

Advanced CNN-Based Ship Detection from SAR Images Using FUSAR Dataset

A PROJECT REPORT

Submitted by

Kosaraju Ramya

ABSTRACT

The detection of ships at multiple scales in large offshore synthetic aperture radar (SAR) images is crucial for both civilian and military purposes, including maritime management and wartime reconnaissance. By utilizing deep learning techniques, a deep neural network can extract multiscale information from SAR images, leading to enhanced detection capabilities. Advanced solutions are needed to address challenges in SAR imagery like speckle noise, varying illumination conditions, and complex sea clutter to ensure reliable ship detection. Deep learning techniques have emerged as a powerful tool for creating deep neural networks that extract multiscale information from SAR images, leading to significant improvements in ship detection capabilities.

The FUSAR-Ship dataset, which utilizes GF-3 SAR technology, has been created by processing 126 different scenes featuring diverse scenarios such as sea, land, coast, river, and islands. This dataset comprises over 5000 ship chips that contain AIS messages, along with examples of various types of clutter such as strong scatterer, bridge, coastal land, islands, sea, and land. FUSAR-Ship has been developed as a publicly available benchmark dataset for the purpose of detecting and recognizing ships and other marine targets.

Modern deep learning models like YOLOv7, YOLOv8, Advanced Convolutional Neural Network, and the most recent ship bounding box algorithms were used in our exhaustive evaluations. The robustness and flexibility of these models are demonstrated by extensive testing on the FUSAR dataset, which covers a wide range of ship examples, scales, and complex marine landscapes. According to the results, ship detection in SAR imagery has the potential to be revolutionized by powerful deep learning algorithms.

TABLE OF CONTENTS

	Page No.
ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF FIGURES	iv
CHAPTER 1- INTRODUCTION	1
CHAPTER 2 – LITERATURE REVIEW	3
CHAPTER 3 – SYSTEM SPECIFICATIONS	5
CHAPTER 4 – SYSTEM DESIGN	7
4.1 High Level Design with Description	7
4.2 Low Level Design with Description	8
CHAPTER 5 – SYSTEM IMPLEMENTATION	10
5.1 DATASET DESCRIPTION	10
5.2 MACHINE LEARNING MODELS USED	11
5.3 IMPLEMENTATION	14
CHAPTER 6 – SYSTEM TESTING	16
6.1 INTRODUCTION	16
6.2 TESTING ENVIRONMENT	16
6.3 TESTING DATA	16
6.4 EVALUATION METRICS	16
6.5 TESTING PROCEDURE	17
6.6 TESTING RESULTS	18
CHAPTER 7 – RESULTS AND ANALYSIS	20
CHAPTER 8 – CONCLUSION AND FUTURE SCOPE	21
REFERENCES	22

LIST OF FIGURES

FIGURE	PAGE.NO
Fig 4.1 High Level Design	7
Fig 4.2 Low Level Design	8
Fig 5.1: 15 top level Categories of Dataset	10
Fig 5.2: Sample image of a dataset	11
Fig 5.3: Network architecture of Yolov5	12
Fig 5.4: Pretrained Checkpoints of YOLOv5	12
Fig 5.5: Network architecture of Yolov7	13
Fig 5.6: C3TR Structure diagram	14
Fig 5.7: Cloning the Yolov5 requirements from GitHub	14
Fig 6.1: Results of Yolov5	18
Fig 6.2: Results of modified Yolov5	18
Fig 6.3: Results of Yolov7	18
Fig 6.4: Bounding Box labeling	19

CHAPTER - 1

INTRODUCTION

One can't emphasize how crucial ship detection systems that use Synthetic Aperture Radar (SAR) images are to a number of vital applications. SAR-based ship detection plays a critical role in maritime management by tracking vessel movements, averting collisions, and guaranteeing safe navigation. SAR technology makes it possible to identify and track maritime activity, which is a crucial military weapon for reconnaissance during times of conflict. SAR also makes it possible to monitor the environmental effects of maritime operations in great detail, which improves regulation and helps to mitigate ecological hazards. A microwave signal is sent from a sensor platform to the ground, and an active system that detects the reflected signal creates SAR images. SAR offers finer spatial resolution than traditional stationary beam-scanning radars by utilizing the motion of the radar antenna over a target region. Usually installed on a moving platform, such as an aircraft or spacecraft, it is derived from a sophisticated type of side-looking aerial radar (SLAR).

Because of its exceptional capacity to image continuously in all weather circumstances, synthetic aperture radar (SAR) is one of the most potent instruments for Earth observation. SAR is becoming more and more crucial to ocean surveillance activities like ship identification, fishery management, and oil spill detection. Significant advancements in the identification of ships using low- and medium-resolution SAR imagery have been made in recent decades. With the recent successful launch of a few high-resolution SAR satellites, we can now classify ship targets into different groups, such as cargo, tanker, tug, and fisher. The Gaofen-3 (GF-3) satellite is the first C-band high-resolution quad-pol SAR spacecraft launched by China with a specific objective to monitor the ocean.

Using a comprehensive strategy, the suggested ship recognition system combines cutting-edge deep learning methods with the strength of two well-known frameworks, YOLOv7 and YOLOv5. These frameworks are selected based on how well they perform object detection tasks, which lays the groundwork for optimizing ship detection in SAR pictures. The FUSAR dataset is used extensively in the research to train the models. During this training phase, issues like speckle noise, changes in lighting, and the complexity caused by marine clutter are addressed.

YOLO is a single-shot detector that processes an image using a fully convolutional neural network (CNN). To forecast an object's presence and location in an image, single-shot object detection does one pass over the input image. They are computationally efficient since they can process an entire image in a single pass. An end-to-end neural network that predicts bounding boxes and class probabilities simultaneously is the method that You Only Look Once (YOLO) suggests using. YOLO makes all of its predictions with the aid of a single fully connected layer, in contrast to methods like Faster RCNN, which operate by first identifying potential regions of interest using the Region Proposal Network and then carrying out recognition on those regions independently. While YOLO only needs one iteration, methods that employ Region Proposal Networks require numerous iterations for the same image.

The more intricate EfficientDet architecture (shown below) is used by YOLO v5, and it is based on the Efficient Net network architecture. YOLO v5 can achieve better accuracy and broader item category coverage by employing a more sophisticated architecture. D5, a more extensive and varied dataset with 600 object categories in total, was used to train YOLO v5. YOLO v5 creates the anchor boxes using a brand-new technique known as "dynamic anchor boxes." To enhance the model's performance on unbalanced datasets, Yolo V5 employs a CIOU loss function.

YOLO V7 uses Extended Efficient Layer Aggregation Network. Because it makes use of nine anchor boxes, YOLO v7 is able to recognize a greater variety of item sizes and shapes. YOLO v7 has made significant progress with the addition of a new loss function known as "focal loss." Additionally, YOLO v7 boasts a greater resolution than earlier iterations. With a pixel resolution of 608 by 608, YOLO v7 is able to process images with more clarity and overall accuracy, making it capable of detecting smaller things.

For now, we are going to train our FUSAR Ship dataset on YOLO V5, YOLO V7 and a modified YOLO V5. In the modified YOLO V5 we are going to add attention modules like C3TR. Then comparing the models for better understanding.

CHAPTER - 2

LITERATURE REVIEW

The capacity of synthetic aperture radar (SAR) to provide all-weather, real-time imagery makes it one of the most effective Earth observation (EO) instruments [1]. Using SAR to detect oil spills, manage fisheries, and identify ships is becoming more and more crucial for ocean monitoring. A lot of progress has been achieved in the last few years in the identification of ships from low- and medium-resolution SAR imagery [1]. The first civil C-band high-resolution SAR satellite launched by China, the Gaofen-3 (GF-3) satellite is dedicated to ocean remote sensing. Up to 1 m is the nominal maximum resolution of GF-3 data. Applications like airplane detection and ship recognition [10] have made extensive use of GF-3 data. Executing the process on a total of 126 GF-3 scenes that span a wide range of sea, land, coast, river, and island scenarios results in the creation of the FUSAR Ship high-resolution GF-3 SAR dataset. Within it are samples of powerful bridges, coastal land, islands, and sea and land clutter, in addition to over five thousand ship chips with AIS communications.

The authors in [3] have employed Multi-CFAR model to find the ships in the images. Multi-CFAR model includes global-CFAR, ship-CFAR, large-CFAR methods in it. Where global-CFAR method is used to find the higher-level target pixels of the ship. Then large-CFAR method is employed to reduce the number of target pixels identified in global-CFAR. Then ship-CFAR method is employed to approximately detect the pixels which will correspond to the potential ship pixels. Then finally they have used a novel CNN based technique to further identify the correct pixels corresponding to the ship. They have also used OTSU based sea-land segmentation method to correctly discriminate between the sea and land. Since, the FUSAR Ship dataset is a black and white colored dataset instead of RGB colored. The future work involves manually correcting the wrong ship types. For example a ship type of cargo is labelled as fishing ship. We have to manually correct these mistakes.

We are working on images, which are of large in size. We need much computation power to process all these images in real time environment. So, instead of using Constant False Alarm Rate (CFAR) and normal CNN models, the authors in [2] have suggested a model Threshold Neural Networks(TNN). In Threshold Neural Network (TNN) instead of processing whole image at a time, we can process a small part within the image. For that a sliding window is used to optimally detect the target ship pixels. Then an False Alarm Suppression Network(FSN) [] is used to further discriminate the target pixels and to accurately detect the ship. The computation power can be reduced

since we are not using the whole image at a time to process. So, TNN model with FSN can be used to reduce the burden on the system.

The dataset we use is not a balanced dataset. It is an imbalanced dataset. Where the class 'Cargo' has 1300 images and the class 'Dredger' has only 120 images. So, the authors of [5] has proposed a minority oversampling method called Clustering-Based Size-Adaptive Safer Oversampling Technique (C-SASO). The C-SASO method has three steps in it. 1). Semi unsupervised clustering, 2). Subclass sizing and 3). synthetic instance generation in safer regions. First, they divided the dataset into subclasses and have allotted certain weights to each subclass to make the dataset somewhat balanced. This method proved to be an important method, since the accuracy has been increased by 4% corresponding to the previous methods.

Ship identification techniques for SAR pictures were primarily semi-automated and based on conventional object detection algorithms. Constant False Alarm Rate (CFAR) algorithm was proposed as traditional method. A number of issues plague these conventional techniques, such as detection accuracy and takes more time to process. Most of the recent object detection algorithms are of two stage and one stage methods. One stage methods are faster than compared to two stage methods like Faster-RCNN, Fast-RCNN and also require less computation power. Paper [4] introduces a one stage detection model called YOLO. YOLO has many versions and they have employed YOLO V5 to detect the ships. Also to increase the accuracy further they have used convolutional block attention module (CBAM) in YOLO V5 layers. Attention modules are used to find the long range dependencies in the image. The improved YOLO V5 gave better mAP score compared to other YOLO versions. The future work involves using a lightweight backbone in the YOLO to increase the speed and accuracy further.

CHAPTER – 3

SYSTEM SPECIFICATIONS

3.1 Software requirements

Google Colab: Google Colab was chosen because of its stable cloud-based execution environment and availability of GPU resources, which are essential for speeding up deep learning activities. Its interoperability with Python and related libraries improves the programming experience overall, and its smooth interaction with Google Drive facilitates effective data storage and retrieval.

Jupyter Notebook: Jupyter Notebook is used as an interactive computing environment that makes it easier to write and run project code in chunks. Iterative testing, process debugging, and real-time output monitoring are made possible by this functionality.

Operating System: Windows

Python Libraries:

- OS
- glob
- PIL
- IPython display
- ultralytics
- TensorFlow
- torch (PyTorch)
- YOLO (You Only Look Once)
- git
- Pillow

3.2 Hardware requirements

CPU: It is advised to use a contemporary multi-core processor, such as an AMD Ryzen 5/7/9 or an Intel i5/i7/i9. In the testing and development stages, its fast clock speed improves multitasking and data processing efficiency.

RAM: It's suggested to have at least 16GB of RAM. This guarantees that the system can manage big datasets in memory, allowing several processes to run simultaneously during the phases of language recognition and model training.

CHAPTER - 4

SYSTEM DESIGN

4.1 High Level Design with Description

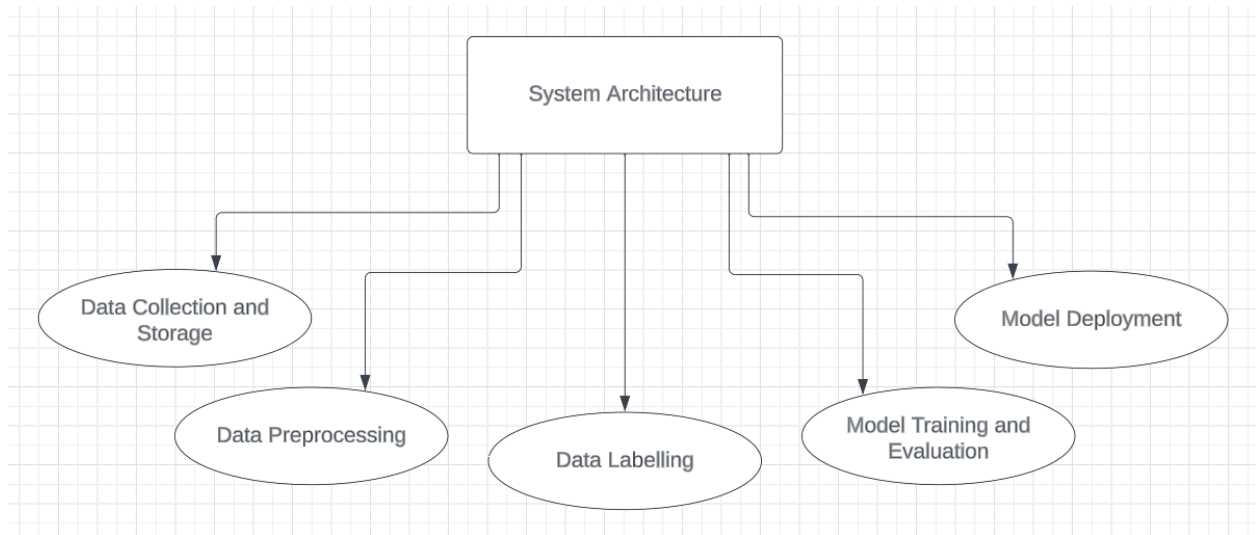


Fig. 4.1 High Level Design

Data Collection and Storage: Preparation of FUSAR Dataset.

Data Preprocessing: Cleaning and enhancing the data to make it ready for training. Like reshaping the images.

Data Labelling: Creating the labels for the images to train them on YOLO models.

Model Training and Evaluation: Referring to Fig. 4.1. Training the data on various Neural Network models. Like Fast RCNN, YOLO V5, YOLO V7, etc. Also evaluating the model based on evaluation metrics such as accuracy, IoU, etc.

Model Deployment: Deploying the model and creating a user interface.

4.2 Low Level Design with Description

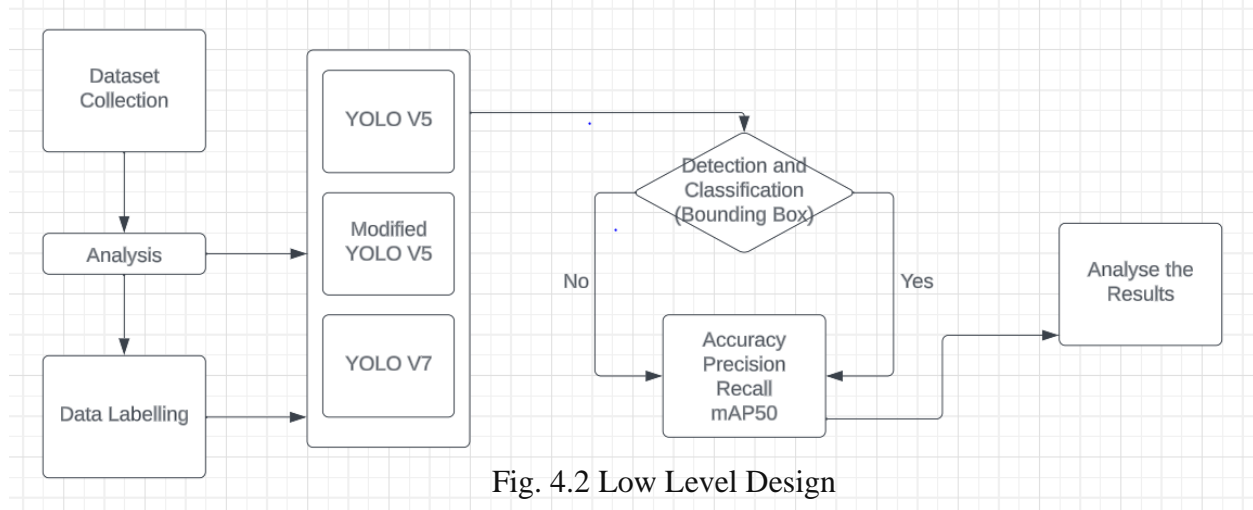


Fig. 4.2 Low Level Design

Dataset Collection: Choosing a best suitable dataset for ship detection model. We have choose FUSAR Ship Detection Dataset. FUSAR Ship Detection dataset has been created using an automated SAR-AIS matching process on more than 100 GF-3 scenes that depict a diverse range of sea, land, coast, river, and island scenarios. There are over 5000 images in the dataset.

Analysis: Analysing the dataset to know the limitations of the dataset such as presence of speckle noise, sea ripples, sea-land variation, etc.

Data Preprocessing: The images are in the size of 512 x 512 in FUSAR-Ship dataset. But the YOLO works on 224 x 224 sized images. So image size reduction has to be made.

Data Labelling: The dataset is not present in the format of YOLO version. So, we have to manually label each image in the dataset. Labelling in the sense, creating a bounding box of the ships in the images to get the coordinates of the ships. Also, we will specify the type of the ship. There are 7 types namely Cargo, Dive Vessel, Dredger, Fishing ship, High Speed Craft, Law Enforcer, other.

Models:

YOLO V5: The dataset is trained over the YOLO V5 pre-trained model to accurately draw a bounding box and classify the ships based on the type of ships. YOLO V5 uses EfficientDet architecture as its backbone and also it uses CIoU as its loss function.

YOLO V7: The dataset is trained over YOLO V7 pre-trained model to accurately draw the bounding box and classify the ships based on the type of ships. YOLO V7 uses Extended Efficient Layer Aggregation Network as its backbone and it uses focal loss as its loss function. Also, YOLO V7 works on a better resolution image to detect the objects clearly.

Modified YOLO V5:

Model Training: The dataset is trained over the above-mentioned YOLO versions and analyzed to pick the best version. Referring to Fig. 4.2 After training the model will generate a bounding box for the images specifying the ships and its type.

Evaluation: Evaluation metrics used are precession, accuracy, recall and mAP50. Yolo versions will automatically generate these evaluation metrics.

Analysis: Finally analyzing the results with existing models.

CHAPTER – 5

SYSTEM IMPLEMENTATION

This chapter focuses on implementing a strong ship detection algorithm in the FUSAR dataset of Synthetic Aperture Radar (SAR) images. The algorithm utilizes various YOLO (You Only Look Once) models, including YOLO V5, V7, and a modified version of YOLOV5. The initial step involves integrating the YOLO V5 model as a foundation, followed by improvements with the inclusion of YOLO V7 and a customized iteration of YOLOV5. This iterative approach aims to enhance ship detection performance in the FUSAR dataset by leveraging the strengths of these different YOLO variants.

5.1 Dataset Description

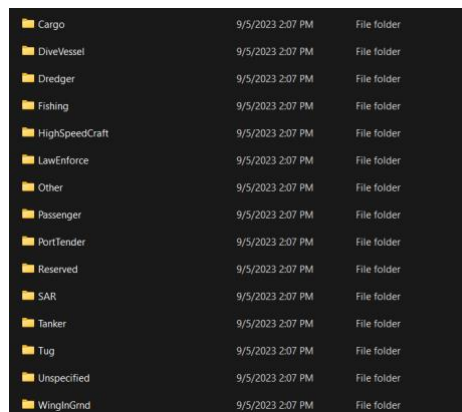
DATASET: FUSAR-Ship

Number of Image: Over 5000

Image Size: 512 X 512

The FUSAR-Ship dataset has been created using an automated SAR-AIS matching process on more than 100 GF-3 scenes that depict a diverse range of sea, land, coast, river, and island scenarios.

Features: Cargo, Dive Vessel, Dredger, Fishing, Highspeed Craft, Law Enforce, Other, Passenger, Port Tender, Reserved, SAR, Tanker, Tug, Unspecified, WingInGrnd. We will be utilizing transfer learning models such as Yolo to identify various types of ships. Although there are subcategories within these features, our current focus is on a broader level.



Cargo	9/5/2023 2:07 PM	File folder
DiveVessel	9/5/2023 2:07 PM	File folder
Dredger	9/5/2023 2:07 PM	File folder
Fishing	9/5/2023 2:07 PM	File folder
HighSpeedCraft	9/5/2023 2:07 PM	File folder
LawEnforce	9/5/2023 2:07 PM	File folder
Other	9/5/2023 2:07 PM	File folder
Passenger	9/5/2023 2:07 PM	File folder
PortTender	9/5/2023 2:07 PM	File folder
Reserved	9/5/2023 2:07 PM	File folder
SAR	9/5/2023 2:07 PM	File folder
Tanker	9/5/2023 2:07 PM	File folder
Tug	9/5/2023 2:07 PM	File folder
Unspecified	9/5/2023 2:07 PM	File folder
WingInGrnd	9/5/2023 2:07 PM	File folder

Fig 5.1: 15 top level Categories of Dataset

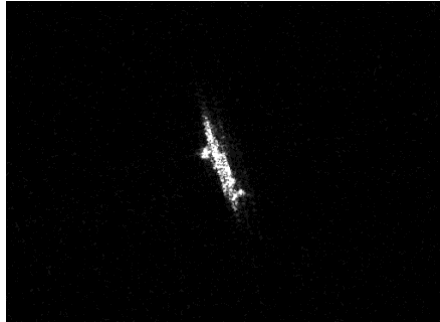


Fig 5.2: Sample image of a dataset

5.2 Machine Learning models used:

1. YOLOv5:

The architecture of YOLOv5 has been simplified and streamlined, making it easier for users to understand and use. It is built on the PyTorch framework, which offers flexibility and seamless integration into different machine learning workflows. YOLOv5 comes in various model sizes, allowing users to balance between speed and accuracy based on their specific needs. It retains the real-time object detection capability of YOLO, providing bounding box coordinates and class predictions for multiple objects in an image. Overall, YOLOv5 is designed to be user-friendly, with simple usage and deployment, making it suitable for a wide range of applications.

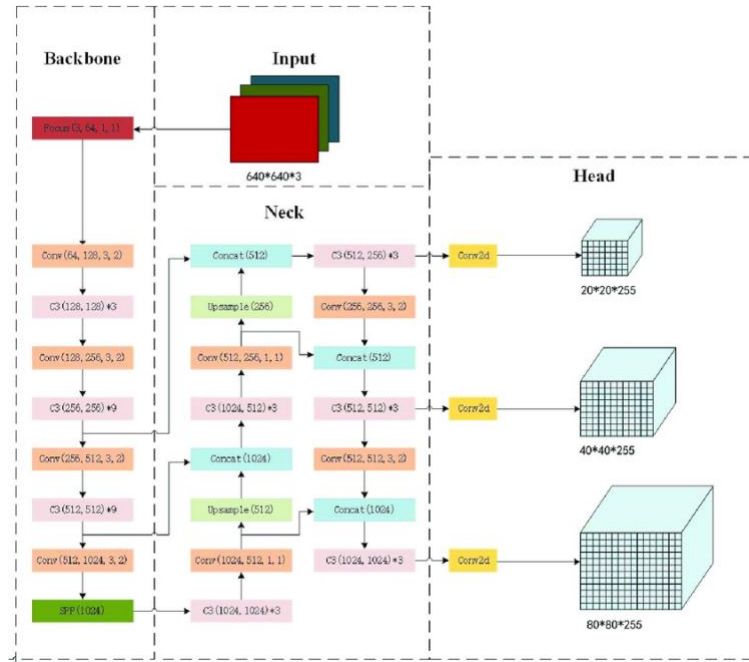


Fig 5.3: Network architecture of YOLOv5

Model	size (pixels)	mAP ^{val} 50-95	mAP ^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6	1280	55.0	72.7	3136	26.2	19.4	140.7	209.8
+ TTA	1536	55.8	72.7	-	-	-	-	-

Fig 5.4: Pretrained Checkpoints of YOLOv5

The models provided above are utilized to train our model and facilitate the identification of images, as well as the creation of bounding boxes. Currently we are working with YOLOv5s.

2. YOLOv7:

YOLOv7 comes in various model sizes, ranging from small to extra-large, to cater to different needs for speed and accuracy. Users can select the appropriate model variant based on their specific requirements. YOLOv7 maintains the real-time object detection capability that YOLO is known for, providing bounding box coordinates and class predictions for multiple objects within an image. This makes it suitable for applications like traffic surveillance and autonomous vehicles. YOLOv7 is designed to be user-friendly, with easy usage and deployment, making it accessible for a wide range of applications. The implementation is well-documented and easy to understand, allowing users to seamlessly integrate the model into their projects. YOLOv7 achieves impressive accuracy, surpassing its predecessor YOLOv5 and other state-of-the-art object detection algorithms. This makes it a reliable choice for tasks that require high precision in object detection. Additionally, YOLOv7 delivers faster inference speeds compared to YOLOv5, making it suitable for real-time or near-real-time object detection applications. It utilizes the PyTorch deep learning framework, which offers flexibility and easy integration into machine learning workflows. The popularity and extensive support of PyTorch make it a convenient choice for developers.

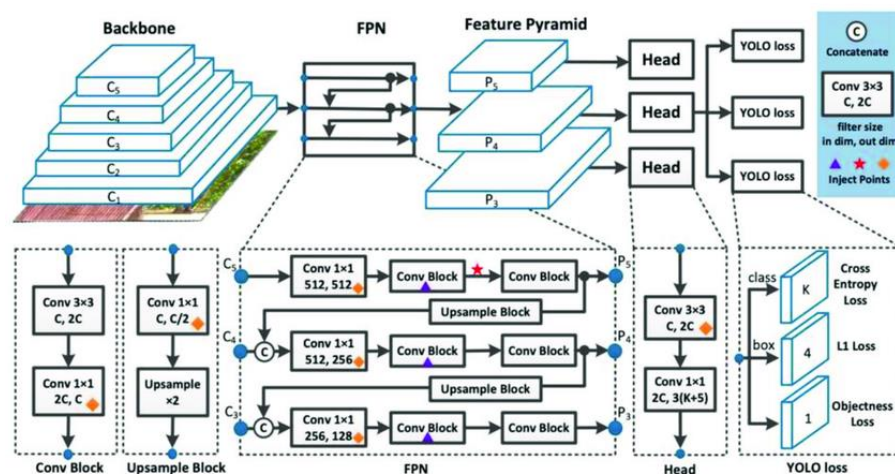


Fig 5.5: Network architecture of YOLOv7

3. Modified YOLOv5:

In the modified yolov5 we modified the backbone of the yolov5s model. We added C3TR attention layer before skip layers so that it had increased accuracy of model when compared to normal Yolov5. The input image is divided into image blocks of a specific size using the Patch Embedding operation by the C3TR module. These divided image blocks are then combined into a sequence and passed to the Transformer Encoder for feature extraction.

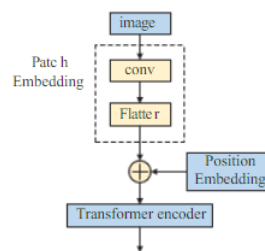


Fig 5.6: C3TR Structure diagram

5.3 Implementation:

- To begin, ensure that all required dependencies such as Python, PyTorch, and other necessary libraries are installed. This can be done by executing the following command in the terminal: "pip install torch torchvision". Next, obtain the YOLOv5 repository by cloning it from either the official GitHub repository or a trusted fork.

```
# Download YOLOv5 repository and install requirements
!git clone https://github.com/ultralytics/yolov5.git
%cd yolov5
!pip install -r requirements.txt
```

Fig 5.7: Cloning the Yolov5 requirements from GitHub

- Data Preparation: Arrange the dataset by adding annotated images and labels in a format compatible with YOLO. Then, create a YAML file that outlines the dataset configuration, including the paths to the training and validation data.

- Revise the training configuration by modifying parameters like batch size, learning rate, and the number of classes in the training configuration file (yolov5/models/yolov5s.yaml for YOLOv5s).

```
!python /content/yolov5/train.py --batch 16 --cfg  
/content/yolov5/models/yolov5s.yaml --epochs 100 --data  
/content/yolov5/data/custom.yaml --weights yolov5s.pt --device 0
```

- The detect.py script utilizes pre-trained weights (best.pt) for YOLOv5 object detection. It takes an input image (Ship_C01S02N0103.tiff) and identifies objects, displaying the detected results as the script runs

Run

```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --  
source /content/Fusar_Yolov7/images/validation/Ship_C01S02N0103.tiff
```

CHAPTER - 6

SYSTEM TESTING

6.1 Introduction

The following chapter presents the system testing strategy employed to measure the effectiveness of the recommended advanced ML models. The objective of the testing process is to assess the models' resilience, precision, and reliability by utilizing real-world data.

6.2 Testing Environment

The system testing was conducted on a computing platform with the following specifications:

- Operating System: iOS and Windows
- Programming Languages: Python
- Platform: Google Collab

6.3 Testing Data

The testing data consisted of 1500 records of images extracted from FUSAR-Ship dataset. The data was randomly divided into 80% for training and 20% for validation.

6.4 Evaluation Metrics

To evaluate the performance of the system, we employed the following metrics:

- Accuracy: The proportion of correct predictions made by the system.
- Precision: The proportion of positive predictions that are correct.

- Recall: The proportion of actual positive cases that are correctly identified by the system.
- F1-score: The harmonic mean of precision and recall.

6.5 Testing Procedure:

Data Partitioning:

Divide the dataset into training, validation, and test sets. Keep the test set separate until the final evaluation to ensure unbiased performance metrics.

Model Loading:

Load the pre-trained YOLOv5 model or the model that was trained during the training phase.

Validation Set Inference:

Run the model on the validation set and analyze the results. Evaluate the model's accuracy in detecting objects and its performance across different classes.

Performance Metrics:

Calculate standard object detection metrics like precision, recall, and F1 score. Utilize tools such as mAP (mean Average Precision) for a comprehensive evaluation.

Generalization Testing:

Assess the model's performance on images that were not seen during training or validation. This tests the model's ability to generalize to new and unseen data.

Error Analysis:

Analyze false positives and false negatives. Understand common failure modes and use this information to iteratively improve the model or fine-tune parameters.

Test Set Evaluation:

Finally, run the model on the designated test set that was not used during training or validation. Calculate and report the final performance metrics.

Documentation and Reporting:

Document the entire testing procedure, including dataset details, configurations, and results. Provide clear documentation on how to reproduce the testing process.

6.6 Testing Results

1. YOLOv5:

```
Fusing layers...
YOLOv5s summary: 157 layers, 7029004 parameters, 0 gradients, 15.8 GFLOPs
      Class      Images  Instances      P      R      mAP50      mAP50-95: 100% 5/5 [00:03<00:00, 1.32it/s]
      all         151       159      0.984    0.436    0.889      0.51
AggregatesCarrier    151         2        1        0    0.828    0.398
      BulkCarrier    151       157    0.967    0.873    0.95     0.622
Results saved to ../yolov5/runs/train/exp
```

Fig 6.1: Results of YOLOv5

2. ModifiedYOLOv5:

```
Fusing layers...
YOLOv5s summary: 193 layers, 8722572 parameters, 0 gradients, 17.7 GFLOPs
      Class      Images  Instances      P      R      mAP50      mAP50-95: 100% 5/5 [00:03<00:00, 1.40it/s]
      all         151       159      0.96     0.439    0.961    0.675
AggregatesCarrier    151         2        1        0    0.995    0.796
      BulkCarrier    151       157    0.919    0.879    0.926    0.555
Results saved to runs/train/exp2
```

Fig 6.2: Results of modified YOLOv5

3. YOLOv7:

```
      Class      Images  Labels      P      R      mAP@.5      mAP@.5:.95: 100% 5/5
      all         151     159    0.481    0.705    0.705    0.454
AggregatesCarrier    151         2    0.457    0.5     0.498    0.398
      BulkCarrier    151     157    0.504    0.911    0.913    0.51
100 epochs completed in 0.804 hours.
```

Fig 6.3: Results of YOLOv7

After training all evaluation metrics for 100 epochs, it was discovered that the Modified YOLOv5 model, which includes a C3TR attention module before the skip layers, outperformed the other models and produced the best results.

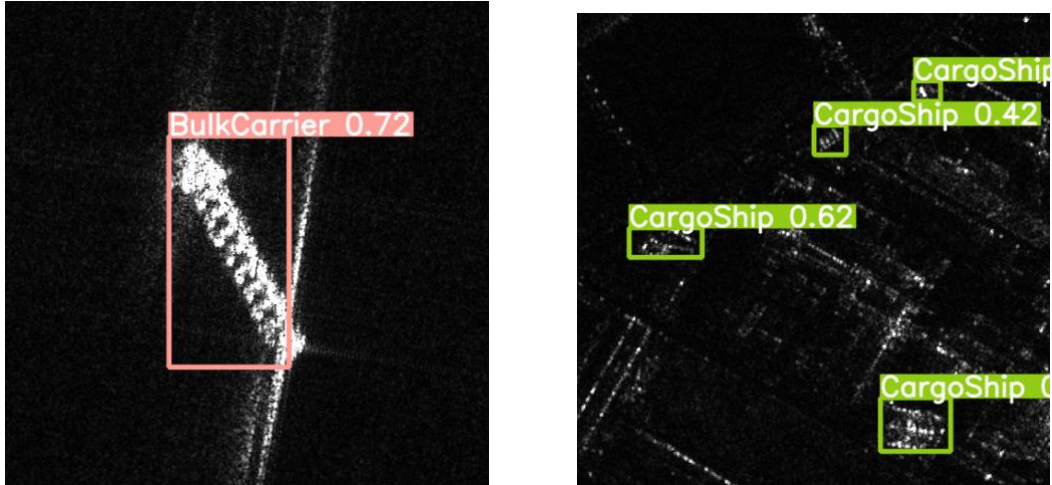


Fig 6.4: Bounding Box labeling

CHAPTER – 7

RESULTS AND ANALYSIS

In the evaluation of three object detection models, namely YOLOv5s, modified YOLOv5, and YOLOv7, we observe distinct performance characteristics. YOLOv5s and the modified YOLOv5 exhibit strikingly similar results, boasting high precision (0.96) and mAP (0.961), signifying accurate object localization and classification.

However, these models show a relatively lower recall (0.439), suggesting a potential trade-off between precision and recall. On the other hand, YOLOv7 demonstrates a different trade-off with a higher recall (0.705) but at the expense of precision (0.481) and mAP (0.705). Notably, all models display class-specific performance variations.

In conclusion, the choice between YOLOv5s, modified YOLOv5, and YOLOv7 hinges on the specific requirements of the application. YOLOv5s excels in tasks demanding high precision, while YOLOv7 may be preferred when capturing more instances is prioritized over precision. The modified YOLOv5, despite its name, mirrors YOLOv5s, indicating that the introduced modifications did not significantly impact its performance.

CHAPTER – 8

CONCLUSION AND FUTURE SCOPE

Conclusion:

In the assessment of three object detection models – YOLOv5s, modified YOLOv5, and YOLOv7 – a nuanced understanding of their strengths and trade-offs has been gained. YOLOv5s and the modified YOLOv5 demonstrate commendable precision and mAP, signifying accurate object recognition, albeit with a trade-off in recall. YOLOv7, on the other hand, prioritizes recall at the expense of precision and mAP. The choice between these models hinges on the specific requirements of an application, balancing precision and recall based on the use case.

Future Scope:

In the quest for further improvements, future research could explore the integration of different activation functions into these object detection models. Activation functions play a pivotal role in shaping the non-linearities within neural networks, and their impact on object detection performance is noteworthy. By experimenting with various activation functions, such as Rectified Linear Unit (ReLU), Leaky ReLU, and Parametric ReLU, researchers can potentially enhance the models' ability to capture intricate features and improve overall performance. This avenue of exploration could lead to models that exhibit more robust generalization across diverse datasets, catering to a broader range of real-world scenarios. Thus, future advancements in object detection could be achieved by refining the activation functions employed in these models, paving the way for more accurate and versatile applications in computer vision.

REFERENCES

- [1] Hou, X., Ao, W., Song, Q. *et al.* FUSAR-Ship: building a high-resolution SAR-AIS matchup dataset of Gaofen-3 for ship detection and recognition. *Sci. China Inf. Sci.* **63**, 140303 (2020)
- [2] J. Cui, H. Jia, H. Wang and F. Xu, "A Fast Threshold Neural Network for Ship Detection in Large-Scene SAR Images," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 6016-6032, 2022
- [3] X. Hou, W. Ao and F. Xu, "End-to-end Automatic Ship Detection and Recognition in High-resolution Gaofen-3 Spaceborne SAR Images," *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, Yokohama, Japan, 2019
- [4] W. Zhao, M. Syafrudin and N. L. Fitriyani, "CRAS-YOLO: A Novel Multi-Category Vessel Detection and Classification Model Based on YOLOv5s Algorithm," in *IEEE Access*, vol. 11, pp. 11463-11478, 2023
- [5] Y. Li, X. Lai, M. Wang and X. Zhang, "C-SASO: A Clustering-Based Size-Adaptive Safer Oversampling Technique for Imbalanced SAR Ship Classification," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1-12, 2022
- [6] Yan, Z.; Song, X.; Yang, L.; Wang, Y. Ship Classification in Synthetic Aperture Radar Images Based on Multiple Classifiers Ensemble Learning and Automatic Identification System Data Transfer Learning. *Remote Sens.* 2022, 14, 5288.

- [7] Guo, H.; Ren, L. A Marine Small-Targets Classification Algorithm Based on Improved Convolutional Neural Networks. *Remote Sens.* 2023, 15, 2917.
- [8] H. Zheng, Z. Hu, J. Liu, Y. Huang and M. Zheng, "Meta Boost: A Novel Heterogeneous DCNNs Ensemble Network With Two-Stage Filtration for SAR Ship Classification," in *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1-5, 2022, Art no. 4509005, Doi: 10.1109/LGRS.2022.3180793.
- [9] X. Shi, S. Fu, J. Chen, F. Wang and F. Xu, "Object-Level Semantic Segmentation on the High-Resolution Gaofen-3 FUSAR-Map Dataset," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 3107-3119, 2021, Doi: 10.1109/JSTARS.2021.3063797.
- [10] Wang Y, Wang C, Zhang H, et al. Automatic ship detection based on retina net using multi-resolution Gaofen-3imagery. *Remote Sens*, 2019, 11: 531

