Imperial College London

Satellite SAR ship detection using convolutional neural network

Department of Earth Science and Engineering, Faculty of Engineering, Imperial College London

Applied Computational Science and Engineering, ACSE 9

Sanaz Salati

University Supervisors: Olivier Dubrule, Lukas Mosser

Internship at CGG

Company Supervisor: Harry McCormack

August 2019

Github alias: sanazss, Email: sanaz.salati18@imperial.ac.uk

Contents

1 Abstract

Detecting ships from satellite images is a cost-effective method in providing valuable information for the energy and environmental sectors. Machine learning technique facilitates ship detection through convolutional neural network (CNN) and provides us with an efficient tool. In this project CNN using You Look Only Once (YOLOv3) architecture has been utilized on Synthetic Aperture Radar (SAR) images to detect ships. Several models were created by changing the image sizes to find an optimal network. Results of these models showed that the size and the dimension of the data influence the mean average precision values. By using the image size of 256×256, we achieved higher accuracy with the mean average precision (mAP) of 0.703. Implementing the hyperparameters set 1 and image size of 256×256 on the YOLOv3 architecture provided us with an optimal network for ship detection from SAR. Additionally, in order to assess the influence of pre-trained weights on the network performance, we used an architecture with initializing the model with the darknet convolutional pre-trained weights. In our experiment, we used the two datasets and the results of the network using 256×256 image size showed a noticeable increase in mAP value compared to other experiments. This architecture achieved an accuracy of 0.739 and has an accurate ship localization. We would recommend machine learning for localization of ships on satellite images.
Keywords: Machine learning, Fully convolutional neural network, YOLOv3, SAR, Ship detection.

2 Introduction

Ship detection plays a significant role in the maritime, environmental and energy sectors. With the development of remote sensing technology, there have been increasing attention to applying earth observation data for ship detection. Synthetic Aperture Radar (SAR) data hold a great promise for this aim, as it is independent of cloud coverage and weather condition and could be collected in day and night (li et al., 2017; Yang et al., 2018; Yang et al., 2012; Jiao et al., 2018). Traditional methods of ship detection using SAR are often associated with difficulties in nearshore areas and with false positives such as icebergs, small islands, and speckle noise (Crisp., 2004).

The deep learning method uses neural networks with many layers to model representations of input data (Goodfellow et al., 2016). Applying deep learning methods with more efficient detection algorithms is a new cost-effective technique in ship detection domain. Deep learning Convolutional Neural Network (CNN) has been widely applied for object detection; however, few attempts have been made on ship detection from SAR imagery (Jiao et al., 2018; Chang et al., 2019; Wang et al., 2019).

There are some challenges using satellite images including spatial resolution along with the geometry of the ships to be detected. Gallego et al (2018) applied optical aerial images to overcome such limitations. They built a dataset of 6000 samples labeled and classified into seven classes. The k-Nearest Neighbor (kNN) method was then applied to classify features extracted by CNN. The limitation of using aerial images is that they only use visible spectrum and are not able to cover ships at night. Besides, changes in weather condition is a challenge for using aerial images. Yang et al (2018) proposed Rotation Dense Feature Pyramid Networks (R-DFPN) to detect ships using images from Google Earth. The R-DFPN is a multiscale detector that builds feature maps with high semantic information by dense connection and using a rotation anchor strategy to avoid the redundancy region and to deal with the difficulty of densely arranged targets.

Compared with other earth observation data, SAR is considered as one of the most suitable datasets for ship detection due to its imagery capability. Jiao et al (2018) proposed a densely connected end to end neural network for multiscale and multiscene SAR ship detection. They used ResNet (He et al., 2015) as the backbone of their network which consists of two sub-networks; region proposal sub-network (RPN) and detection sub-network. In their method, each feature map is connected to every other feature maps from top to down to generate Region of Interest (RoIs) from the fused feature map. Higher feature maps of CNN have a lower spatial resolution, while lower feature maps have higher resolution, which makes them suitable to detect large-scale and small-scale object detection, respectively. Also, an improved loss function was introduced to prevent the inference of inshore complex background that are highly like the ships. The suggested loss function is multitasked function as faster RCNN consisting of two parts; the first one is classification loss which is the softmax loss (L_{cls}) for two classes and the second function is for loss of bounding box regression (L_{reg}). To deal with easy examples which are dominant in the training process they multiply a modulating factor to the cross-entropy. However, the applied network prediction led to miss-classifying objects such that there were false positives and missing ships in prediction results.

Chang et al (2019) applied YOLOv2 to detect ships on SAR images and could achieve an accuracy of 90 percent. With comparison to the top detection method, faster region-based CNN (faster R-CNN), YOLOv2 (Redmon et al., 2017) has shown better performance for ship detection.

The size of the dataset is important to train a good model in CNN so that there would be sufficient representatives for each class to avoid over-fitting and obtain discriminant features. One problem in some research is using a low number of samples which required the regularization (Tang et al., 2015; Yang et al., 2012). Few studies used a large number of data collected from Google Earth (Zou and Shi., 2016; Zhang et al., 2016). However, it is difficult to access these datasets as they are not public. A large number of SAR data could be used to learn powerful models and improve object detection performance. For this project, 2000 SAR satellite images were collected from different locations in the world and various satellite platforms such as ALOS, Cosmo-SkyMed, Envisat, ERS, Radarsat-1, and Seasat. This dataset is with more diversity in ships and having various SAR image resolution. However, due to the low quality of some of the training data, a small set of this collection was chosen based on two criteria; the existence of visible ships in images and the proximity to ports. Thus, we would be able to precisely evaluate the network outcomes and prediction results.

The CGG company has products and services, which incorporate ship identification using SAR data. This can be used for a range of purposes from general situational awareness to pollution studies to combating illegal fishing. Currently, in most cases, a CFAR method is used for automated bright spots detection in SAR. While this is a fast method, it is also heavily prone to false positives when used for ship detection. It is expected that state-of-the-art machine learning techniques such as convolutional neural networks will be able to outperform CFAR and CGG is interested in exploring the viability of this approach.

The main objective of this project is to design and train CNN to process the SAR images and identify ship locations. In this project, we apply the most advanced YOLOv3 method (Redmon and Farhadi, 2018) because unlike other approaches; it sees the entire image during training and testing periods and therefore encodes contextual information about classes as well as their appearance. It has three predictor layers to detect large, medium, and small size objects. Ships often appear as small size objects on the satellite data, therefore, applying YOLOv3 could yield reliable detection. This research provides a framework to assess

the performance of the YOLOv3 for ship detection from satellite data. After comparing obtained results by tuning hyperparameters and using different image sizes, an optimal architecture for detecting small ships in large satellite data was chosen to build a pipeline for ingesting SAR imagery and outputting ship locations.

The detection results illustrate that the network could localize ships precisely and in most cases, it neglects nearshore areas look like ships. Future work would be providing the network with accurate labels and much larger data to produce robust models for ship localization with high certainty.

3 Data

3.1 SAR Images

SAR images resolution are ranging from 10 to 20 meters. The collected image sizes were too large to be used by the YOLOv3. The size of images can be up to 20000 $\times$ 20000 pixels which could make holding in memory and doing convolutions unfeasible using available GPUs. YOLOv3 accepts smaller image sizes; therefore, images of this collection were tiled into sub-images (Figure 1a). At the first attempt, the image size of 512$\times$512 was chosen to create tiles from SAR images. This is a good trade off between maintaining enough information for context and getting enough images in GPU memory. Images of a fixed size of height and width were used to be processed in batches. A set of 2422 images containing ships were created and divided into training and validation set with the proportion of 8:2.

To experiment with the influence of the image size on the network performance, the second set of sub-images with the size of 256$\times$256 were created from the same SAR image collection. From the second division, 3211 images containing ships were available for our experiments. The dataset was also divided into a training set and validation set with the proportion of 8:2.

3.2 SAR Labels

The SAR data were collected from different sensors with various spatial resolutions. Images contain ships of different sizes, open water, wharf, coastal zones, wind farm, and offshore islands. Smaller ships often only cover a few pixels within a SAR image. Other objects such as oil platforms and icebergs produce similar signals leading to ambiguity in the identification of small ships. Manual interpretation of ships is time-consuming and difficult due to this ambiguity which can lead to mislabeling of data. In recent years there has been a significant increase in the volume of data available magnifying these issues further. To prevent bias and because of time constraints, an automatic ship labeling method was chosen for our project.

Labels were provided by a constant false alarm rate (CFAR) method with all detections assumed to be ships. The CFAR techniques have been widely applied in ship detection using SAR data. The CFAR method is used to set a threshold so that we can identify targets above the background signal while maintaining a constant false alarm rate. First, a distribution function is fitted to a region of interest centred on a pixel to be tested. The average power of the region is then calculated while excluding the pixel to be tested and its adjacent pixels to ensure any potential target does not bias the average. This leads to an adaptive threshold across the SAR image which is necessary for taking variable ocean and atmospheric conditions into account. Pixels with power above the threshold are deemed a target, or in this instance a ship. The CFAR method has

shortcomings in high non-homogeneity of sea clutter caused by complicated sea conditions. The probability detection, which is a function of the signal to noise ratio varies depending on a defined threshold.

The outputs of the CFAR implementation on SAR are polygons (ESRI shapefile) indicating the presence of ships. In object detection, labels are fed to the algorithm as bounding boxes around targets. The bounding boxes were created from detected polygons (ships) using their center, width and height coordinates. These coordinates were then normalized concerning the image dimension (Figure 1b).

Images were mostly selected near ports and shipping lanes to maximize the likelihood that bright spots were ships. Thus, we could compare the detected ships with reliable ground truths to evaluate the performance of the YOLOv3.

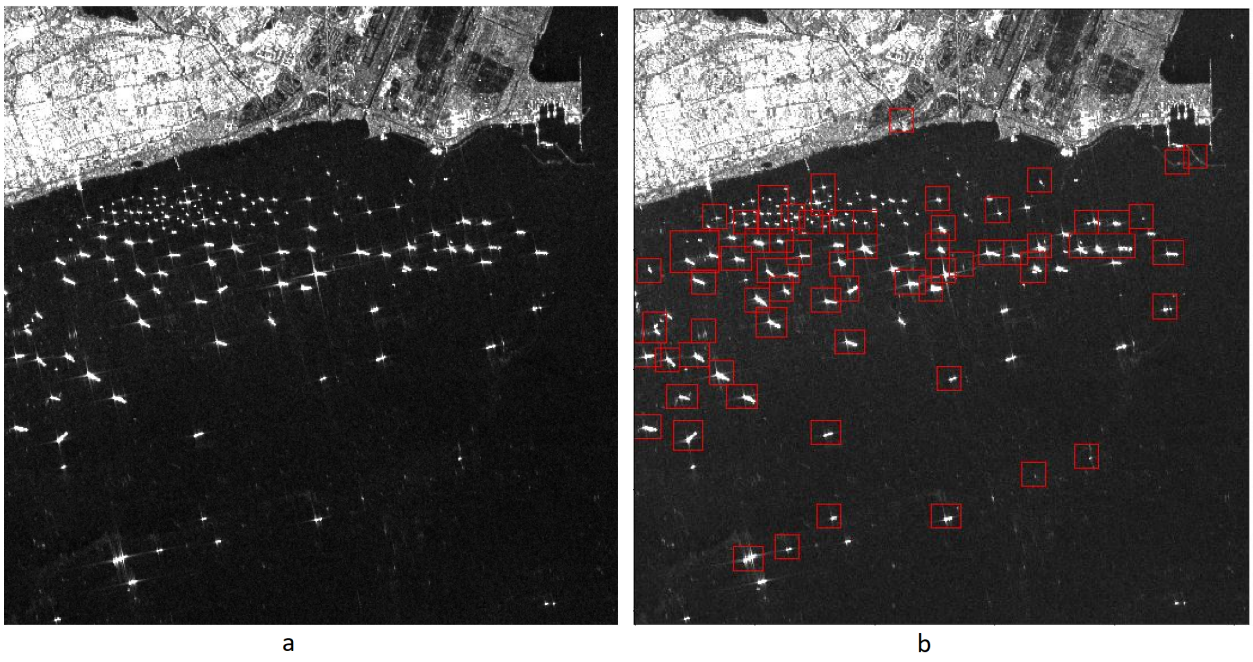

Figure 1: A SAR imagery (a) overlaid by ground truth bounding boxes (b)

4 Method

4.1 Software Development Management Model

The Agile-Waterfall hybrid software development method (Agifall)(Fromson, 2013) was chosen for this project management. This provided the benefits of waterfall sequential and iterative process of Agile. While the Waterfall approach was applied for planning, requirements gathering, and documenting the project's process, Agile was used during the development phase meaning that phases were completed through Agile methodologies. Applying this model allowed rapid iteration and parallel tasks to be completed resulting in efficiency gains. For example, I applied the method on a set of data set with size 512×512 while pre-processing another set of data with the image size of 256×256. The following criteria were considered to select the hybrid method:

• The requirements of the project including design, function, and goals were recorded; therefore, the specification of input and output were defined in the requirement phase (initial proposal by the CGG company).
• Second phase includes research and study of the first step to design a framework for the software, technology, and system requirements (proposal).
• Third step involved coding in small units and integrate them to a complete product (YOLOv3-for SAR ship detection)
• Fourth step consists of testing the product. We have tested the YOLOv3 on data of interest (SAR) for our goal .
• The modification was applied based on obtained results so that we made changes in previous phases to improve the final output.
• Decision on modifications and changes were made based on regular meeting and discussion with the project supervisors.
• The product is installed on GitHub.

4.2 YOLOv3

4.2.1 Network Design

YOLOv3 is a fully convolutional neural network (FCN) using the features learned by deep convolutional layers to detect objects. The input of YOLOv3 are images of size w×h×c, where w and h are width and height of images and c is the number of channels. We applied YOLOv3 for two sized data; thus, the inputs are 512×512×3 and 256×256×3. The channels of SAR grayscale images were stacked together leading to three identical channels. The outputs of the algorithm are predicted bounding box, objectness and class predictions. The size of input image should be an integer multiple of 32 (such as 416×416, 512×512, 256×256) because of the total of five steps for downsampling operation leading to the most stride of 32.

YOLOv3 uses DARKNET-53 for performing feature extraction (Figure 2) (Redmon and Farhadi 2018). For the task of detection, 53 more layers are added to the network giving us 106 layer underlying architecture for YOLOv3. The architecture consists of skip connections and up-sampling. The YOLOv3 applies skip connections to overcome with vanishing gradient problem of deep neural networks and uses up-sampling and concatenation to preserve fine-grained features of small objects. Each convolutional layer is followed by a batch normalization aimed at normalizing the input of each layer to overcome the internal covariate shift problem. The network consists of small-sized convolutional filters of 1×1 and 3×3. The 1×1 convolutions performs weighted summation along depth dimension which ultimately results in a single feature map produced by the layer. It can shrink the number of channels and save the computation (Howard et al., 2017).

YOLOv3 applies no form of pooling to prevent losing low-level features which are attributed to pooling.A convolutional layer with stride 2 is used to downsample the feature maps. We use the following leaky rectified linear activation as an activation function of the network:

$$\phi(x) = \begin{cases} x & \text{if x>0} \\ 0.1x & \text{otherwise} \end{cases} \tag{1}$$

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53

Figure 2: The Darknet-53 feature extractor architecture including convolutional layers, residual blocks, and strides (Redmon and Farhadi, 2018)

4.2.2 Detection Across Scales

Ships appear on SAR images at the different scales due to changes in satellite resolution and different size of ships(e.g. tanker and boat). Therefore, detection at variable scales would allow a generalized detector.

The network has three YOLO layers as detectors at three different scales obtained from different places of the network. (Figure 3). Detection layers make detection at feature maps with strides 32, 16, and 8.

From the base feature extractor, several convolutional layers are added and the last of these predicts a 3-d tensor encoding bounding box, objectness, and class predictions. Based upon scale for each grid cell three bounding boxes are predicted so the tensor is S×S×[3×(4 +1 +1)] for the 4 bounding box offsets (x, y, w, h), 1 objectness prediction, and 1 class prediction (in our project we only have one class). S indicates the size of the grid, which is calculated as image size divided by stride. For example for an image size of 256×256, S would be 8 (256/32) and the tensor is 8×8×18. at first detection layer.

Then we take the feature map from 2 layers previous and upsampled it by a factor of 2. This feature map is concatenated with a feature map of previous layers having identical feature map size. Further, few convolutional layers are added to process this combined feature map, and eventually predict similar tensor but now twice the size (16×16×18 in comparison to an image size of 256). The same procedure is performed one more time to predict boxes for the final scale. The final prediction benefits from the prior computation as well as fine-grained features from early on in the network (Redmon and Farhadi, 2018). The predicted tensor at final scale is 32×32×18 in comparison to image size of 256×256.

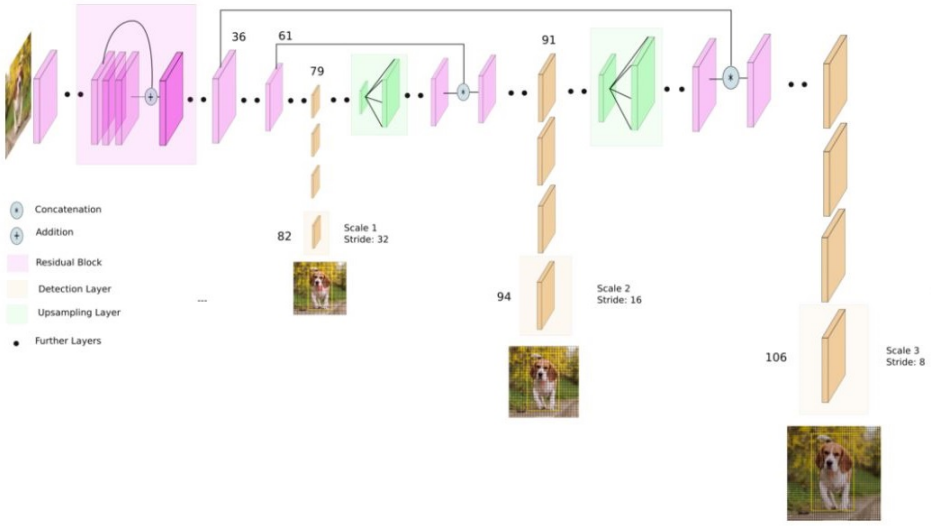

Figure 3: The schematic diagram of YOLOv3 showing the convolutional layers, upsampling layers, residual blocks, and detection layers. Detection layers are placed at three different locations of the network with strides of 32, 16, and 8 to detect objects at at three different scales. The * sign illustrates the concatenation of feature maps and the plus sign (+) indicates the addition of feature maps (credit to Kathuria, 2018).

4.2.3 Bounding box Prediction

It makes sense to predict the width and height of the bounding boxes; however, in practice that yields to unstable gradient during training. In modern object detection methods, anchors which are offsets to predefined default bounding boxes are used to obtain a prediction. YOLOv3 has three anchors to predict the three bounding boxes per cell. The bounding box whose anchor has the highest Intersection over Union (IoU) with ground truth box is responsible to detect the object. The network output is transformed to obtain bounding box predictions through following formula:

$$b_x = \sigma((t_x) + c_x$$

$$b_y = \sigma((t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

where b_x, b_y, b_w and b_h are the x and y center coordinates, and width and height of predictions. t_x, t_y, t_w, and t_h are network output and c_x and c_y are top left coordinates of the grid. p_w and p_h are dimensions of anchors of the box. Sigmoid function is used to predict the center coordinates and output values range between 0 and 1. YOLOv3 predicts offsets, which are relative to the top-left corner of the grid cell (predicting the object) and normalized by dimension of the cell from the feature map (is one). The resultant predictions b_w and b_h are normalized by the height and width of the image.

Non Maximum Suppression method (NMS)(Felzenszwalb, 2010) was then applied to retain only one box

per group corresponding to a precise local maximum of the response function, obtaining only one detection per ship. The number of true detections varies in different images. For example, there might be 1, 10, and 20 true detections for three different images. The NMS threshold was initialized to the value of 0.5 so that only boxes with NMS values equal and/or above this threshold would be kept to detect ships.

4.2.4 Training

We trained the network on SAR data with a single ship class. First, We trained the network for two sets of different sized data from scratch (sections 5.1 and 5.2). Further, we trained the model by initializing it with darknet53.conv.74 weights to investigate the influence of pre-trained weights on the accuracy of the model (section 5.3).

Several loss functions are linearly combined to calculate training and test loss. The classification, confidence, and GIoU loss were calculated and summed up to gain a total train loss and a total test loss. In addition, xy and wh loss were calculated (as an optional alternative to GIoU loss) to give the user the opportunity of choosing between GIoU loss and xy-wh loss.

Generalize intersection over union (GIoU) (Rezatofighi et al, 2019) were used in measuring a loss function for bounding boxes. The GIoU is the state of the art in defining a loss function for bounding boxes. The IoU is often applied to compare similarity between ground truth (label) and the predicted bounding box. However, in cases where IoU is zero, it doesn't reflect whether two bounding boxes are in the vicinity or very far from each other. The GIoU considers the smallest convex hull that encloses both ground truth and predicted bounding box and is calculated as :

$$GIoU = \frac{|A \cap B|}{|A \cup B|} - \frac{|C \setminus (A \cup B|}{|C|} = IoU - \frac{|C \setminus (A \cup B|}{|C|}$$

where A and B are the prediction and the ground truth bounding boxes and C is the smallest convex hull. Negative values of GIoU occur when the area enclosing both bounding boxes (C) is greater than IoU. As the IoU component increases, the value of GIoU converges to IoU. When IoU has no value (no intersection) there will not be gradient. However, GIoU is always differentiable and allows us to calculate a loss function for IoU.

Another localization error which could be considered includes two parameters; xy and wh. The xy and wh loss are also calculated using mean square error because it is one of the loss function in the official YOLOv3. We haven't used this function during training as GIoU is applied to calculate the bounding box error (localization error). A user of the code has been given this opportunity to choose one of the above localization error function (compute loss code in utils.py).

The YOLOv3 predicts an objectivness score using logistic regression. The predicted confidence score is zero if there is no ship in that cell and should be one if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. The confidence score (objectness) loss is calculated using binary cross-entropy with logits (includes sigmoid activation). Also, the classification loss is calculated through binary cross entropy with logits (includes sigmoid activation).

We trained the network using Stochastic Gradient Decent (SGD) optimizer for 300 epochs on the training and validation data sets. Throughout training we used a learning rate of 0.0001, batch size of 16, an accumulate of 4, a momentum of 0.944 and a decay of 0.0005. We set the initial and final learning rates

as hyperparameters hyp['lr0'] and hyp['lrf']. The final learning rate is initial learning rate $\times$ (10^{lrf}). For example, if the initial learning rate is 0.001 and the final learning rate is 100 times (1e-2) smaller then the hyp['lrf'] would be -2.

To avoid over-fitting, we used extensive augmentation such as random scaling, translations, shearing, and flipping.

4.2.5 Evaluation

Precision, recall, F1 score, and mean average precision (mAP) were measured on validation set to evaluate the accuracy of the model. The calculation is based on calculated true positive (TP), false positive (FP), and false negative (FN).We aim to compare the IoU between predicted bounding boxes and ground truth. With an IoU greater than threshold, the test result is a TP, and the values less than the threshold is called FP. The FN indicates that the model predicts no ship in the image but the image contains ships. Precision (P) is calculated as:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{n}$$

where n indicates true positive (TP) + false positive (FP), and is the total number of images recognized by the system. Precision shows how accurate is our prediction and has a zigzag pattern. It goes up with true positives and goes down with false positives. Recall (R) represents how good we found the all positives and is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

The F1 score is a weighted mean of precision and recall and measured as:

$$F1 = \frac{2.precision.recall}{(precision + recall)}$$

The mAP measure is the area under the recall and precision curve, which is ranging between 0 to 1. Larger values of mAP exhibits better prediction accuracy.

4.2.6 Inference

Predicting detections for test images require one network evaluation. YOLOv3 is very fast at inference time since it requires a single network evaluation. The confidence threshold values of detection depend on inference image quality and ship size and are often ranging between 0.1 to 0.9. Experiments had shown that initializing higher values of confidence threshold is suitable to detect ships for images containing larger size ships. These value are decreased for images with small size ships and images containing ships in a cluster in nearby pixels. The confidence threshold for such images is ranging between 0.1 to 0.3 to get a reliable ship localization.

4.2.7 Metadata

Several Python libraries were used in this project including gdal, opencv, torch, geopandas, shapely, sklearn.-model_selection, and other known Python libraries. The satellite data and labels required pre-processing

using GDAL and geopandas. These are commonly used geospatial libraries. The SAR data was opened and tiled using GDAL. The labels were provided in ESRI shapefile format and were opened and re-projected to match the SAR using geopandas and then converted to bounding boxes for YOLO. Maintaining the geospatial information of the data is essential to converting model output back into geographic coordinates.

The backbone of the model is the Darknet network, which has been written in a C programming language by Redmon and Farhadi (2018). In this research, the code has been converted to pytorch format in python language. There are different GitHub pages on application of YOLOv3 on the COCO dataset in pytorch such as https://github.com/ultralytics/yolov3 and https://github.com/ayooshkathuria/YOLO_v3_tutorial_from_scratch. The start up code for SAR ship detection using YOLOv3 is inspired by these repositories.

Several modifications have been applied to be able to import SAR data and achieve the project aims. The Darknet requires special treatment of the dataset in which several text, shape, and data type files should be created. YOLOv3 access the data through the path provided in two txt files for images and labels. The algorithm uses a configuration file to construct the network architecture. The official configuration file was modified to address single class ship localization. The official configuration file of YOLO layer has 255 outputs. There are 85 outputs per anchor calculated by:

$$(4box coordinates + 1objectconfidence + nclassconfidence) \times 3anchors$$

In our research n indicating the number of classes is 1 so that the output filter size is 18. This correction was implemented in all three layers of YOLOv3. The configuration file of this project is named by YOLOv3-1cls and placed in the cfg folder.

Following is a brief description of several Python files, which have been used to build the desired models.

The splitdatato-train-validation.py file has been designed to split data into a training set and validation set and includes codes for drawing ground truth bounding boxes on images and codes to filter files based on defined image and label id.

Several python files were created to implement YOLOv3 on SAR. The model.py includes the Darknet architecture. The train.py, test.py and detect.py are used to train the data (train.py), test the data (test.py), and to detect ships from inference images (detect.py) using the model weights. Two model weights are saved after each training; the best and the last weights. The best weight of the model is used to test the model and to detect ships.

The utils.py file include helper functions such as compute loss, non maximum suppression, build targets, IoU functions, and accuracy metric functions.

The dataset.py consists of mainly two classes named by LoadImages and LoadImagesAndLabels to read and load images and labels for training, testing and inference. SAR data was opened and read by Open cv with a single greyscale channel. To adopt images with the network structure, channels of SAR have been stacked to create three identical channel images. The dataset.py include augmentation methods as well as letterbox function to padding and re-sizing the images.

The parse_config.py, and torch_utils.py contain the codes for data, path, and device configurations.

A file named by main.py was created to call the train.py, test.py, and detect.py. The configuration file, image size, batch size, and accumulate number have been initialized at each call of the python files.

4.2.8 Installing the software

The software is available on GitHub. The first step is providing all required text files including the name.DATA file, name.NAMES file (name of classes), shape.SHAPES (dimension of images), train.txt containing the path to all training images, validation.txt including path to all validation images. The image path replaced to label path in the LoadImageAndLabels class during trainng and testing.

The second step is changing torch_utils.py based on available devices. Further, the configuration file in cfg folder should be adopted based on the number of classes, channels, and output filters of the YOLO layers.

From the main.py run following steps to get the results.

Run !python3 train.py –data data/ship-obj.data –img-size 256 (or 512) –batch-size 16 –accumulate 4.

The values for batch size, img-size, and accumulate could be changed based on the desired format. This would train the training data and provide different loss plots for training as well as metric measures plots for testing.

Run !python3 test.py –data data/ship-obj.data –img-size 256 (or 512) to test the model and acquire the mean average precision.

Run utils.plot_results() to plot the results of training and testing.

Run !python3 detect.py on inference images to detect ships. One should keep in mind that the confidence threshold should be changed for different images. This is subjective and depends on the quality of the image and the size of the ship.

5 Results

We evaluated the feasibility and performance of the YOLOv3 architecture for SAR ship detection using two sizes of the training dataset. Different scenarios were defined to achieve our aims. The learning rate of 0.0001 and two sets of hyperparameters were used to train several models. For the sake of fair comparison, all network architectures were trained using one GPU and to 300 epochs with a batch size of 16.

The result section is divided into three subsections; The first and second sub-sections (5.1 and 5.2) explain our experiments training the YOLOv3 network without initializing the model with the pre-trained weights. The first sub-section reports the results of such training using the image size of 512×512 and the second sub-section presents the training results using image size of 256×256. In all scenarios, the model is first run on data without any augmentation and then augmentation was applied on the SAR using those hyperparameters that return higher mAP values. Applying augmentation would help us to check whether the loss and the evaluation metric would be improved by artificially increasing the numbers of images and introducing more variations in the data.

The third sub-section (5.3) describes our experiment of training YOLOv3 network by initializing the model with the darknet53.conv.74 weights using the two different sized images. This allows us to investigate the influence of using pre-trained weights on the performance of YOLOv3 network for SAR ship detection and compare it with previous models.

5.1 Experiments with the image size of 512×512

At the first attempt, the network was trained on images with the dimension of 512×512. Table 1 shows dataset distribution.

Table 1: Ship detection dataset distribution (dimension of 512×512)

Data sets	Number of Samples
Training Set	1937
Validation Set	485
Total	2422

Table 2 represents the results of the training on 1937 data and evaluation metric values of 485 validation data using two sets of hyperparameters. The loss gain hyperparameters are multiplied by computed losses and the iou_t is the IoU target-anchor threshold which is used for building targets so that anchors below this threshold would be rejected to build a target. The hyperparameter set 1 returns a better accuracy with mAP value of 0.532 and best fitness value of 0.515. The resultant model using hyperparameter set 2 has a mAP value of 0.522 and best fitness value of 0.487.

Table 2: The evaluation results of trained models using image size of 512×512.

Set	Best fitness	Best fitness epoch	Giou loss gain	xy loss gain	Wh loss gain	Obj loss gain	iou_t	mAP
Set 1								
	0.515	197	1.2	4.062	0.185	20.0	0.194	0.532
Set 2								
	0.487	204	1.43	4.27	0.265	11.5	0.281	0.522

Figure 4 presents the results of loss functions and metric values of generated models using hyperparameters set 1 and set 2. The GIoU loss converges for the two models and the train and test loss plots gradually decline. As expected the classification loss doesn't show variation because we have a single class object localization. The wh loss values stay at zero value because instead of xy-wh loss we used the GIoU loss. One can switch to xy-wh loss by changing the setting of the function of computing loss (in utils.py). We use the GIoU loss in all other experiments.

As shown in Figure 4, applying hyper-parameters set 1 lead to a higher value of mAP. It is expected to obtain a zigzag pattern for the precision curve as it goes up with true positive and goes down with false positive. A good object detector shows high values of precision while recall increases meaning that if we vary the confidence threshold precision and recall will still be high. Figure 4 shows that the precision curve of set 2 declines sharply for the first 100 epochs, while the precision curve of set 1 has a steady trend. The mAP, recall, and precision curves reflect a more stable model for hyperparameter set 1.

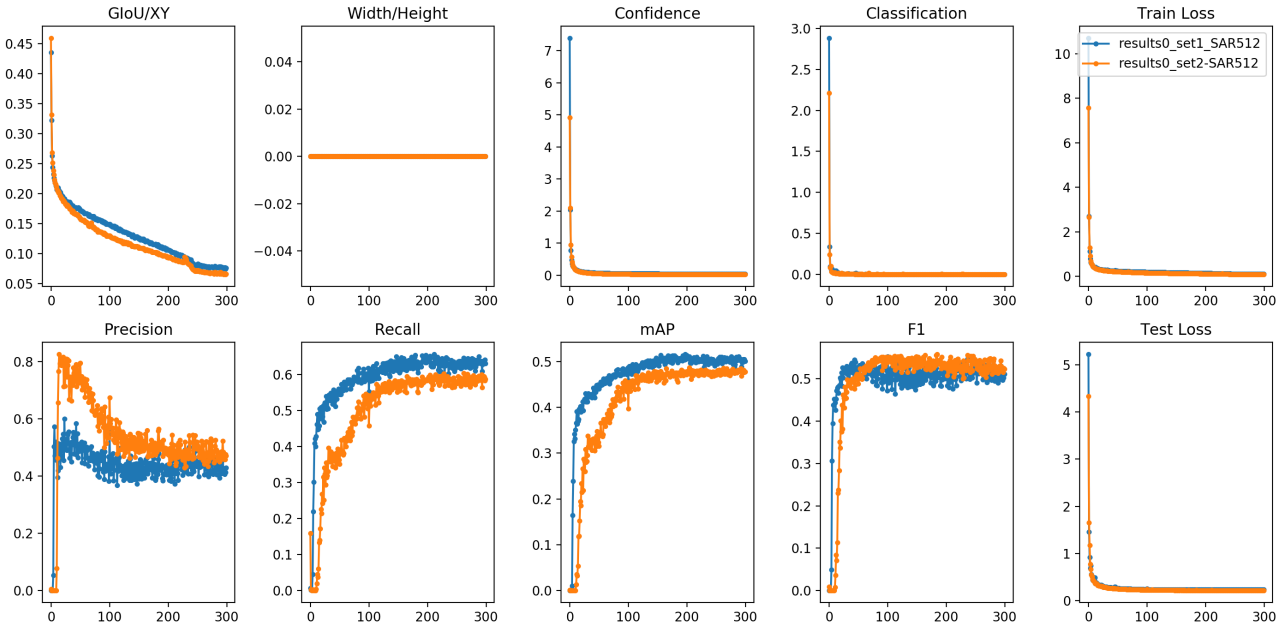

Figure 4: The loss and metric plots of three models using hyperparameters set 1 and set 2. The above curves present the giou, classification, confidence, and train loss while the below curves show the measured precision, recall, mAP, F1, and test loss.

Comparing the outcomes of both sets and evaluating the best fitness values, set one show a better performance with a higher fitness value. The hyperparameters of the set 1 with the best fitness value of 0.515 were selected as the best set of hyperparameters of the network.

Several experiments have been made to check the SAR ship detection and the chosen model provided us with better results using inference images. Figures 6 a, c, e, and g illustrate training data and ground truth bounding boxes, while figures 6 b, d, f, and h show results of the best model prediction in the sea and coastal regions. Although the model shows difficulties in detecting closely spaced small ships (Figures 5f and h), it can precisely detect ships in the sea (Figures 5b and d).

Augmentation methods including translating, shearing, flipping, and rotating were applied on training data to check whether the performance of the models would be improved. Table 3 exhibits the results of two models after applying augmentation methods. As shown in Table 3, the mAP and the best fitness values of the models were not improved by applying augmentation. We expect this due to the inherent random orientation of ships in the SAR data.

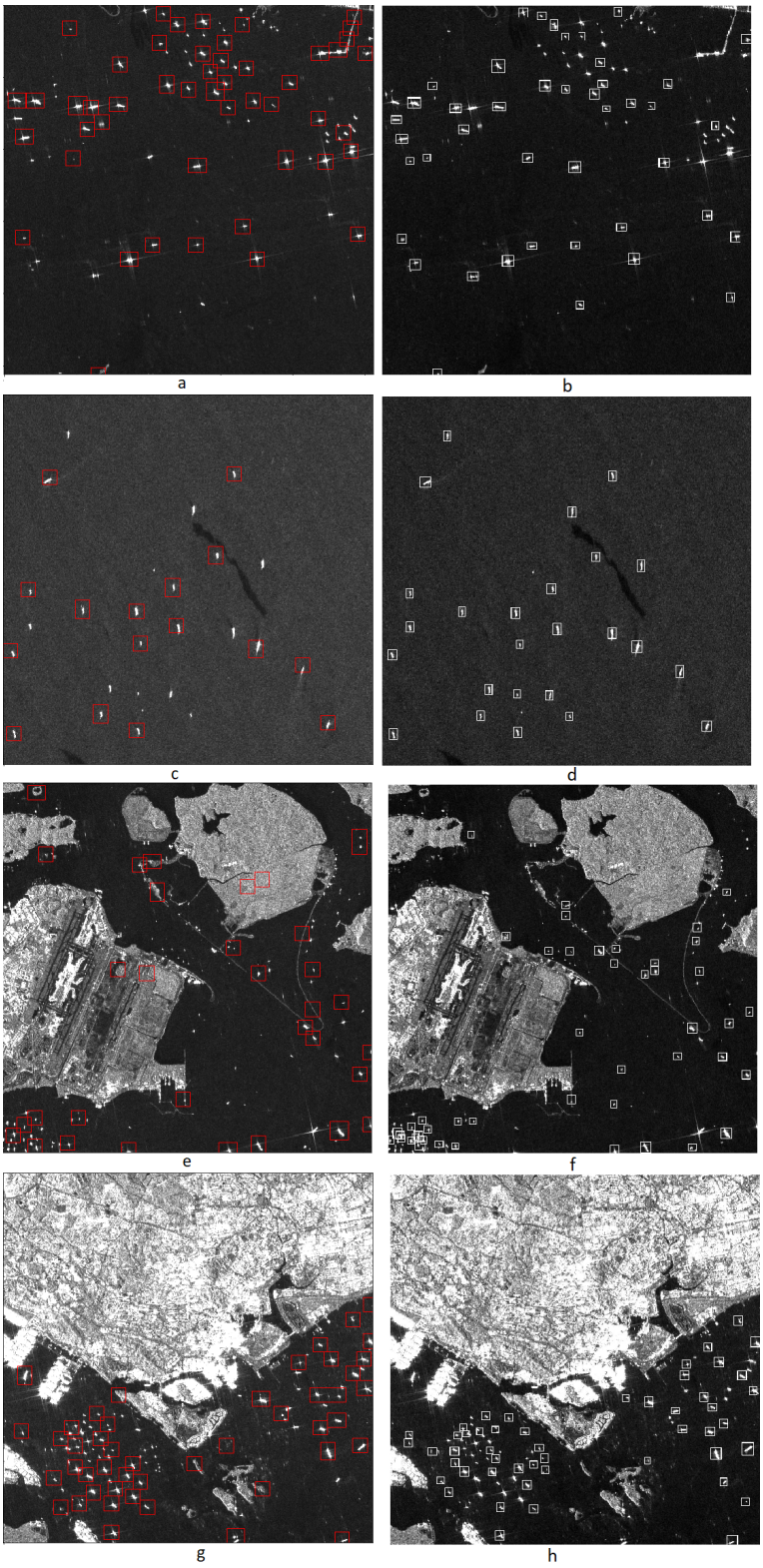

Figure 5: Training data and ground truth bounding boxes (a, c, e, and g) and prediction images (b, d, f, and h) using the image size of 512×512.

Table 3: Evaluation results of the models using augmentation using image size of 512×512

Set	Best fitness	Best fitness epoch	Augmentation	mAP
1	0.482	225	Flip-translate-shear-resize-rotate	0.495
2	0.465	225	Flip-translate-shear-resize-rotate	0.49

5.2 Experiments with the image size of 256×256

At the second attempt, we evaluated the performance of YOLOv3 on 3211 SAR images with the dimension of 256×256. Table 4 exhibits the dataset distribution for the second experiment.

Table 4: Ship detection dataset distribution (dimension of 256×256)

Data sets	Number of Samples
Training Set	2568
Validation Set	643
Total	3211

The network was trained using two sets of hyperparameters and the learning rate of 0.0001. Table 5 shows the results of network performance. Set 1 of hyperparameters produced a best fitness value of 0.686 and mAP value of 0.703, which are slightly higher than the best fitness value of 0.680 and mAP of 0.702 of hyperparameters set 2.

Table 5: The evaluation results of trained models using image size of 256×256.

Set	Best fitness	Best fitness epoch	Giou loss gain	xy loss gain	Wh loss gain	Obj loss gain	iou_t	mAP
Set 1								
1	0.686	110	1.2	4.062	0.185	20.0	0.194	0.703
Set 2								
2	0.680	156	1.43	4.27	0.265	11.5	0.281	0.702

As shown in Figure 6, the accuracy of the model was dramatically increased by reducing the size of the images to 256×256. The precision and F1 curves illustrate a steady increase using hyperparameters set 1. The recall and mAP curves of set 1 decline after 110 epochs. Applying hyperparameters set 2 led to a lower precision than set 1 and needs more epochs to reach high accuracy. Using both sets, the GIoU loss curves decline and confidence and classification loss curves don't change.

Table 6 presents the results of the network evaluation metrics after applying augmentation methods. The results show no improvement in the outcomes of the network and the mAP value is lower than mAP value of training the model without augmentation.

The detection results of the network were produced on inference images to evaluate the model performance on ship localization. Although there are few missing ships and errors in detections, The results demonstrate a reliable ship localization (Figure 7). Figures 7 b, and d exhibit the results of ship detection in the sea, while Figures 7 f and h show the detected position of ships in SAR images hosting sea and land regions. It has shown that the network could learn about small-sized ships using the image size of 256×256, leading a more powerful model of ship detection.

Table 6: Evaluation results of the models after augmentation using image size of 256×256

Set	Best fitness	Best fitness epoch	Augmentation	mAP
1	0.653	233	Flip-translate-shear-resize-rotate	0.665
2	0.647	233	Flip-translate-shear-resize-rotate	0.662

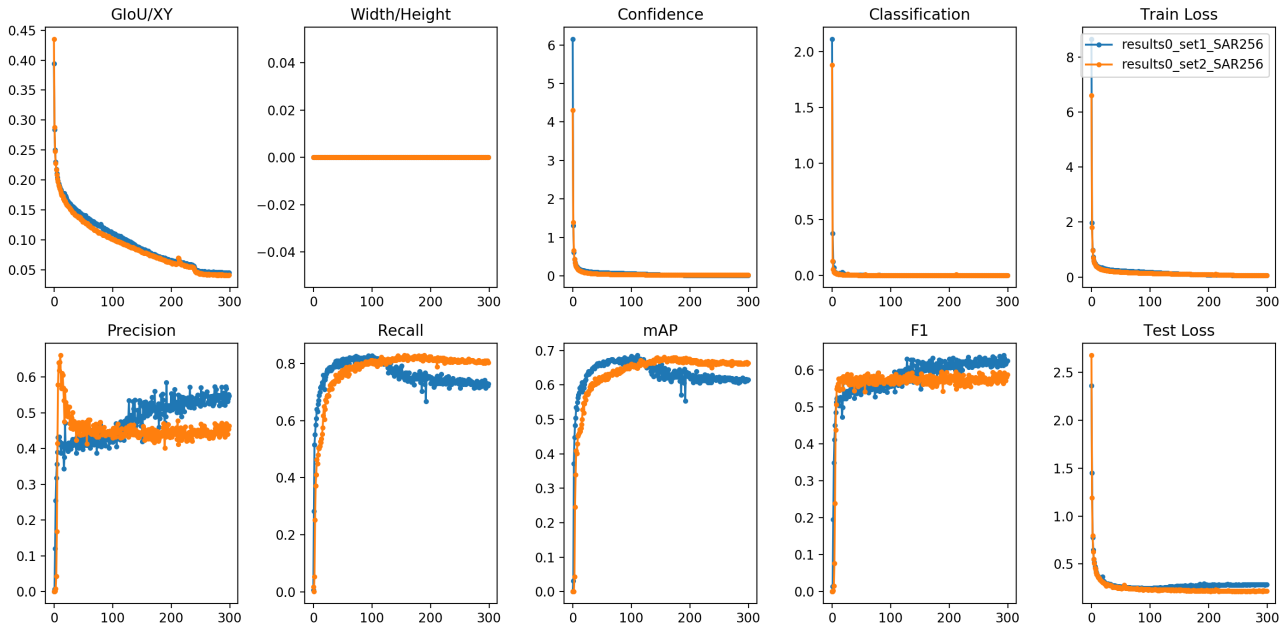

Figure 6: The loss and metric plots of two models for images with dimensions of 256×256 using two sets of hyperparameters. The above plots present the giou, confidence, classification, and train loss values while the below plots show the precision, recall, F1, mAP, and test loss.

Although the network detected ships with high accuracy, there are few missing ships in the detection results. Detection errors occur in areas where small ships are labeled as a group in neighboring pixels in the training data as well as in coastal zones where it is difficult to discriminate between ships and other objects.

An approach to overcome these limitations of training data is paying attention to choosing the proper confidence threshold level and initializing suitable inference image resolution in the detection step. We applied different confidence threshold ranging between 0.1-0.9 to detect ships on various SAR images. The lower values were chosen to detect small ships, while higher confidence threshold values were used to detect larger ships and disregard land areas.

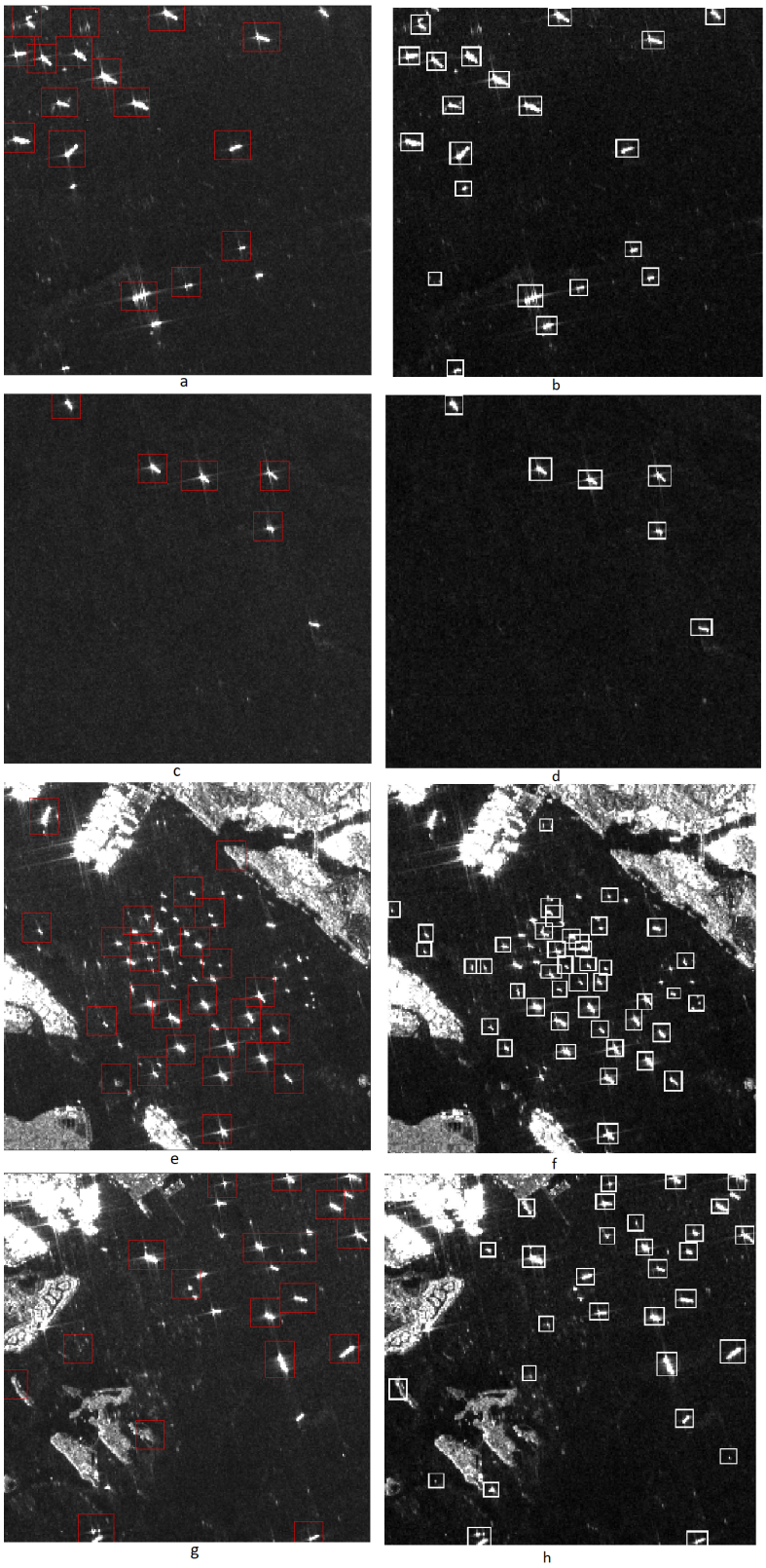

Figure 7: Training data and ground truth bounding boxes (a, c, e, and g) and prediction images (b, d, f, and h) using the image size of 256×256.

5.3 Experiments of SAR ship detection by initializing the model with darknet53.conv.74 weights

This section presents our experiment to investigate the influence of initializing the model with darknet53.conv.74 weights on the performance of the network for SAR ship detection. The YOLOv3 network was trained and evaluated using two sized datasets and ship detection results were produced to assess the value of initializing the models with pre-trained weights.

Table 7 presents the results of trained models evaluation using the image size of 512×512 and two sets of hyperparameters. Compare with the previous results presented in section 5.1 we achieved higher accuracy with the mAP value of 0.57 and the best fitness value of 0.558 for set 1 and mAp value of 0.574 and the best fitness value of 0.549 for set 2.

Results of the evaluation of trained models using two sets of hyperparameters and image size of 256×256 are shown in Table 8. The obtained results show a high accuracy value of 0.739 with the best fitness value of 0.722 for set 1 and mAP value of 0.729 and best the fitness value of 0.713 for set 2, which shows a better performance of the network in comparison with our experiment in section 5.2.

The trained models were applied on inference images to precisely detect ships as individual objects. Figures 8 and 9 show the ground truths and detection results of the obtained models using the image size of 512×512 and 256×256, respectively. With comparison to results of previous experiments, the obtained results show lower errors in localizing ships. Figures 8 b and d illustrate the ship detection results in the sea, while Figures 8 f and h show the ship localization near coastal zones using images size of 512×512. There are few missing ships (Figure 8h) in detection results due to the shortcomings of training data. The detection results of the model using the image size of 256×256 indicate a precise location of ships in the sea and nearshore areas (Figures 9 b, d, f, and h). Compared to previous experiments there are fewer errors in detection results. Unlike CFAR which group nearby ships into one group (Figures 9 e and g), YOLOv3 predict ships as individual objects and didn't detect nearshore areas as ships. YOLOv3 could detect ships which were not labeled in training data. Besides, areas of land which were identified as ships by CFAR did not appear in the detection results.

The evaluation results of both datasets and the ship detection results demonstrate the suitability of using pre-trained weights as a better starting point in training models.

Table 7: The evaluation results of trained models using image size of 512×512.

Set	Best fitness	Best fitness epoch	Giou loss gain	xy loss gain	Wh loss gain	Obj loss gain	iou_t	mAP
Set 1								
1	0.558	165	1.2	4.062	0.185	20.0	0.194	0.57
Set 2								
2	0.549	268	1.43	4.27	0.265	11.5	0.281	0.574

Table 8: The evaluation results of trained models using image size of 256×256.

Set	Best fitness	Best fitness epoch	Giou loss gain	xy loss gain	Wh loss gain	Obj loss gain	iou_t	mAP
Set 1								
1	0.722	74	1.2	4.062	0.185	20.0	0.194	0.739
Set 2								
2	0.713	207	1.43	4.27	0.265	11.5	0.281	0.729

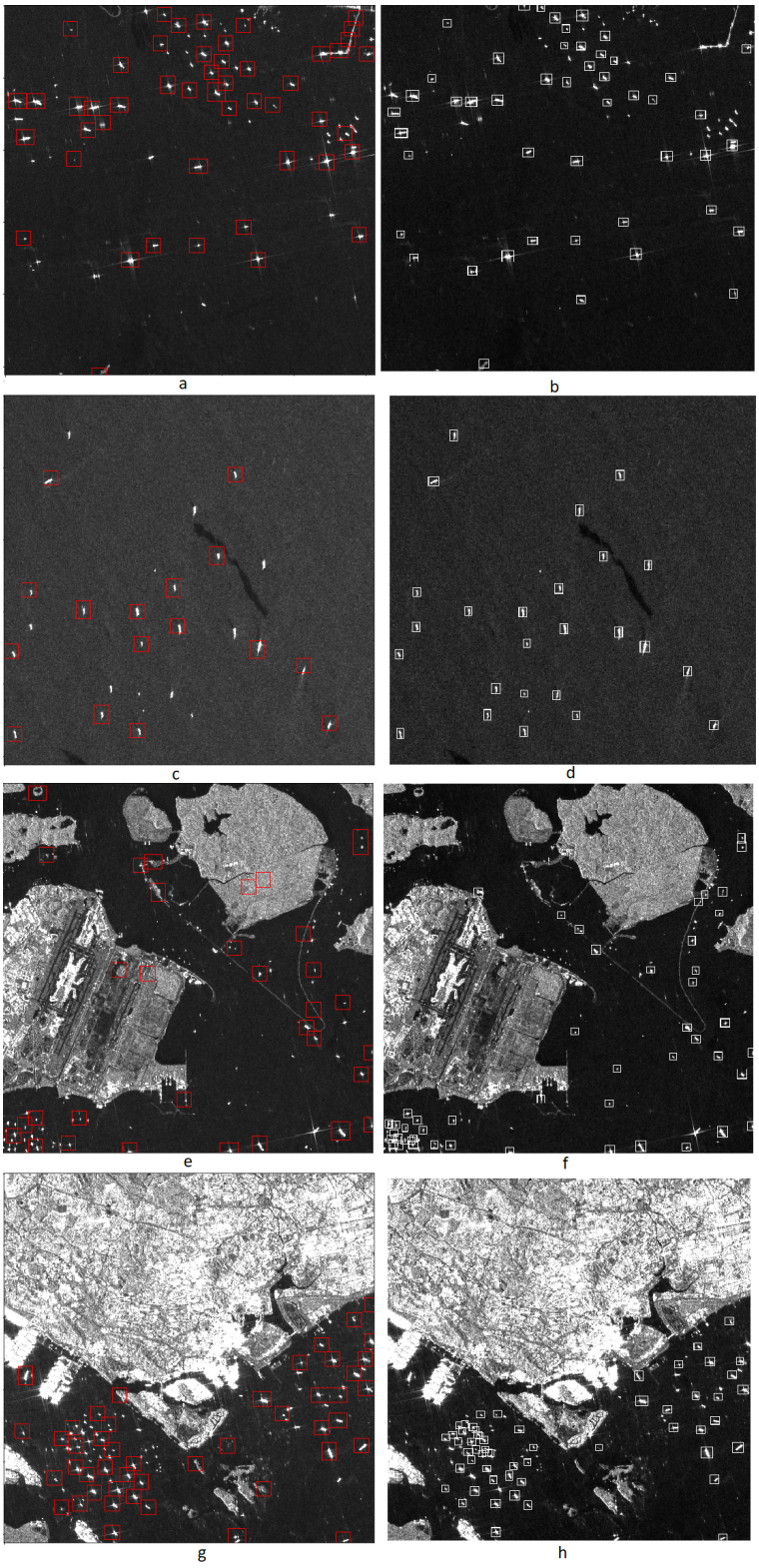

Figure 8: Training data and ground truth bounding boxes (a, c, e, and g) and prediction images (b, d, f, and h) using the image size of 512×512. Models were initialized by darknet53-conv-74 weights.

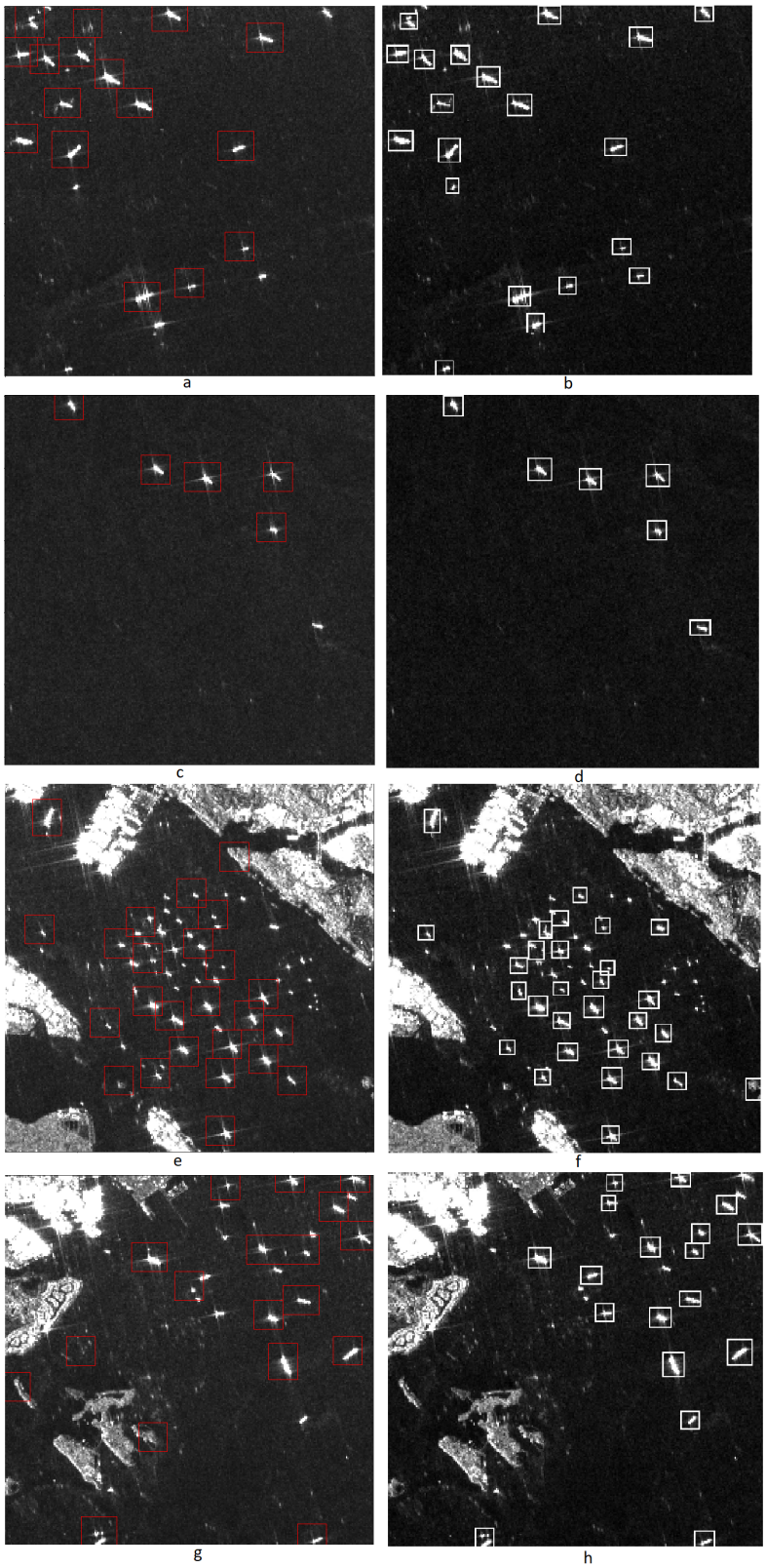

Figure 9: Training data and ground truth bounding boxes (a, c, e, and g) and prediction images (b, d, f, and h) using the image size of 256×256. Models were initialized by darknet53-conv-74 weights

6 Discussion

In this project, we evaluated the performance of the YOLOv3 deep network architecture to detect ships based on various scenarios. According to obtained results, YOLOv3 is suitable for SAR image ship detection. The experimental results on SAR images show that YOLOv3 method has some advantages over the CFAR method.

Based on our observation of CFAR results, some of the small closely spaced ships are clustered into one polygon (Figure 2b). Moreover, there are polygons in nearshore areas making it difficult to discriminate between land and ships. YOLOv3 predicts only one bounding box per object, therefore, predictions would not be grouped into one cluster and this leads high certainty in SAR ship localization. Also, in most cases, with initializing a suitable confidence threshold based on image quality and sizes of ships, the YOLO detectors didn't predict bounding boxes on the land.

We investigated the performance of the YOLOv3 architecture with different image sizes and assessed the impact of the image size on the detection performance. For this purpose, we selected the SAR dataset with two different image sizes of 512×512 and 256×256. We found that the image size had a significant influence on the accuracy of the model. The detection results of using image size of 512×512 is promising; however, the network shows difficulties detecting small ships in neighboring pixels. To overcome this limitation, images with the size of 256×256 were used and experimental results of this set of images demonstrated that YOLOv3 architecture return higher mAP values and a better detection accuracy. The reason is using image size of 256×256 the network is able to learn about small ships leading a higher accuracy and a better detection of small size ships.

Optimizing hyperparameters is a key step in making accurate predictions, as they define characteristics of the model that can impact model accuracy. One of the limitation of SAR ship detection using YOLOv3 is that there are not specific sets of hyperparameters for SAR ship detection and they should be chosen based on several experiments which might even not result in finding an optimal set. As a part of our search to select accurate hyperparameters, two sets of hyperparameters specified for the COCO dataset were initialized to train the network and to adjust the hyperparameters. The experiments carried out by initializing these sets of hyperparameters on small size images obtained high mAP; however, using hyperparameters of set 1 returned better mAP.

The YOLOv3 has not yet been applied for SAR ship detection; therefore, the evaluation could not be compared with an existing published work.

Although YOLOv3 network showed a reasonable performance for detection of ships, we applied several augmentation methods including rotating, flipping, and shearing to improve the network performance and to gain higher accuracy values. Results showed that applying these techniques is not suitable for SAR ship detection because SAR data contain ships of different sizes aligned in different directions and the network could learn such kind of variations from the original dataset without augmentation.

To investigate the influence of initializing the model with darknet-52-conv-74 weights on the performance of YOLOv3, we made another experiment on both datasets. We found that using the pre-trained weights for initializing the model returns higher accuracy (Figures 10 and 11) and a better ship localization. The reason behind it is the darknet has strong generic object detection capability, that is then refined by the training.

In our experiences, we observed that errors often occurs in areas with very small size ships in neighboring

pixels. It is worth noting that the accuracy and performance of the network is constrained by shortcomings of the training data. Providing the network with precise ship labels stay as a challenging task. Generation of hand labeled data is prohibitively expensive and time consuming and could result in mislabeling SAR data due to the feature similarity (e.g. brightness, shape, and size) of ships and other objects in the sea such as icebergs, platforms, and wind farm.

Future research work includes incorporating accurate labels and ancillary information into our ship localization system to obtain ship position with high certainty on SAR images. This could be achieved by incorporation ship characteristics such as length, heading and speed and by using unambiguous training data which could be provided by co-incident of SAR and AIS data from the NovaSAR or PAZ. Also, using more than one polarization (e.g., more channels) and adding more images to the current dataset makes it larger and more robust results in generating more powerful models.

In conclusion, we showed the feasibility of YOLOv3 for SAR ship detection. Through different experiments we presented that YOLOv3 is a suitable convolutional neural network for localizing ships and it's detection capacity would be significantly improved by using more accurate training data and including more images in the network. By this project, we proposed a framework for SAR ship detection.

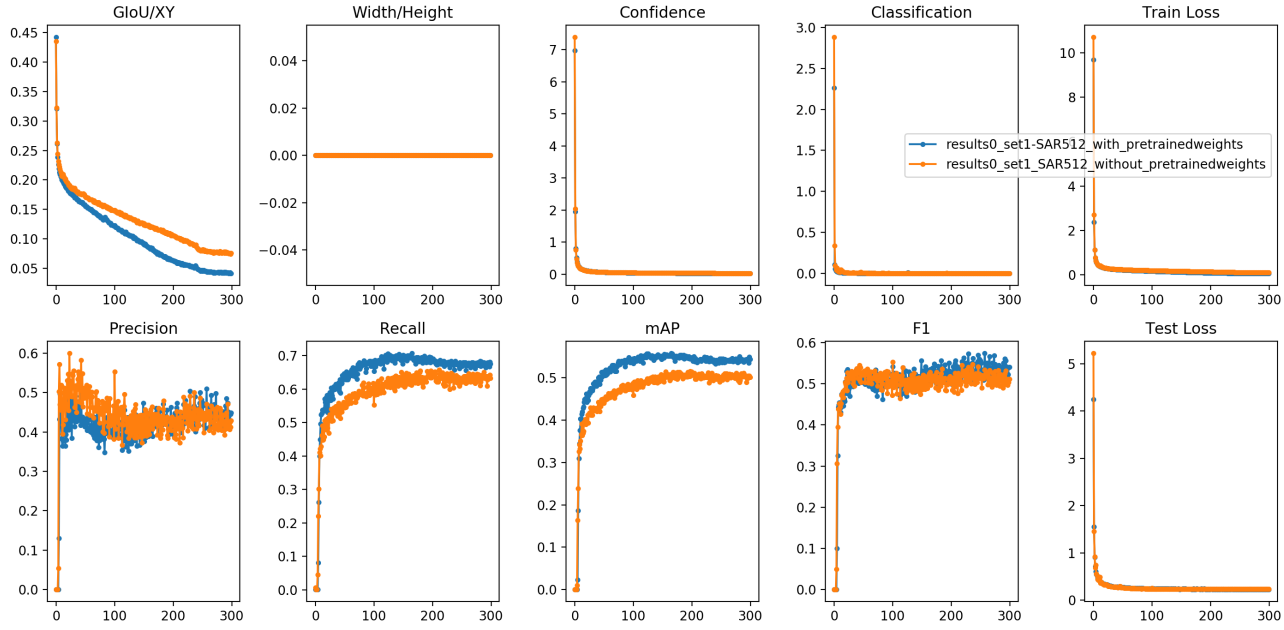

Figure 10: Comparing the loss and metric plots of two models trained without initializing to pre-trained weights and trained with initializing to pre-trained weights using image size of 512×512 and hyperparameters set 1.

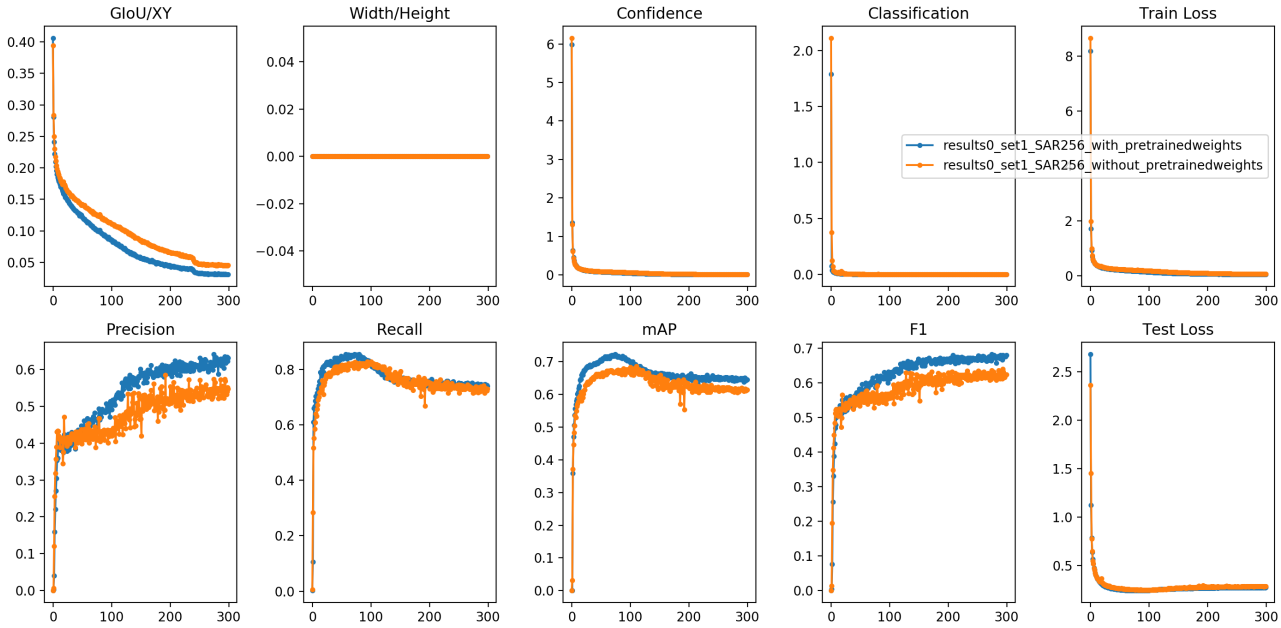

Figure 11: Comparing the loss and metric plots of two models trained without initializing to pre-trained weights and trained with initializing to pre-trained weights using image size of 256×256 and hyperparameters set 1.

7 References

Chang, Y., Anagaw, A., Chang, L., Wang, Y., Hsiao, C., Lee, W. (2019). Ship detection based on YOLOv2 for SAR imagery. Remote Sensing, 11, 768, doi: 10.3390/rs11070786.

Crisp, D. (2004). The State-of-the-Art in Ship Detection in Synthetic Aperture Radar Imagery. Australian Government, Department of Defense: Canberra, Australia, p. 115.

Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.(2010). Object detection with discriminatively trained part based models. TPAMI, 32, 9, 1627–1645.

Fromson, M. (2013). Agifall-Comibing Waterfall and Agile Development Process for Digital and Software project. https://www.slideshare.net/markfromson/agifall-presentation.

Gallego, A., Pertusa, A., Gil, P. (2018). Automatic ship classification from optical aerial images with convolutional neural networks. Remote Sensing, 10, 251.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning, The MIT Press, 800 pp, ISBN: 0262035618.

He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv:1512.03385v1.

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017). MoblieNets: Efficient Convolutional Neural Network for Mobile Vision Application. arXiv:1704.04861v1.

https://github.com/pjreddie/darknet.

https://github.com/ultralytics/yolov3.

https://github.com/ayooshkathuria/YOLO_v3_tutorial_from_scratch.

Jiao, J., Yang, X., Hong, W. (2018). A Densely Connected End-to-End Neural Network for Multiscale and

Multiscene SAR Ship Detection. IEEE Access, doi: 10.1109/ACCESS.2018.2825376.
Kathuria, A. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.
Li, J., Qu, C., Shao, J. (2017). Ship detection in SAR images based on an improved faster R-CNN. In Proceedings of the 2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA), Beijing, China, 13–14 November, 1–6.
Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
Redmon, J., Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
Redmon, J., Farhadi, A. (2018). YOLOv3: An incremental Improvement. Arxiv.
Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S. (2019). Generalized Intersection over Union: A Metric and a Loss for Bounding Box Regression. Arxvix:1902.09630v2.
Tang, J., Deng, C., Huang, G.B., Zhao, B. (2015). Compressed-Domain Ship Detection on Spaceborne Optical Image Using Deep Neural Network and Extreme Learning Machine. IEEE Trans. Geosci. Remote Sensing, 53, 1174–1185, doi:10.1109/TGRS.2014.2335751.
Wang, Y., Wang, C., Zhang, H., Dong, Y., Wei, S. (2019). A SAR dataset ship detection for deep learning under complex backgrounds. Remote Sensing, 11 (7), 765.
Yang,W., Dai, D., Triggs, B., Xia, G.S. (2012). SAR-Based Terrain Classification Using Weakly Supervised Hierarchical Markov Aspect Models. IEEE Trans. Image Process, 21, 4232–4243, doi:10.1109/TIP.2012.2199127.
Yang, X., Sun, H., Fu, K., Yang, J., Sun, X., Yan, M., Guo, Z. (2018). Automatic ship detection in remote sensing images from Google Earth of complex scenes based on multiscale rotation dense feature pyramid networks. Remote Sens, 10 (1), 132, 2018.
Zhang, R.; Yao, J.; Zhang, K.; Feng, C.; Zhang. (2016). J. S-CNN Ship Detection from High-Resolution Remote Sensing Images. ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci, 423–430, doi: 10.5194/isprs-archives-XLI-B7-423-2016.
Zou, Z., Shi, Z. (2016). Ship Detection in Spaceborne Optical Image with SVD Networks. IEEE Trans. Geosci. Remote Sens, 54, 5832–5845, doi:10.1109/TGRS.2016.2572736.