B. Tech Capstone Project - Review 2

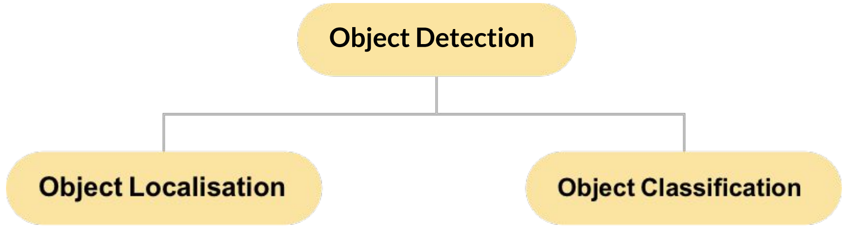
Remote Sensing, Object Detection & Classification using Deep Learning

Centre for Airborne Systems, DRDO, Bangalore

Ananya K R - 17BEC0857 - SENSE
Guide - Dr. Valarmathi, Dr. Rajesh (DRDO), Dr. Dhipu (DRDO)

Current Status and Motivation

1. The problem definition of object detection is to determine where objects are located in a given image (object localization) and which category each object belongs to (object classification).



2. My responsibilities at DRDO include -
   a. Understanding the existing methodologies to perform object detection **(reading base papers)**
   b. Implementation of these methods on small-scale databases
   c. Observing results, playing with parameters and fine-tuning models
   d. Extension of the proposed model to a large-scale dataset (MARVEL)
   e. Publishing novel results

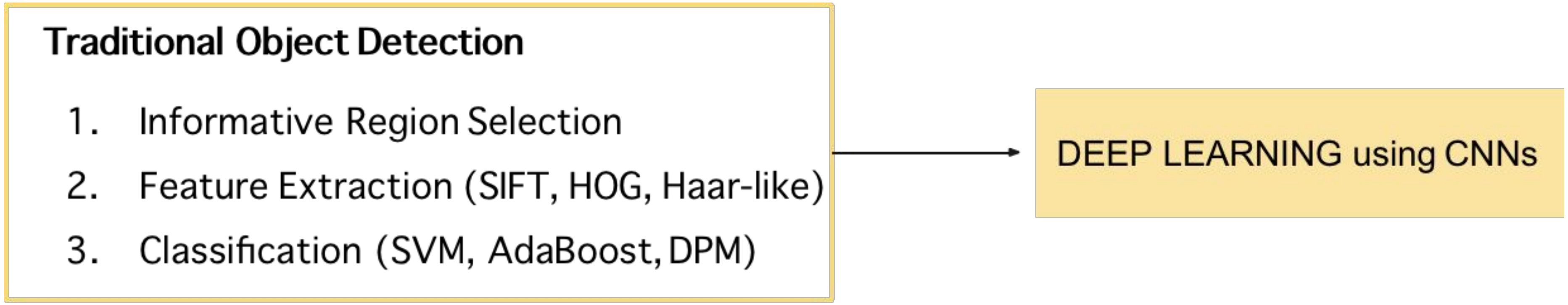
Object Detection - Traditional Methods Literature Survey

Informative region selection. As different objects may appear in any positions of the image and have different aspect ratios or sizes, it is a natural choice to **scan the whole image with a multi-scale sliding window**. Disadvantages - computationally expensive, numerous redundant windows.

Feature extraction. To recognize different objects, we need to **extract visual features** which can provide a semantic and robust representation. Disadvantages - Due to the diversity of appearances, illumination conditions and backgrounds, it's difficult to **manually design a robust feature descriptor** to perfectly describe all kinds of objects.

Classification. A classifier is needed to classify the existing crops using the obtained feature descriptor values and assign a crop to a given class. Simultaneously, drawing a bounding box around an object in a given image. Disadvantages - These **models need far more information about a class** and so, tedious tweaking is needed to get good results.

When Deep Learning came into the picture!



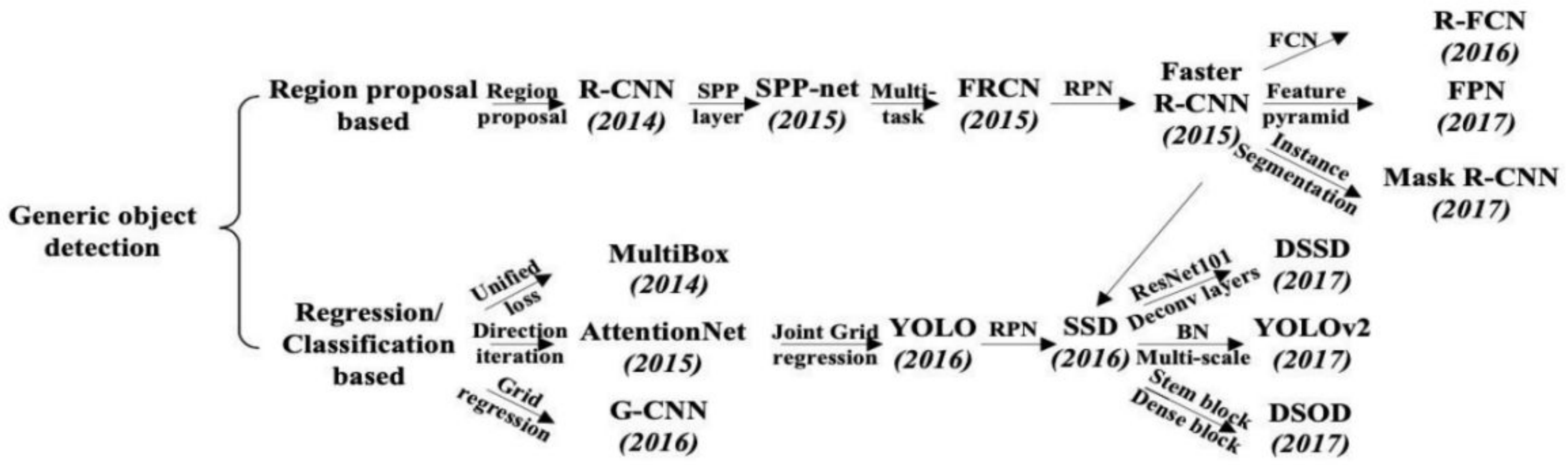
Traditional Object Detection

1. Informative Region Selection
2. Feature Extraction (SIFT, HOG, Haar-like)
3. Classification (SVM, AdaBoost, DPM)

$\longrightarrow$

DEEP LEARNING using CNNs

The Transition!



My region of Interest - R-CNN, Mask R-CNN, YOLO

Project #1 - Implementation of R-CNN

Software - Tensorflow, Keras, Google Colab

R-CNNs (Region-Based CNNs)

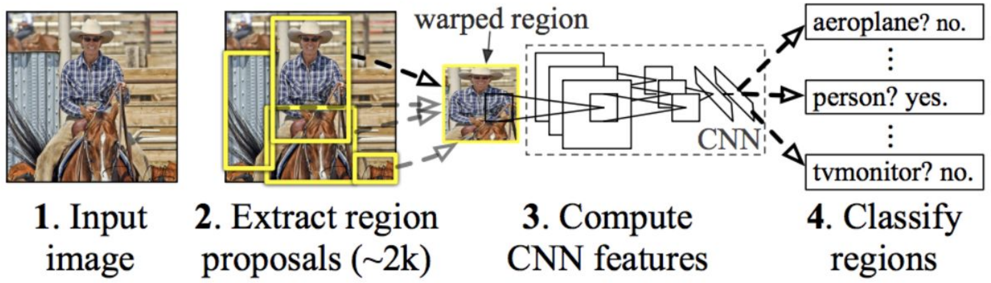
One of the first large and successful application of convolutional neural networks to the problem of object localization, detection, and segmentation. The approach was demonstrated on benchmark datasets, achieving then state-of-the-art results on the **VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset**.

Their proposed R-CNN model is comprised of three modules; they are:

Module 1: Region Proposal. Generate and extract category independent region proposals, e.g. candidate bounding boxes.

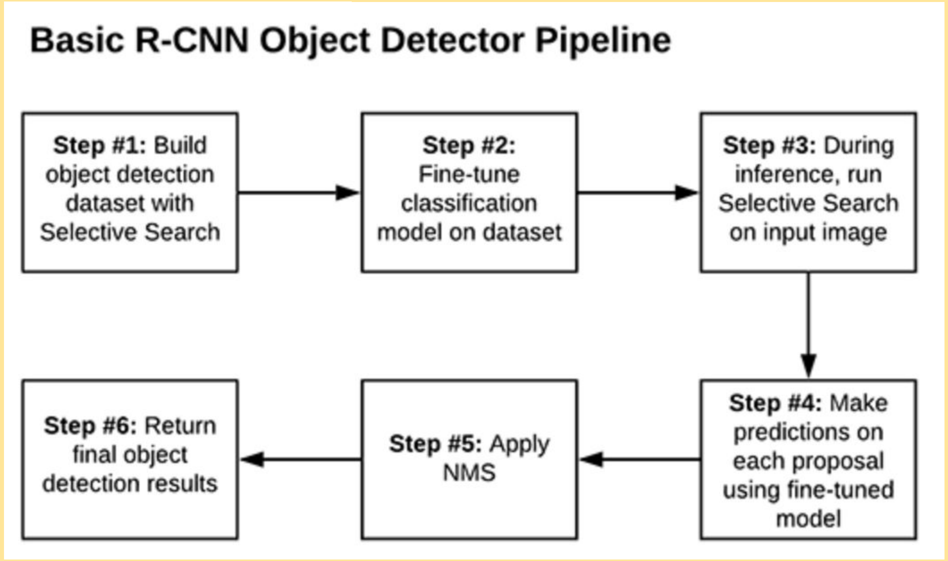
Module 2: Feature Extractor. Extract feature from each candidate region, e.g. using a deep convolutional neural network.

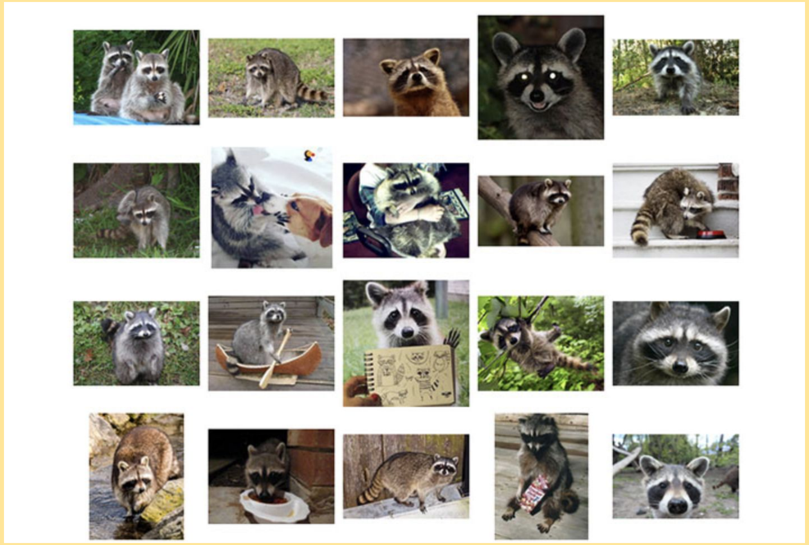
Module 3: Classifier. Classify features as one of the known class, e.g. linear SVM classifier model.



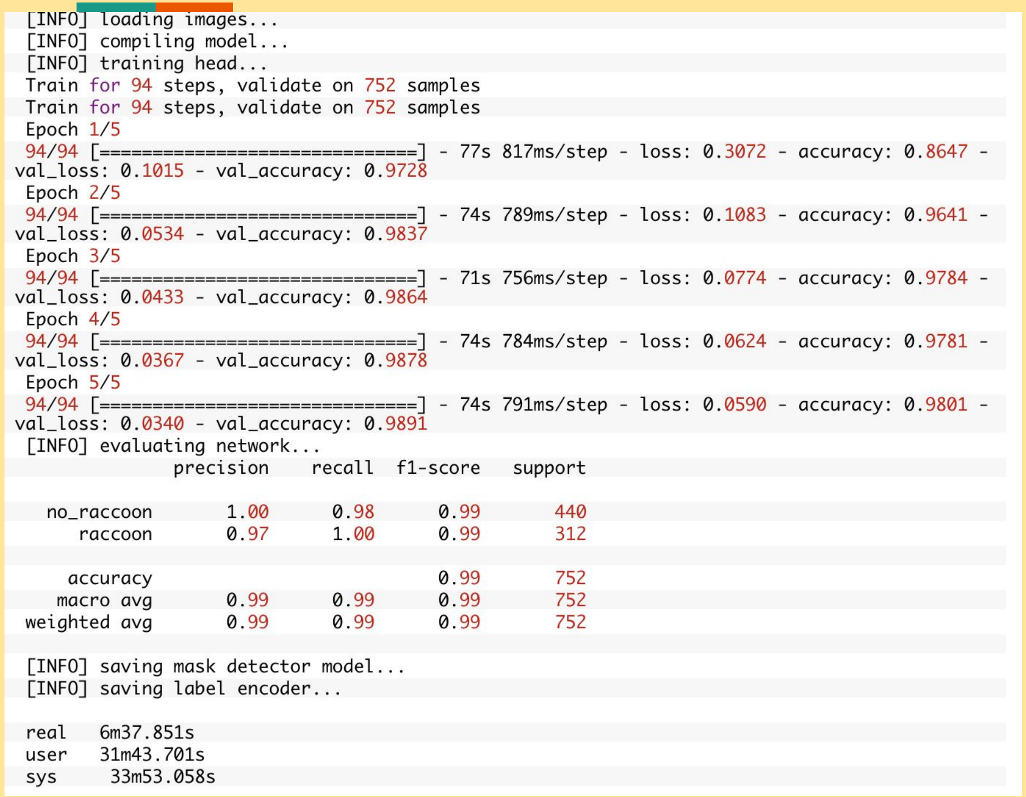
Project Overview - 1

Using the Raccoon object detection dataset curated by **Dat Tran**,
R-CNN object detection with Keras, TensorFlow, and Deep Learning

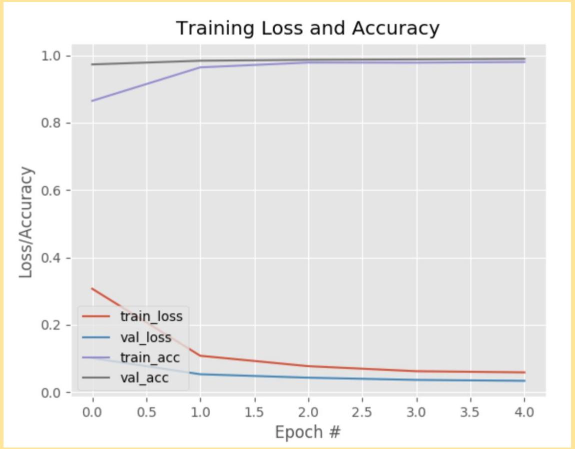

Basic R-CNN Object Detector Pipeline
Step #1: Build object detection dataset with Selective Search
Step #2: Fine-tune classification model on dataset
Step #3: During inference, run Selective Search on input image
Step #4: Make predictions on each proposal using fine-tuned model
Step #5: Apply NMS
Step #6: Return final object detection results

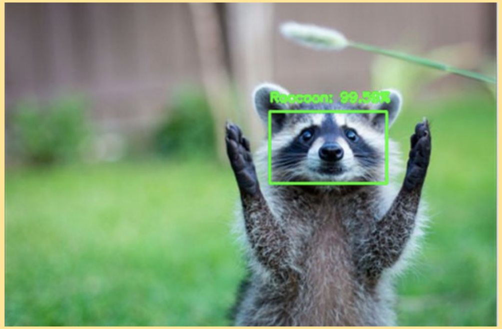


Results of the Raccoon Detector using RCNN



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

   no_raccoon       1.00      0.98      0.99       440
      raccoon       0.97      1.00      0.99       312

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

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

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





Results of the Raccoon Detector using RCNN - Key Takeaways

1. The Raccoon project taught me how to implement and use Tensorflow as well as Keras. It was my first project at DRDO, so it also help me get accustomed to the working environment, and pick up a few skills that I wasn't used to before.

2. Even though the results and accuracy shown by R-CNN was promising, I was advised by mentors at DRDO to try my model on databases that involved air-borne objects such as helicopters, aeroplanes as well as the ones that involved ships, since it suited the problem statement more.

3. I was also told to look up performance metrics, since accuracy is not always a good parameter to judge the performance of an object detection model.

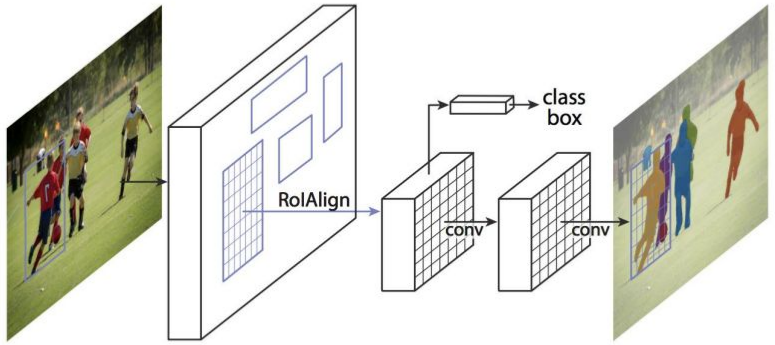
4. Even though the accuracy of my model was great, it was mostly because of the smaller size of the dataset, with easily distinguishable objects in it.

Project #2 - Implementation of Mask R-CNN

Software - Tensorflow, Keras, Google Colab

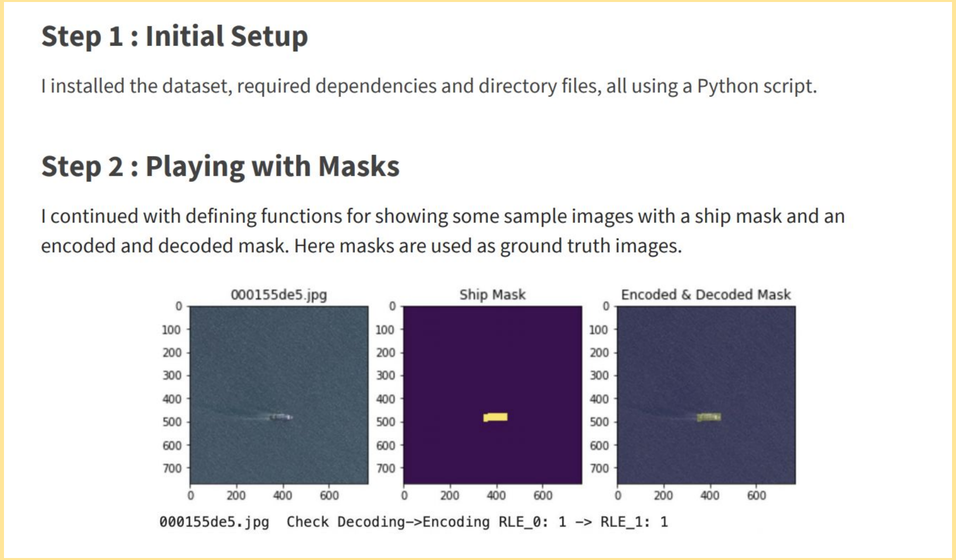
Object Detection using Mask R-CNN

Mask R-CNN is an instance segmentation technique which locates each pixel of every object in the image instead of the bounding boxes. **It has two stages: region proposals and then classifying the proposals and generating bounding boxes and masks.** It does so by using **an additional fully convolutional network on top of a CNN** based feature map with input as feature map and gives a matrix with 1 on all locations where the pixel belongs to the object and 0 elsewhere as the output.



Project Overview - 2

I used “**The Kaggle’s Airbus Detection Challenge Dataset**” as the dataset. In this project, the objective is to automatically identify whether a remotely sensed target is a ship or not. To do so, I was required to locate ships in images, and put an aligned bounding box segment around the ships located. There are many images that do not contain ships, and those that contain multiple ships. Ships within and across images differ in size (sometimes significantly) and are located in open sea, at docks, marinas, etc.



Step 1 : Initial Setup

I installed the dataset, required dependencies and directory files, all using a Python script.

Step 2 : Playing with Masks

I continued with defining functions for showing some sample images with a ship mask and an encoded and decoded mask. Here masks are used as ground truth images.

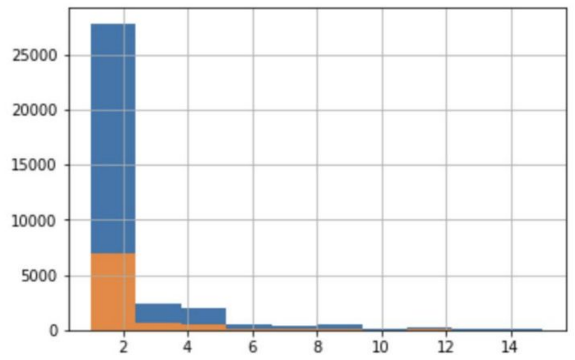
```
000155de5.jpg  Check Decoding->Encoding RLE_0: 1 -> RLE_1: 1
```

Project Overview - 2

Step 3 : Splitting my dataset and setting up the Mask-RCNN model

I divided my dataset into training and validation sets using a python script to count the number of ships in them and divide in a random manner.

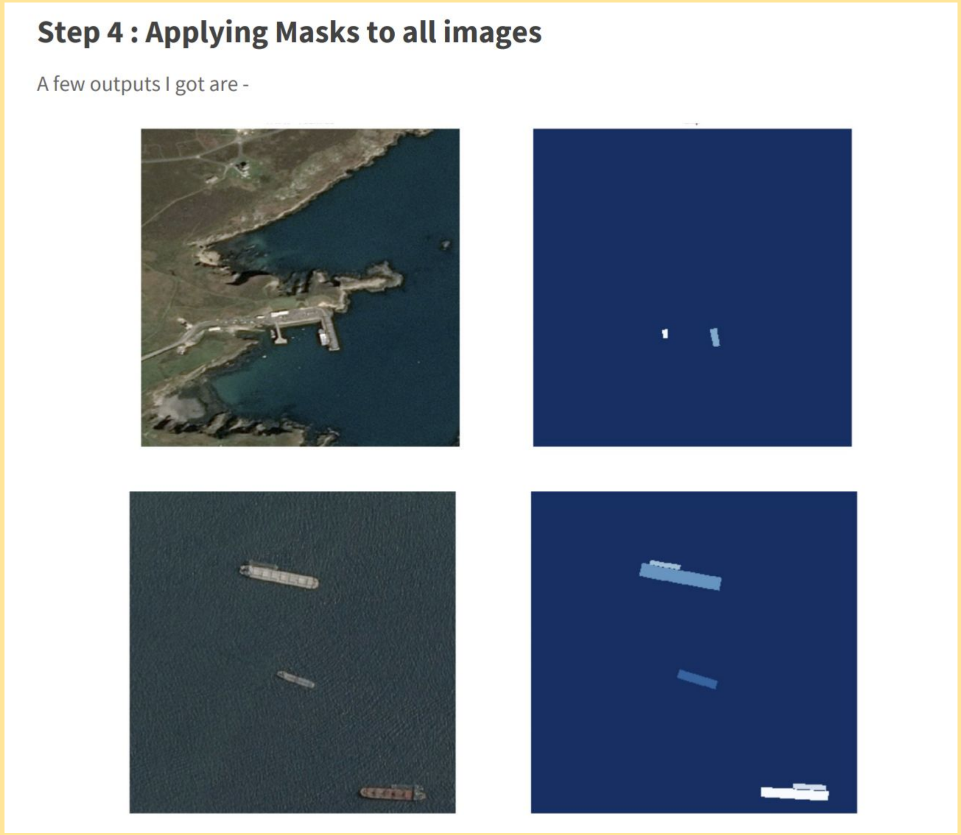
```
34044 training masks
8512 validation masks
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c8345cba8>
```



Blue denotes the training set and orange denotes the validation set. I generated 34,044 training images and 8,512 validation images. The X-axis denotes the number of ship masks in each image.

Next, I forked the original MASK-RCNN repo from here - https://github.com/matterport/Mask_RCNN

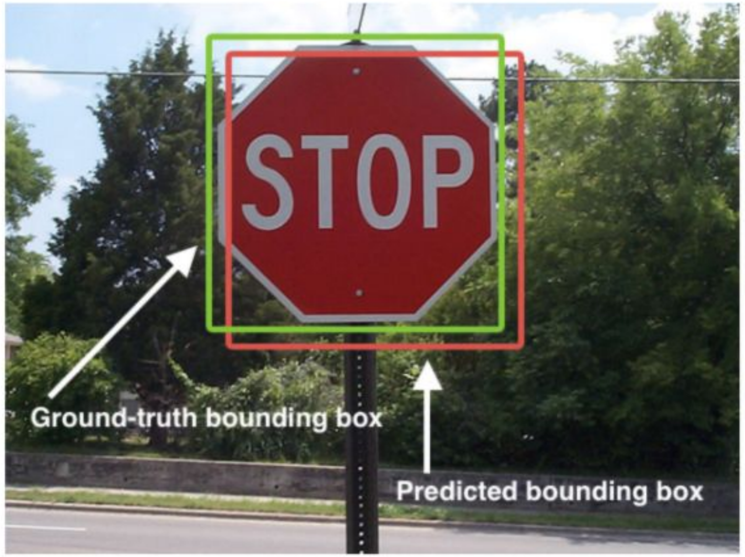
Project Overview - 2

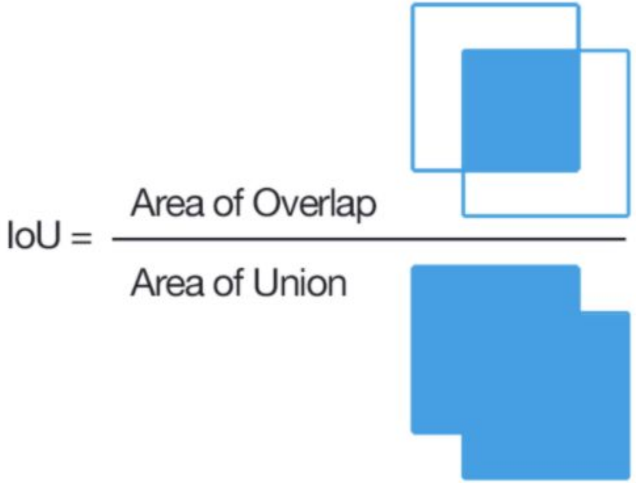

Step 4 : Applying Masks to all images

A few outputs I got are -

Project Overview - 2

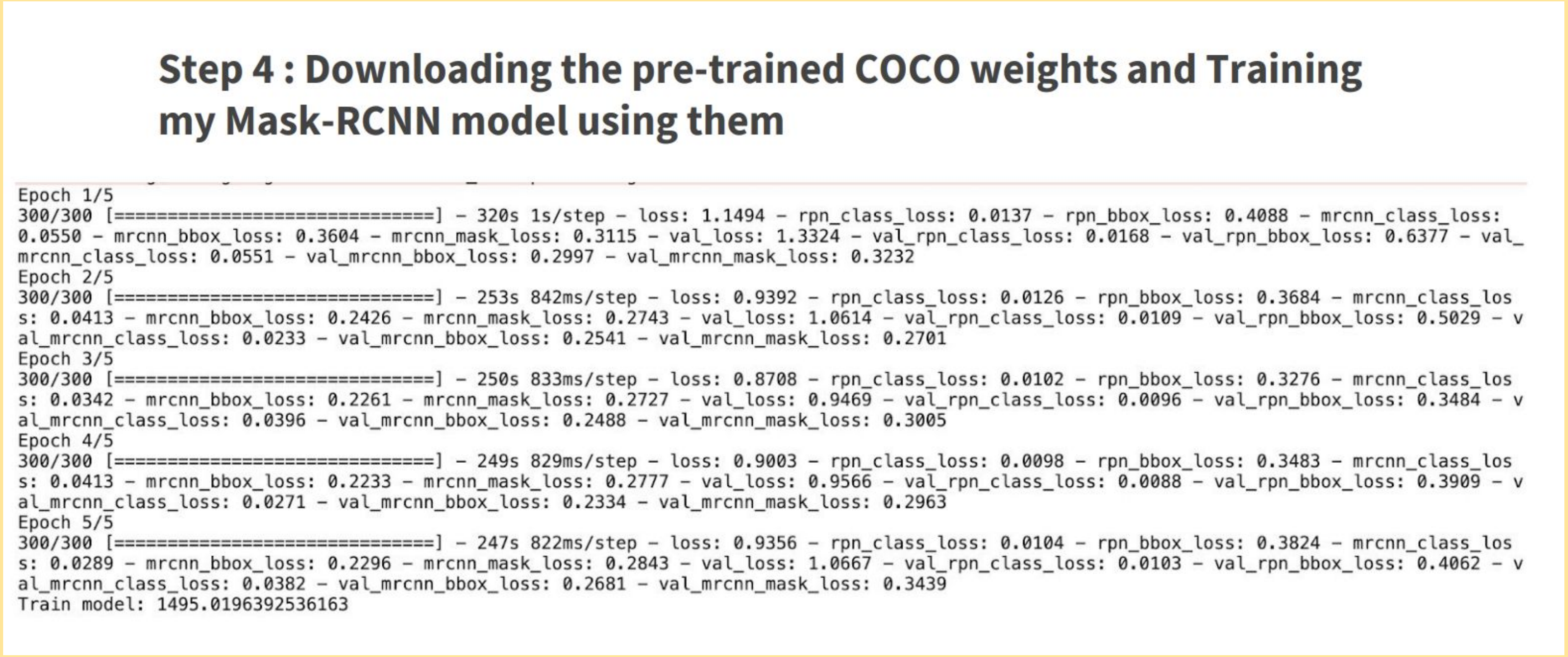
Metric used to find accuracy and create dataset **- IOU (Intersection over Union)**





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

Project Overview - 2

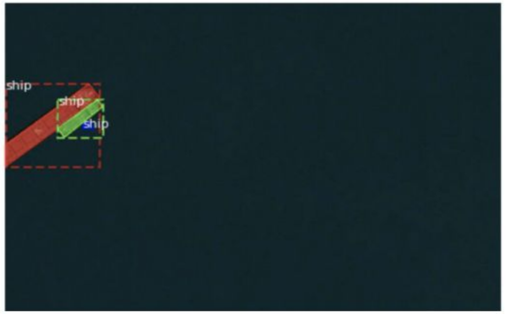


Step 4 : Downloading the pre-trained COCO weights and Training my Mask-RCNN model using them

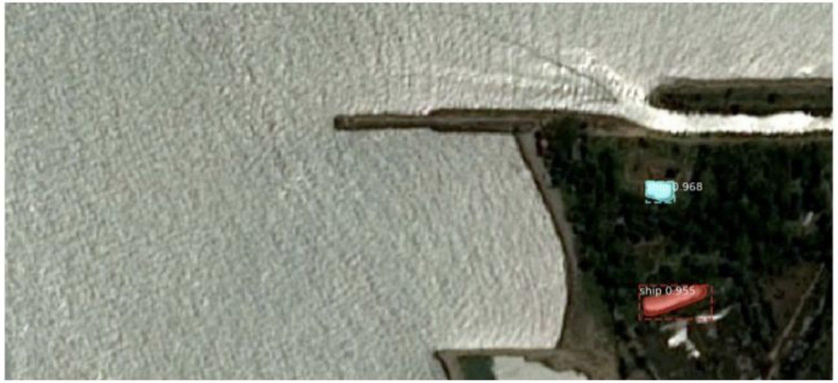
```
Epoch 1/5
300/300 [==============================] - 320s 1s/step - loss: 1.1494 - rpn_class_loss: 0.0137 - rpn_bbox_loss: 0.4088 - mrcnn_class_loss:
0.0550 - mrcnn_bbox_loss: 0.3604 - mrcnn_mask_loss: 0.3115 - val_loss: 1.3324 - val_rpn_class_loss: 0.0168 - val_rpn_bbox_loss: 0.6377 - val_
mrcnn_class_loss: 0.0551 - val_mrcnn_bbox_loss: 0.2997 - val_mrcnn_mask_loss: 0.3232
Epoch 2/5
300/300 [==============================] - 253s 842ms/step - loss: 0.9392 - rpn_class_loss: 0.0126 - rpn_bbox_loss: 0.3684 - mrcnn_class_los
s: 0.0413 - mrcnn_bbox_loss: 0.2426 - mrcnn_mask_loss: 0.2743 - val_loss: 1.0614 - val_rpn_class_loss: 0.0109 - val_rpn_bbox_loss: 0.5029 - v
al_mrcnn_class_loss: 0.0233 - val_mrcnn_bbox_loss: 0.2541 - val_mrcnn_mask_loss: 0.2701
Epoch 3/5
300/300 [==============================] - 250s 833ms/step - loss: 0.8708 - rpn_class_loss: 0.0102 - rpn_bbox_loss: 0.3276 - mrcnn_class_los
s: 0.0342 - mrcnn_bbox_loss: 0.2261 - mrcnn_mask_loss: 0.2727 - val_loss: 0.9469 - val_rpn_class_loss: 0.0096 - val_rpn_bbox_loss: 0.3484 - v
al_mrcnn_class_loss: 0.0396 - val_mrcnn_bbox_loss: 0.2488 - val_mrcnn_mask_loss: 0.3005
Epoch 4/5
300/300 [==============================] - 249s 829ms/step - loss: 0.9003 - rpn_class_loss: 0.0098 - rpn_bbox_loss: 0.3483 - mrcnn_class_los
s: 0.0413 - mrcnn_bbox_loss: 0.2233 - mrcnn_mask_loss: 0.2777 - val_loss: 0.9566 - val_rpn_class_loss: 0.0088 - val_rpn_bbox_loss: 0.3909 - v
al_mrcnn_class_loss: 0.0271 - val_mrcnn_bbox_loss: 0.2334 - val_mrcnn_mask_loss: 0.2963
Epoch 5/5
300/300 [==============================] - 247s 822ms/step - loss: 0.9356 - rpn_class_loss: 0.0104 - rpn_bbox_loss: 0.3824 - mrcnn_class_los
s: 0.0289 - mrcnn_bbox_loss: 0.2296 - mrcnn_mask_loss: 0.2843 - val_loss: 1.0667 - val_rpn_class_loss: 0.0103 - val_rpn_bbox_loss: 0.4062 - v
al_mrcnn_class_loss: 0.0382 - val_mrcnn_bbox_loss: 0.2681 - val_mrcnn_mask_loss: 0.3439
Train model: 1495.0196392536163
```

Step 5 : Output

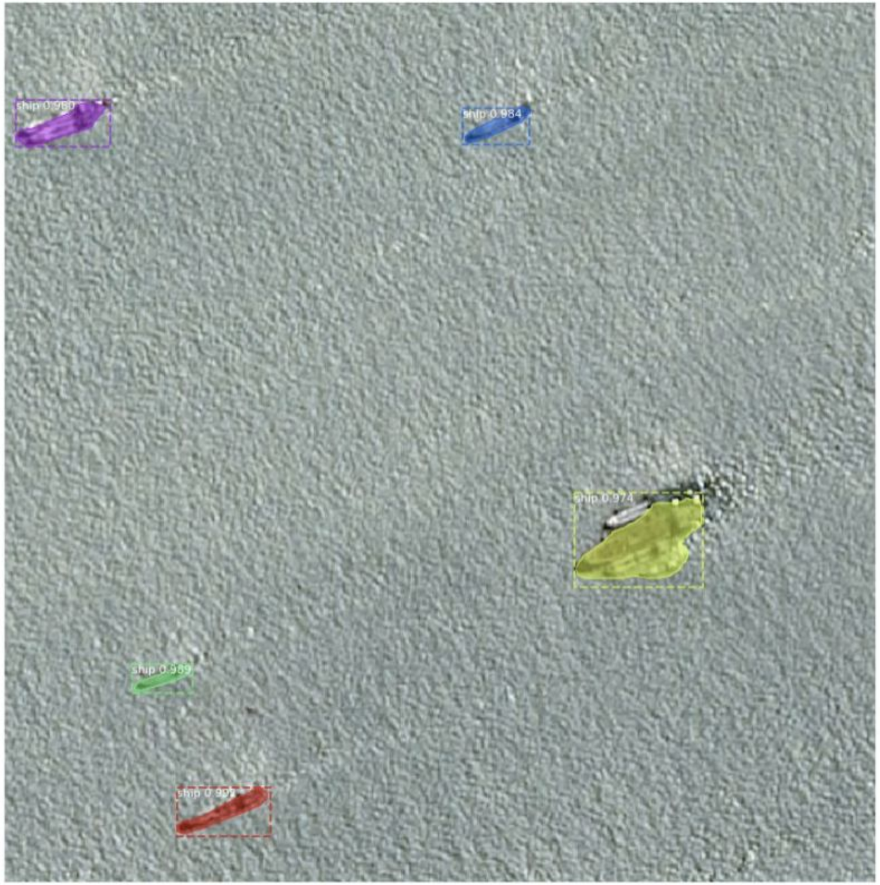
```
Re-starting from epoch 3
original_image           shape: (768, 768, 3)        min:    0.00000  max:  250.00000  uint8
image_meta               shape: (14,)                min:    0.00000  max: 7698.00000  int64
gt_class_id              shape: (3,)                 min:    1.00000  max:    1.00000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



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



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

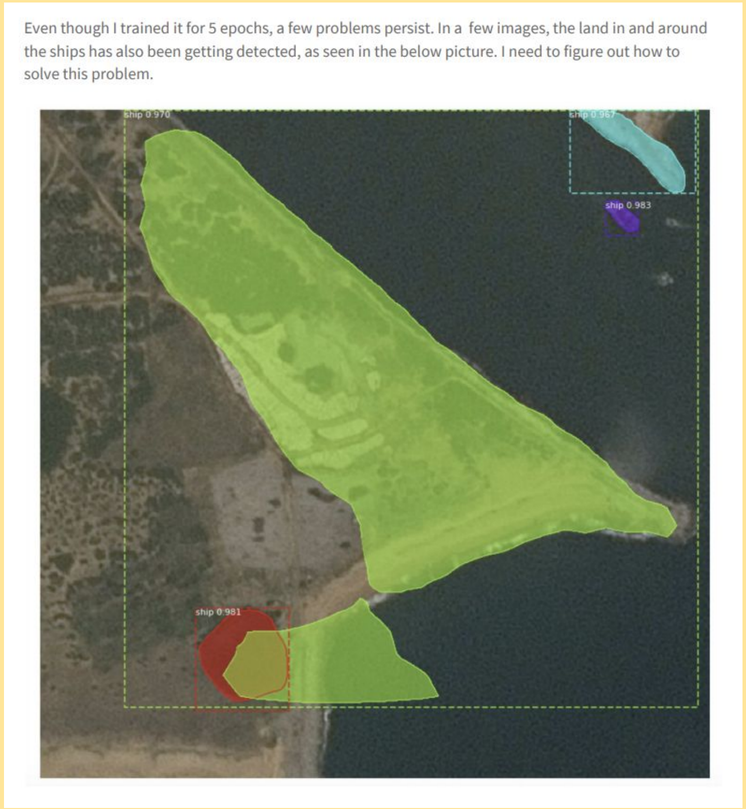


Results of Ship Detection using Mask RCNN - Key Takeaways

1. There were quite a few false positives detected by my model, as seen in the image to the right.

2. Moreover, MASK R-CNN involves generation of mask as images that takes up a lot of storage space as well as computation power. It is also extremely slow to execute.

3. I was advised by my guide at DRDO to look at alternative models to perform object detection due to the above issue.


Even though I trained it for 5 epochs, a few problems persist. In a few images, the land in and around the ships has also been getting detected, as seen in the below picture. I need to figure out how to solve this problem.

Project #3 - Implementation of YOLO v3

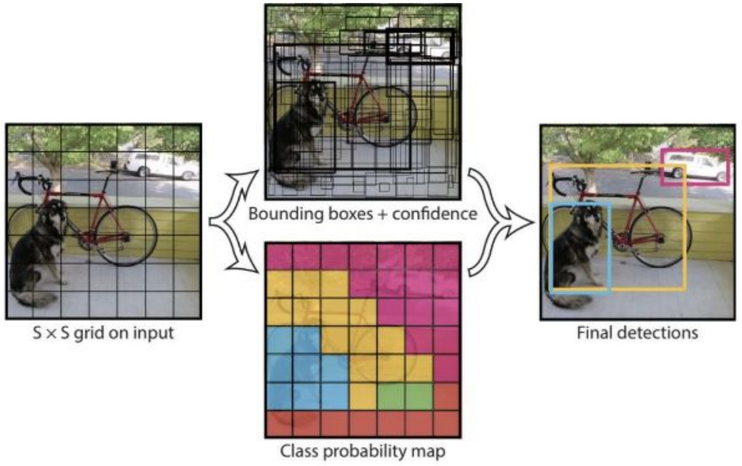
Software - Tensorflow, Keras, Google Colab

Object Detection using YOLO v3

The R-CNN models may be generally more accurate, yet the YOLO family of models are fast, much faster than R-CNN, achieving object detection in real-time.

The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g. more localization errors), although operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model.

The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within it. A class prediction is also based on each cell.



Project Overview - 3 : Under Progress

1. I’m currently in the process of developing this particular project.

2. I am using the Marvel Dataset, provided by DRDO. It contains ship satellite images, and I need to implement YOLO v3.

3. Using the YOLO v3 model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Mask R-CNN.

4. Currently generating ground truth images using a tool called LabelMe.

Timeline for the Internship

Nov '20

Introduction to the Journey

I was allotted this particular project based on my previous area of interest, and was given a few base papers to read and understand so that I get the idea of what is expected out of me.

Dec '20

Picking up skills and playing with them

Most of December was spent brushing up my Python skills, implementing Tensorflow and Keras, and reading up papers related to object detection.

Jan '21

Built my RCNN model from scratch

Feb '21

Built my Mask RCNN model using ships dataset.

Mar '21

Currently working on YOLO v3

References + Papers I referred to

1. Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
2. G. Cheng, X. Xie, J. Han, L. Guo and G. -S. Xia, "Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 3735-3756, 2020, doi: 10.1109/JSTARS.2020.3005403.
3. H. Fu, Y. Li, Y. Wang and P. Li, "Maritime Ship Targets Recognition with Deep Learning," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 9297-9302, doi: 10.23919/ChiCC.2018.8484085.
4. D. Zhao and X. Li, "Ocean ship detection and recognition algorithm based on aerial image," 2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 2020, pp. 218-222, doi: 10.1109/IPEC49694.2020.9115112.
5. M. Songhui, S. Mingming and H. Chufeng, "Objects detection and location based on mask RCNN and stereo vision," 2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Changsha, China, 2019, pp. 369-373, doi: 10.1109/ICEMI46757.2019.9101563.
6. B. A. Krinski, D. V. Ruiz, G. Z. Machado and E. Todt, "Masking Salient Object Detection, a Mask Region-Based Convolutional Neural Network Analysis for Segmentation of Salient Objects," 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), Rio Grande, Brazil, 2019, pp. 55-60, doi: 10.1109/LARS-SBR-WRE48964.2019.00018.
7. G. Li, Z. Song and Q. Fu, "Small Boat Detection for Radar Image Datasets with YOLO V3 Network," 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP), Chongqing, China, 2019, pp. 1-5, doi: 10.1109/ICSIDP47821.2019.9173163.
8. X. Li and K. Cai, "Method research on ship detection in remote sensing image based on Yolo algorithm," 2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS), Xi'an, China, 2020, pp. 104-108, doi: 10.1109/ISPDS51347.2020.00029.
9. H. Li, L. Deng, C. Yang, J. Liu and Z. Gu, "Enhanced YOLO v3 Tiny Network for Real-Time Ship Detection From Visual Image," in IEEE Access, vol. 9, pp. 16692-16706, 2021, doi: 10.1109/ACCESS.2021.3053956.
10. H. Shin, K. Lee and C. Lee, "Data Augmentation Method of Object Detection for Deep Learning in Maritime Image," 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Korea (South), 2020, pp. 463-466, doi: 10.1109/BigComp48618.2020.00-25.

Thank you.