

PERFORMANCE INVESTIGATION OF HYBRID YOLO BASED SHIP DETECTION FRAMEWORK USING SAR IMAGES



A PROJECT REPORT

Submitted by

DEVADHARSHINI S

(421116104024)

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

IFET COLLEGE OF ENGINEERING, VILLUPURAM -605 108

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2020

BONAFIDE CERTIFICATE

Certified that this project report “**PERFORMANCE INVESTIGATION OF HYBRID YOLO BASED SHIP DETECTION FRAMEWORK USING SAR IMAGES**” is the bonafide work of “**DEVADHARSHINI.S (421116104024)**” who carried out the project work under my supervision.

SIGNATURE

P.KANIMOZHI (Ph.D),

HEAD OF THE DEPARTMENT,

ASSOCIATE PROFESSOR,

Department of Computer Science and
Engineering,

IFET College of Engineering,

Villupuram – 605 108.

SIGNATURE

R.RAJMOHAN (Ph.D),

SUPERVISOR,

ASSOCIATE PROFESSOR,

Department of Computer Science and
Engineering,

IFET College of Engineering,

Villupuram – 605 108.

The project report submitted for the viva voce held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I thank the almighty, for the blessings that have been showered upon me to bring forth the success of the project. I would like to express my sincere gratitude to our chairman **Mr.K.V.Raja**, vice chairman **Mr.A.MohamedIlyas** and our secretary **Mr.K.Shivram Alva** for providing us with an excellent infrastructure and necessary resources to carry out this project and I extend my gratitude to our principal **Dr. G. Mahendran**, for his constant support to my work.

I also take this opportunity to express my sincere thanks to our vice principal **Dr.S.Matilda** and our dean-placement **Prof.J.Asha**, who has provided all the needful help for me in executing the project successfully.

I wish to express my thanks to our Head of the Department, **Prof.P.Kanimozhi, Ph.D.**, for her encouragement and support to complete this project. I express my heartfelt gratitude to my guide **Mr.R.Rajmohan,Ph.D.**, Associate Professor, Department of Computer Science and Engineering for his priceless guidance and motivation which helped me to bring this project to a perfect shape who encouraged me in each and every step of this project to complete it successfully.

I express our deep sense of thanks to all faculty members in my department for their cooperation and interest shown at every stage of our endeavour in making a project work success.

Last but not least, our sincere thanks to our lovely parents and friends who had been the constant source of our strength throughout our life.

ABSTRACT

Synthetic Aperture Radar (SAR) images are realized as encouraging data information for checking oceanic activities and its function for oil and ship recognizable proof, which is the focal point of numerous past research considers for better spatial goals. Several article discovery strategies extending from customary to deep learning approaches are proposed. Ship detection framework in deep learning technique accomplish top execution, benefitting from a SAR free open dataset (SFOD). Nonetheless, a dominant part of them are computationally dangerous and have exactness issues. The main problem identified is when the number of images increases, performance may decrease. To overcome this, we propose a technique called Hybrid YOLO, which realizes K-Means Clustering and WordTree for object identification and image classification. Hybrid YOLO also realizes SEPD for the improvement between sea clutter and ship targets and bounding box for the probability update network. The proposed model is implemented using Conda, used with Tensorflow and Keras Framework utilizing the SAR Ship Dataset. The performance of the Hybrid YOLO model is improved in terms of accuracy and F-measure when compared with other existing models.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF TABLES	viii
	LIST OF ABBREVIATION	ix
1	INTRODUCTION	1
	1.1 Scope of the project	2
	1.2 Objective of the project	2
	1.3 Review Summary	2
2	LITERATURE SURVEY	3
	2.1 Introduction	3
	2.1.1 Ship Detection Based on YOLOv2 for SAR Imagery	3
	2.1.2 Precise and Robust Ship Detection for High-Resolution SAR Imagery	4
	2.1.3 Embedded Deep Learning for Ship Detection and Recognition	5
	2.1.4 Automatic Ship Detection of Remote Sensing Images from Google Earth in Complex Scenes	6
	2.1.5 Ship Classification in High-Resolution SAR Images	7
	2.1.6 Ship Detection and Classification on Optical Remote Sensing Images Using Deep Learning	8
	2.1.7 High-Speed Ship Detection in SAR Images Based on a Grid Convolutional Neural Network	9
3	EXISTING SYSTEM	10
	3.1 Ship Detection Model	10
	3.2 Literature Conclusion	13

4	PROPOSED WORK	14
	4.1 Introduction	14
	4.2 Ship Detection Framework	15
	4.3 YOLO-VGG16 Algorithm	17
	4.3.1 YOLO Architecture	17
	4.3.1.1 Pseudo code for YOLO algorithm	18
	4.3.2 YOLO based VGG-16 Architecture	19
	4.3.2.1 Pseudo code for primary YOLO based VGG-16 algorithm	19
	4.4 Framework Modules	20
5	SYSTEM SPECIFICATION	22
	5.1 Software Requirement	22
	5.1.1 Anaconda	22
	5.1.2 Tensorflow	22
	5.1.3 Keras	23
	5.1.4 NumPy	23
	5.1.5 Pillow	23
	5.1.6 Pandas	23
	5.1.7 Pyyaml	24
	5.2 Hardware Requirement	24
	5.3 Installation procedure	24
	5.4 Dataset Description	27
6	IMPLEMENTATION AND RESULTS	30
7	PERFORMANCE COMPARISON	35
8	CONCLUSION AND FUTURE SCOPE	38
	APPENDIX – I	39
	APPENDIX – II	56
	APPENDIX – III	59

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURES	PAGE NO
3.1	Architecture of Ship detection using RetinaNet with FPN	12
3.2	Output for Ship detection under complex backgrounds	13
4.1	Architecture of Hybrid YOLO – VGG16	16
4.2	Architecture of YOLO object detection	18
4.3	Architecture of primary YOLO based VGG-16	19
4.4	Proposed Model Flow	20
5.1	Anaconda Navigator	25
5.2	Installation of Anaconda Packages	26
5.3	Sample SAR Images from the dataset	29
6.1	Training the model	30
6.2	Accuracy	31
6.3	Loss function	31
6.4	Confusion Matrix	33
6.5	Sample Input Image	33
6.6	Classified Ship Image	34
7.1	Accuracy Comparison	35
7.2	Precision Comparison	36
7.3	Recall Comparison	36
7.4	F1-Score Comparison	37

LIST OF TABLES

TABLE NO	NAME OF THE TABLES	PAGE NO
1	Software Specifications	22
2	Hardware Specification	24
3	Sample Distribution of each ship class	28
4	Performance Analysis	32
5	Comparison Analysis	35

LIST OF ABBREVIATIONS

S. NO	ABBREVIATION	EXPANSION
1	YOLO	You Only Look Once
2	VGG	Visual Geometry Group
3	RNN	Region based Neural Network
4	CNN	Convolution Neural Network
5	SEPD	Spatial Based Pixel Descriptor
6	SAR	Synthetic Aperture Radar
7	SSD	Single Shot multibox Detector

CHAPTER 1

INTRODUCTION

Ship location is significant for marine observation in zones, for example, illicit fishing, oil slick discovery, marine traffic the executives, and sea theft. The SAR images in ship detection [1] are depicted as having significant standards limit, not being dependent upon the climatic case and self-governing of flight height. It consistently gives quality pictures at any position due to its automatic light capacity and has a ton of uses in remote detecting. The idea of deep learning taking in begins from the investigation of artificial neural systems. Deep learning [17] has made momentous accomplishments in the field of image processing, particularly for object discovery. Ship detection using SAR images in the deep learning approach is advantageous with its performance and accuracy. Convolution network having more number of layers is used for training the images. YOLO [2] utilizes a very surprising methodology. It applies a unique neural structure to the entire picture. This framework isolates the map into zones and predicts bounding boxes in the form of grid and class probabilities for every district. The foreseen probabilities weight these bounding boxes. YOLO is a best in class ongoing item recognition framework, which beats Faster R-CNN, RetinaNet, and SSD strategies. Specifically, the focus is on unsolved problems like computational time, high performance, high accuracy, and enhanced technology for ship detection framework. To diminish computational time with moderately aggressive discovery exactness, another design is built with less number of layers called Hybrid YOLO. Hybrid YOLO is the YOLO-VGG16 algorithm added with Spatial Based Pixel Descriptor (SEPD) and K-means clustering. This framework is proposed by using the Anaconda tool, which is combined with Tensorflow, Keras, Numpy, h5py, Pillow, and Python3. The TensorFlow and Keras framework plays a vital role for this ship detection. The

performance that is measured in this ship detection framework using Hybrid YOLO is precision, recall, accuracy, specificity, F-measure, and Kappa statistics. The optimization algorithm used in this ship detection model is RMSprop and the activation function used is LeakyReLU and Softmax functions.

1.1 SCOPE OF THE PROJECT

The main scope of this project is to develop an efficient framework to detect the ship from the various Synthetic Aperture Radar Images.

1.2 OBJECTIVE OF THE PROJECT

Ship detection has been playing a significant role in the field of remote sensing for a long time but it is still full of challenges. The main limitations of traditional ship detection methods usually lie in the complexity of application scenarios, the difficulty of intensive object detection and the redundancy of detection region. In order to solve such problems above, we propose a framework called Hybrid YOLO-VGG16 which can effectively detect ship in different scenes including ocean and port.

1.3 REPORT SUMMARY

The project report is organized as follows: the chapter 2 narrates the related work done in this Ship Detection domain, chapter 3 explains the existing system and methodologies, chapter 4 depicts the architecture and design of the proposed methodology, chapter 5 discuss the system implementation requirements, chapter 6 describes the simulation and discussion for the ship detection model, the chapter 7 details the conclusion and future work of this ship detection model, and the last section provides the references and appendix.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

The various research works on the existing ship detection model using SAR images are discussed and analysed.

2.1.1. Ship Detection Based on YOLOv2 for SAR Imagery

The proposed method by Yang Lang et al [8], realizes you only look once version 2 (YOLOv2) deep learning framework, which is a well-known sliding window based deep learning model in the field of computer vision, as a base to implement vessel detection and adjust the parameters to achieve high accuracy performance in near real-time. In addition, we introduced a new architecture, YOLOv2-reduced, having fewer layers due to elimination of some of the unrequired layers. The proposed architecture has less computational time compared with YOLOv2 on NVIDIA TITAN X GPU. YOLOv2-reduced is best for real time object detection problem. The performance of the YOLOv2 approach is evaluated on two different datasets and its performance is compared with region proposed approach Faster R-CNN. The performance of YOLOv2-reduced is evaluated on SSSD dataset and it reduces the computational time significantly. YOLOv2 test results showed an increase in accuracy of ship detection as well as a noticeable reduction in computational time compared to Faster R-CNN. From the experimental results, the proposed YOLOv2 architecture achieves an accuracy of 90.05% and 89.13% on the SSDD and DSSDD datasets respectively. The proposed YOLOv2-reduced architecture has a similarly competent detection performance as YOLOv2, but with less computational time on a NVIDIA TITAN X GPU. The experimental results

shows that the deep learning can make a big leap forward in improving the performance of SAR image ship detection.

2.1.2. Precise and Robust Ship Detection for High-Resolution SAR Imagery

The proposed method by Shunjun Wei et al [14], realizes a ship detection method based on a high-resolution ship detection network (HR-SDNet) for high-resolution SAR imagery is proposed. The HR-SDNet adopts a novel high-resolution feature pyramid network (HRFPN) to take full advantage of the feature maps of high-resolution and low-resolution convolutions for SAR image ship detection. In this scheme, the HRFPN connects high-to-low resolution subnetworks in parallel and can maintain high resolution. Next, the Soft Non-Maximum Suppression (Soft-NMS) is used to improve the performance of the NMS, thereby improving the detection performance of the dense ships. Then, introduced the Microsoft Common Objects in Context (COCO) evaluation metrics, which provides not only the higher quality evaluation metrics average precision (AP) for more accurate bounding box regression, but also the evaluation metrics for small, medium and large targets, so as to precisely evaluate the detection performance of the method. Finally, the experimental results on the SAR ship detection dataset (SSDD) and TerraSAR-X high-resolution images reveal that (1) our approach based on the HRFPN has superior detection performance for both inshore and offshore scenes of the high-resolution SAR imagery, which achieves nearly 4.3% performance gains compared to feature pyramid network (FPN) in inshore scenes, thus proving its effectiveness; (2) compared with the existing algorithms, this approach is more accurate and robust for ship detection of high-resolution SAR imagery, especially inshore and offshore scenes; (3) with the Soft-NMS algorithm, the network performs better, which achieves nearly 1% performance gains in terms of AP; (4) the COCO evaluation

metrics are effective for SAR image ship detection; (5) the displayed thresholds within a certain range have a significant impact on the robustness of ship detectors.

2.1.3. Embedded Deep Learning for Ship Detection and Recognition

The proposed method by Hongwei Zhao et al [13], realizes a ship detection and recognition for smart monitoring of ships in order to manage port resources effectively. However, this is challenging due to complex ship profiles, ship background, object occlusion, variations of weather and light conditions, and other issues. It is also expensive to transmit monitoring video in a whole, especially if the port is not in a rural area. To overcome this, an on-site processing approach, which is called Embedded Ship Detection and Recognition using Deep Learning (ESDR-DL). In ESDR-DL, the video stream is processed using embedded devices, and designed a two-stage neural network named DCNet, which is composed of a DNet for ship detection and a CNet for ship recognition, running on embedded devices. The video stream in ESDR-DL is connected to a nearby TX2 through a LAN. To ensure real-time performance of video surveillance, each TX2 receives only one or two video streams. When the system is running, a Video Stream Receiver in TX2 is responsible for receiving the video stream accessed by the current device, decoding the video stream through a Video Stream Decoder, and inputting the decoded images to an Image Processor for detection. In the Image Processor, the DCNet model is used to detect and identify key parts of a ship (the bow, the cabin, and the stern), and classify the ship's identity based on these key parts, and output three prediction results. These prediction results are then used in a Voter for the decision of the ship's identity. We use NVIDIA Jetson TX2 as it is an industry-leading embedded computing device. The ESDR-DL is deployed at the Dongying port of China, which has been running for over a year and demonstrates that it can work reliably for practical usage. They have extensively evaluated ESDR-DL the accuracy of the

system is dropping sharply in rain and smoggy weather, while performing well at dusk.

2.1.4. Ship Detection of Remote Sensing Images in Complex Scenes

The proposed method by Xue Yang et al [15], realizes an end-to-end detection framework in ship detection called Rotation Dense Feature Pyramid Networks (R-DFPN). The framework is based on a multi-scale detection network, using a dense feature pyramid network, rotation anchors, multi-scale ROI Align and other structures. Compared with other rotation region detection methods such as RRPN and R2CNN, this framework is more suitable for ship detection tasks, and has achieved the state-of-the-art performance. The main contributions include:

- i. A new ship detection framework based on rotation region which can handle different complex scenes, detect intensive objects and reduce redundant detection region.
- ii. The feature pyramid of dense connections based on a multi-scale detection framework, which enhances feature propagation, encourages feature reuse and ensures the effectiveness of detecting multi-scale objects.
- iii. They adopt rotation anchors to avoid side effects of non-maximum suppression and overcome the difficulty of detecting densely arranged targets, and eventually get a higher recall.
- iv. The multi-scale ROI Align to solve the problem of feature misalignment instead of ROI pooling, and to get the fixed-length feature and regression bounding box to fully keep the completeness of semantic and spatial information through the horizontal circumscribed rectangle of proposal. Experiments based on remote sensing images from Google Earth for ship detection show that our detection method based on R-DFPN representation has a state-of-the-art performance.

2.1.5. Ship Classification in High-Resolution SAR Images

The proposed method by Changchong Lu et al [6], realizes a technique called CNN, which often requires a large amount of data to train adequately. However, as with the SAR dataset, enough images for training doesn't exist. Therefore, we must do something to avoid the problem of over-fitting. One effective way is data augmentation. There are many methods to achieve our requirement, including adding noise, changing colours, flipping, etc. The original SAR ship images and some common ways of data augmentation. One of the most popular methods is random crop. It can significantly increase the amount of data. Recently, researchers usually use some fixed frame to do random crop. For example, using a frame of 224×224 to do random crop in a picture of 256×256 . It not only increases the data, but it also retains the most information of the data. However, not all traditional ways of data processing can be helpful when dealing with SAR images. The operation of data processing may loss original information and amplify noise information. Sometimes, random crop may lose some important information. When the main information of images diverges of the centres, using random crop cannot perform well. However, when concerned with deep neural networks, such as Resnet-50, the repeated images cannot give more contributions for training. Thus the performance cannot have more improve. Rotating is also a popular way for data augmentation, it can keep main information of images. To solve this problem, we expansion the images in their edges with pixel pads to remove the black. By using the proposed method, we not only increase the number of images, but we can also retain the important information of the images. Some noise is still in the background, but concerning SAR images, that noise of the sea surface can be accepted. The data augmentation of all images allows the network to produce better feature representation.

2.1.6. Ship Detection and Classification on Optical Remote Sensing Images

The proposed method by Ying LIU et al [19], realizes a Ship detection by deep learning technique is the proposed method. It detects actual ships from all the ship candidates and then the actual ships are classified into different types by CNN. The state-of-the-art ship detection approaches extract features using feature operators or feature descriptors, then use traditional machine learning methods for detection. Features extracted by these methods generally have some fundamental limitations in practical applications. For example, they may have poor performances when the images are corrupted by blur, distortion, or illumination which commonly exist in remote sensing images. So the processed images may contain various pseudo-targets, e.g., islands, clouds, sea waves, etc. Traditional machine learning algorithms, e.g., support vector machine (SVM), may have difficulties in efficiently handling such highly varying inputs. When dealing with highly variant conditions, the computation is exponentially increased. Relatively, automatically learned features by deep learning from images can help to tackle these issues. Recent works have shown that the features extracted by deep learning outperform those manually designed ones on target detection. That is, the machine learning approach and the convolution network is used in this experiment for Optical Remote Sensing Images. These experiments were conducted on a server with Intel Core i5- 4460 CPU @3.20GHz, 8.00GB RAM and GTX TitanX card. Matlab2014a and cuda 7.0 were used. Experiments showed that CNN, as a deep neural network is a good model for automatically feature learning and extraction. And up to 75 h speedup was achieved on a server with a GTX TitanX GPU which indicates its potential for real-time processing.

2.1.7. High-Speed Ship Detection in SAR Images

The proposed method by Tianwen Zhang et al [20], realizes a Ship detection approach for high-speed ship detection in SAR images based on a grid convolutional neural network (G-CNN). This method improves the detection speed by meshing the input image, inspired by the basic thought of you only look once (YOLO), and using depthwise separable convolution. G-CNN is a brand new network structure proposed by us and it is mainly composed of a backbone convolutional neural network (B-CNN) and a detection convolutional neural network (D-CNN). First, SAR images to be detected are divided into grid cells and each grid cell is responsible for detection of specific ships. Then, the whole image is input into B-CNN to extract features. Moreover, the dataset must have been correctly labeled. Additionally, the dataset was divided into a training set and a test set. According to the parameters of the known real ships in the training set, G-CNN fits these parameters by iteration many times to minimize the error. Finally, ship detection is completed in D-CNN under three scales. We experimented on an open SAR Ship Detection Dataset (SSDD) used by many other scholars and then validated the migration ability of G-CNN on two SAR images from RadarSat-1 and Gaofen-3. The experimental results show that the detection speed of our proposed method is faster than the existing other methods, such as faster-regions convolutional neural network (Faster R-CNN), single shot multi-box detector (SSD), and YOLO, under the same hardware environment with NVIDIA GTX1080 graphics processing unit (GPU) and the detection accuracy is kept within an acceptable range. Our proposed G-CNN ship detection system has great application values in real-time maritime disaster rescue and emergency military strategy formulation.

CHAPTER 3

EXISTING SYSTEM

3.1 Ship detection model

In the Existing System [4] the SAR images, which is a coherent imaging process, leads to inherent characteristics such as foreshortening, layover, and shadowing is used. Apart from the imaging mechanisms, ships in SAR images appear differently with regard to size and background. For the former, the same ship in different resolutions varies widely, and ships of various shapes in the same resolution have different sizes. For the latter, targets on the ocean, near an island, or in a harbor that have similar backscattering mechanisms to ships produce high false-positive rates. Therefore, it is necessary to construct a dataset that is as complex as possible to improve the application of deep learning for ship detection in SAR images.

A dataset is constructed first. Specifically, it contains 102 Gaofen-3 images and 108 Sentinel-1 images that were used to create 59,535 ships in 43,819 ship chips. They vary in terms of polarization, resolution, incidence angle, imaging mode, and background. Second, modified object detectors are adapted for ship detection and can be baselines. The contributions are as follows.

1. A SAR ship detection dataset under complex backgrounds is constructed. This dataset can be the catalyst for the development of object detectors in SAR images without land-ocean segmentation, thus helping the dynamic monitoring of marine activities.
2. Modified state-of-the-art object detectors, including Faster regions with convolutional neural networks (R-CNN), single shot multiBox detector (SSD), and RetinaNet are adapted to ship detection and can be baselines.

There are two main factors for SAR ship detection with deep learning, including ship size and background influencing detection performance. Compared with those objects in the optical dataset [9], the ship size is relatively quite small. Therefore, the ship may be a pixel and miss the characteristics of ships on a feature map after subsampling several times in deep convolutional networks (ConvNets), thus making it difficult to distinguish ships. Since the targets on the backgrounds of the ships, such as a building, harbor, noise, and islands may share similar backscattering mechanisms to ships, thus leading to false alarms. Currently, there are many object detectors in deep learning for computer vision, such as R-CNN, Faster R-CNN, feature pyramid networks (FPN), SSD, YOLO, and RetinaNet. These models can be divided into two-stage and one-stage models. Faster R-CNN is the top two-stage performer, and RetinaNet has the best performance among one-stage models. These two models are the baseline for this dataset. Since SSD, Faster R-CNN, and RetinaNet can share the same backbone network to extract features for further object classification and bounding-box regression, one kind of backbone networks will be chosen. Faster R-CNN consists of three main networks: the ConvNets to extract feature maps, the region proposal network (RPN) for generating region proposals, and a network using these proposals for object classification and bounding-box regression. First, an image is fed to the deep convolutional networks (ConvNets) to obtain its feature maps, and then the RPN is employed to acquire a set of the rectangular object proposal and its corresponding object score. After that, region of interest (ROI) pooling resizes the rectangular object proposals to the same shapes. Finally, these converted object proposals are passed to the classifier to output their corresponding classes and the bounding boxes that include the ships. VGG16 was chosen as the ConvNets for ship detection.

The architecture of RetinaNet contains three components: a backbone network and two subnetworks (one for classification and the other for box

regression) as shown in figure 3.1. FPN is used to distill multiscale features. FPN has two pathways—a bottom-up pathway and top-down pathway. The former usually consists of ConvNets such as VGG16, ResNet, and Inception, and employs them to extract hierarchical features. The latter constructs the high-resolution, multiscale layers from the top layer in the bottom-up pathway. For the top-down, 1×1 convolution is adapted to reduce the number of feature map channels to 256, and skip-connection is utilized for lateral connections between the corresponding feature maps and the reconstructed layers, which enables the object detector to better predict the location. It acts as a feature extractor with the consideration of the low-level high resolution and high-level low-resolution semantic meaning. Specifically, RetinaNet uses a backbone network such as ResNet, VGG, Inception, or DenseNet to extract higher semantic feature maps, and then FPN is applied to extract the same dimension features with various scales. After that, these pyramidal features are fed to the two subnets to classify and locate objects.

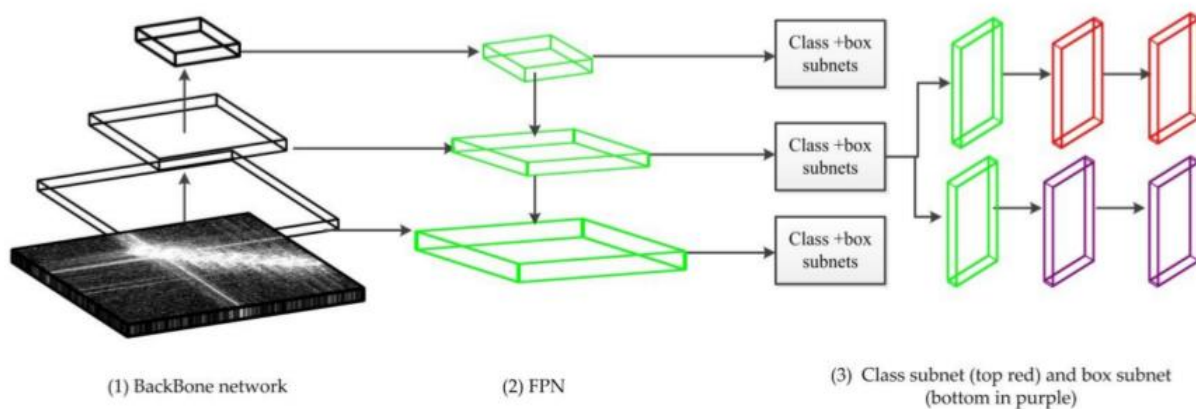


Fig. 3.1 Architecture of Ship detection using RetinaNet with FPN

Figure 3.2 shows the ship detection results under complex backgrounds. Red rectangles, green ellipses, and yellow ellipses indicate the detected ships, missing ships, and false alarms, respectively.



Fig. 3.2 Output for Ship detection under complex backgrounds

3.2 Literature Conclusion

From our survey, it is obvious that Faster R-CNN and RetinaNet achieve the worst and the best mean average precision (mAP), respectively. This may be because RetinaNet exploits multiscale features for ship detection, whereas Faster R-CNN does not. Considering that the bounding-box sizes that include the ships are relatively small, these three layers, Conv7_2, Conv8_2, and Conv9_2 are removed to construct the new ship detection model for SSD-300 and SSD-512. The modified SSD-300 can detect almost all the multiscale ships, except those that are side by side in the harbor.

The disadvantage of the existing models has less computational efficiency and accuracy. Ship detection is performed on the two-stage detector, where it first detects and make the classification. This is a time-consuming process. The main disadvantage is that less number of dataset images gives less accuracy.

CHAPTER 4

PROPOSED WORK

4.1 Introduction

Ship location is significant for marine observation in zones, for example, illicit fishing, oil slick discovery, marine traffic the executives, and sea theft. The SAR [1] images in ship detection are depicted as having significant standards limit, not being dependent upon the climatic case and self-governing of flight height. It consistently gives quality pictures at any position due to its automatic light capacity and has a ton of uses in remote detecting. The idea of deep learning taking in begins from the investigation of artificial neural systems. Deep learning [2] has made momentous accomplishments in the field of image processing, particularly for object discovery. Ship detection using SAR images in the deep learning approach is advantageous with its performance and accuracy. Convolution network having more number of layers is used for training the images. YOLO utilizes a very surprising methodology. It applies a unique neural structure to the entire picture. This framework isolates the map into zones and predicts bounding boxes in the form of grid and class probabilities for every district. The foreseen probabilities weight these bounding boxes. YOLO (You Only Look Once) is the object detectors, they take images and detect objects within the image instead of classifying the image according to a class. To be more specific, YOLO uses a single neural network pretrained on ImageNet to predict bounding boxes, which define where the object is in the image, and class probabilities. YOLO is the improved version in detection that works like the original network, but it runs faster and detects small objects better due to a new and improved network. During pretraining, the network was pretrained on 224 x 224 images, but during detection, YOLO works on images that are double the resolution. YOLO [3] is a best in class ongoing item recognition framework, which

beats Faster R-CNN, RetinaNet, and SSD strategies [4]. The backbone network in YOLO is the VGG16 network, was to evaluate the network's accuracy on the Ship SAR dataset. VGG16 is an object recognition network that uses a convolutional neural network (CNN) with 16 layers to recognize a 224 x 224 image as input [10]. It possesses 3 fully-connected layers, the last of which is a Softmax layer that contains the output features corresponding to each of the seven image classes. It is pretrained on ImageNet. Specifically, the focus is on unsolved problems like computational time, high performance, high accuracy, and enhanced technology for ship detection framework. To diminish computational time with moderately aggressive discovery exactness, another design is built with less number of layers called Hybrid YOLO. Hybrid YOLO is the YOLO-VGG16 algorithm added with Spatial Based Pixel Descriptor (SEPD) and K-means clustering. This framework is proposed by using the Anaconda tool, which is combined with Tensorflow, Keras, Numpy, h5py, Pillow, and Python3. The TensorFlow and Keras framework plays a vital role for this ship detection. The performance that is measured in this ship detection framework using Hybrid YOLO is precision, recall, accuracy, specificity, F-measure, and Kappa statistics. The optimization algorithm used in this ship detection model is RMSprop and the activation function used is LeakyReLU and Softmax functions.

4.2 Ship Detection Framework

The proposed framework realizes Hybrid YOLO is implemented by the ship detection and classification that involves a two-stage process, namely Spatial Based Pixel Descriptor (SEPD) and K-Means Clustering. Hybrid YOLO realizes YOLO-VGG16 [4] is a standardized system for identification over many classes by together advancing recognition and characterization by hierarchical classification. The WordTree consolidate information from different sources and the joint enhancement

method to prepare all the while on SAR pictures. Spatial Based Pixel Descriptor (SEPD) is used for the detection of improvement between ship targets and sea clutter.

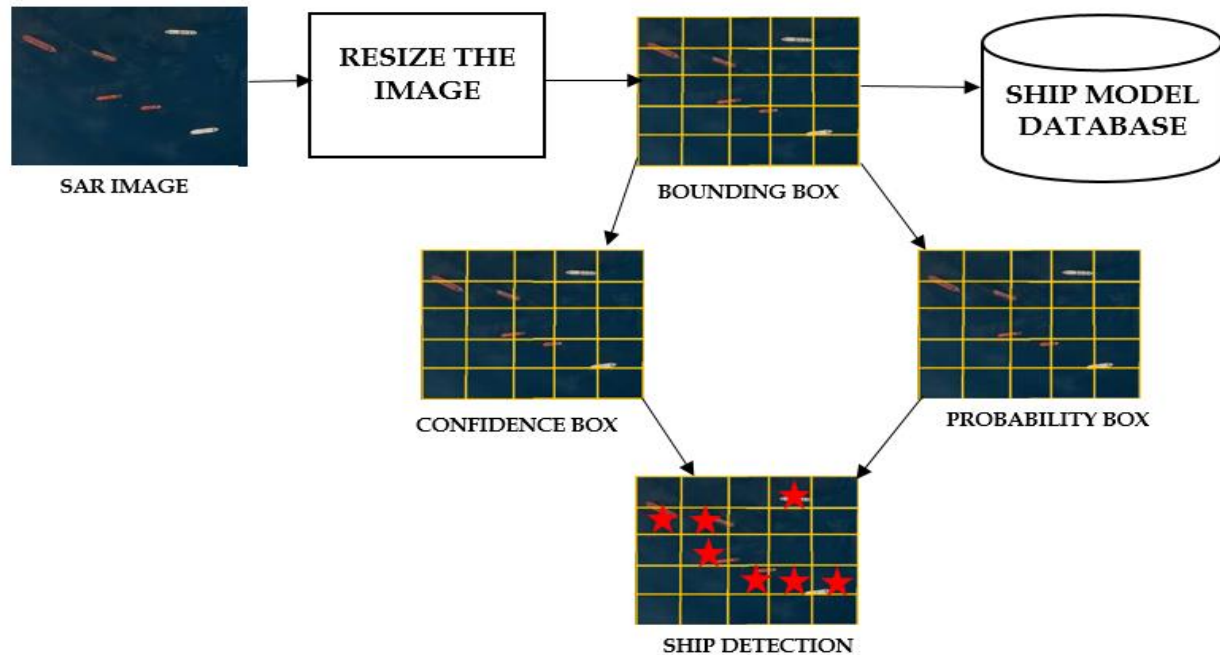


Fig. 4.1 Architecture of Hybrid YOLO – VGG16

Figure 4.1 realizes the classification and clustering of the ship detection framework by using the SAR images. These SAR images can be collected from the SAR Free Open Dataset (SFOD). Since the SAR images are collected from various sources, it may vary in its size. For this size variation of images, resizing an image is done. Resizing of image can be done by data pre-processing. After resizing the image, a bounding box is performed. The bounding box is a technique that splits the image as a 5X5 grid. This bounding box involves two categories. One is the confidence box, which detects all the objects in the image, and the other is the probability box, which realizes the class probability for the detection of a ship from the SAR images. Further, ship detection model is performed, and that ship detection model is stored in the database.

To improve the accuracy of the ship detection model, Spatial Enhanced Pixel Descriptor (SEPD) [18] is used to find the difference between the sea clutters and ship targets. The ship is categorized into ten classes and each class consists of five images per class. The ship class which is detected is clustered by using K-Means Clustering, which improves more accuracy in the ship detection model.

4.3 YOLO-VGG16 Algorithm

4.3.1 YOLO Architecture

YOLO (You Only Look Once) is a single stage object detector, they take images and detect objects within the image instead of classifying the image according to a class. To be more specific, YOLO uses a single neural network pretrained on ImageNet to predict bounding boxes, which define where the object is in the image, and class probabilities. YOLO is the improved version in detection that works like the original network, but it runs faster and detects small objects better due to a new and improved network. During pretraining, the network was pretrained on 224 x 224 images, but during detection, YOLO works on images that are double the resolution. YOLO is a best in class ongoing item recognition framework, which beats Faster R-CNN, RetinaNet, and SSD strategies.

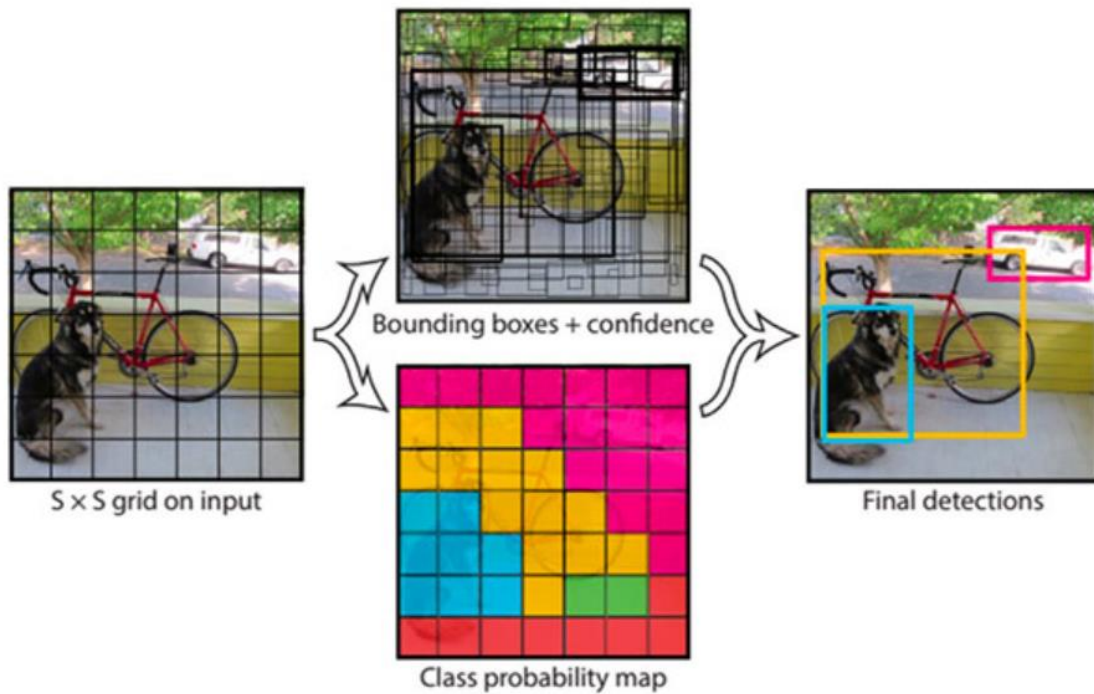


Fig. 4.2 Architecture of YOLO object detection

4.3.1.1 Pseudo code for YOLO algorithm

```

image = readImage()
NoOfCells = 7
NoOfClasses = 4
threshold = 0.7
step = height(image)/NoOfCells
prediction_class_array = new_array(size(NoOfCells,NoOfCells,NoOfClasses))
predictions_bounding_box_array =
new_array(size(NoOfCells,NoOfCells,NoOfCells,NoOfCells))
final_predictions = []
for (i<0; i<NoOfCells; i=i+1):
    for (j<0; j<NoOfCells; j=j+1):
        cell = image(i:i+step,j:j+step)
        prediction_class_array[i,j] = class_predictor(cell)
        predictions_bounding_box_array[i,j] =
bounding_box_predictor(cell)
        best_bounding_box = [0 if predictions_bounding_box_array[i,j,0,
4] > predictions_bounding_box_array[i,j,1, 4] else 1]

```

```

        predicted_class =
index_of_max_value(prediction_class_array[i,j])
        if predictions_bounding_box_array[i,j,best_bounding_box, 4] *
max_value(prediction_class_array[i,j]) > threshold:
            final_predictions.append([predictions_bounding_box_array[i,j,best_boun
ding_box, 0:4], predicted_class])
print final_predictions

```

4.3.2 YOLO based VGG-16 Architecture

The backbone network in YOLO is the VGG16 network, was to evaluate the network's accuracy on the Ship SAR dataset. VGG16 is an object recognition network that uses a convolutional neural network (CNN) with 16 layers [11] to recognize a 224 x 224 image as input. It possesses 3 fully-connected layers, the last of which is a Softmax layer that contains the output features corresponding to each of the seven image classes. It is pretrained on ImageNet [5].

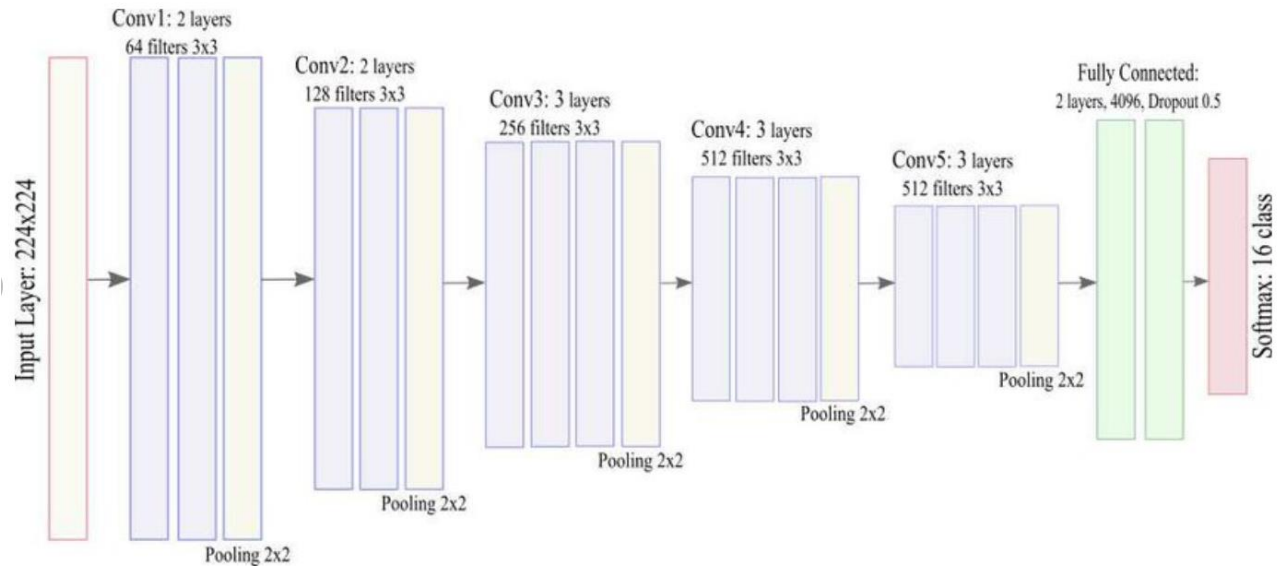


Fig. 4.3 Architecture of primary YOLO based VGG-16

4.3.2.1 Pseudo code for primary YOLO based VGG-16 algorithm

```

vgg16 = applications.VGG16(include_top=False, weights='imagenet')

```

```

bottleneck_features_train = vgg16.predict_generator(generator,
predict_size_train)
np.save('bottleneck_features_train.npy', bottleneck_features_train)

bottleneck_features_validation = vgg16.predict_generator(
    generator, predict_size_validation)
np.save('bottleneck_features_validation.npy', bottleneck_features_validation)

bottleneck_features_test = vgg16.predict_generator(
    generator, predict_size_test)
np.save('bottleneck_features_test.npy', bottleneck_features_test)

```

4.4 Framework Modules

The primary algorithm for the YOLO is VGG-16, which is mainly used for the ship detection framework which can easily make the analysis using the convolution layers in the network, which gives top accuracy and computational efficiency.

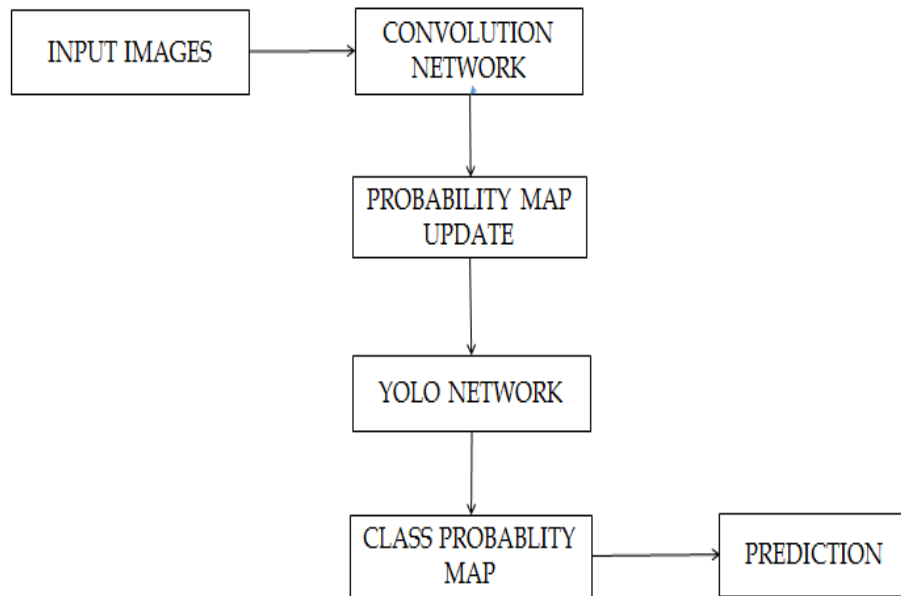


Fig. 4.4 Proposed Model Flow

The modules in this ship detection framework realize the following steps:

Step 1: Data Collection which collects the data from the SAR free open dataset (SFOD) named SAR Ship Dataset.

Step 2: Data preprocessing is a technique that transforms raw data into an understandable format.

Step 3: Training the model which realizes image classification that is ten classes is categorized, each level takes five images

Step 4: Object detection which achieves a hybrid YOLO algorithm, which uses YOLO-VGG16 for object detection added with Spatial Based Pixel Descriptor (SEPD) and K-Means Clustering.

Step 5: Optimizer Analysis is then used for better screening and training of images.

Activation function determines the output for deep learning network whether it is activated or not, the input is relevant for the prediction of the model. The output of each neuron has the finite value ranges between 0 to 1 or -1 to 1. It has its computational efficiency. In this case LeakyRelu and Softmax are used as activation functions. LeakyRelu activation function is a hyperparameter, whose value is set before the learning process begins. Instead of being zero, leaky Relu has a slight non-zero limits. Softmax activation function calculates the probability dissemination of each objective class over all the probable objective modules, and then finally the probabilities calculated will be helpful in determining the objective modules for the given inputs.

Optimizer helps in minimizing or maximizing the error function. It trains the model effectively and efficiently. Optimization algorithm produces accurate results. In the simulation, RMSprop and Cross entropy Loss are realized as optimization functions. RMSprop optimizer is used for increasing the learning rate in the ship detection framework and this optimizer algorithm takes loftier phases in horizontal direction congregating more rapid learning rate and efficiency. Cross Entropy loss function is utilized for estimating the presentation of problems in classification of ship. Cross-entropy loss increments the anticipated probability, in the output value ranges between 0 and 1.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 Software Requirement

Table I. Software Specifications

S. No	Software	Version	URL
1	Anaconda	3.7	https://www.anaconda.com/distribution/
2	Tensorflow	2.0	https://www.tensorflow.org/install/pip
3	Keras	2.3.0	https://keras.io/
4	Numpy	1.11.3	https://numpy.org/
5	Pillow	4.0.0	https://pypi.org/project/Pillow/
6	Pandas	0.19.2	https://pandas.pydata.org/
7	Pyyaml	3.12	http://pyyaml.org/download/pyyaml/

5.1.1 Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and MacOS.

5.1.2 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a

symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

5.1.3 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit. It focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

5.1.4 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

5.1.5 Pillow

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

5.1.6 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations

for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

5.1.7 Pyyaml

YAML is a data serialization format designed for human readability and interaction with scripting languages. PyYAML is a YAML parser and emitter for the Python programming language.

5.2 Hardware Requirement

Table II. Hardware Specification

Processor	IntelCore
Operating System	Windows 10 (64-bit)
RAM	8 GB

5.3 Installation procedure

Step 1 — Install the dependencies for Windows

1. Download & install Anaconda package 64-bit version [7] and choose the Python 3.6 version. This automatically installs Python and many popular data scientist/ML libraries (*NumPy*, *Scikit-Learn*, *Pandas*, *R*, *Matplotlib*...), tools (*Jupyter Notebook*, *RStudio*) and hundreds of other open source packages for your future projects. For example, the Anaconda Jupyter Notebook is used for all experiments. OpenCV library is not included though and we will install it separately as it is needed for real-time computer vision tasks.

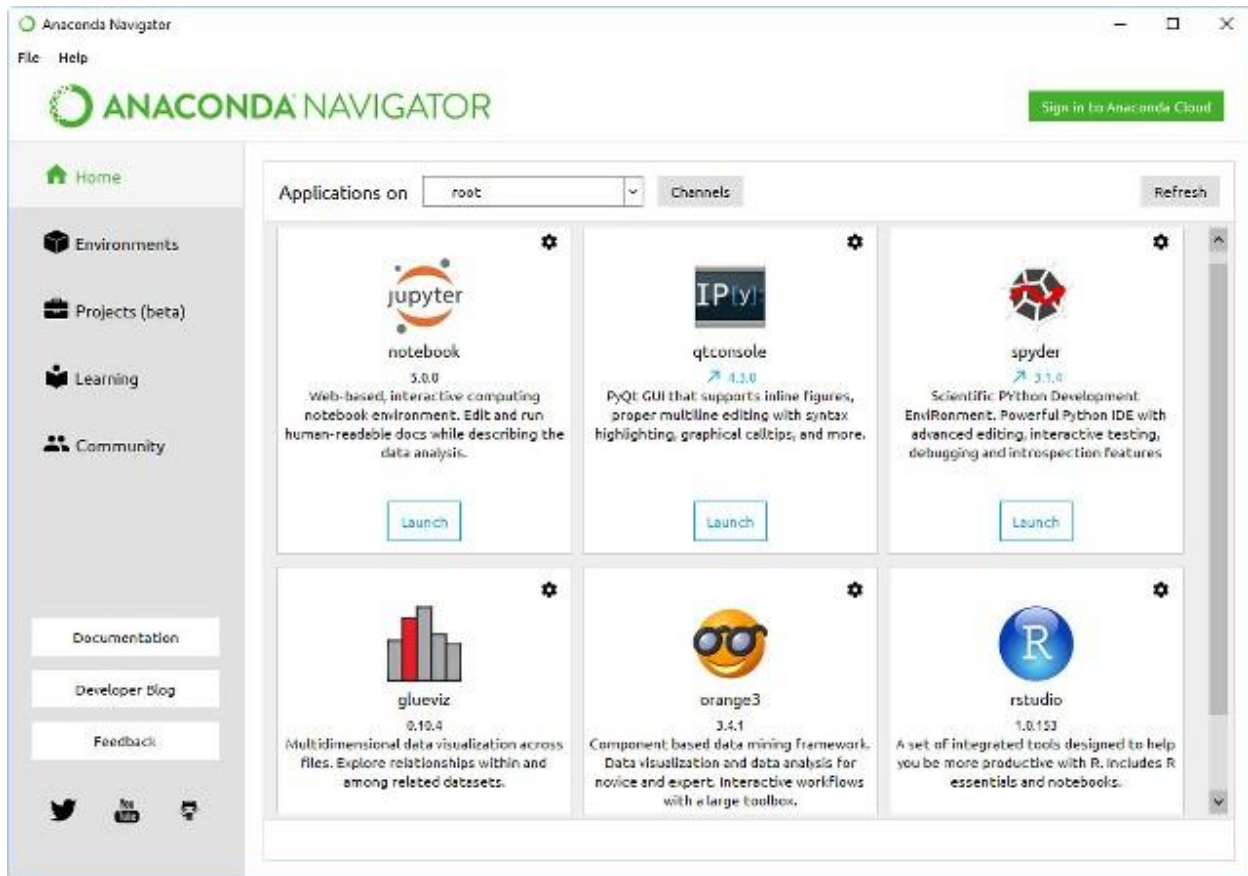


Fig. 5.1 Anaconda Navigator

2. Install Tensorflow and Keras. TensorFlow is the most popular AI software library and is created/maintained by Google. Keras is another highly popular & high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation.

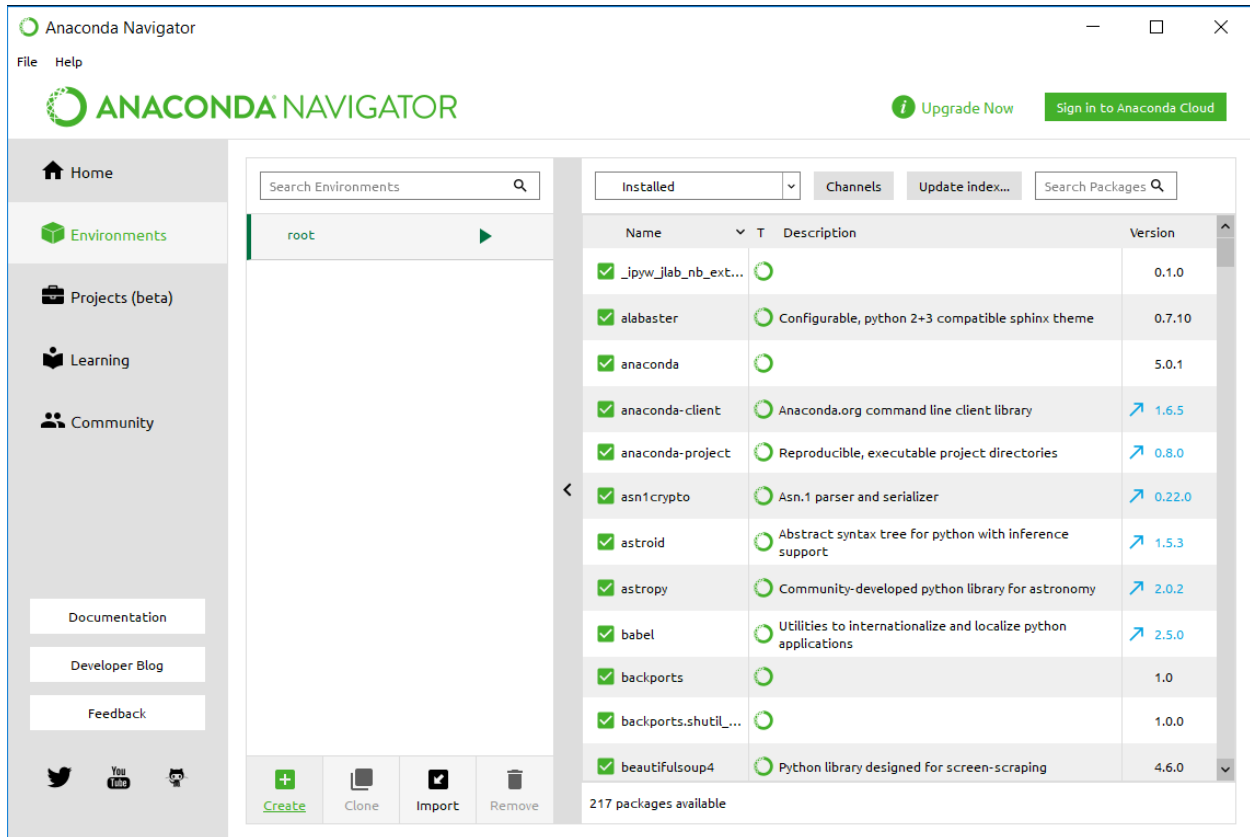


Fig. 5.2 Installation of Anaconda Packages

Step 2 — Install the DarkNet/YOLA, Darkflow stuff

DarkNet: Originally, YOLO algorithm is implemented in DarkNet framework by Joseph Redmon. Darknet is an open source custom neural network framework written in C and CUDA. It is fast, easy to install, and supports both CPU and GPU computations.

Darkflow: It is a nickname of an implementation of YOLO on TensorFlow. Darknet models are converted to Tensorflow and can be installed on both Linux and Windows environments.

Open your anaconda prompt and clone the darkflow github repository.

```
git clone https://github.com/thtrieu/darkflow
```

Alternative is to basically go to the **DarkFlow GitHub** page and download the master repository to your local

```
# If you have not already created a new virtual environment in Step 1, then create a
conda environment for darkflow installation.
conda create -n your_env_name python=3.6
# Activate the new environment using anaconda prompt.
activate your_env_name
# You can install the needed OpenCV with a conda-forge repository. conda-forge is
a github organization containing repositories of conda libraries.
conda config --add channels conda-forge
conda install opencv
# Build the Cython extensions in place. This is a widely used Python to C compiler
and wrapper that helps us to call the DarkNet C-code from Python.
python setup.py build_ext --inplace
or try the following as alternative
pip install --e
```

5.4 Dataset Description

The dataset used here is MASATI (MARitime SATellite Images) dataset [12]. This dataset provides maritime scenes of optical aerial images from visible spectrum. The MASATI dataset contains color images in dynamic marine environments, and it can be used to evaluate ship detection methods. Each image may contain one or multiple targets in different weather and illumination conditions. The datasets is composed of 7,389 satellite images labeled according to the following seven classes: land, coast, sea, ship, multi, coast-ship, and detail. In addition, labeling with the bounding box for the location of the vessels is also included.

Table III. Sample Distribution of each class

Main class	Sub-class	Samples	Description
Ship	Ship	1027	Sea with a ship (no coast).
	Detail	1789	Ship details.
	Multi	304	Multiple ships.
	Coast & ship	1037	Coast with ships.
Non-ship	Sea	1022	Sea (no ships).
	Coast	1132	Coast (no ships).
	Land	1078	Land (no sea)

For evaluation we have defined three additional sets by grouping samples of several classes as follows:

- **Set 1:** Ship on high sea and ocean or high sea without ship.
- **Set 2:** Set 1 plus two new subsets: ship on sea close to coast (then coast is visible), and coast (sea scene with coast visible but without ship).
- **Set 3:** Set 2 plus three new subsets: ship image acquired at lower altitude compared with the set 1, land (inland this is without coastal areas), and multi (multiple instances of ships).

The satellite images were acquired from Bing Maps in RGB and with different sizes, as size is dependent on the region of interest to be registered in the image. In general, the average image size has a spatial resolution around 512 x 512 pixels. The images are stored as PNG where pixel values represent RGB colors. The distance between targets and the acquisition satellite has also been changed in order to obtain captures at different altitudes.

To label the category of each image, an organization divided into folders was used, where each folder represents a category.



(a)



(b)



(c)

Fig. 5.3(a), (b), (c) Sample SAR Images from the dataset

CHAPTER 6

IMPLEMENTATION AND RESULTS

Keras framework is used for the ship detection model is experimented using deep learning technique. The pre-training model used in this Hybrid YOLO – VGG16 to prepare the system of the network. 700 images fewer than 7 classes with each 10 images is trained with 25k iterations and the learning rate for this ship detection framework is 0.0001.

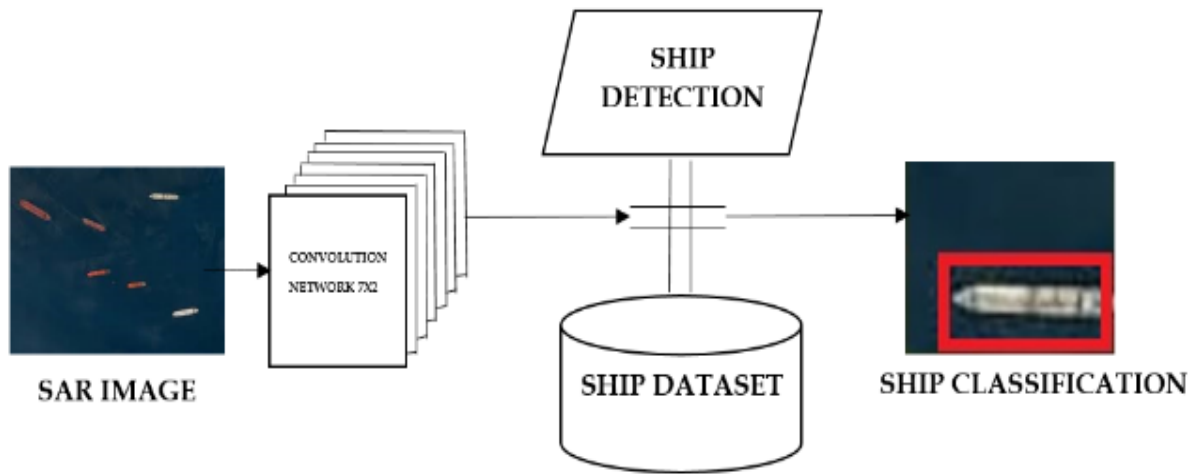


Fig 6.1 Training the model

Optimizer Analysis is for better screening and training of images. The optimization algorithm used for this ship detection framework is RMSProp, and the activation function used is Leaky ReLu and Softmax functions.

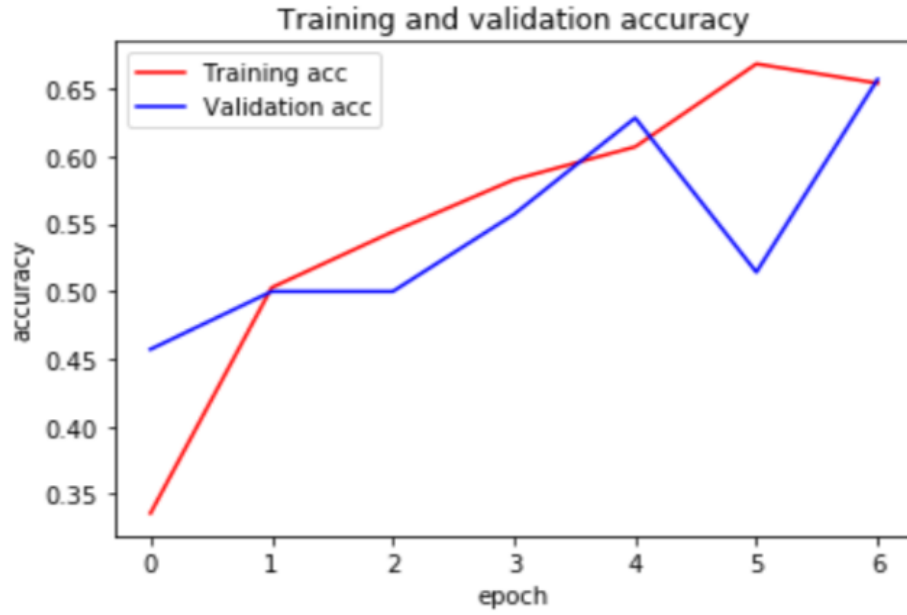


Fig. 6.2 Accuracy

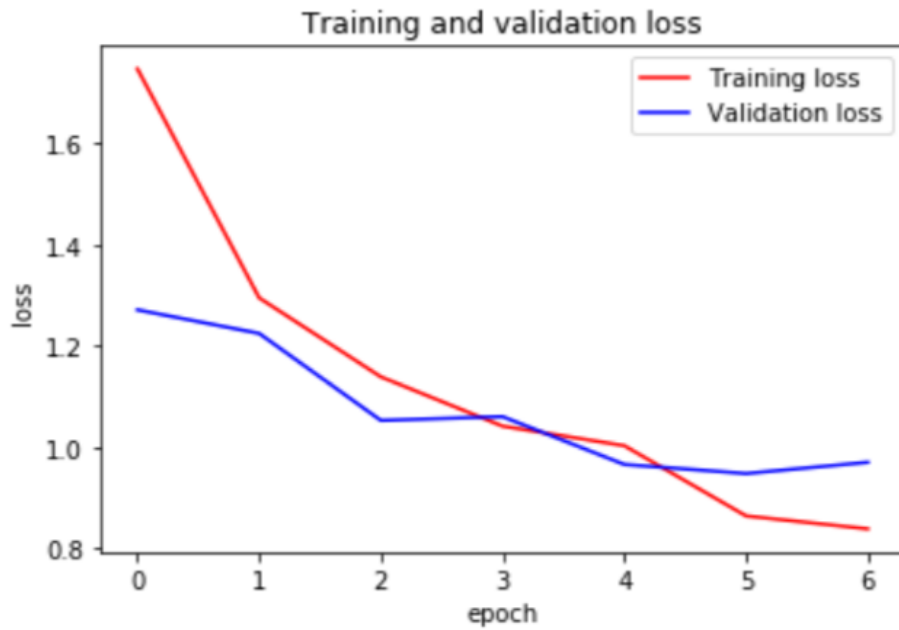


Fig. 6.3 Loss function

Performance of the proposed framework is evaluated with Ship based SAR images for the performance metrics namely precision, recall, f1-score and, support. Table I summarizes the investigational outcomes for figure 5 in differentiation of ships from coastal and sea areas.

Table IV. Performance Analysis

	Precision	Recall	F1-score	Support
Coast	0.75	0.60	0.67	10
Coast-ship	1.00	0.20	0.33	10
Detail	1.00	0.80	0.89	10
Land	0.89	0.80	0.84	10
Multi	0.00	0.00	0.00	10
Ship	0.00	0.00	0.00	10
Water	0.80	0.40	0.53	10
Micro Avg	0.85	0.40	0.54	70
Macro Avg	0.63	0.40	0.47	70
Weighted Avg	0.63	0.40	0.47	70
Samples Avg	0.40	0.40	0.40	70

Specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table). The confusion matrix obtained for this ship detection framework is shown below

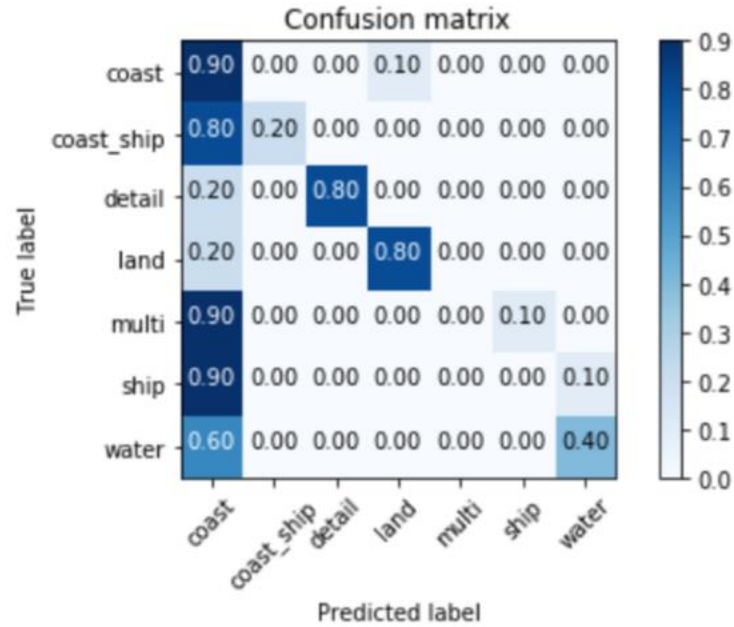


Fig.6.4 Confusion matrix



Fig. 6.5 Sample Input Image

The above figure.6 is feeded to the proposed hybrid YOLO framework for landscape differentiation. The learning ratio of the proposed framework for the figure 6 is

described below and the image is classified under the label: Cruise for ship classification.



Fig. 6.6 Classified Ship Image

ID: 0, Label: coast 2.69%

ID: 1, Label: coast-ship 4.65%

ID: 2, Label: detail 8.57%

ID: 3, Label: land 3.59%

ID: 4, Label: multi 11.49%

ID: 5, Label: ship 36.09%

ID: 6, Label: cruise

From the above results in figure.7, it is viewed that the exactness of the proposed environment increases densely, if more number of images is embedded at the training phase.

CHAPTER 7

PERFORMANCE COMPARISON

The proposed model is compared with existing frameworks for ship detection such as Faster RNN, CNN and RetinaNet [16]. Table II depicts the performance analysis of the ship detection using ShipSAR dataset which was performed using various existing algorithms and the proposed Hybrid YOLO-VGG16 model.

Table V. Comparison Analysis

Models	Accuracy	Precision	Recall	F1-score
CNN	0.57	0.70	0.50	0.67
RNN	0.62	0.73	0.60	0.33
Faster-RNN	0.65	0.79	0.70	0.69
RetinaNet	0.67	0.85	0.80	0.64
Hybrid YOLO-VGG16	0.72	0.89	0.80	0.80

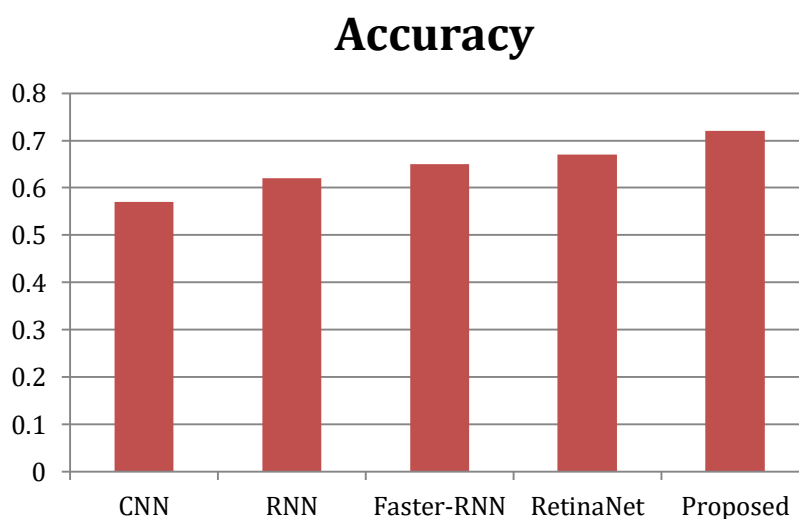


Fig. 7.1 Accuracy Comparison

Figure.8 inscribes the accuracy performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.72 which is better than other existing frameworks.

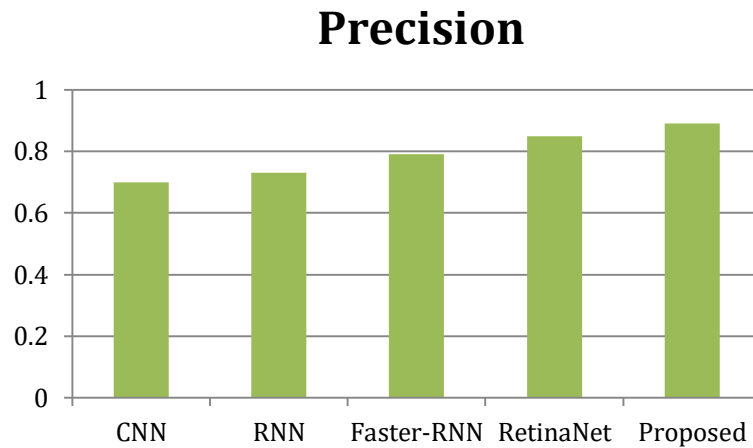


Fig. 7.2 Precision Comparison

Figure.9 engraves the Precision performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.89 which outperforms all other existing frameworks.

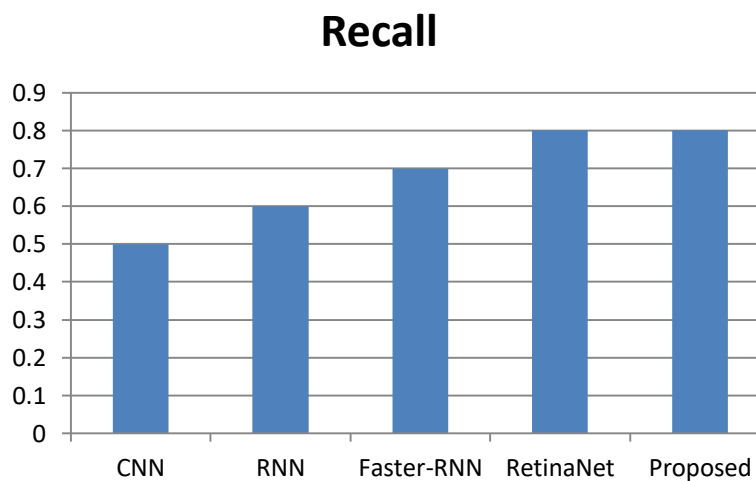


Fig. 7.3 Recall Comparison

Figure.10 incises the recall performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.80 which beats up the other existing frameworks.

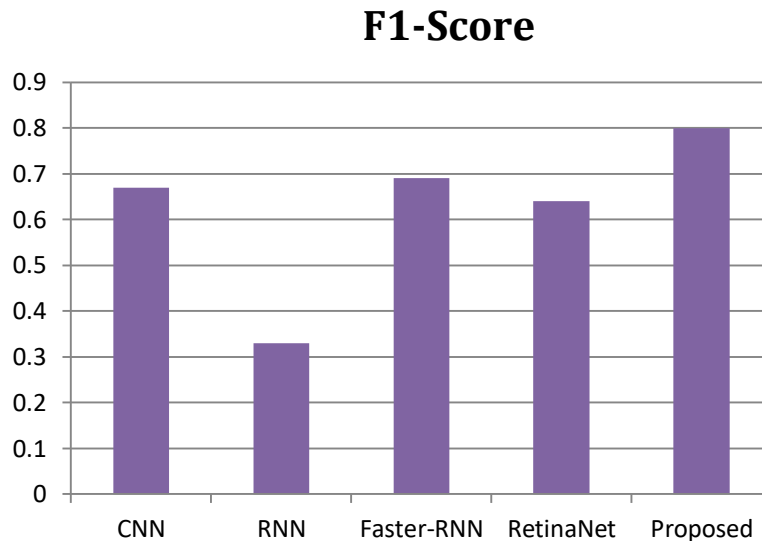


Fig. 7.4 F1-Score Comparison

Figure.11 etches the F1-Score performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.80 which top performs than other existing frameworks.

From the above performance analysis, the proposed hybrid YOLO-VGG16 outperforms all other efficient existing ship detection algorithms in terms of accuracy, precision, recall and F1-score.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In this project, how SAR images can be useful for Ship Detection model and how it can be used for examining the oceanic activities is explored. Despite the advancement in Ship Detection using various techniques and algorithms, they still face some issues in the accuracy and computational efficiency. The proposed framework is Hybrid YOLO which is the combination of YOLO-VGG16, SEPD and K-Means Clustering. The parameter determined in this ship detection framework is optimization algorithm, activation function, loss function, and hidden layer. This Hybrid YOLO gives more accuracy and computational efficiency than other existing models because, the ships detected from the SAR images is categorized into ten classes and five images per class is trained for this framework. Thereby it detects the ship from the images, and clusters the images model with same classes, proving its high accuracy and top computational efficiency.

APPENDIX – I

CODING

Dataset Split

```
import os
import numpy as np
import shutil

root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'coast'
os.makedirs(root_dir + '/train' + imagetype)
os.makedirs(root_dir + '/val' + imagetype)
os.makedirs(root_dir + '/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/" + imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src + '/' + name for name in train_FileNames.tolist()]
val_FileNames = [src + '/' + name for name in val_FileNames.tolist()]
test_FileNames = [src + '/' + name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
```



```

print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)
root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'coast_ship'
os.makedirs(root_dir + '/train' + imagetype)
os.makedirs(root_dir + '/val' + imagetype)
os.makedirs(root_dir + '/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/" + imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src + '/' + name for name in train_FileNames.tolist()]
val_FileNames = [src + '/' + name for name in val_FileNames.tolist()]
test_FileNames = [src + '/' + name for name in test_FileNames.tolist()]

```

```

print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)
root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'detail'
os.makedirs(root_dir + '/train' + imagetype)
os.makedirs(root_dir + '/val' + imagetype)
os.makedirs(root_dir + '/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/"+imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])

```

```

train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
val_FileNames = [src+'/'+ name for name in val_FileNames.tolist()]
test_FileNames = [src+'/'+ name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)
root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'land'
os.makedirs(root_dir +'/train' + imagetype)
os.makedirs(root_dir +'/val' + imagetype)
os.makedirs(root_dir +'/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/"+imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)

```

```

train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
val_FileNames = [src+'/'+ name for name in val_FileNames.tolist()]
test_FileNames = [src+'/'+ name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)
root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'multi'
os.makedirs(root_dir +'/train' + imagetype)
os.makedirs(root_dir +'/val' + imagetype)
os.makedirs(root_dir +'/test' + imagetype)
currentCls = imagetype

```

```

src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/"+imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
val_FileNames = [src+'/'+ name for name in val_FileNames.tolist()]
test_FileNames = [src+'/'+ name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)
root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'ship'
os.makedirs(root_dir +'/train' + imagetype)

```

```

os.makedirs(root_dir + '/val' + imagetype)
os.makedirs(root_dir + '/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/"+imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
val_FileNames = [src+'/'+ name for name in val_FileNames.tolist()]
test_FileNames = [src+'/'+ name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train"+currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val"+currentCls)
for name in test_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/test"+currentCls)

```

```

root_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/'
imagetype = 'water'
os.makedirs(root_dir + '/train' + imagetype)
os.makedirs(root_dir + '/val' + imagetype)
os.makedirs(root_dir + '/test' + imagetype)
currentCls = imagetype
src = "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/" + imagetype
allFileNames = os.listdir(src)
np.random.shuffle(allFileNames)
train_FileNames, val_FileNames, test_FileNames =
np.split(np.array(allFileNames), [int(len(allFileNames)*0.8),
int(len(allFileNames)*0.9)])
train_FileNames = [src + '/' + name for name in train_FileNames.tolist()]
val_FileNames = [src + '/' + name for name in val_FileNames.tolist()]
test_FileNames = [src + '/' + name for name in test_FileNames.tolist()]
print('Total images: ', len(allFileNames))
print('Training: ', len(train_FileNames))
print('Validation: ', len(val_FileNames))
print('Testing: ', len(test_FileNames))
for name in train_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/train" + currentCls)
for name in val_FileNames:
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/val" + currentCls)

```

```
for name in test_FileNames:
```

```
    shutil.copy(name, "C:/Users/DHIVYA/SAR Radar Ship Detection final  
project/DATASET/test"+currentCls)
```

Ship Detection

```
import pandas as pd
```

```
import numpy as np
```

```
import itertools
```

```
import keras
```

```
from sklearn import metrics
```

```
from sklearn.metrics import confusion_matrix
```

```
from keras.preprocessing.image import ImageDataGenerator, img_to_array,  
load_img
```

```
from keras.models import Sequential
```

```
from keras import optimizers
```

```
from keras.preprocessing import image
```

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

```
from keras import applications
```

```
from keras.utils.np_utils import to_categorical
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
%matplotlib inline
```

```
import math
```

```
import datetime
```

```
import time
```

```
import warnings
```



```

import PIL as pillow
warnings.simplefilter("ignore")
img_width, img_height = 224, 224
top_model_weights_path = 'bottleneck_fc_model.h5'
train_data_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/TRAIN'
validation_data_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/VALIDATION'
test_data_dir = 'C:/Users/DHIVYA/SAR Radar Ship Detection final
project/DATASET/TEST'
epochs = 7
batch_size = 50
vgg16 = applications.VGG16(include_top=False, weights='imagenet')
warnings.simplefilter("ignore")
datagen = ImageDataGenerator(rescale=1. / 255)
start = datetime.datetime.now()
generator = datagen.flow_from_directory(
train_data_dir,
target_size=(img_width, img_height),
batch_size=batch_size, class_mode=None, shuffle=False)
nb_train_samples = len(generator.filesnames)
num_classes = len(generator.class_indices)
predict_size_train = int(math.ceil(nb_train_samples / batch_size))
bottleneck_features_train = vgg16.predict_generator(generator, predict_size_train)
np.save('bottleneck_features_train.npy', bottleneck_features_train)
end= datetime.datetime.now()
elapsed= end-start

```

```

print ('Time: ', elapsed)
start = datetime.datetime.now()
generator = datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)
nb_validation_samples = len(generator.filesnames)
predict_size_validation = int(math.ceil(nb_validation_samples / batch_size))
bottleneck_features_validation = vgg16.predict_generator(
    generator, predict_size_validation)
np.save('bottleneck_features_validation.npy', bottleneck_features_validation)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
warnings.simplefilter("ignore")
start = datetime.datetime.now()
generator = datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)
nb_test_samples = len(generator.filesnames)
predict_size_test = int(math.ceil(nb_test_samples / batch_size))
bottleneck_features_test = vgg16.predict_generator(

```

```

    generator, predict_size_test)
np.save('bottleneck_features_test.npy', bottleneck_features_test)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
generator_top = datagen.flow_from_directory(train_data_dir,
target_size=(img_width, img_height),batch_size=batch_size,
class_mode='categorical', shuffle=False)
nb_train_samples = len(generator_top.fileNames)
num_classes = len(generator_top.class_indices)
train_data = np.load('bottleneck_features_train.npy')
train_labels = generator_top.classes
train_labels = to_categorical(train_labels, num_classes=num_classes)
generator_top = datagen.flow_from_directory(validation_data_dir,
target_size=(img_width, img_height), batch_size=batch_size, class_mode=None,
shuffle=False)
nb_validation_samples = len(generator_top.fileNames)
validation_data = np.load('bottleneck_features_validation.npy')
validation_labels = generator_top.classes
validation_labels = to_categorical(validation_labels, num_classes=num_classes)
generator_top = datagen.flow_from_directory(test_data_dir,
target_size=(img_width, img_height), batch_size=batch_size, class_mode=None,
shuffle=False)
nb_test_samples = len(generator_top.fileNames)
test_data = np.load('bottleneck_features_test.npy')
test_labels = generator_top.classes
test_labels = to_categorical(test_labels, num_classes=num_classes)

```

```

start = datetime.datetime.now()
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Dense(50, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])
history = model.fit(train_data, train_labels, epochs=7, batch_size=batch_size,
validation_data=(validation_data, validation_labels))
model.save_weights(top_model_weights_path)
(eval_loss, eval_accuracy) = model.evaluate(validation_data, validation_labels,
batch_size=batch_size, verbose=1)
print("[INFO] accuracy: {:.2f}%".format(eval_accuracy * 100))
print("[INFO] Loss: {}".format(eval_loss))
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
warnings.simplefilter("ignore")
model.summary()
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

```

```

plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
model.evaluate(test_data, test_labels)
print('test data', test_data)
preds = np.round(model.predict(test_data),0)
print('rounded test_labels', preds)
signs = ['coast', 'coast_ship', 'detail', 'land', 'multi', 'ship', 'water']
classification_metrics = metrics.classification_report(test_labels, preds,
target_names=signs )
print(classification_metrics)
categorical_test_labels = pd.DataFrame(test_labels).idxmax(axis=1)
categorical_preds = pd.DataFrame(preds).idxmax(axis=1)
confusion_matrix= confusion_matrix(categorical_test_labels, categorical_preds)
def plot_confusion_matrix(cm, classes,
                        normalize=False,

```

```

        title='Confusion matrix',
        cmap=plt.cm.Blues):
    """prints pretty confusion metric with normalization option """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color="white"
        if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

plot_confusion_matrix(confusion_matrix, ['coast', 'coast_ship', 'detail', 'land',
'multi', 'ship', 'water'])
plot_confusion_matrix(confusion_matrix,
                      ['coast', 'coast_ship', 'detail', 'land', 'multi', 'ship', 'water'],
def read_image(file_path):

```

```

print("[INFO] loading and preprocessing image...")
image = load_img(file_path, target_size=(224, 224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image /= 255.

return image

def test_single_image(path):
    signs = ['coast', 'coast_ship', 'detail', 'land', 'multi', 'ship', 'water']
    images = read_image(path)
    time.sleep(.5)
    bt_prediction = vgg16.predict(images)
    preds = model.predict_proba(bt_prediction)
    for idx, radar, x in zip(range(0,6), signs , preds[0]):
        print("ID: { }, Label: { } { }%".format(idx, radar, round(x*100,2) ))
    print('Final Decision:')
    time.sleep(.5)
    for x in range(3):
        print('.'*(x+1))
        time.sleep(.2)
    class_predicted = model.predict_classes(bt_prediction)
    class_dictionary = generator_top.class_indices
    inv_map = {v: k for k, v in class_dictionary.items()}
    print("ID: { }, Label: { }".format(class_predicted[0],
    inv_map[class_predicted[0]]))
    return load_img(path)

```

```
path = 'C:/Users/DHIVYA/Desktop/SAR Radar Ship Detection final project/SAR  
Radar Ship Detection final project/DATASET 7 CLASSES OF 100 IMAGES  
EACH/TRAIN/trainship/s0019.png'  
test_single_image(path)
```


APPENDIX – II

REFERENCES

- [1] Miguel M.Pinto, Renata Libonati, Ricardo M.Trigo, Isabel F.Trigo, Carlos C.DaCamara, A deep learning approach for mapping and dating burned areas using temporal sequences of satellite images, ISPRS Journal of Photogrammetry and Remote Sensing Volume 160, February 2020, Pages 260-274.
- [2] Nina W., Condori W., Machaca V., Villegas J., Castro E. (2020) Small Ship Detection on Optical Satellite Imagery with YOLO and YOLT. In: Arai K., Kapoor S., Bhatia R. (eds) Advances in Information and Communication. FICC 2020. Advances in Intelligent Systems and Computing, vol 1130. Springer, Cham.
- [3] Caglar Gulcehre, Deep Learning, 2015, <http://deeplearning.net/>.
- [4] Yuanyuan Wang, Chao Wang, Hong Zhang, Yingbo Dong, and Sisi Wei “A SAR Dataset of Ship Detection for Deep Learning under Complex Backgrounds” Remote Sensing 2019.
- [5] Krizhevsky A, Sutskever I, Hinton G E “ImageNet classification with deep convolutional neural networks” International Conference on Neural Information Processing Systems, 2012.
- [6] Changchong Lu and Weihai Li “Ship Classification in High-Resolution SAR Images via Transfer Learning with Small Training Dataset” mdpi, 2018.
- [7] Anaconda tool available online on <https://www.anaconda.com/distribution/>.
- [8] Yang-Lang Chang, Amare Anagaw, Lena Chang, Yi Chun Wang, Chih-Yu Hsiao, and Wei-Hong Lee”Ship Detection Based on YOLOv2 for SAR Imagery” Remote Sensing 2019.

- [9] Arnold E. Kiv, Serhiy O. Semerikov, Vladimir N. Soloviev, Andrii M. Striuk, Convolutional neural networks for image classification, November 2019, <http://elibrary.kdpu.edu.ua/xmlui/handle/123456789/3682>.
- [10] Michal Segal-Rozenhaimer, Alan Li, Kamalika Das, Ved Chirayath, Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (CNN), Remote Sensing of Environment, Volume 237, February 2020.
- [11] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T “Convolutional architecture for fast feature embedding” 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
- [12] MASATI dataset (v2) - MAritime SATellite Imagery dataset, <https://www.iuii.ua.es/datasets/masati/>
- [13] Hongwei Zhao, Weishan Zhang, Haoyun Sun, Bing Xue “Embedded Deep Learning for Ship Detection and Recognition” future internet, 2019.
- [14] Shunjun Wei, Hao Su, Jing Ming, Chen Wang, Min Yan, Durga Kumar, Jun Shi, Xiaoling Zhang “Precise and Robust Ship Detection for High-Resolution SAR Imagery Based on HR-SDNet” Remote Sensing, 2019.
- [15] Xue Yang, Hao Sun, Kun Fu1, Jirui Yang, Xian Sun, Menglong Yan, Zhi Guo “Automatic Ship Detection of Remote Sensing Images from Google Earthin Complex Scenes Based on Multi-Scale Rotation Dense Feature Pyramid Networks.
- [16] Pasquale Iervolino, Raffaella Guida, Parivash Lumsdon, Jürgen Janoth, Melanie Clift, Andrea Minchella, Paolo Bianco “Ship Detection in SAR Imagery: A Comparison Study” Research Gate 2017.
- [17] Abdelrahman Hosny, Anthony Parziale “A Study On Deep Learning”.

- [18] Haitao Lang, Yuyang Xi, Xi Zhang“Ship Detection in High-Resolution SAR Images by Clustering Spatially Enhanced Pixel Descriptor” IEEE Transactions on Geoscience and Remote Sensing 2019.
- [19] Ying Liu, Hong-Yuan Cui, Zheng Kuang and Guo-Qing Li “Ship Detection and Classification on Optical Remote Sensing Images Using Deep Learning” published in “ITM Web of Conferences 12, (2017)”.
- [20] Tianwen Zhang and Xiaoling Zhang “High-Speed Ship Detection in SAR Images Based on a Grid Convolutional Neural Network” Remote Sensing 2019.

APPENDIX – III

PUBLICATIONS

[1] Devadharshini S, Kalaipriya R, Rajmohan R “Performance Investigation of Hybrid YOLO – VGG16 based Ship Detection Framework using SAR Images” 3rd International Conference on Systems, Computation, Automation and Networking (ICSCAN), July 2020. [Accepted for publication].

[2] Devadharshini S, Kalaipriya R, Rajmohan R “Deep Learning Models for Image Analysis” Indo - Israel International Joint Conference on Sustainable Cities (IICSC), January 2020.

Performance Investigation of Hybrid YOLO-VGG16 Based Ship Detection Framework Using SAR Images

S.Devadharshini¹, R.Kalaipriya², R.Rajmohan³, M.Pavithra⁴, Dr.T.Ananthkumar⁵

Research Student^{1,2}, Associate Professor³, Assistant Professor^{4,5}

Department of CSE^{1,2,3,4,5}

IFET College of Engineering^{1,2,3,4,5}

Villupuram, India

dhharshu0018@gmail.com, r.99kalai@gmail.com, rjmohan89@gmail.com, pavimuthu27@gmail.com, ananth.eec@gmail.com

Abstract—Synthetic Aperture Radar (SAR) images are realized as encouraging data information for checking oceanic activities and its function for oil and ship recognizable proof, which is the focal point of numerous past research considers for better spatial goals. Several article discovery strategies extending from customary to deep learning approaches are proposed. Ship detection framework in deep learning technique accomplishes high execution, which benefits from a SAR free open dataset (SFOD). Nonetheless, a dominant part of them are computationally dangerous and have exactness issues. The main problem identified is when the number of images increases, performance may decrease. To overcome this, we propose a technique called Hybrid YOLO, which realizes K-Means Clustering and WordTree for object identification and image classification. Hybrid YOLO also realizes SEPD for the improvement between sea clutter and ship targets and bounding boxes for the probability update network. The proposed model is implemented using Conda, used with Tensorflow and Keras Framework utilizing the SAR Ship Dataset. The presentation of the Hybrid YOLO model is enhanced in rapports of accuracy and performance measures when compared with other existing models.

Keywords—SAR; Deep learning; Ship detection; Hybrid YOLO; VGG16

I. INTRODUCTION

Ship location is significant for marine observation in zones, for example, illicit fishing, oil slick discovery, marine traffic the executives, and sea theft. The SAR [1] images in ship detection are depicted as having significant standards limit, not being dependent upon the climatic case and self-governing of flight height. It consistently gives quality pictures at any position due to its automatic light capacity and has a ton of uses in remote detecting. The idea of deep learning taking in begins from the investigation of artificial neural systems. Deep learning [3] have made momentous accomplishments in the field of image processing, particularly for object discovery. Ship detection using SAR images in the deep learning approach is advantageous with its performance and accuracy. Convolution network having more number of layers is used for training the images. YOLO utilizes a very surprising methodology. It applies a unique neural structure to the entire picture. This framework isolates the map into zones and predicts bounding boxes in the form of grid and class probabilities for every district. The foreseen probabilities weight these bounding boxes. YOLO [2] is a best in class

ongoing item recognition framework, which beats Faster R-CNN, RetinaNet, and SSD strategies [9, 10]. Specifically, we focus on unsolved problems like computational time, high performance, high accuracy, and enhanced technology for ship detection framework. To diminish computational time with moderately aggressive discovery exactness, we build up another design with less number of layers called Hybrid YOLO. Hybrid YOLO is the YOLO-VGG16 algorithm added with Spatial Based Pixel Descriptor (SEPD) and K-means clustering. This framework is proposed by using the Anaconda tool, which is combined with Tensorflow, Keras, Numpy, h5py, Pillow, and Python3. The TensorFlow and Keras [12] framework play a vital role in this ship detection. The performance that is measured in this Hybrid YOLO ship detection framework is precision, recall, F1-score, accuracy, and specificity. The optimization algorithm used in this ship detection model is RMSProp, and the activation function used is Leaky ReLu and Softmax functions.

This paper is systematized in trails such as, the section II narrates the related work done in this Ship Detection domain, the section III explains the architecture and design of the proposed methodology, the section IV describes the simulation and discussion for the ship detection model, the section V demonstrates the implementation and results of the ship detection framework, the section VI details the performance compassion of existing algorithms with the proposed model, section VII declares the conclusion and future work of our ship detection model, and the last section provides the references.

II. RELATED WORK

Hinton *et al.* [5] proposed an SSD is a run of the mill one phase identifier, which procedures pictures in a single system, and has excellent efficiency and precision. Changhong Lu *et al.* [6] proposed a CNN model that applies for ship allocation and location by using SAR images with small datasets. Shaoming Zhang *et al.* [7] proposed a technique called R-CNN, which is a pioneering approach that first creates a 2K area proposition and recognizes the object inside every district proposition. This R-CNN uses the ROI (Region of Interest) images gives the single feature map for the task of Ship Detection. Yuanyuan Wang *et al.* [4] proposed a Faster R-CNN is a two-stage locator, which utilizes RPN (Region Proposal Network) to create an excellent locale proposition and afterward identify them. It doesn't feed 2K regions every time. Instead, convolution operation once per

image and the feature map is generated. RetinaNet is the backbone network that has a convolution feature map of an input image. It is categorized into two subnets. One subnet performs classification, and another subnet performs convolution bounding box regression, which predicts localization in object detection. *Castro E et al* [2] proposed a single neural system named YOLO, which forsakes grapple boxes and forecasts hurdling containers and class probabilities legitimately from a full picture in one assessment. YOLO considers object identification as a relapse issue to anticipate bounding boxes and class probabilities. It tends to be improved as start to finish straightforwardly with excellent recognition execution. *Yang-Lang Chang et al.* [8] proposed a Fast YOLO that can process 155 edges for every second. Contrasted and other best in class discovery calculations, YOLO makes more limitation mistakes. YOLOV2 depends on YOLO. YOLOV2 expels the completely associated layers from YOLO and utilizations grapple boxes to anticipate bouncing boxes. The YOLOV2 model can run with different picture sizes, and it is

anything but difficult to make an exchange off among speed and precision. YOLOV2 is quicker than YOLO, which can process 200 edges for every second with the tiny model.

Hongwei Zhao et al. [13] proposed a technique called ESDR-DL, where it is used for the video stream is prepared utilizing installed gadgets and structured a two-level DCNet, made out of a DNet and a CNet, seriatim on inserted gadgets. *Shunjun Wei et al.* [14] proposed a technique named HR-SD Net is used for high-resolution ship detection network for its detection. NMS is used for its high-performance detection in dense ships. This high-resolution ship image provides accurate results for ship detection. *Xue Yang et al.* [15] proposed R-DFPN viably recognize send in various sights comprising of sea and coast. In particular, we set forward the (DFPN), which is planned for tackling the issue that came about because of the restricted width of the ship with Contrasted and past multi-scale finders.

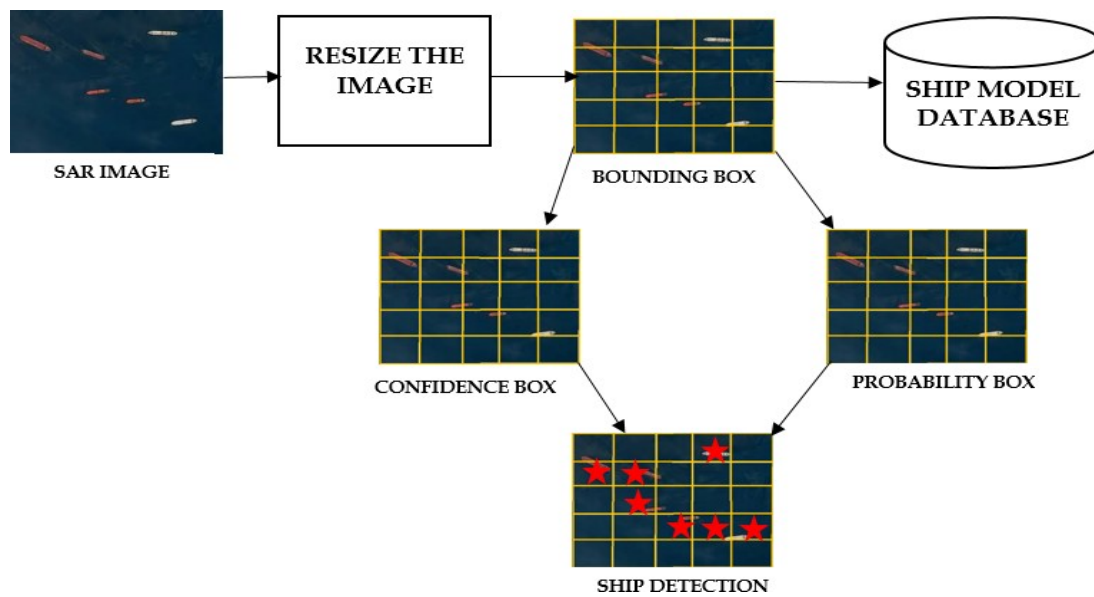


Fig. 1. The architecture of Hybrid YOLO-VGG16

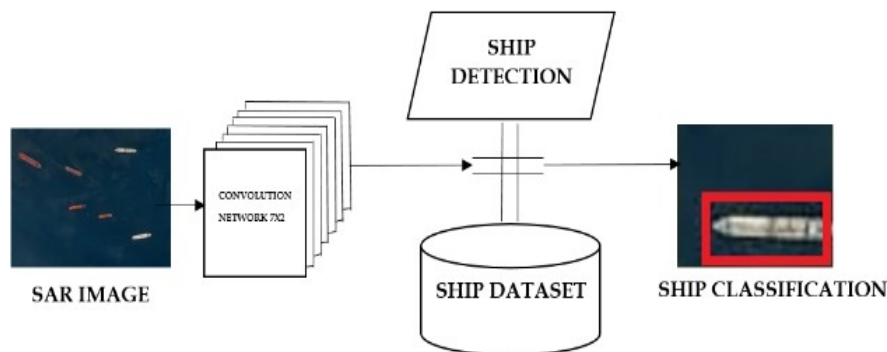


Fig. 2. Training Model Flow

III. PROPOSED METHODOLOGY

The proposed framework figure.1 realizes Hybrid YOLO is implemented by the ship detection and classification that involves a two-stage process, namely Spatial Based Pixel Descriptor (SEPD) and K-Means Clustering. Hybrid YOLO realizes YOLO-VGG16 is a standardized system for identification over many classes by together advancing recognition and characterization by hierarchical classification. The WordTree is used to consolidate information from different sources and our joint enhancement method to prepare all the while on SAR pictures. Spatial Based Pixel Descriptor (SEPD) is used for the detection of improvement between ship targets and sea clutter.

Hybrid YOLO realizes the classification and clustering of the ship detection framework by using the SAR images. These SAR images can be collected from the SAR Free Open Dataset (SFOD). Since the SAR images are collected from various sources, it may vary in its size. For this size variation of images, resizing an image is done. Resizing of the image can be done by data pre-processing. After resizing the image, a bounding box is performed. The bounding box is a technique that splits the image as a 5X5 grid. This bounding box involves two categories. One is the confidence box, which detects all the objects in the image, and the other is the probability box, which realizes the class probability for the detection of a ship from the SAR images. Further, the ship detection model is performed, and that ship detection model is stored in the database.

To improve the accuracy of the ship detection model, a Spatial Enhanced Pixel Descriptor (SEPD) is used to find the difference between the sea clutters and ship targets. The ship is categorized into ten classes, and each class consists of five images per class. The ship classes which are detected is clustered by using K-Means Clustering, which improves more accuracy in the ship detection model.

IV. SIMULATION MODEL

The ship detection framework in figure.2 and figure 3 is implemented using the Anaconda tool [12] and TensorFlow framework utilizing the SAR Ship Dataset [11]. Our experiments for this ship detection framework were conducted on IntelCore i7 64980U CPU and 8.00 GB of with Windows 64-bit OS (Version 10). The software embedded for the ship detection model is Darknet and Darkflow framework with Numpy, h5py, Pillow, and Python3.

Darknet is a groundwork used to train the neural networks. This Darknet has been ported into TensorFlow as Dark flow. The Darknet plays a major role in this Ship Detection Model, Tensorflow and Keras framework is used to implement the ship by bounding box.

The primary algorithm for the YOLO is VGG-16, which is mainly used for the ship detection framework which can easily make the analysis using the convolution layers in the network, which gives top accuracy and computational efficiency.

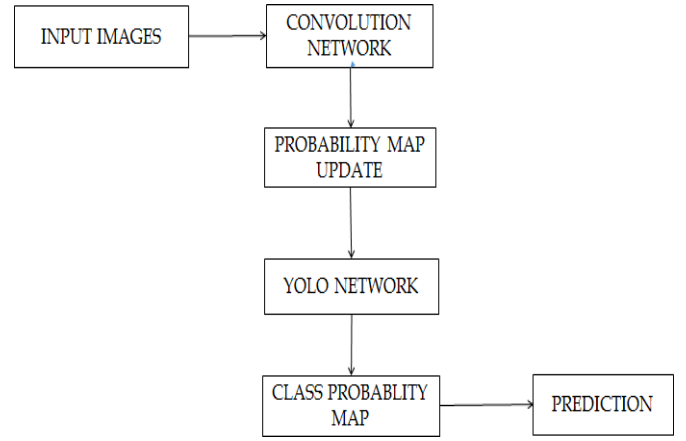


Fig. 3. Proposed Model Flow

The modules in this ship detection framework realize the following steps:

Step 1: Data Collection, which collects the data from the SAR free open dataset (SFOD) named SAR Ship Dataset.

Step 2: Data pre-processing is a technique that transforms raw data into an understandable format.

Step 3: Training the model which realizes image classification that is ten classes is categorized, each level takes five images.

Step 4: Object detection which achieves a hybrid YOLO algorithm, which uses YOLO-VGG16 for object detection added with Spatial Based Pixel Descriptor (SEPD) and K-Means Clustering.

Step 5: Optimizer Analysis is then used for better screening and training of images.

Activation function determines the output for a deep learning network; whether it is activated or not, the input is relevant for the prediction of the model. The output of each neuron has the finite value ranges between 0 to 1 or -1 to 1. It has its computational efficiency. In our case, LeakyRelu and Softmax are used as activation functions. LeakyRelu activation function is a hyperparameter, whose value is set before the learning process begins. Instead of being zero, leaky Relu has a slight non-zero limits. Softmax activation function calculates the probability dissemination of each objective class over all the probable objective modules, then finally, the probabilities calculated will be helpful in determining the objective modules for the given inputs.

Optimizer helps in minimizing or maximizing the error function. It trains the model effectively and efficiently. The optimization algorithm produces accurate results. In our simulation, RMSprop and Cross entropy Loss are realized as optimization functions. RMSprop optimizer is used for increasing the learning rate in the ship detection framework, and this optimizer algorithm takes larger steps in the horizontal direction, converging faster learning rate and efficiency. Cross-Entropy loss function is utilized for

estimating the presentation of problems in the classification of ship. Cross-entropy loss increments as the anticipated probability, where the output of the probability, values between 0 and 1.

V. IMPLEMENTATION AND RESULTS

Keras framework is used for the ship detection model is experimented using deep learning techniques. The pre-training model used in this Hybrid YOLO – VGG16 to prepare the system of the network. 700 images under 7 classes with every 10 images is trained with 25k iterations and the learning rate for this ship detection framework is 0.0001. Optimizer Analysis which is for better screening and training of images. The optimization algorithm used for this ship detection framework is RMSProp, and the activation function used is Leaky ReLu and Softmax functions.

Figure.4 and figure.5 shows the test results of our proposed framework, we can see that our hybrid YOLO achieves higher accuracy with low epoch and low function loss with high epoch.

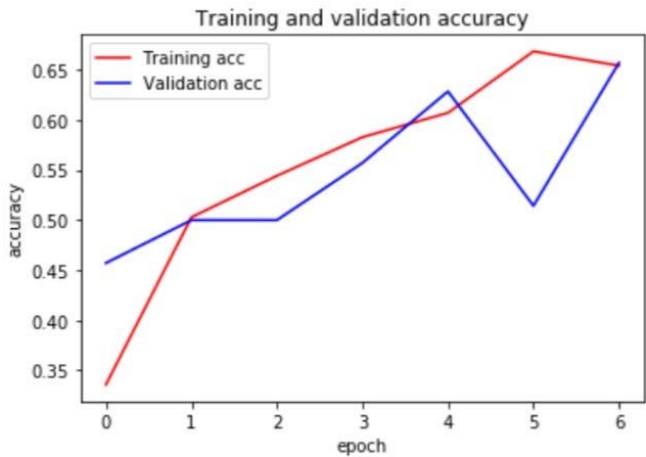


Fig. 4. Accuracy

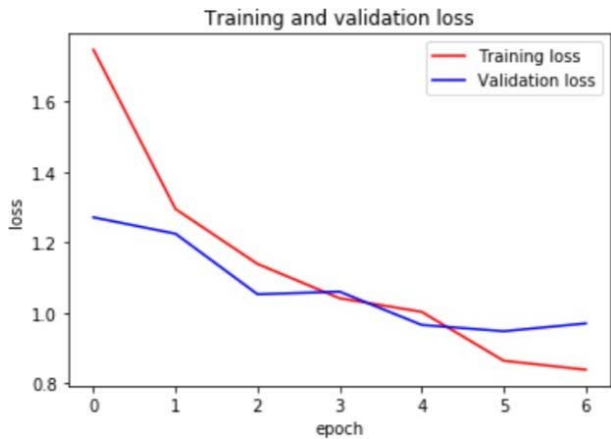


Fig. 5. Loss function

The performance of the proposed framework is evaluated with ship based SAR images for the performance metrics, namely precision, recall, f1-score, and support.

Table I summarizes the investigational outcomes for figure 5 in the differentiation of ships from coastal and sea areas.

TABLE I. PERFORMANCE ANALYSIS

	Precision	Recall	F1-score	Support
Coast	0.75	0.60	0.67	10
Coast-ship	1.00	0.20	0.33	10
Detail	1.00	0.80	0.89	10
Land	0.89	0.80	0.84	10
Multi	0.00	0.00	0.00	10
Ship	0.00	0.00	0.00	10
Water	0.80	0.40	0.53	10
Micro Avg	0.85	0.40	0.54	70
Macro Avg	0.63	0.40	0.47	70
Weighted Avg	0.63	0.40	0.47	70
Samples Avg	0.40	0.40	0.40	70



Fig. 6. Sample Input Image

The above figure.6 is fed to the proposed hybrid YOLO framework for landscape differentiation. The learning ratio of the proposed framework for figure 6 is described below, and the image is classified under the label: Cruise for ship classification.



Fig. 7. Classified Ship Image

ID: 0, Label: coast 2.69%
ID: 1, Label: coast-ship 4.65%
ID: 2, Label: detail 8.57%

ID: 3, Label: land 3.59%
ID: 4, Label: multi 11.49%
ID: 5, Label: ship 36.09%
ID: 6, Label: cruise

From the above results in the figure.7, we achieved the exactness of the proposed environment increases densely, if number of images is embedded at the training phase.

VI. COMPARISON AND REVIEW

The proposed model is compared with existing frameworks for ship detection such as Faster RNN, CNN, and RetinaNet. Table II depicts the performance analysis of the ship detection using ShipSAR dataset, which was performed using various existing algorithms and the proposed Hybrid YOLO-VGG16 model.

TABLE II. COMPARISON ANALYSIS

Models	Accuracy	Precision	Recall	F1-score
CNN	0.57	0.70	0.50	0.67
RNN	0.62	0.73	0.60	0.33
Faster-RNN	0.65	0.79	0.70	0.69
RetinaNet	0.67	0.85	0.80	0.64
Hybrid YOLO-VGG16	0.72	0.89	0.80	0.80

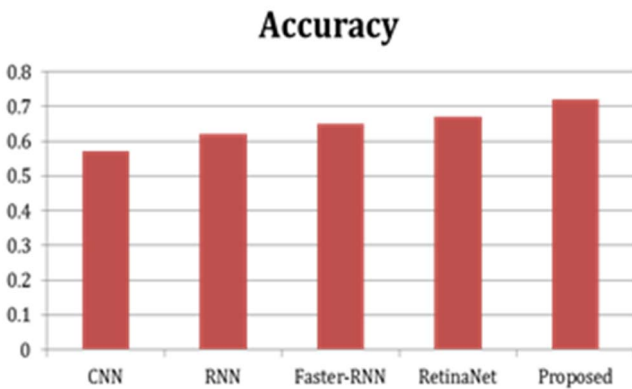


Fig. 8. Accuracy Comparison

Figure.8 inscribes the accuracy performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.72 which is better than other existing frameworks.

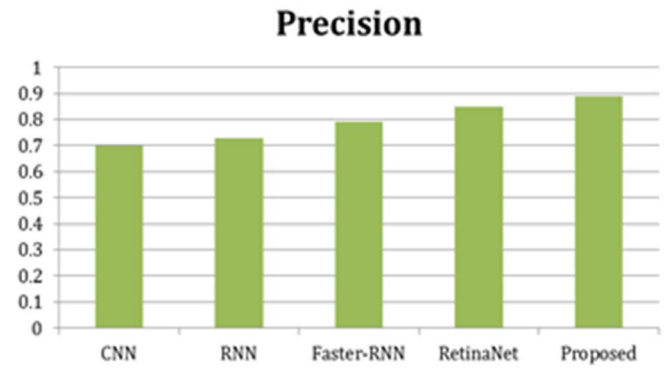


Fig. 9. Precision Comparison

Figure.9 engraves the Precision performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.89 which outperforms all other existing frameworks.

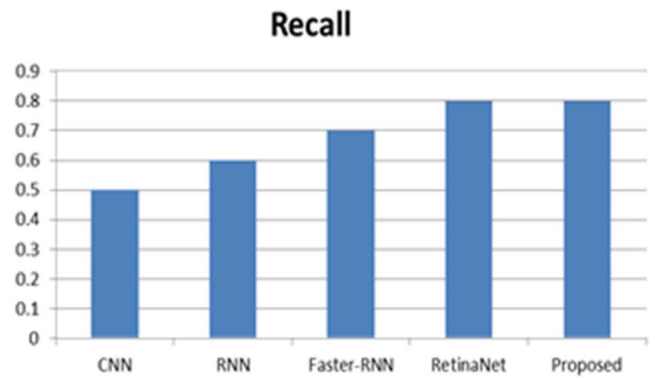


Fig. 10. Recall Comparison

Figure.10 incises the recall performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.80 which beats up the other existing frameworks.

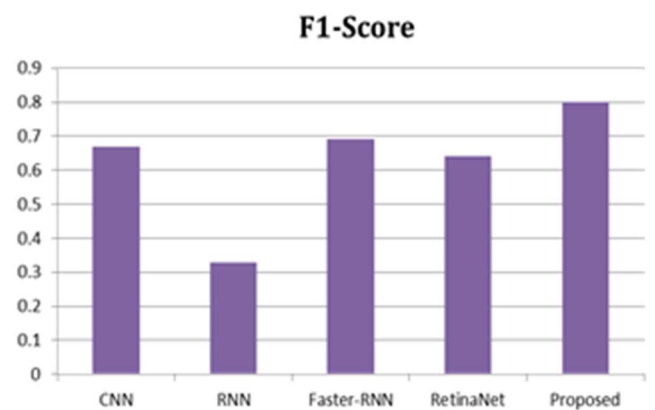


Fig. 11. F1-Score Comparison

Figure.11 etches the F1-Score performance of proposed hybrid YOLO-VGG16 compared with existing algorithms. It proves that the proposed model achieves an accuracy of 0.80 which top performs than other existing frameworks.

From the above performance analysis, we find that our proposed hybrid YOLO-VGG16 outperforms all other efficient existing ship detection algorithms in terms of accuracy, precision, recall and F1-score.

VII. CONCLUSION AND FUTURE WORK

In this paper, we discussed of how SAR images can be useful for Ship Detection model and how it can be used for examining the oceanic activities. Despite the advancement in Ship Detection using various techniques and algorithms, they still face some issues in accuracy and computational efficiency. Our proposed framework is Hybrid YOLO, which is the combination of YOLO-VGG16, SEPD, and K-Means Clustering. The parameters determined in this ship detection framework is optimization algorithm, activation function, loss function, and hidden layer. This Hybrid YOLO gives more accuracy and computational efficiency than other existing models because the ships detected from the SAR images are categorized into ten classes, and five images per class are trained for this framework. Thereby detects the ship from the images, and clusters the model of the image with the same classes, prove its high accuracy and top computational efficiency.

REFERENCES

- [1] Miguel M.Pinto, Renata Libonati, Ricardo M.Trigo, Isabel F.Trigo, Carlos C.DaCamara, A deep learning approach for mapping and dating burned areas using temporal sequences of satellite images, ISPRS Journal of Photogrammetry and Remote Sensing Volume 160, February 2020, Pages 260-274.
- [2] Nina W., Condori W., Machaca V., Villegas J., Castro E. (2020) Small Ship Detection on Optical Satellite Imagery with YOLO and YOLT. In: Arai K., Kapoor S., Bhatia R. (eds) *Advances in Information and Communication. FICC 2020. Advances in Intelligent Systems and Computing*, vol 1130. Springer, Cham.
- [3] Caglar Gulcehre, Deep Learning, 2015, <http://deeplearning.net/>.
- [4] Yuanyuan Wang, Chao Wang, Hong Zhang, Yingbo Dong, and Sisi Wei "A SAR Dataset of Ship Detection for Deep Learning under Complex Backgrounds" Remote Sensing 2019.
- [5] Krizhevsky A, Sutskever I, Hinton G E "ImageNet classification with deep convolutional neural networks" International Conference on Neural Information Processing Systems, 2012.
- [6] Changchong Lu and Weihai Li "Ship Classification in High-Resolution SAR Images via Transfer Learning with Small Training Dataset" mdp, 2018.
- [7] Shaoming Zhang, Ruize Wu, Kunyuan Xu, Jianmei Wang, Weiwei Sun "R-CNN-Based Ship Detection from High Resolution Remote Sensing Imagery" Remote Sensing, 2019.
- [8] Yang-Lang Chang, Amare Anagaw, Lena Chang, Yi Chun Wang, Chih-Yu Hsiao, and Wei-Hong Lee "Ship Detection Based on YOLOv2 for SAR Imagery" Remote Sensing 2019.
- [9] Michal Segal-Rozenhaimer, Alan Li, Kamalika Das, Ved Chirayath, Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (CNN), Remote Sensing of Environment, Volume 237, February 2020.
- [10] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T "Convolutional architecture for fast feature embedding" 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
- [11] MASATI dataset (v2) - MARitime SATellite Imagery dataset, <https://www.iuii.ua.es/datasets/masati/>
- [12] Ananconda Tool available at <https://www.anaconda.com/distribution/>
- [13] Hongwei Zhao, Weishan Zhang, Haoyun Sun, Bing Xue "Embedded Deep Learning for Ship Detection and Recognition" future internet, 2019.
- [14] Shunjun Wei, Hao Su, Jing Ming, Chen Wang, Min Yan, Durga Kumar, Jun Shi, Xiaoling Zhang "Precise and Robust Ship Detection for High-Resolution SAR Imagery Based on HR-SDNet" Remote Sensing, 2019.
- [15] Xue Yang, Hao Sun, Kun Fu1, Jirui Yang, Xian Sun, Menglong Yan, Zhi Guo "Automatic Ship Detection of Remote Sensing Images from Google Earth in Complex Scenes Based on Multi-Scale Rotation Dense Feature Pyramid Networks.



MANAKULA VINAYAGAR INSTITUTE OF TECHNOLOGY

(Approved by AICTE , Affiliated to Pondicherry University and Accredited by NBA)

Kalitheerthalkuppam, Puducherry-605 107




*3rd IEEE-International Conference on
Systems, Computation, Automation and Networking*

ICSCAN-2020

Certificate of Presentation

This is to certify that S. DEVADHARSHINI of IFET has
presented a paper on PERFORMANCE INVESTIGATION OF HYBRID YOLO-VGG16 BASED SHIP DETECTION FRAMEWORK USING SAR IMAGES
in the 3rd IEEE - International Conference on Systems, Computation, Automation and Networking (Virtual Conference),
organized by Manakula Vinayagar Institute of Technology on 3rd and 4th July 2020.


Dr. R. Valli
Conference Chair
ICSCAN 2020


Dr. S. Malarkkan
Principal


Shri. M. Dhanasekaran
Chairman & Managing Director



MANAKULA VINAYAGAR INSTITUTE OF TECHNOLOGY

(Approved by AICTE , Affiliated to Pondicherry University and Accredited by NBA)

Kalitheerthalkuppam, Puducherry-605 107



*3rd IEEE-International Conference on
Systems, Computation, Automation and Networking*

ICSCAN-2020

Certificate of Presentation

This is to certify that R. RASMOHAN of TAET has
presented a paper on PERFORMANCE INVESTIGATION OF HYBRID YOLO-VGG16 BASED SHIP DETECTION FRAMEWORK USING SAR IMAGES
in the 3rd IEEE - International Conference on Systems, Computation, Automation and Networking (Virtual Conference),
organized by Manakula Vinayagar Institute of Technology on 3rd and 4th July 2020.


Dr. R. Valli
Conference Chair
ICSCAN 2020


Dr. S. Malarkkan
Principal


Shri. M. Dhanasekaran
Chairman & Managing Director