Ship Detection From Aerial Images

Ravi Pushkar (2018MT60790)
Zuhaib Ul Zamann (2018MT60798)
Instructor: Prof. Niladri Chatterjee

Course: MTL782(Data Mining)

1. Abstract

Image segmentation is one of the fundamental problems in computer vision. A lot of algorithms have been developed to solve the problem accurately and efficiently. We developed models through machine learning methods(U-net, YOLO) that segments an image into ship containing and non-ship containing portions (Detection as well as Segmentation) in aerial images. We developed modified U-net model for the problem which on training gives a binary accuracy of99%. However even though the model accuracy is high, the model is not accurate enough in terms of IOU. We developed YOLO object detection model to test its accuracy against our U-net model.

2. Introduction

Shipping traffic is growing day by day. More shipping traffic increases the chances of infractions at sea like environmentally devastating ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. This has compelled many organizations, from environmental protection agencies to insurance companies and national government authorities, to have a closer watch over the open seas. Also, Comprehensive maritime monitoring services helps support the maritime industry to increase knowledge, anticipate threats, trigger alerts, and improve efficiency at sea. Thus, there is a need for an automatic extractor of objects from images taken from an aerial view.
Through this assignment, we plan to develop a model that accurately and quickly extracts ships from an aerially taken image. We also plan to develop a model that predicts an image has a ship or does not have a ship. This classification problem is challenging because boats are very small in satellite images and hence hard to detect. We developed a U-net model for segmenting the image and also a YOLO model for predicting the bounding boxes around a ship in an image.

3. Past Works

In recent years, machine learning and artificial intelligence have been a topic of great interest. It has achieved great success in various areas including Natural Language Processing and Computer Vision. Also, many remarkable achievements have been made in the area of image classification, such as AlexNet, ResNet and DenseNet. However, there are many drawbacks in these models such as high computation requirements to train, requirement of very large datasets and sometimes, they are slow as well. In addition, it is a *Kaggle Challenge* problem and several attempts have been made by some Kaggle users.

4. Dataset and Features

4.1. Data Description

We obtained public dataset provided on kaggle competition "Airbus Ship Detection Challenge". The dataset consists of 193K 768x768 satellite images with total size exceeding 29GB. The dataset is highly imbalanced in the sense that approximately 78% of the images (15000 out of 19000) contain no ships at all. Some of the images in dataset are corrupted. Along with these images, there is an annotations file(CSV file) which contains the labelling of images in run length encoding (loss less encoding to save memory). Not having pixel coordinates means the image doesn't have any ships.The run length encoding depicts the pixel coordinates for the segment bounding boxes for a ship in an image. Further details can be found at the following link
Airbus Data

Image Id	Encoded Pixels
000155de5.jpg	264661 17 265429 33 266197 33 ...
000194a2d.jpg	360486 1 361252 4 ...
0001124c7.jpg	
000194a2d.jpg	198320 10 199088 10 ...

Table 1: Rows of csv file

Table 1 shows the encoded pixels for the corresponding image id. Here, imageId is the image name in the kaggle working directory and the encoded pixels is the segmentation masks corresponding to a ship in the image. The mask is in Run Length Encoding.

4.2. Run Length Encoding

Run Length Encoding(RLE) is a form of lossless encoding in which runs of data are stored as single data value and count rather than as the original run(Wikipedia). This is most efficient on data that contains many such runs, for example simple graphic images such as icons, line drawings etc. In our case run-length encoding consists of pairs of numbers (starts and lengths) where starts are the coordinates of the pixels in flattened image and the lengths are the total run of pixels from start which we should include in our mask. For example, consider a 4x4 image and suppose we want to include pixles (2,1), (2,2) and (2,3) in our mask. First, we will flatten our 2d array in 1d array of size 16. Corresponding to (2,1) we have 4*(2-1)+1 = 5 in our array. Hence the run-length encoding of this run will be 5 3, as shown in Figure 1.

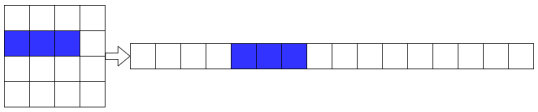

Figure 1: Mapping of 2d Array to 1d Array

4.3. Data Preprocessing

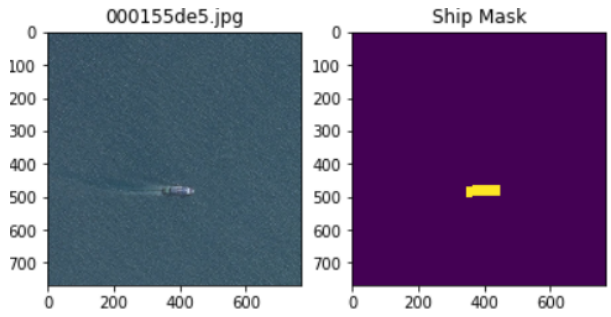

Figure 2: Image of dataset and generated mask

Due to very large size of the dataset, pre-processing was done at the time of training. For this, we created a custom image data generator class which took a batch of size of 48 from the dataset. All the pre-processing was done in that generator. We converted the given Run Length Encoding into an image mask and overlaid all masks of an image to create one single mask that separated the image into ship containing and non-containing segments. We resized both image and mask into a 256x256x3 image in order to reduce the parameters of our U-net model. Using Keras ImageDataGenerator class, we augmented images data with rotation, adjusting brightness, shifting, zooming and flipping. Some of the

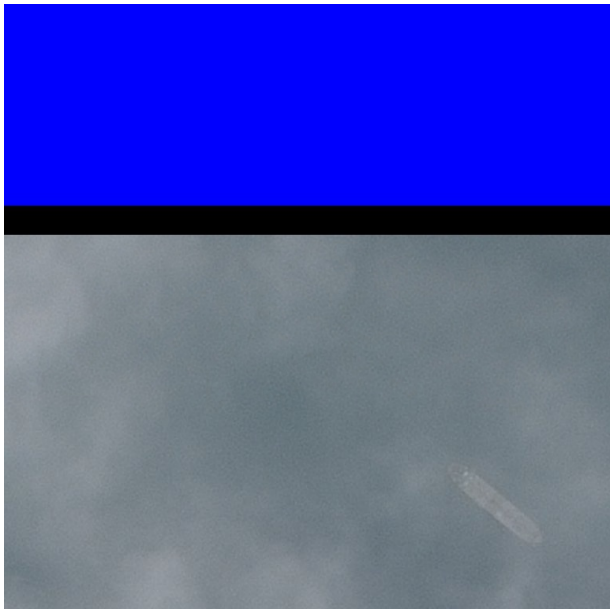

Figure 3: A Corrupt Image

images in our dataset were corrupted and led to incorrect predictions during training and validation. So, we removed these images by removing all such files, which had an approximate size of 55KB. As the number of images without ships are more than 75% of the dataset, training the model on the whole dataset lead to model predicting no ship at all.

So, we downsampled our data by including only 2000 images of those classes having more than 2000 images in the dataset. The dataset still remained skewed with images having no ship or 1 or 2 or 3 ships being the most common classes of images in the dataset. However, on this dataset, the predictions of our model were much better. We split this dataset into a training and validation dataset (80:20 split). Our final dataset consisted of around 7000 images in training set and around 1800 images in Validation set. We also created two separated datasets, one without any images having no ships and one with those images and checked the predictions. The predictions in the former case were more refined. We did separate Preprocessing for training YOLO model. We converted run length encoding to a bounding box and the converted the bounding box in YOLO format.

A bounding box is described by four numbers. xc,yc which represent the coordinates of the centre of the bounding box and h, w which represent the height and the width of the bounding box. Hence a bounding box is a tuple of numbers [xc, yc, h, w]. However the YOLO format for a bounding box is slightly different. Instead

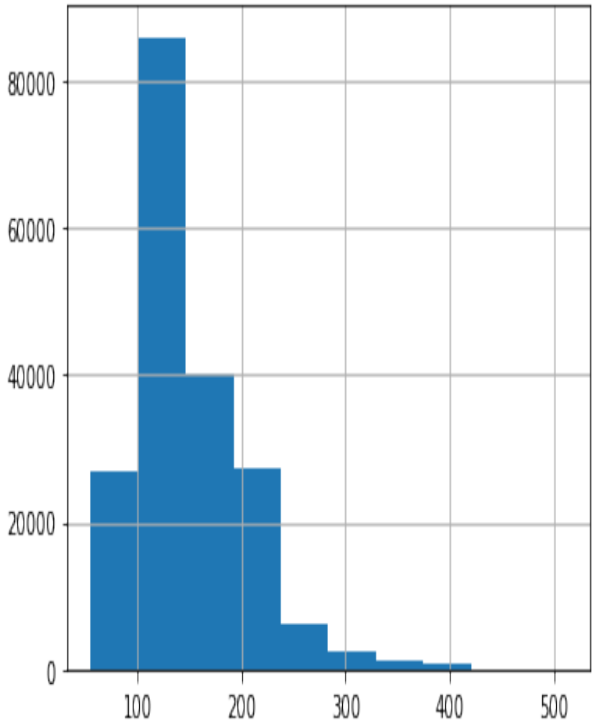

Figure 4: Images vs File size in KB

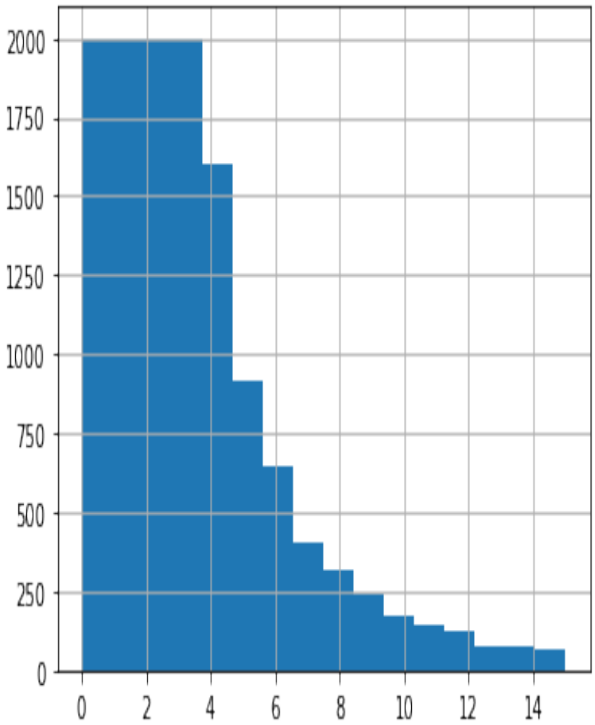

Figure 6: Down sampled Data

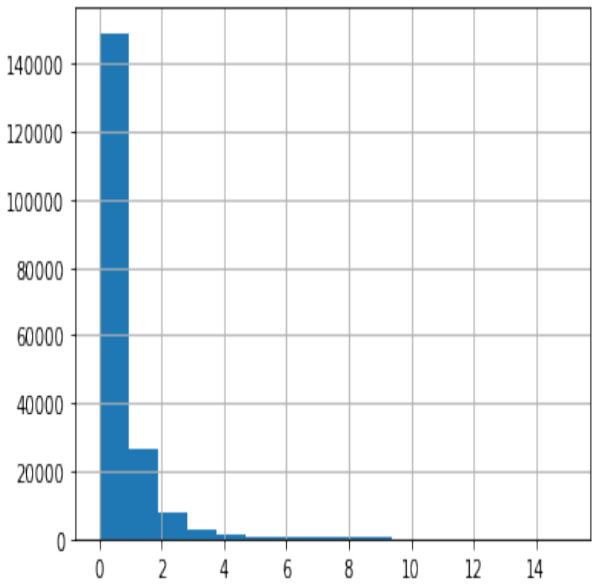

Figure 5: Images vs ships

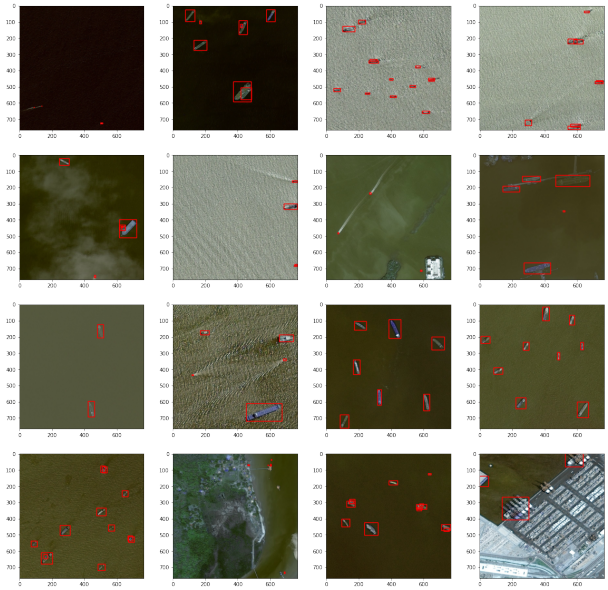

Figure 7: Bounding Boxes

of giving actual coordinates in terms of indices in 2d Array, we consider the top left corner as (0,0) and bottom right corner as (1,1). Hence we report xc as $xc* = \frac{xc}{W}$, yc by $yc* = \frac{yc}{H}$, h by $h* = \frac{h}{H}$ and w by $w* = \frac{w}{W}$ where image is of shape (W,H). We also append the class ID to the left in bounding box in YOLO format, Hence the final YOLO format of a bounding box is [Class ID, xc*, yc*,h*,w*].

ClassID	xc	yc	h	w
0	0.5	0.04	0.09	0.06
0	0.39	0.0	0.1	0.12
0	0.54	0.10	0.21	0.13
0	0.25	0.32	0.3	0.3

Table 2: Sample YOLO format

4.4. ER Diagram

The relationships between the various attributes (given and derived) is given in the ER diagram shown in Figure 8.

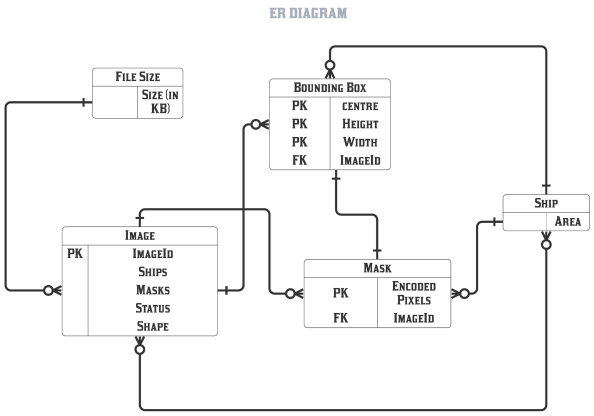

Figure 8: ER Diagram

4.5. Some Other Uses of Our Dataset

Our datasets can also be used for tracking ships by the shipping companies. There are various containers that shipment companies want to track.They can use this datasets to do so. However, tracking requires some additional work apart from detection. Nonetheless, it will be very helpful in case of emergencies such as war time, blocked traffic, etc. In addition, the datasets can also be used to check the ship traffic at ports.

5. Methods and Results

5.1. Models

We made the following two models.

5.1.1. U-net

We formed a modified U-net model with two extra layers of Average pooling and Gaussian Noise. We also kept the padding as same in between Convolutional layers instead of valid. The architecture of the model can be found here Image U-net Model. We trained the model from scratch on two different loss functions and compared the corresponding predictions.

5.1.2. YOLO

We loaded a pretrained YOLO model (Darknet) available on githubWe changed the configuration file of the model. Set the number of filters of the convolutional layer before YOLO layer to 18 from 255 and changed number of classes to 1 from 80. We changed the total number of training epochs to 4000 from 50500. Then we trained the model on our dataset in Colab on Tesla T4.The model had to be trained 3 times due to kernel getting disconnected at iteration number 1000, iteration number 2500 and iteration number 2800.

5.2. Loss Functions

5.2.1. U-net

1. **IoU Loss**: As the name suggests this loss is the ratio of size of intersection between Prediction and Ground truth to the size of union between prediction and ground truth. Consider a prediction area A by the model and let B be the corresponding ground truth area, then the IoU of the prediction is given by
$$\frac{|A \cap B|}{|A \cup B|}$$
.Since we need to maximize the IoU, we try to minimize negative of IoU. The corresponding Loss function is know as IoU loss. The loss vs iterations is shown below

2. **Dice Loss**: Consider a prediction area A by the model and corresponding ground truth area B, then the Dice coefficient of the prediction is given by
$$\frac{2|A \cap B|}{|A| + |B|}$$
. The value of dice coefficient lies between 0-1. The higher the dice coefficient, better the prediction. So we try to minimize 1-Dice Coefficient. The corresponding loss function is known as soft dice loss.

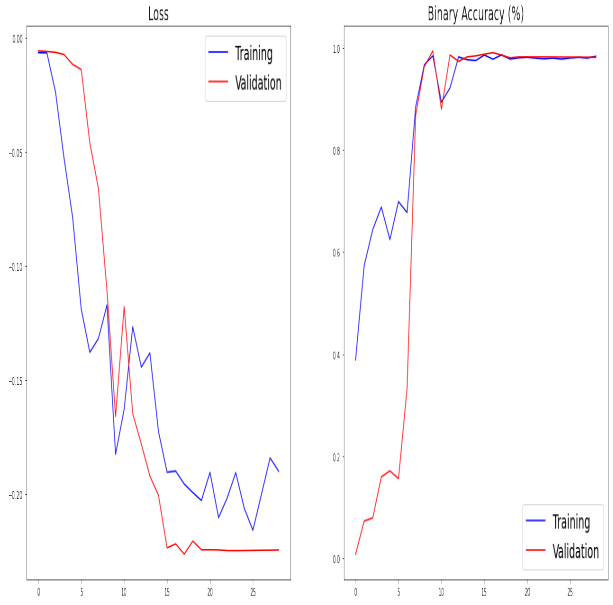

Figure 9: IoU Loss History U-net

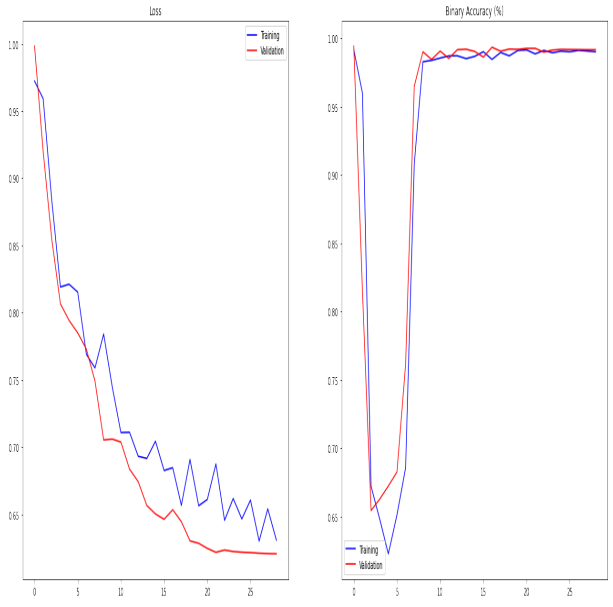

Figure 10: Dice Loss History U-net

5.2.2. YOLO

YOLO model uses IoU loss function and we have trained our model using the same loss function. The kernel disconnected 3 times during the training, so we do not have the full training loss curve. The graph for the loss function vs iterations from iteration number 2800 is shown below. The loss of YOLO started from 24 at iteration 0 and decreased to 1.6 at iteration 4000. The last trained weights can be found here YOLO Weights 4000. The Link to the darknet master folder where the image and label dataset is loaded can be found here Darknet Master

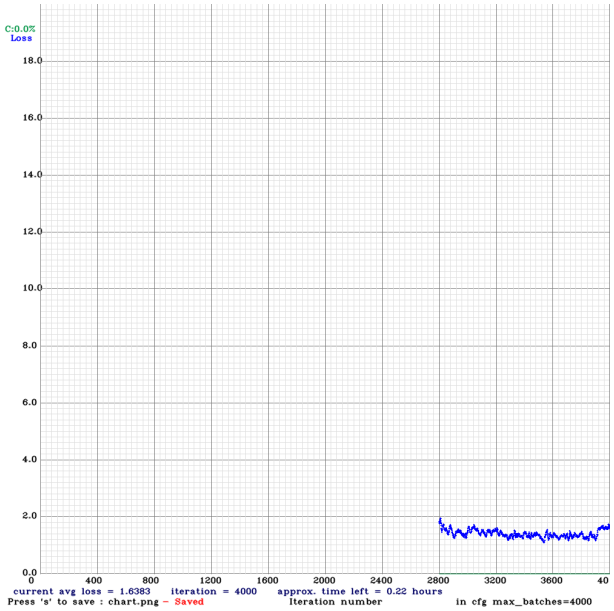

Figure 11: Loss vs iterations YOLO

5.3. Predictions

5.3.1. U-net

IoU Loss function: The binary accuracy of this model for testing dataset is 99%. However even though the model has high binary accuracy, the average IoU is small (0.62). The main reason for this is that the ships in the images are very small hence corresponding to binary accuracy no significant deviation is observed as compared to IoU. To illustrate this consider a Mask(Table 3)

and let the prediction be as shown in Table 3. The binary accuracy of the prediction will be $\frac{8}{9} = 88\%$ where as the IoU accuracy of the prediction will be 0(Lowest possible).

Dice Loss Function: The binary accuracy of the model in this case was a little bit lower (96%), however the IoU accuracy was better (0.68)

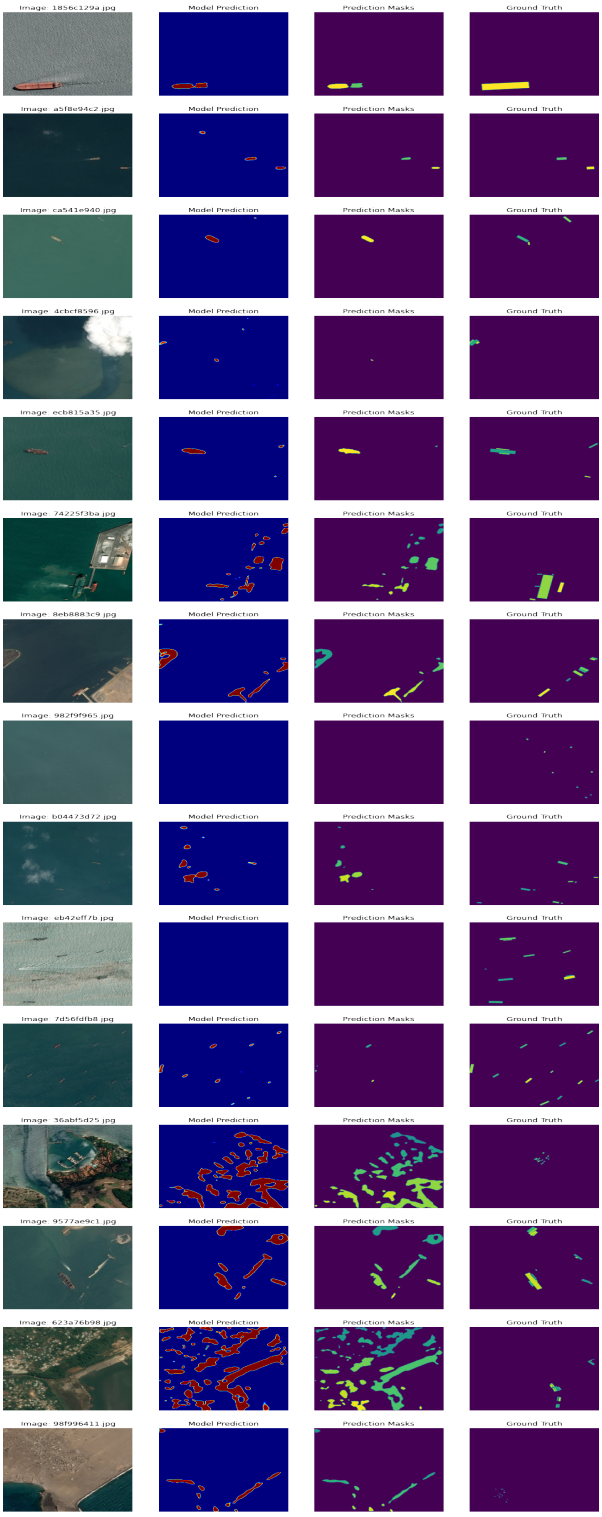

Figure 12: Predictions: Loss = IOU

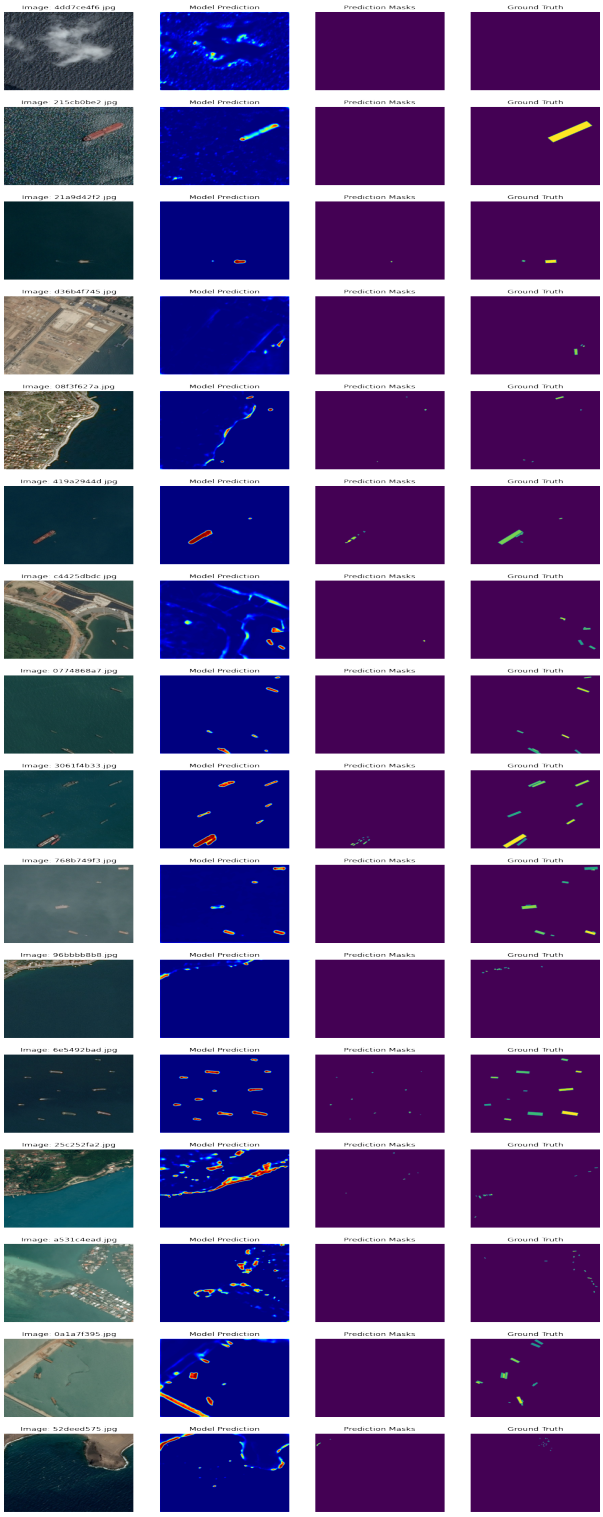

Figure 13: Predictions: Loss = Dice Loss

Mask Vs Prediction					
Mask			Prediction		
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0

Table 3: Mask and Prediction

5.3.2. YOLO

The Average precision of YOLO model on training data was 75% in terms of bounding boxes. However the segmentation accuracy was not high enough which was expected as YOLO does not segment the image rather predicts a bounding box around the object.

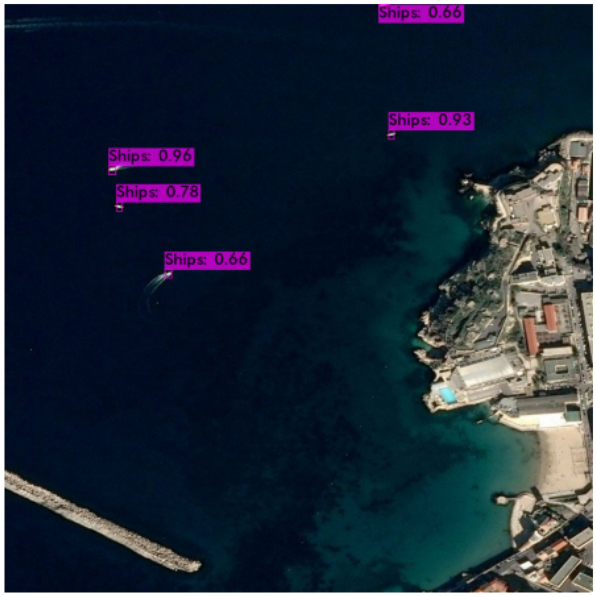

Figure 14: YOLO predictions

6. Novel Aspects and Ideas

The U-net model developed for this assignment had some novel features including a Gaussian noise layer and a net scaling layer at the end. Also, the dimensions of the various layers were chosen carefully so that the model predicts accurately and quickly and the kernel does not run out of memory during training. The model was easy and faster to train than the traditional U-net model and it also predicted acceptable results. We also hypothesise that combining the results from YOLO model and U-net model will produce better results, The combining is as follows:

1. Choose the centre pixel predicted as the centre of the bounding box by YOLO model.
2. In the mask predicted by U-net model select all the pixels which are connected to the centre pixel (a simple graph connectivity algorithm will be used here).
3. Do this for all bounding boxes. The resulting mask produced is the prediction of the combined model.

We were unable to implement such a model and test its accuracy due to increased complexities and limited coding abilities of the group, but we hypothesise that the results will be better. This hypothesis is based on the observation that the YOLO model is better at identifying the object location(hence centre will be accurate) and the U-net model is better in segmentation(and hence the run of the pixels will be accurate).
We also developed a detector model from scratch whose architecture was simple linear (a decoder and encoder like one for PCA), but the model failed in providing viable predictions.

7. Conclusion and Future works

Model	Binary Accuracy	IoU Accuracy
U-net	99%	62%
YOLO	-	75%

Table 4: Models and Accuracies

We developed an image segmentation model that segmented the image into ship containing and non-ship containing segments with high accuracy(Binary accuracy 99%, IoU accuracy 68%). We also developed a ship detector which detected the ships in an image and produced bounding boxes around them with an AP score of 75%. The models can be made more accurate by training on kernels which support batches of larger size and for larger number of iterations. The loss functions can also be modified for improving the rate of convergence of the U-net model. Instead of using YOLO for transfer learning one can use segmentation models like mask R-CNN, especially detectron 2 by facebook for achieving better accuracy.

8. Project Code

The codes of the project are written in a markdown file which then we exported to pdf and have uploaded on

drive, The link of the pdf can be found here
YOLO Training codes U-net and YOLO Training.

References

[1] Joseph Redmon et. al (2016) *You Only Look Once*: Unified, Real-Time Object Detection. URL: YOLO

[2] Ross Grischik et. al(2014) Rich feature hierarchies for accurate object detection and semantic segmentation. URL: R-CNN

[3] Olaf Ronneberger et. al(2015) *U-Net*: Convolutional Networks for Biomedical Image Segmentation. URL: U-Net

[4] Alexy et. al (2018) *Darknet*:Yolo v4, v3 and v2 for Windows and Linux. URL: Alexy Darknet

[5] MatterPort et. al *Mask R-CNN for Object Detection and Segmentation*: MatterPort R-CNN

[6] Scott Mader *Baseline u-net model* Baseline U-Net

[7] Siddharta *Airbus EDA*. URL: Airbus EDA

[8] Aakarsh Yeliseti *Detectron Tutorial*, URL: Detectron Tutorial

[9] Ravi Kaushik et. al (2019): Image Segmentation Using Convolutional Neural Network. URL: Ravi Kaushik

[10] Yuhen et. al, Image Segmentation Algorithms Overview, URL: Yuheng

[11] *Fizyr Retinanet*:Tensorflow Keras implementation of RetinaNet object detection, URL: Retina Net

[12] Yuxin Wu and Alexander Kirillov and Francisco Massa and Wan-Yen Lo and Ross Girshick, Detectron2(2019): URL: Detectron Github