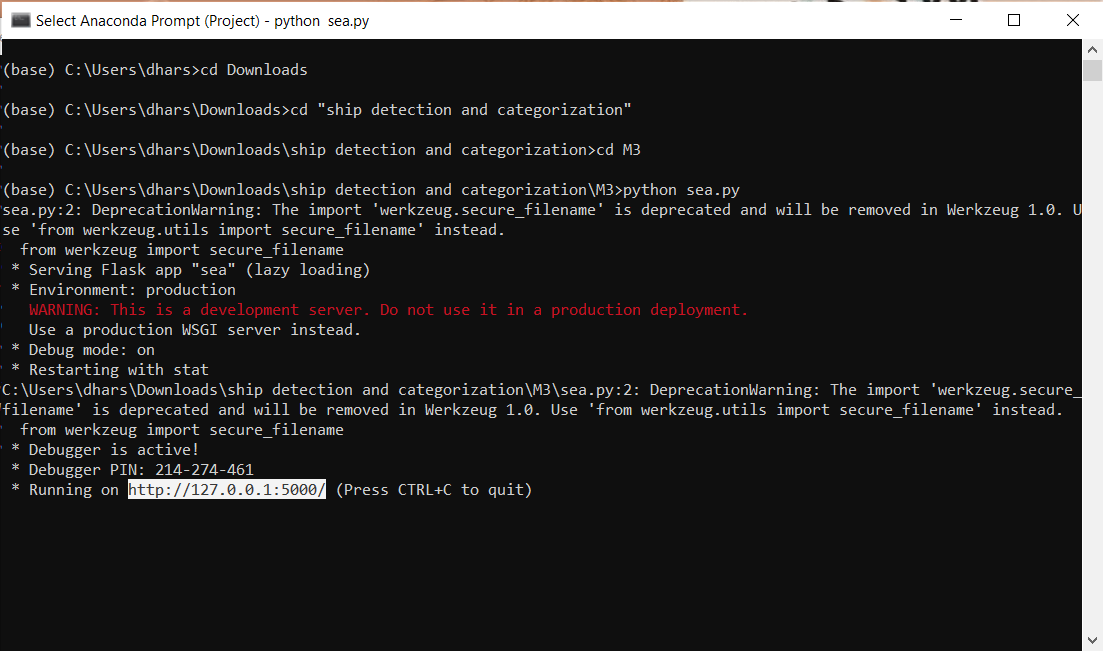
# STEP 5:

* + 1. **Screenshot of rectangular box around the detected ship**



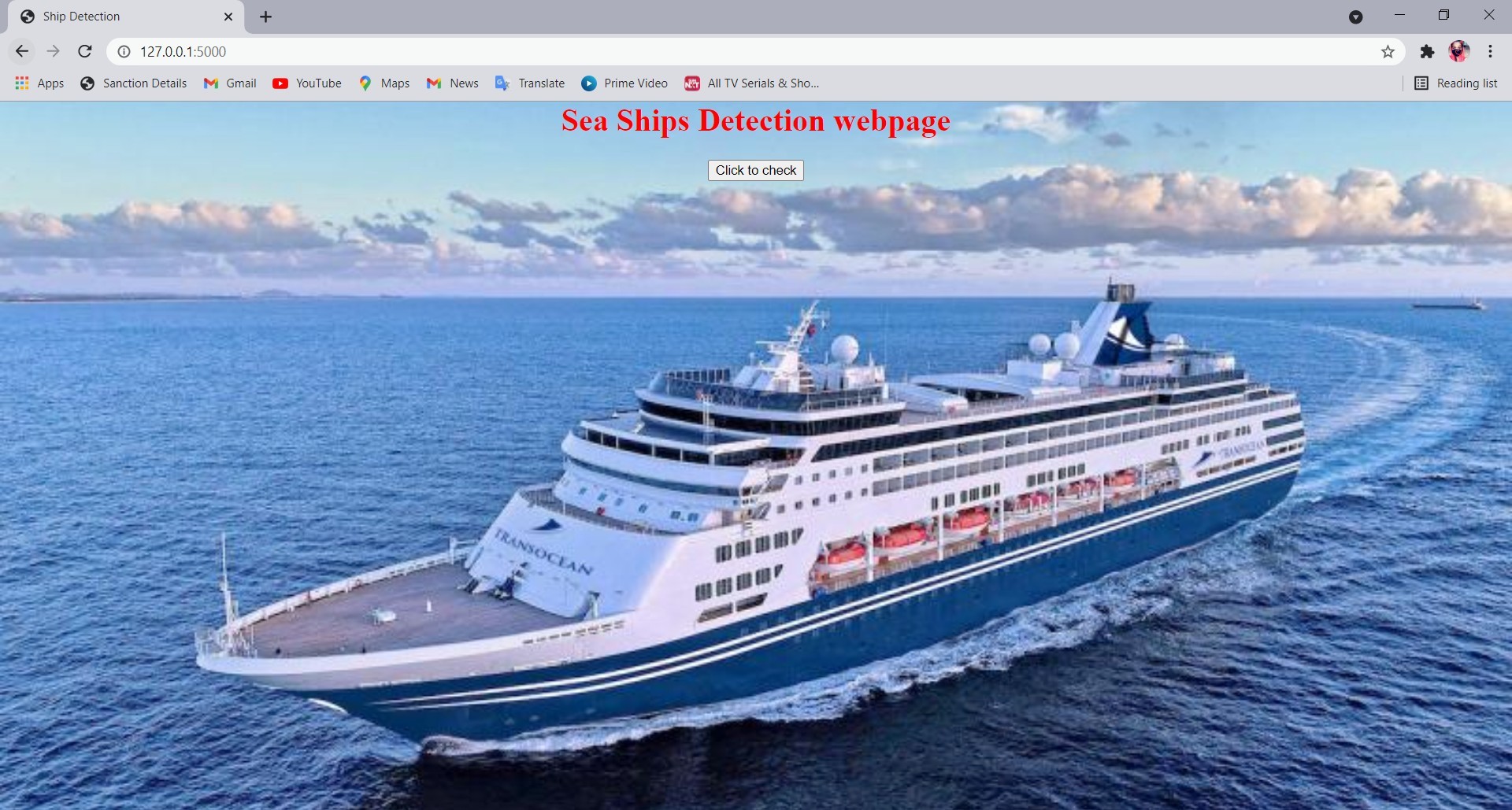
* + 1. **Screenshot to run the categorization code**

# STEP 6:



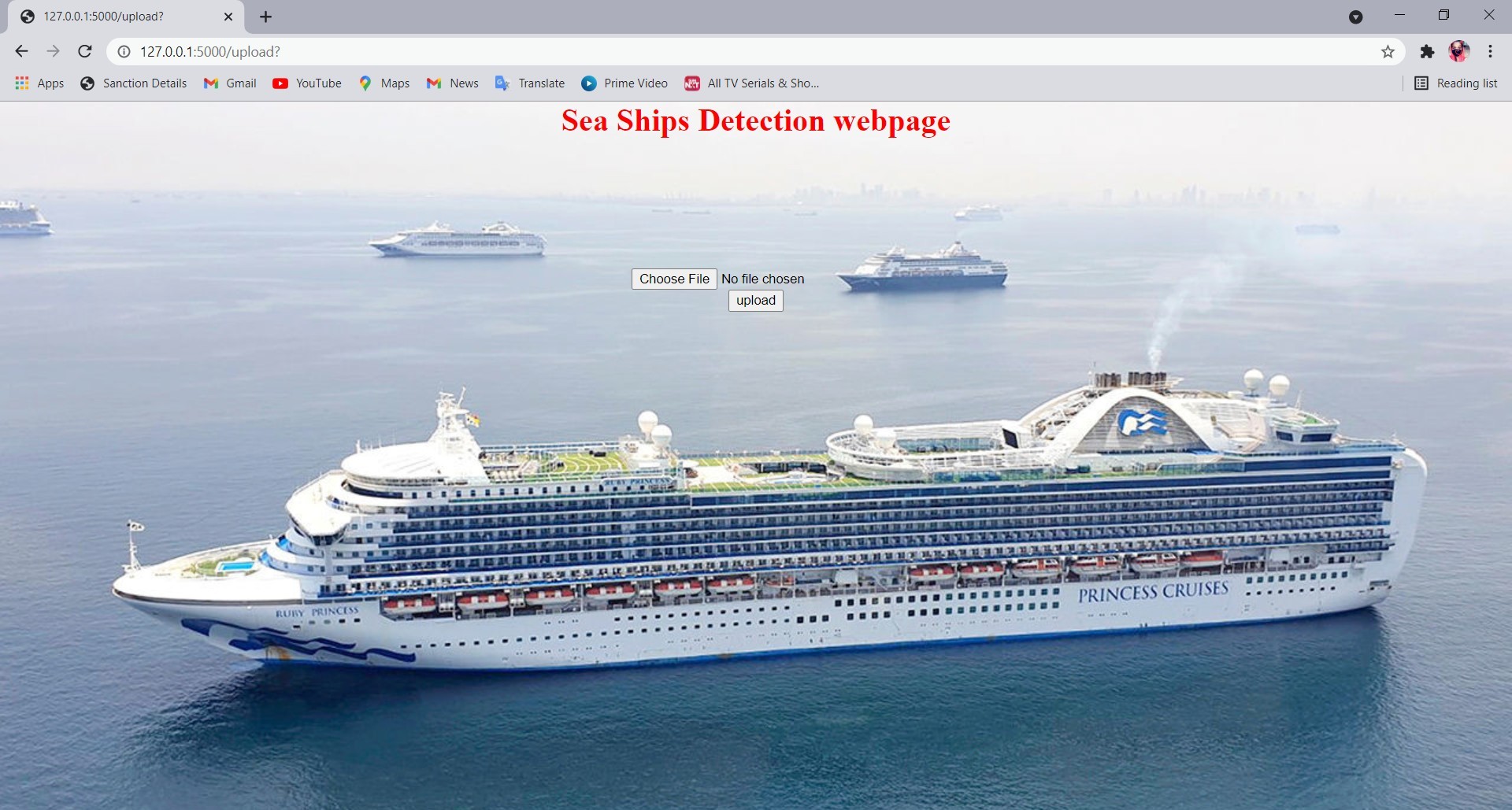
**STEP 7:**

* + 1. **Screenshot of Getting the URL for categorizing the ships**



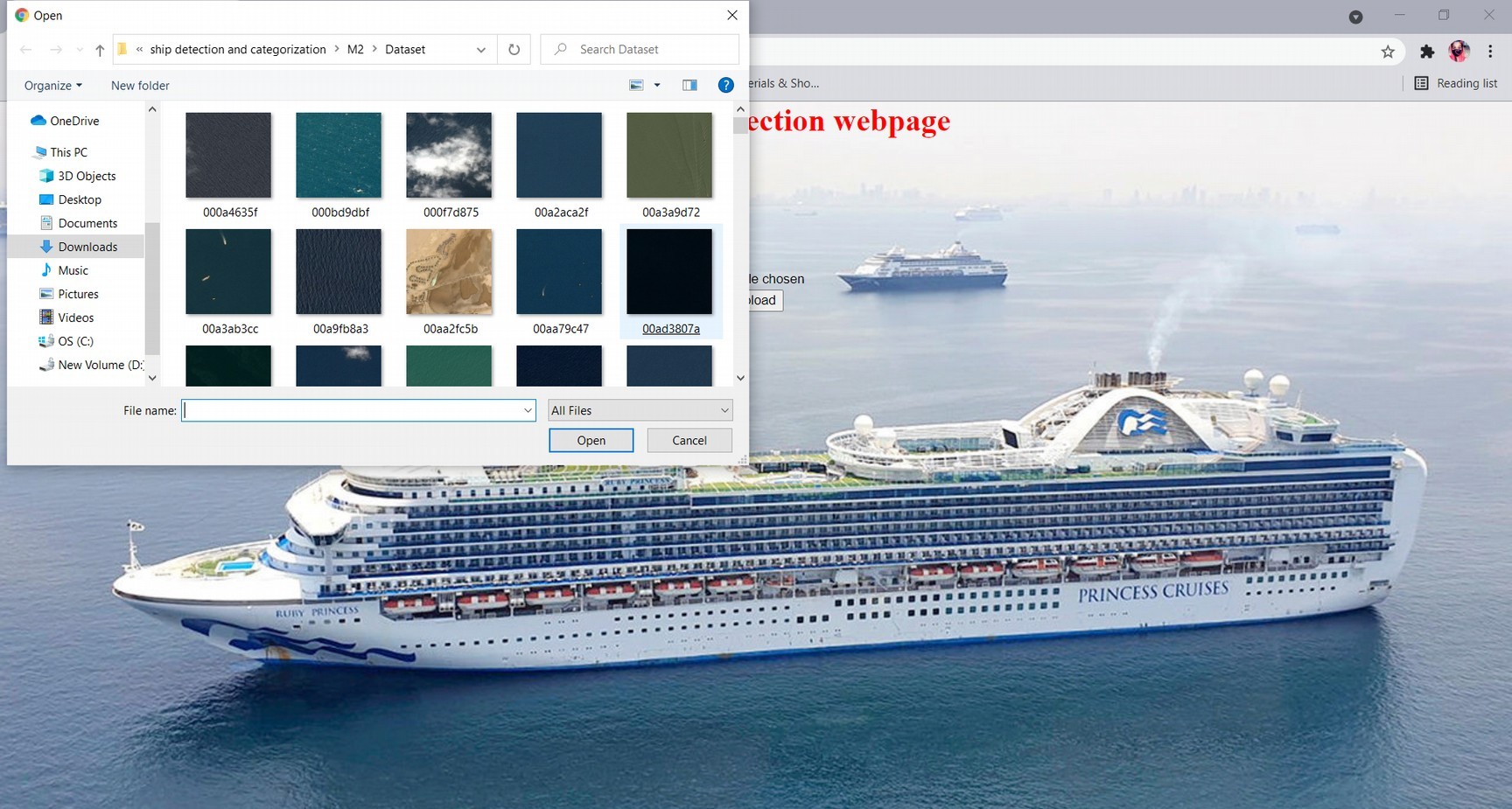
# STEP 8:

* + 1. **Screenshot of web page for categorizing the ships**



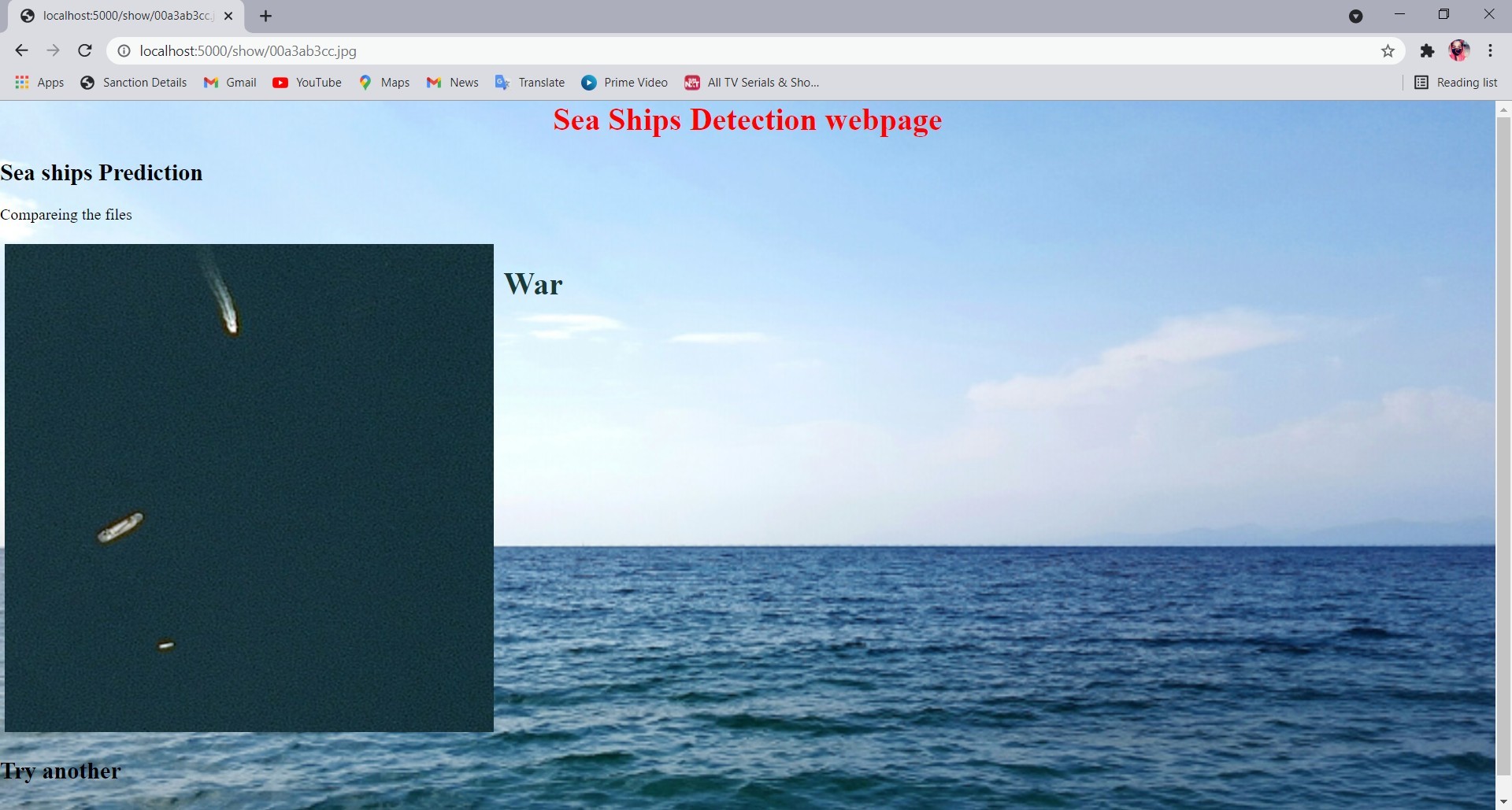
# STEP 9:

* + 1. **Screenshot of image uploading page**



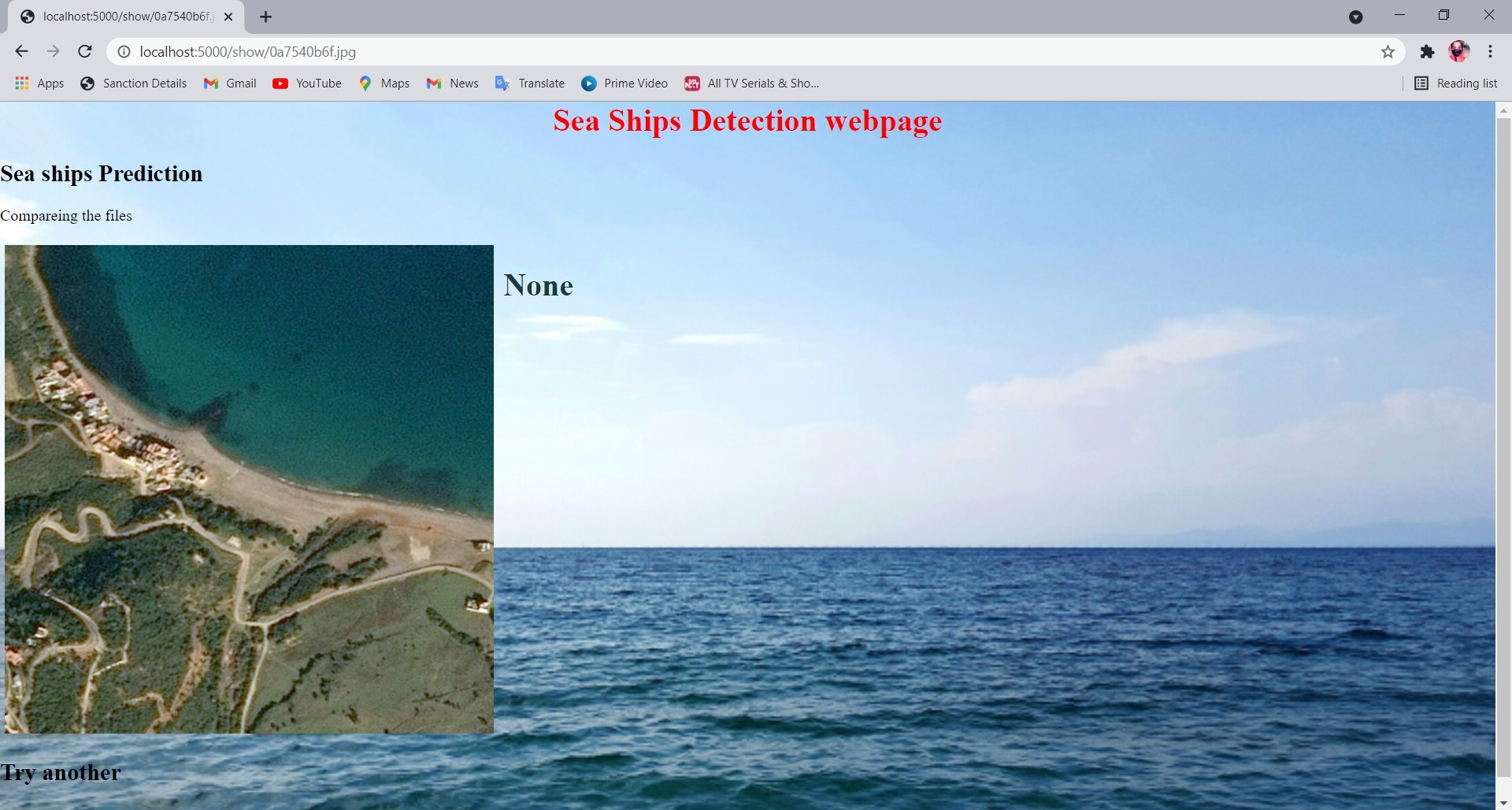
* + 1. **Screenshot of choosing the image**

# STEP 10:



**STEP 11:**

* + 1. **Screenshot for categorizing the detected ships**



* + 1. **Screenshot of no ships detected**

# PUBLICATION

This project was published as a paper in Journal of Emerging Technologies and Innovative Research on 25.04.2021 by Hemamalini R, Dharshini A, Jayashree V. **Published issue**: Vol 8 issue 4, April 2021

**DOI Details**: https://[www.jetir.org/view?paper=JETIRES06024](http://www.jetir.org/view?paper=JETIRES06024)

# REFERENCES

1. D. Zeiler, Graham W. Taylor and Rob Fergus,”Adaptive Deconvolutional Networks For Mid And High Level Feature,”International Conference on Computer Vision, 2016
2. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,”Spatial Pyramid Pooling In Deep Convolutional Networks For Visual Recognition,”IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 April 2015
3. Karen Simonyan & Andrew Zisserman,” Very Deep Convolutional Networks for Large-Scale Image Recognition,” Computer Vision and Pattern Recognition (cs.CV) ,2015
4. Olaf Ronneberger, Philipp Fischer, and Thomas Brox,” U-Net: Convolutional Networks for Biomedical Image Segmentation,” International Conference on Medical Image Computing and Computer-Assisted Intervention,18 May 2015
5. Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla,” Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 10 Oct 2016
6. Evan Shelhamer\_, Jonathan Long\_, and Trevor Darrell” Fully Convolutional Networks for Semantic Segmentation,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),”20 May 2016
7. Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu ”Object Detection with Deep Learning,” IEEE Transactions on Neural Networks and Learning Systems,16 Apr 2019
8. Shaoqing Ren\_ Kaiming He Ross Girshick Jian Sun,’’Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks,’’Computer Vision and Pattern Recognition,2019
9. Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun,’’Deep Residual Learning for Image Recognition,”IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2015
10. Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold W. M. Smeulders,’’Selective Searchfor Object Recognition,’’International Conference on Computer Vision,2012