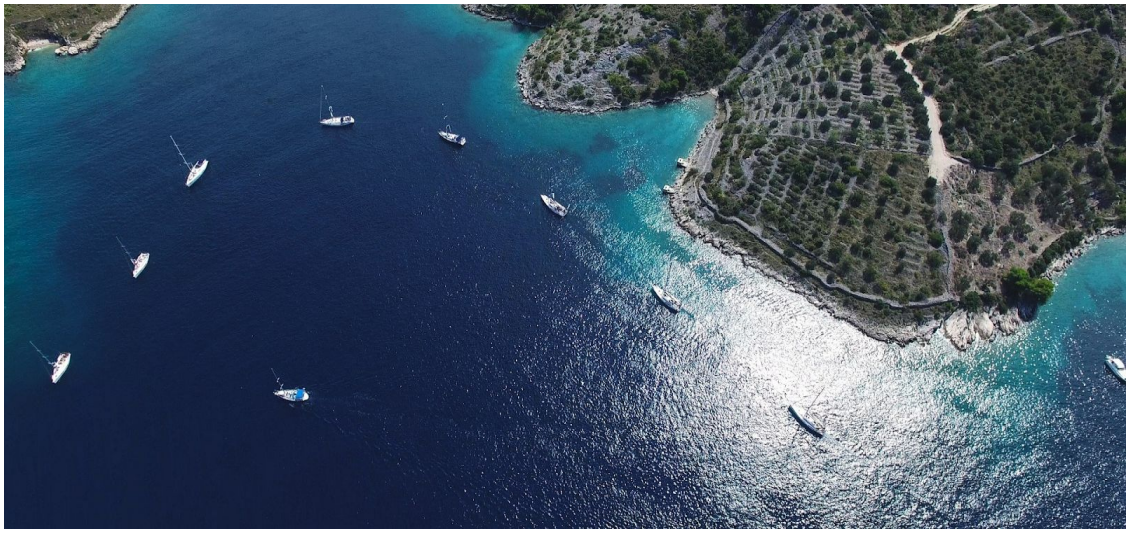
Detecting Ships from Satellite Images



EE599:Deep Learning

Darun Jayavel, Omer Solakli, Arushi Kapurwan

Date: 05.07.2019

INDEX

1. ABSTRACT

The safety and security of the landless areas are one the most challenging fields in the current era and it becomes important to establish a system that can monitor illegal activities taking place at inaccessible areas and take necessary actions to conquer the unlawful acts.

Our proposal takes a stance on the safety aspect of the sea by detecting the ships in an image and placing an aligned boundary box around it, thereby detecting and localising the ship. The model present here takes the input as the satellite image and tries to return an image with bounding boxes of all the ships in that image using a convolutional neural network architecture.

2. INTRODUCTION

2.1 PROBLEM TYPE,STATEMENT AND GOALS

The past century has observed an immense increase in the number of marine relevant activities. With the increase in the demand of food, medical care and boom in the research of marine species there has been an exponential increase in the number of ships in the sea. The ships increment is directly proportional to the number of illegal activities at sea. Some of them comprise of illegal fishing, drug trafficking, illegal cargo movements and devastating ship accidents. To combat such pursuit it becomes important to design a mechanism that can counter such activities and make an effective contribution to the surveillance aspect of the sea.

Many companies and agencies invest millions of dollars for the protection of the marine industry and it becomes useful to adapt the deep learning neural network architecture into the prediction and analysis of the images and objects.

The goal of the project is to design a model covering the safety parameter of the marine. The proposal is to detect the ships in the satellite images ranging from a minute or timid picture to the enlarged one. The model and its architecture will contribute in image detection and predict the ships in any image by boundary boxing it.

The study explores that several machine learning methods are used to tackle semantic segmentation problem for ships occuring in satellite images. Semantic image segmentation is a dense prediction problem where the goal of the process is to classify each pixel corresponding to its class [1]**.** The usage areas of the solution ranges from autonomous cars, medical image diagnosis and geospatial information analysis.

There is a belief that with our plan there can be a major contribution to comprehensive monitoring; support to the maritime industry can be done to anticipate threats, trigger alerts and improve the efficiency at sea.

2.2 LITERATURE REVIEW

Some of the publications and the research papers used in the project are:

a) Keras Documentation

b) Stack Exchange and Quora for hyperparameter tuning.

c) U-Net Architecture Research papers.

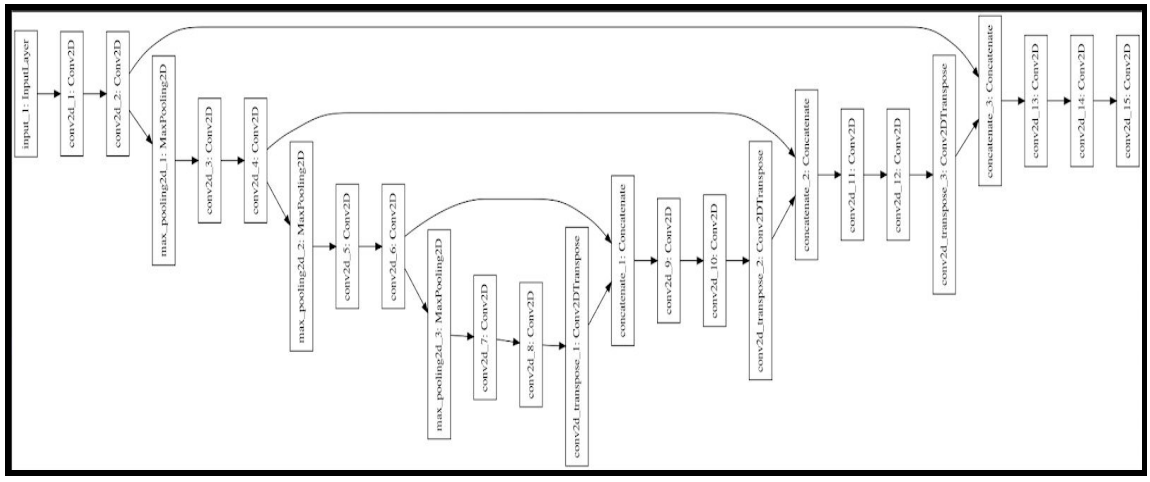
2.3 PRIOR AND RELATED WORK

A thorough understanding of the data as contributed by Kaggle through the courtesy of Airbus was conducted. The images were understood to design the later stages of the project.

2.4 OVERVIEW OF APPROACH

The idea is to experiment with different U-Net architectures and be able to classify each pixel correctly which uses a Fully Convolutional Network. The Pixel level image classification will be utilized through U-Net. The layers of U-Net being Contraction, Bottleneck and Expansion will predict each pixel's class and contribute to Image Segmentation.

The following architecture represents the model that will be trained to produce the predicted images with bounding boxes:



3. IMPLEMENTATION

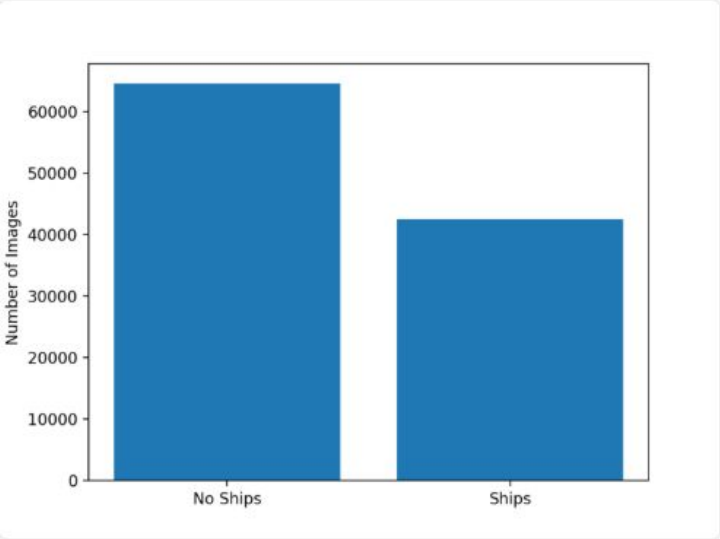
3.1 DATA SET

The data set is collected from Kaggle and comprises of the training and test data provided by Airbus. The data is in the form of images in the jpeg format. The composition of the data is as follows: 100k+ images for the training data and 15k+ images for the test data.

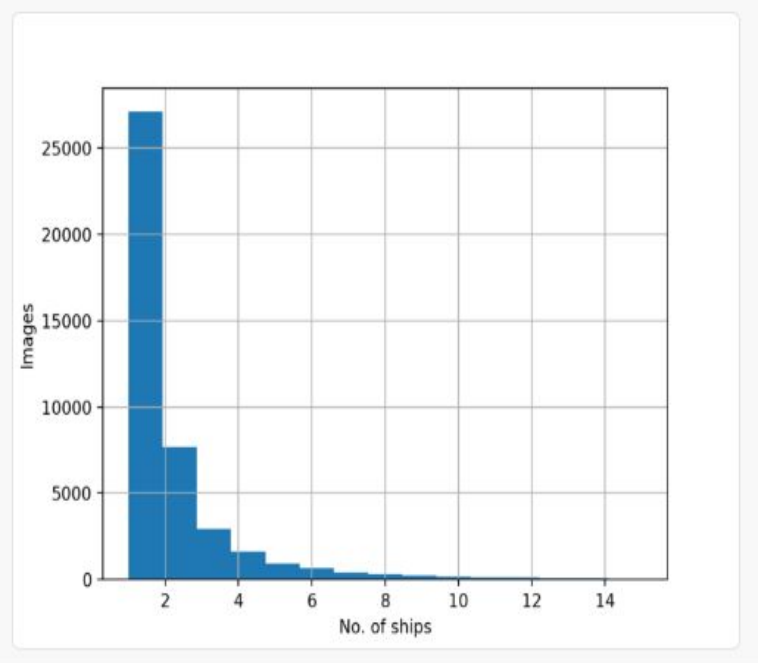
3.2 DATA PREPROCESSING

When we look at the data presented by Airbus, we saw that there are many samples without having any ships. The ratio of samples having ship and not having any ship was 4:1 initially, it was reduced to the ratio of 3:2. The final dataset consists of 109,000 training images with 60% of images having no ships and rest 40% having one or multiple ships.

The following histogram represents the Number of images with Ships and No-Ships:



The following histogram represents the total number of ships corresponding to the number of images with that set of ships(as in the training data):



The following table represents other features about the data:

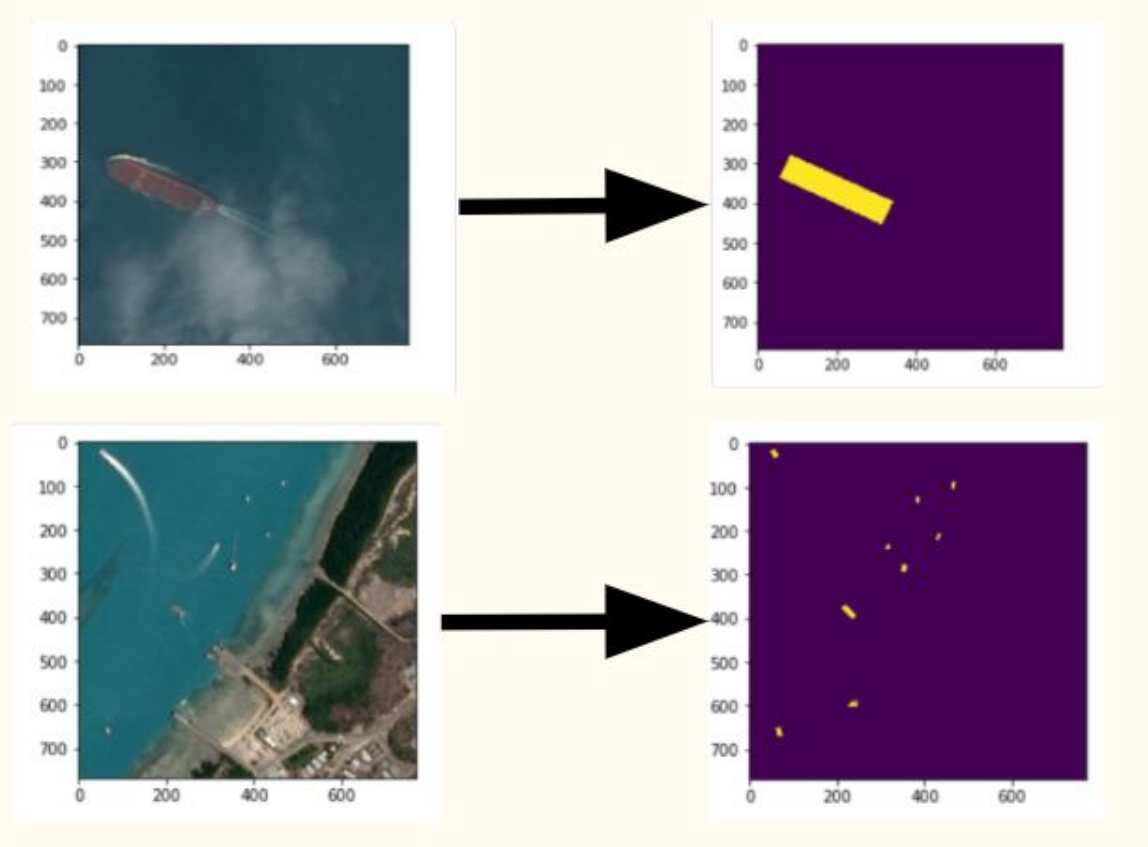
TOTAL COUNT (with ships)	42536
MEAN	1.92
STANDARD DEVIATION	1.90
MINIMUM NO. OF SHIPS	1.00
FIRST QUARTILE (25%)	1.00
SECOND QUARTILE (50%)	1.00
THIRD QUARTILE (75%)	2.00
MAXIMUM NO. OF SHIPS	15.00

Each image has a corresponding field called Encoding Pixel. This encoding pixel contains the information of the bounding box of all the ships in a given image. For example,

00021ddc3.jpg; 139644 2 140408 6 141174 9 141942 9 142711 6 143479 2

Here for the image '00021ddc3.jpg' we have its corresponding encoded pixels of the bounding box. Decoding this will give the bounding box of the ship.

The following diagram represents the masked images from the training data images:



For Data Augmentation, we used the Keras Image generator function and a custom data augmentation function. It consists of performing different transformations on existing images in order to obtain images that make the machine learning model more robust. First, we normalize the images by dividing each pixel by 255 (maximum range of the pixel value) and additionally, we flip the image, increase the brightness and resize it to reduce complexity and computational power.

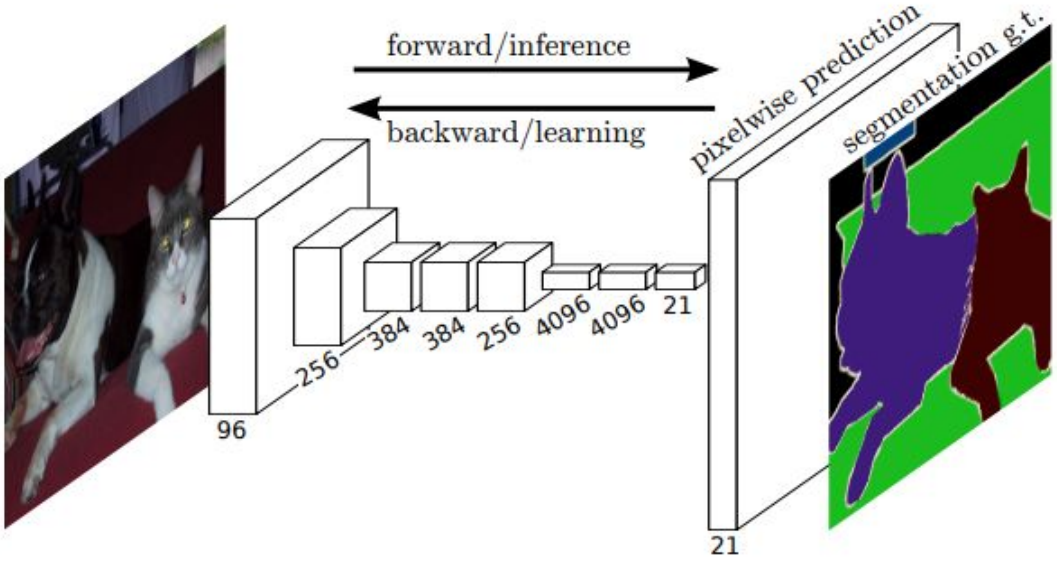
3.3 MODEL SELECTION

3.3.1 U-Net MODEL

As we mentioned before, we used U-Net to segmentate the ships from the background. The architecture consists of a contraction, bottleneck and expanding path that enables localization of the feature maps where the information on what is present in the image is learnt. Then the image is reconstructed and each pixel is classified using skip connections between the contraction path and expanding path. In the expanding path the pooling layers are simply changed by upsampling layers to increase the resolution of the output.

The network is Full Convolutional so that the output is also an image where the output image has the same dimensions as that of the input.

The high-level visualization of the network is presented as follows:



3.3.2 MODEL IN DETAIL

Each layer in the contraction path consists of 2-3x3 convolution layers followed by a ReLu layer and a 2x2 max pooling operation afterwards. There is a bottleneck layer between the contraction and expanding path which consists of 2 convolutions without max pooling. The expanding path starts with transposed convolution layer where the image size is doubled for reconstruction and is followed by concatenation (skip connection) with the corresponding convolutional layer in the contraction path.

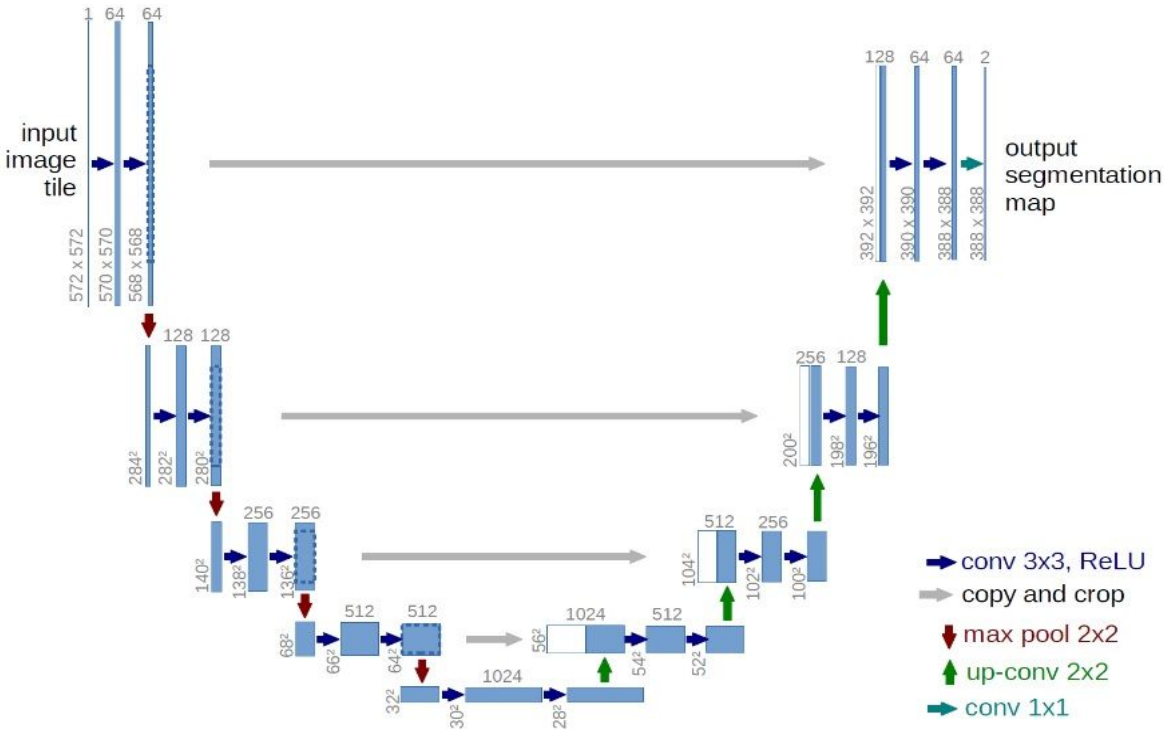
Convolutional layers in the model learns what is present in the image by going deeper in each layer. This is accomplished by increasing the number of channels. The max pooling layers are used to decrease the

computational load of the system-lowering the resolution of the images and discriminating the classes from a low-resolution point of view to get a better classification.

The expanding path (also called upsampling path) is used for extracting the positional information of the data to generate the image back with correct labels. As we mentioned above, transposed convolutions upsample the image dimensions by a factor of 2. The convolution layers followed by it reduces the channel gradually to 1 in order to regenerate the image.

The transposed convolution layer is essential because of the fact that it enables learnt upsampling rather than using a predefined method such as Nearest Neighbor Interpolation or Max Unpooling. They are simply the opposite of the convolution operation where the filter weights are multiplied with the single value of a low resolution output, hence generating a one-to-many relationship.

The original U-Net architecture is as follows:



U-Net model is considered computationally less expensive than the networks other than convolutional image segmentation networks where the model is trained using a sliding window method. It provides a good segmentation map by combining the local information from the downsampling path with the contextual information of the upsampling path.

3.4 TRAINING PROCESS

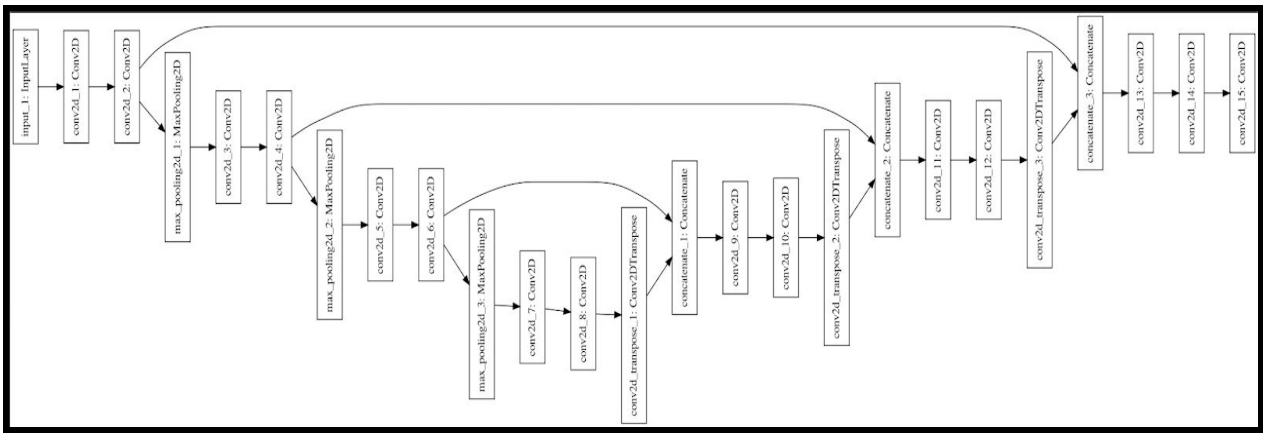
To get the best working model in our use case we tried different U-Net architectures and we tuned the hyperparameters accordingly. The model is trained using Amazon p2.8xlarge EC2 instance using Tensorflow with CUDA.

3.4.1 INITIAL MODEL

We stick loyal to the original model in terms of the number of down blocks and up blocks, which is 3 in our case. For each downsampling block the convolution layers (3x3 convolution) are used and the number of channels is increased in each block by 2 (8, 16, 32, 64). For the upsampling blocks the number of channels are decreased by 2 which has 1 channel output in the output layer (64, 32, 16,8,1).

All the convolutions are used with full padding to be able to capture the instances on edges. For this model, the original image sizes are used without any resizing.

Our baseline model is presented below.



The initial model took 4 hours to train per epoch and yielded 0.05 IoU which was far below than what we expected and it seems that the model was able to predict the instances having less noise (clouds, storms, lands) well but the instances with complex background and small ship size was classified incorrectly.

Alternatively, the batch size was reduced leading to the decrease in learning rate and increase in momentum and the model still did not learn properly. The problem seemed to be the depth of the model (channel size). The filters weren't able to capture the features of the ships when the noise is present in the images. Hence, a deeper model was required to resolve the issue.

3.4.2 SECOND MODEL

When the original U-Net paper was released, which was around May 2015 the batch normalization had been used only for 3 months thus the original paper did not consider using batch normalization which leads to an increase in the convergence of the deep learning models. Hence we used batch normalization after each up and down blocks to increase convergence of our models.

Furthermore,we needed deeper model to capture what is present in the image and to achieve that, we increased the filter numbers for each convolutional layer. In order to be more computationally and memory efficient, the image size had to be decreased.

In the second model the image size is reduced to 384x384 from 768x768. The model has the same number of down blocks and up blocks, 3 in the second model and in the original model. For each downsampling block, the convolution layers (3x3 convolution) are used and the number of channels is increased in each block by a factor of 2 (12, 24, 48, 96). For the upsampling blocks, the number of channels are decreased by 2 having 1 channel output in the output layer (96, 48, 24,12,1).

The second model is trained with different batch sizes spanning [4 8 16 32] and with the learning rates spanning [0.0001 0.0005 0.0010 0.0040]. Adam optimizer and SGD with momentum is used for this model.

The second model took around 1 and a half hours to train per epoch and it yielded around 0.20 IoU after 2 epochs which was a significant increase compared to the initial model.

3.4.3 LOSS FUNCTION SELECTION

The original U-Net model uses stochastic gradient descent optimization method using an energy function computed by pixel-wise soft-max over the final feature map combined with the cross-entropy loss function. In our model, the loss function has been implemented in the form of dice loss function, Intersection over Union Loss function and cross-entropy loss function.

A loss function is defined as the measure of how good a prediction model does in terms of being able to predict the expected outcome. It is also referred as the cost function that maps an event or values of one or more

variables onto some real number intuitively representing the “cost” associated with the event.

Loss function is typically used for parameter estimation and the function used is the difference between the estimated and true values of the instance of the data. Some of the loss functions are determined by its estimates and sometimes by the output of a random variable X.

3.4.3.1 DICE LOSS FUNCTION

The Sorensen Dice coefficient is a statistic used to gauge the similarity of two samples. There are several other names that are used to describe this function and some of them include Sorensen-Dice Index or simple Sorensen Index or sometimes as Dice’s Coefficient. Other variations include the “similarity coefficient” or the index as the Dice Similarity Coefficient.

Given two sets as X and Y, the Dice Loss Function is defined as:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

The Loss Function can also be defined in terms of False Positive(FP), False Negative(FN) and True Positive(TP) as the following:

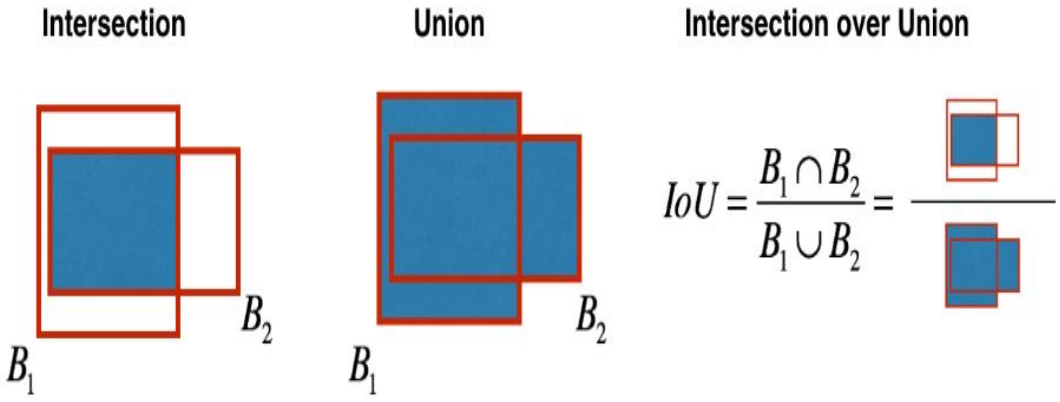
$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

It is different from the Jaccard Index as the index only counts true positives once in both the numerator and the denominator. DSC is a quotient of similarity that ranges between 0 and 1.

A complete overlap produces the result as 1 while a complete non-overlap produces 0 as the result.

3.4.3.2 INTERSECTION OVER UNION LOSS FUNCTION

The Intersection over Union metric is involved in computing the accuracy and the loss of our models.



Let B1 be the ground truth mask of a ship and let B2 be the predicted mask. When the matching is perfect, the metric value is 1 while for full imperfect matching the metric value is 0 as it is the ratio of the common overlapped image to the combination of the images along with its common overlapped areas.

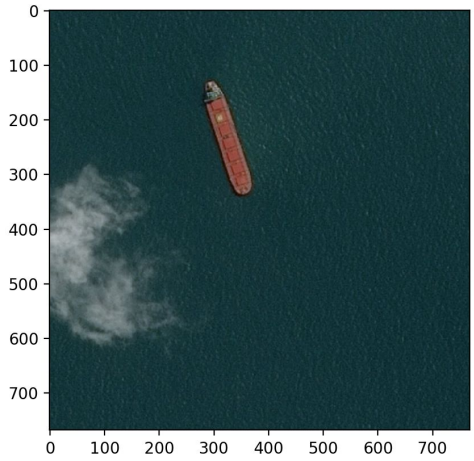
We use the above Dice Coefficient and IoU (Intersection over Union) loss function as the output metric to define the goodness of the model. The model is good if it has a high Dice coefficient or if the value of IoU is close to 1.

4. RESULTS

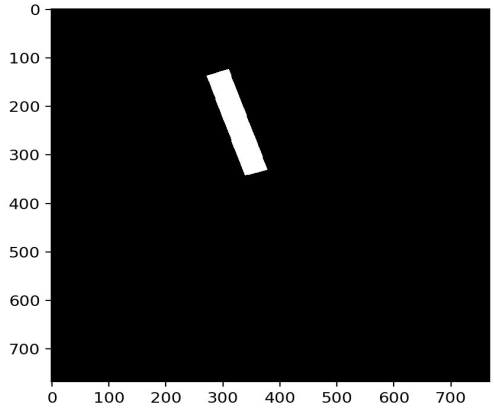
Below, the way in which the model classifies some of the instances are presented for each model. The first image is the original training image, the second is the ground truth and the third is the output predicted by the model.

For the initial model, the results are presented as following:

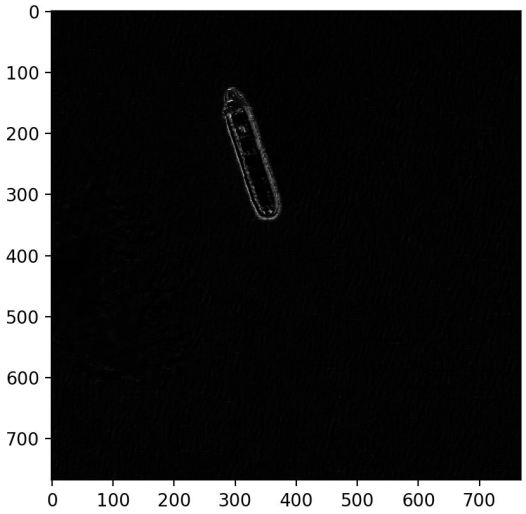
a) ORIGINAL IMAGE:



b) ACTUAL BOUNDING BOX:

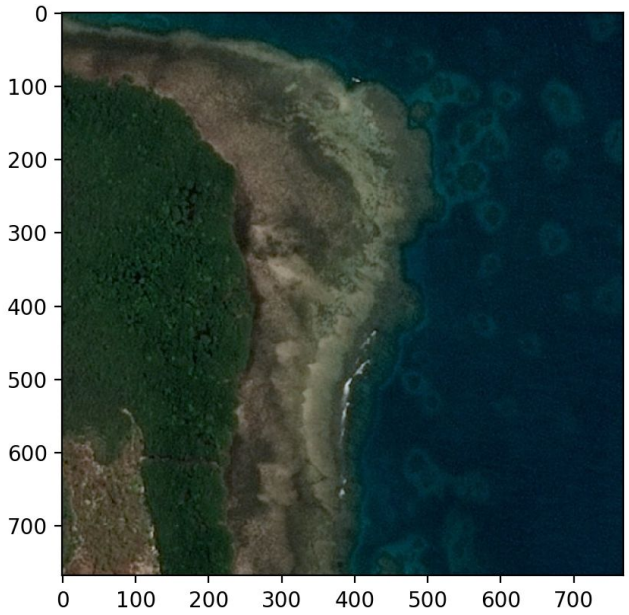


c) PREDICTED BOUNDING BOX:

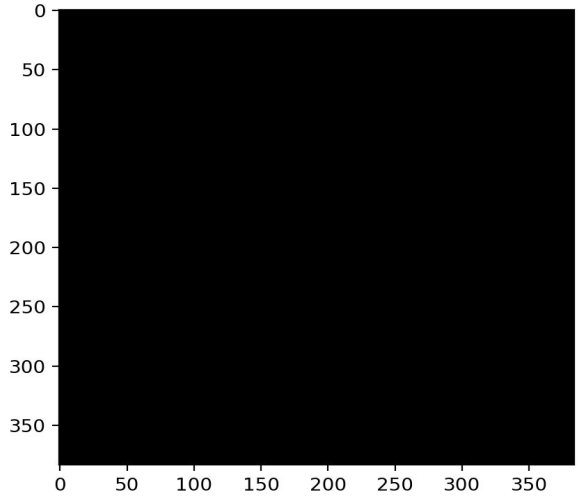


For another image the following results are presented:

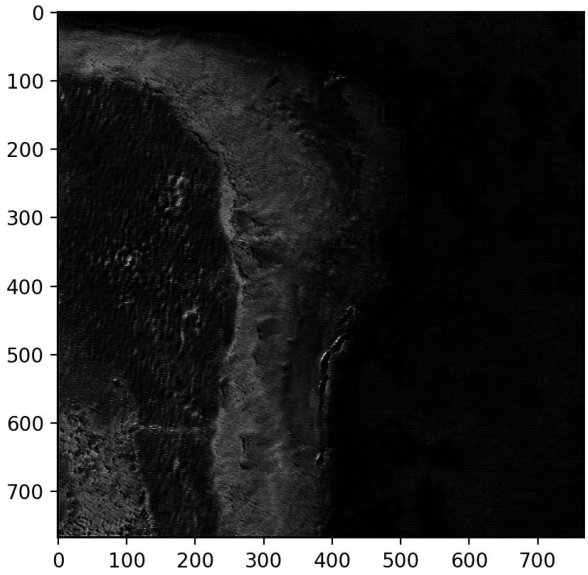
a) ORIGINAL IMAGE:



b) ACTUAL BOUNDING BOX:



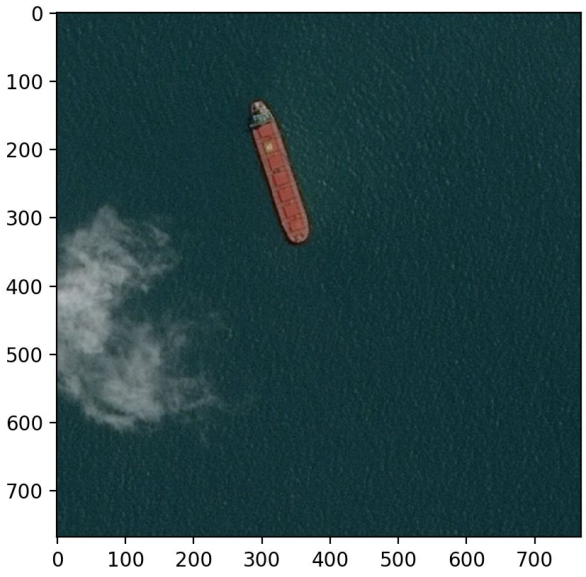
c) PREDICTED BOUNDING BOX:



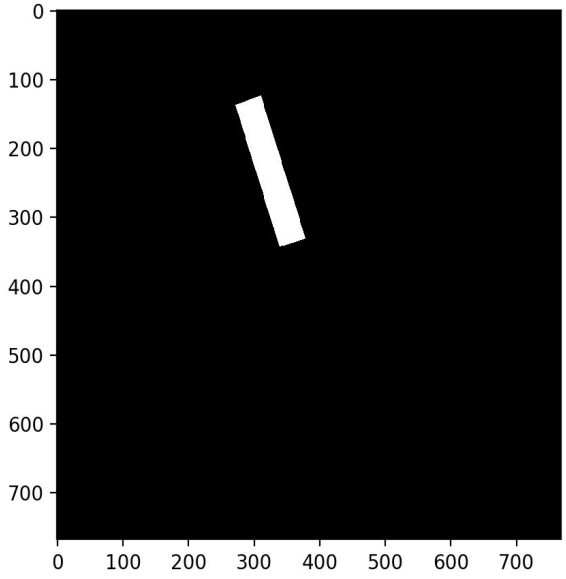
It is clear that the model was able to regenerate the images in a good fashion and was able to detect the edges. But the problem asks for not only edge detection but also filling the edges with the corresponding class. Further, for the second image it is not able to detect the background because there are lots of noise in the image (the land is considered as noise because there can't be any ship in the land).

For the improved model, the results are presented. First image is the sample, second one is the ground truth and the third one is the predicted by model.

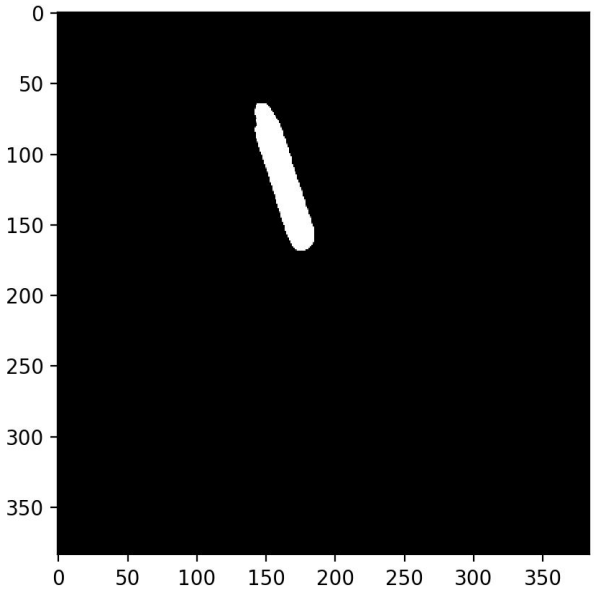
a) ORIGINAL IMAGE:



b) ACTUAL BOUNDING BOX:



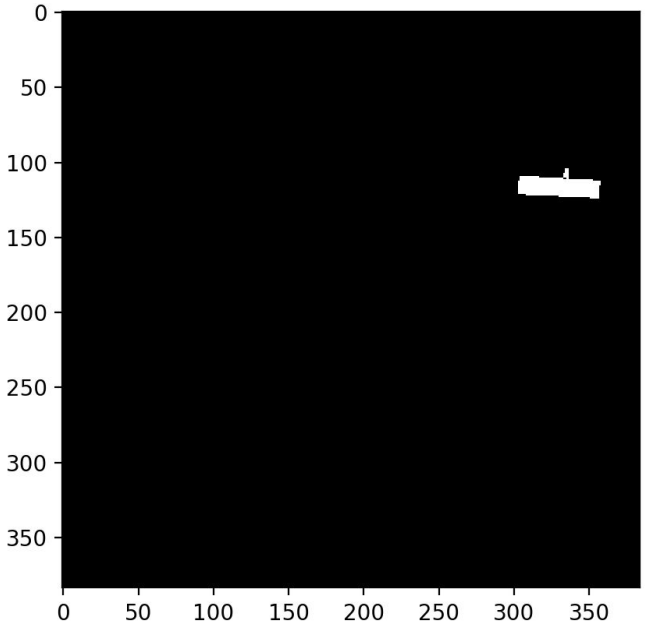
c) PREDICTED BOUNDING BOX:



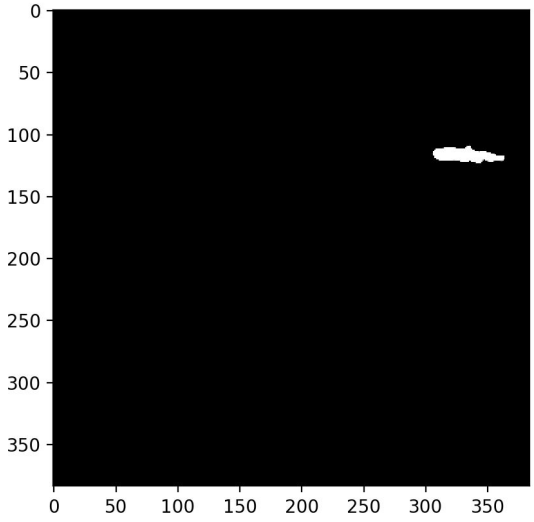
The results of following image are as follows:

a) ORIGINAL IMAGE:

b) ACTUAL BOUNDING BOX:



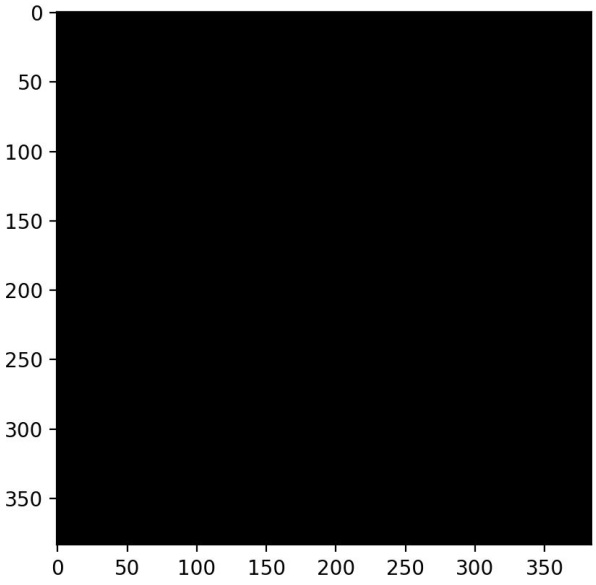
c) PREDICTED BOUNDING BOX:



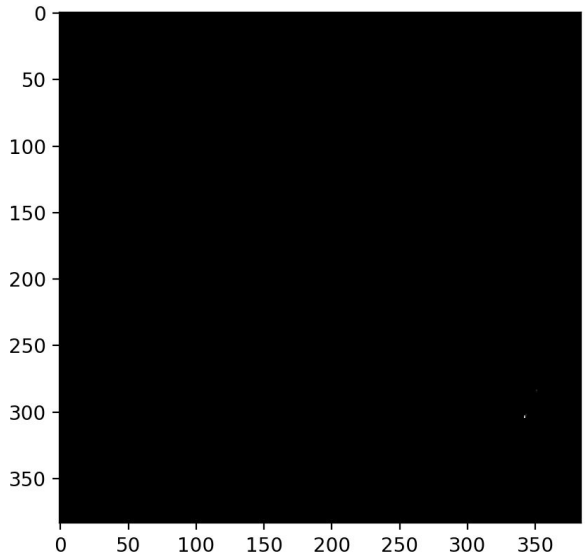
It can clearly be seen that the second model outperforms the first one. The first model was able to capture edges for the instances having a clearer image. Where as second model is able to fill the box and achieve semantic segmentation. Let's explore how it works with the image with noise.

a) IMAGE WITH NOISE:

b) GROUND TRUTH IMAGE:



c) PREDICTED BOUNDING BOX:

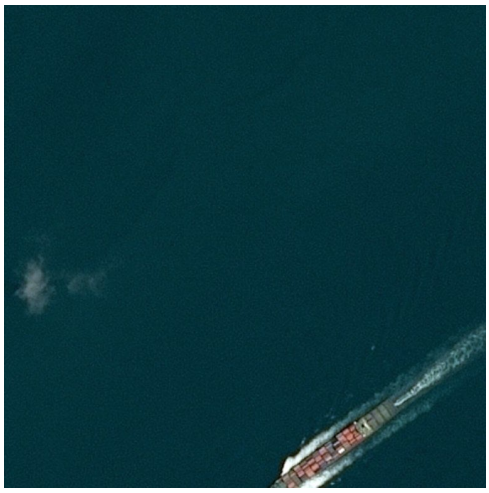


Looking at the image, the ground truth and the prediction of the second model is able to segment much better than the first one.

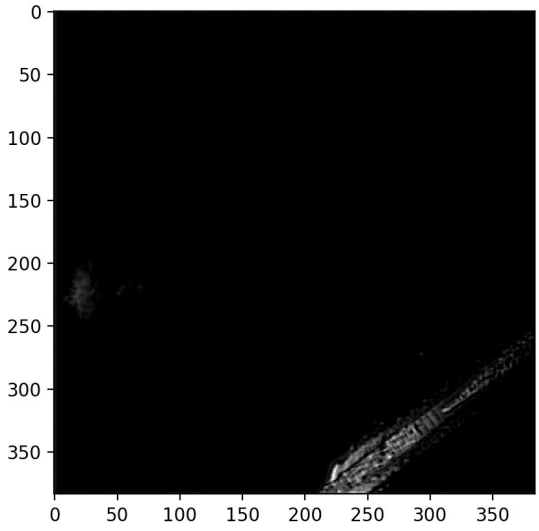
The IoU score for the first model was 0.05 where as the IoU score of the second model was 0.20.

Also, to be able to capture how model learns we present the output of each block (output of the intermediate layers) where a filter selected randomly from each block and the output is plotted.

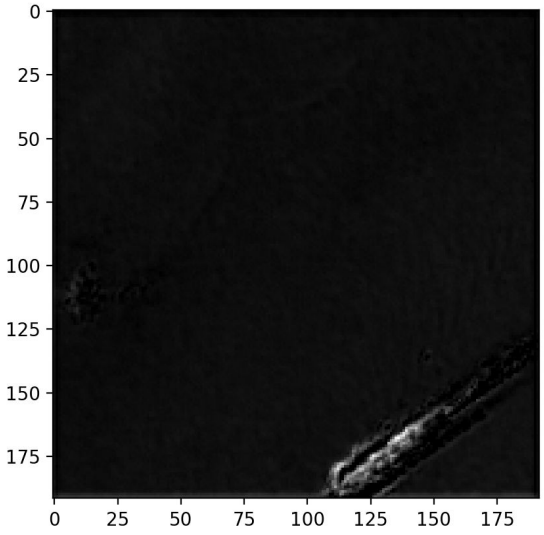
a) ACTUAL IMAGE:



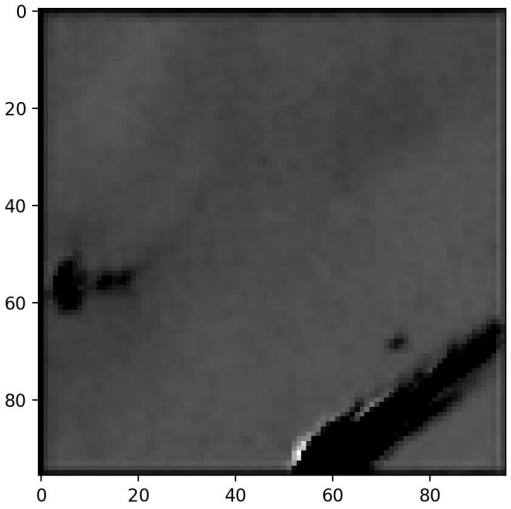
b) OUTPUT OF THE 1ST DOWN BLOCK:



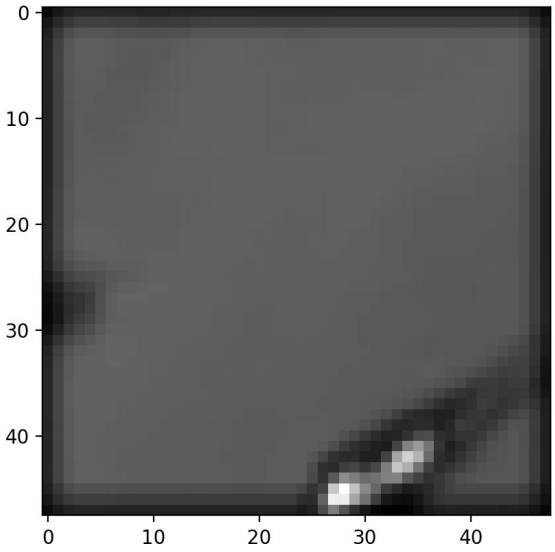
c) OUTPUT OF THE 2ND DOWN BLOCK:



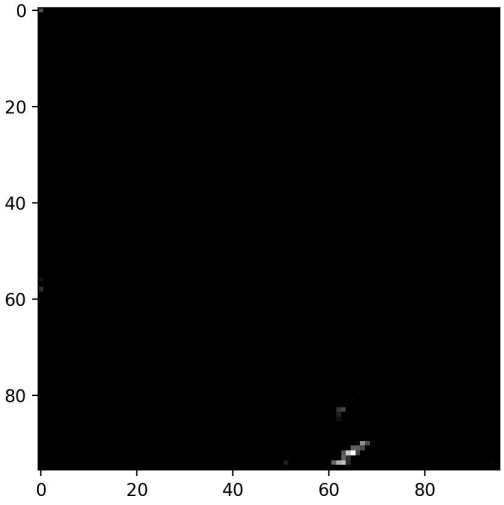
d) OUTPUT OF THE 3RD DOWN BLOCK:



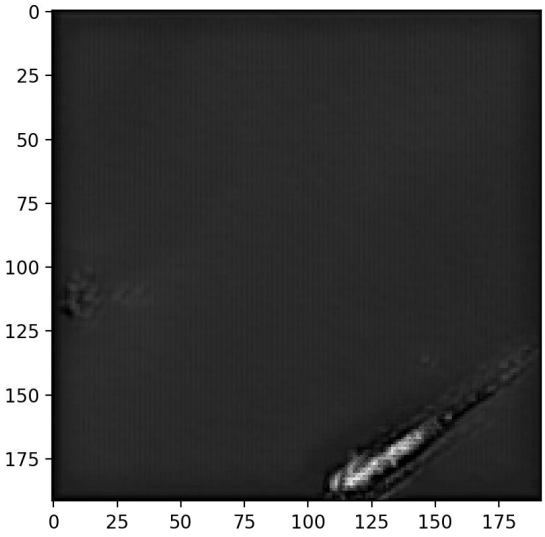
e) OUTPUT OF THE 1ST UP BLOCK:



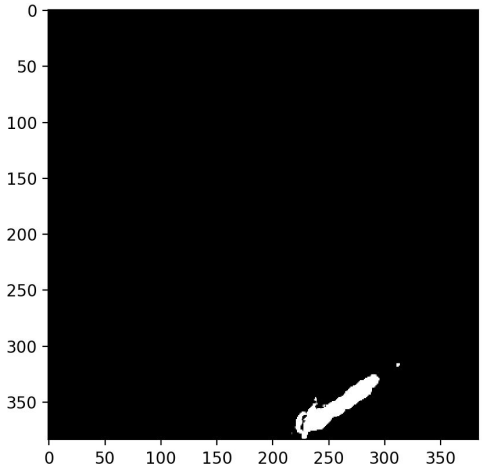
f) OUTPUT OF THE 2ND UP BLOCK:



g) OUTPUT OF THE 3RD UP BLOCK:



h) FINAL OUTPUT:



5. SUMMARY AND CONCLUSIONS

In this project we tried to tackle semantic image segmentation of ships from satellite images. There are different solutions for the problem in the literature and it is still an ongoing extensive research to find better solutions. We used deep learning techniques to implement and hence we used U-Net model which is a Fully Convolutional model.

The improved second model with batch normalization, increased number of channels and resized image can detect the ships better than the first. From the results above, we can infer that the improved model is able to predict and localise the ship from a given full color RGB image.

6. FUTURE WORK

For the future research work, an improvement to the model can be made by fine tuning the hyper parameters of the model. Also, the number of filters can be increased in the model which might help in improving the model. However, the increase in the number of filters comes with a high computation cost.

Furthermore, we see that the outputs of the intermediate layers are presented as a checkerboard artifact because of the overlap of the transposed convolution (upsampling) layers. This can be tackled using more channels and original image size but this comes with high computation cost. A possible improvement to the model could be using Nearest Neighbor Interpolation followed by a convolution layer for upsampling [8].

7. REFERENCES

[1] Dense Prediction on Sequences with Time-Dilated Convolutions for Speech Recognition by Tom Sercu and Vaibhava Goel,2016.

[2] "Airbus Ship Detection Challenge"
https://www.kaggle.com/c/airbus-ship-detection/data

[3] "SAR ship detection using sea-land segmentation-based convolutional neural network" Yang Liu, Miao-hui Zhang, Peng Xu, Zheng-wei Guo.

[4] "EdgeFlow: A Technique for Boundary Detection and Image Segmentation "Wei Ying Ma and B. S. Manjunath.

[5]"How to do semantic segmentation using Deep Learning"
https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef, James Lee, May 3 2018

[6] U-Net: Convolutional Networks for Biomedical Image Segmentation

By Olaf Ronneberger, Philipp Fischer, Thomas Brox,18 May 2015

[7] Understanding Semantic Segmentation with UNET-
https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[8] "Deconvolution and Checkerboard Artifacts" , Chris Olah, Oct. 17 2016

https://distill.pub/2016/deconv-checkerboard/