Airbus Ship Detection Challenge

Bazsó Sándor (FYKXN2, sanyi12b@gmail.com), Fényes Balázs (CAU3ZN, f.balazs96@gmail.com)

***Abstract* — Our problem is from Kaggle (https://www.kaggle.com/c/airbus-ship-detection) We give a solution using the Single Shot Detector neural network to the problem of detecting ships in satellite images.**

I. Introduction

Airbus is excited to challenge Kagglers to build a model that detects all ships in satellite images as quickly as possible. Can you find them even in imagery with clouds or haze?
Here's the backstory: Shipping traffic is growing fast. More ships increase 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.
Airbus offers comprehensive maritime monitoring services by building a meaningful solution for wide coverage, fine details, intensive monitoring, premium reactivity and interpretation response. Combining its proprietary-data with highly-trained analysts, they help to support the maritime industry to increase knowledge, anticipate threats, trigger alerts, and improve efficiency at sea.
A lot of work has been done over the last 10 years to automatically extract objects from satellite images with significative results but no effective operational effects. Now Airbus is turning to Kagglers to increase the accuracy and speed of automatic ship detection.

II. About the task

In this competition, you are required to locate ships in images, and put an aligned bounding box segment around the ships you locate. Many images do not contain ships, and those that do may contain multiple ships. Ships within and across images may differ in size (sometimes significantly) and be located in open sea, at docks, marinas, etc.
For this metric, object segments cannot overlap. There were a small percentage of images in both the Train and Test set that had slight overlap of object segments when ships were directly next to each other. Any segments overlaps were removed by setting them to background (i.e., non-ship) encoding. Therefore, some images have a ground truth may be an aligned bounding box with some pixels removed from an edge of the segment. These small adjustments will have a minimal impact on scoring, since the scoring evaluates over increasing overlap thresholds.

III. About the dataset

About 200000 satellite images are available together with the bounding boxes of the ships on the images. Each image has a dimension of 768 by 768 pixels, and has 3 (RGB) color channels. The majority of images contain no ships. Around half of the images which contains any ships have some kind of land in the image, i.e. a port. A few images only contain the landscape and no open water. There are many clouds obstructing the water surface. To make the task even more difficult the ships are varying heavily in size. For example there are images about large freighters and small boats.

IV. Choosing the network

The following architectures were considered:

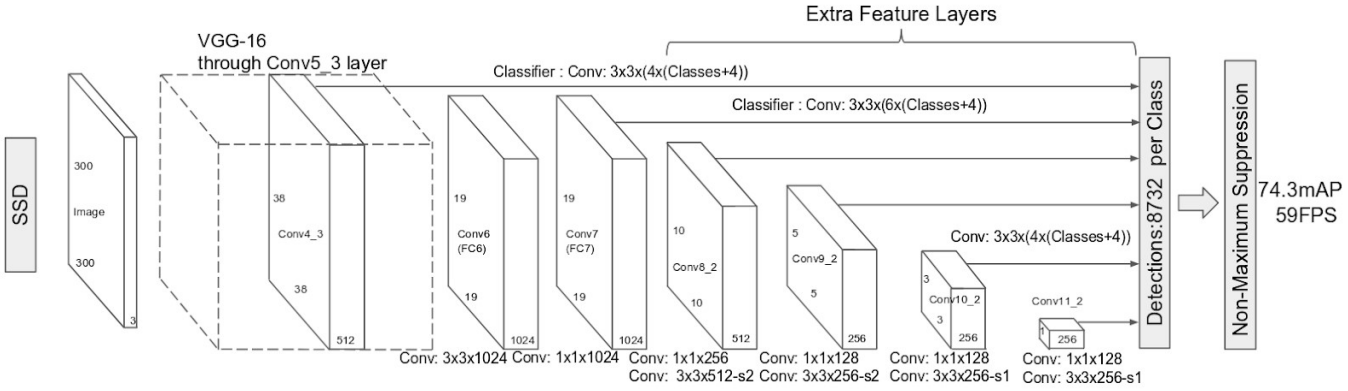
- R-CNN (Regions with CNN features): proposes rectangular regions and then classifies them, but is very slow (having a speed of around 1 minute/image)
- YOLO (You Only Look Once): creates an N×N grid of the image, and predicts the most likely class for each area (fast, but have a low spatial resolution)
- SSD (Single Shot Detector): similar to YOLO, but uses different sized grids and the bounding box need not be a square.

V. About the single shot detector

From the original paper:

SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and

produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes



Description of how this network works: (source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors.html)

Summarising the strategy of these methods

1. *Train a CNN with regression(bounding box) and classification objective (loss function).*
2. *Normally their loss functions are more complex because it has to manage multiple objectives (classification, regression, check if there is an object or not)*
3. *Gather Activation from a particular layer (or layers) to infer classification and location with a FC layer or another CONV layer that works like a FC layer.*
4. *During prediction use algorithms like non-maxima suppression to filter multiple boxes around same object.*
5. *During training time use algorithms like IoU to relate the predictions during training the the ground truth.*

On this kind of detector it is typical to have a collection of boxes overlaid on the image at different spatial locations, scales and aspect ratios that act as "anchors" (sometimes called "priors" or "default boxes").

VI. Processing the Data

The bounding boxes were provided in run-length encoding and we converted them to a format (coordinates) that can be understood by the network. An example of the original ground truth file:
"0570217ba.jpg, 555509 4 556273 8 557041 9 557809 4"

- The first element is the filename
- The second one is the run-length code. It contains pair of numbers in a sequence the first one is the starting pixel (the top left corner is 1), than going down and right. Second number is the run length from the starting position.
- There is at least one line for every image, with empty second value for shipless images
- And each line represents only one ship therefor it contains multiple line for images containing multiple ships

The *annotation/create_annotations.ipynb* notebook generates the bounding boxes from the given run-length encoding masks.

VII. Training the Network

We used an implementation of the SSD made for the Keras framework. Source: https://github.com/pierluigiferrari/ssd_keras

The codes had to be modified (e.g. changing directory paths).

To save time on the training, we used a pretrained network, which has been trained on 300×300 images on the COCO image database. The pretrained weights are available in the ssd_keras github repository. We used Google Cloud to run the training, which took about 1 day on a NVIDIA Tesla K80 GPU.

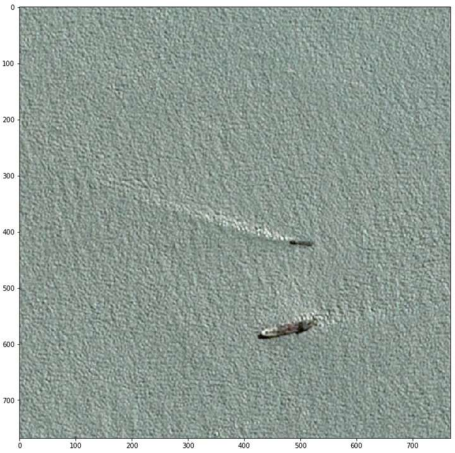
We tried different parameters for the training and the best model is included at: *weights/ShipDetection-120_loss-2.8902_val_loss-2.7863.h5*

An example training script is located at: *training/training.py*, an example log is located at: *training/mycrop_training_log.csv*

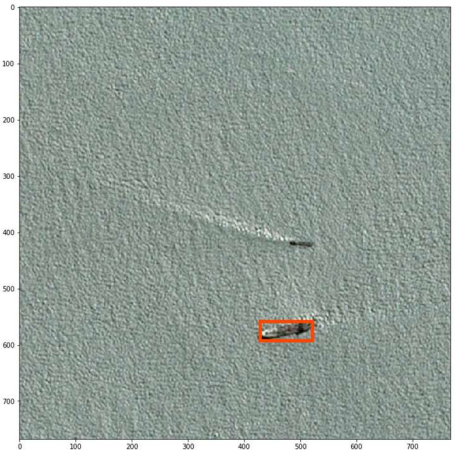
VIII. Evaluating the results

The network detects only large ships, and may detect islands as ships. Our network only works for images of size 300×300, therefore the satellite images had to be resized before giving it to the network. If a ship is small, then it will be even smaller and the network will not detect it. To solve this problem, we decided to cut the images into a few overlapping 300×300 images and then combine the predictions.

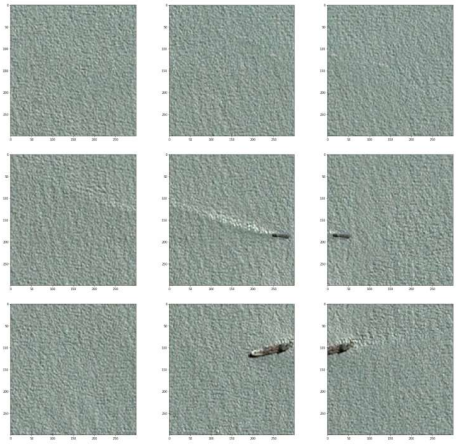
An example image (size: 768×768):



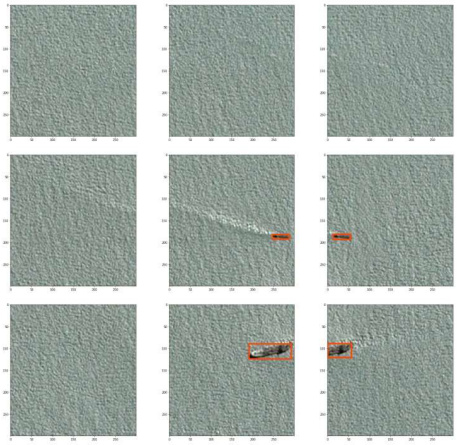
Predicted objects (image resized to 300×300):



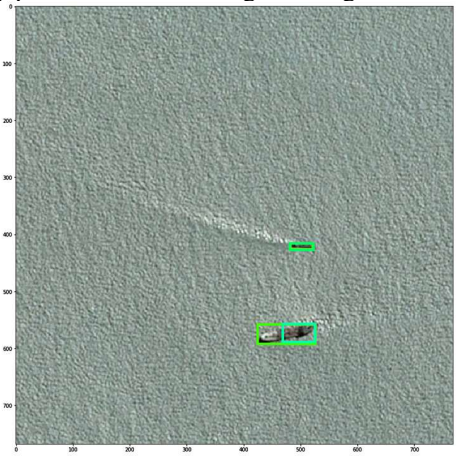
Segment image to avoid loss of small features due to resizing:



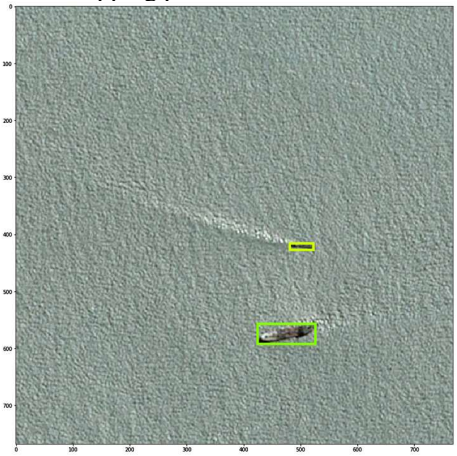
Prediction on the smaller segments is more successful:



Display predictions on the original image:



Combine overlapping predictions:



IX. Comparison with other solutions

The main drawback of our model is that it can only predict rectangular areas, while the competition scoring is depends on the correctly predicted pixels of the ships. The most successful teams used the U-Net network, which is a fully convolutional neural network with residual (skip) connections.

References

[R-CNN] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, UC Berkeley, 2013, https://arxiv.org/pdf/1311.2524.pdf

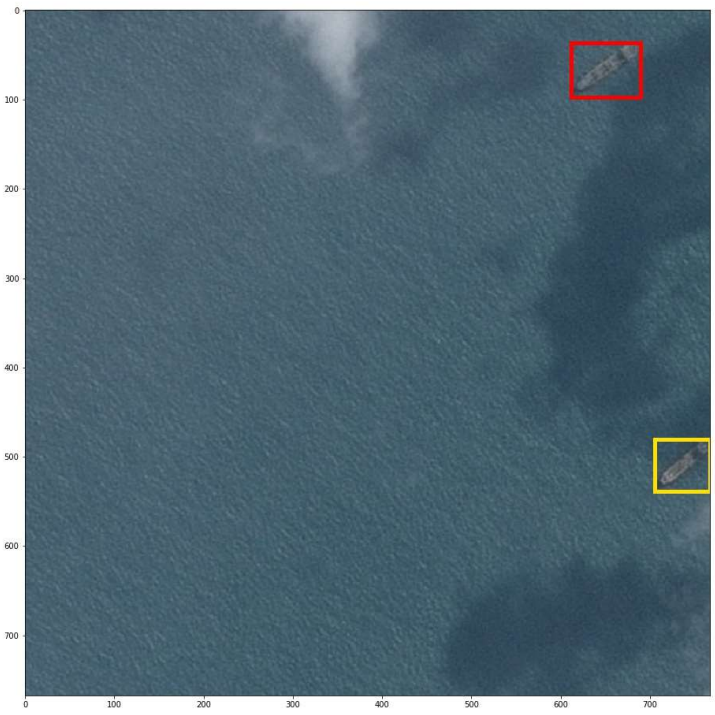
[YOLO] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object* DetectionW.-K. Chen, *Linear Networks and Systems*, University of Washington, Allen Institute for AI, Facebook AI Research, 2015, https://arxiv.org/pdf/1506.02640v5.pdf

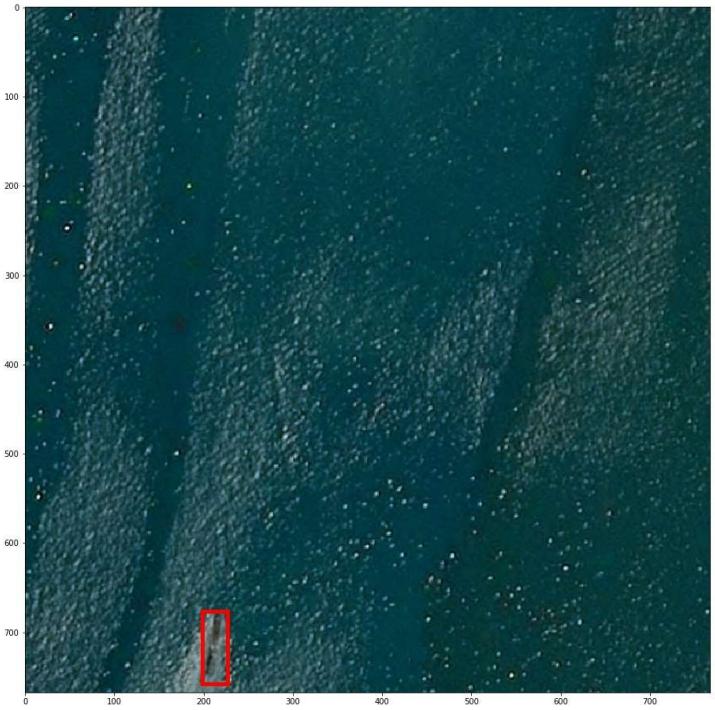
[SSD] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, *SSD: Single Shot MultiBox Detector*, UNC Chapel Hill, Zoox Inc., Google Inc., University of Michigan, Ann-Arbor, 2015, https://arxiv.org/pdf/1512.02325.pdf

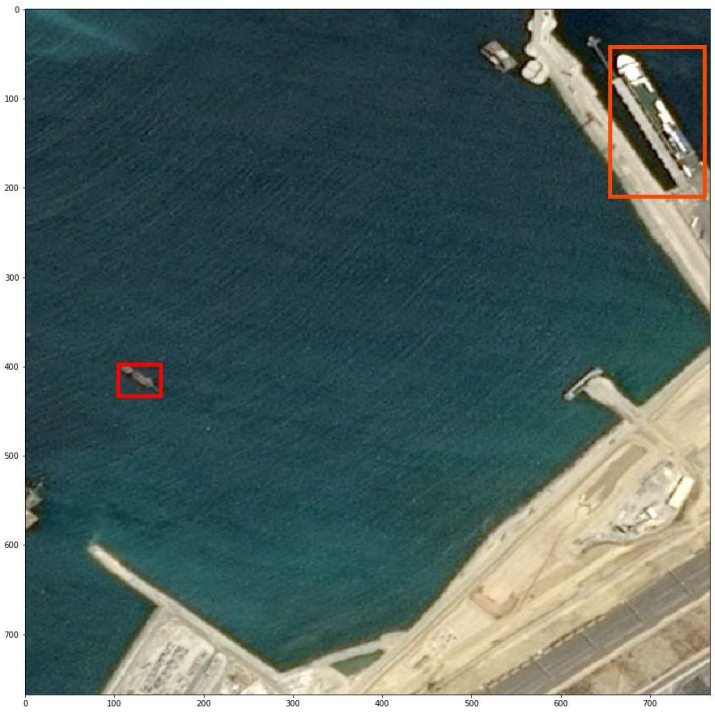
[U-Net] Olaf Ronneberger, Philipp Fischer, Thomas Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, 2015, https://arxiv.org/pdf/1505.04597.pdf

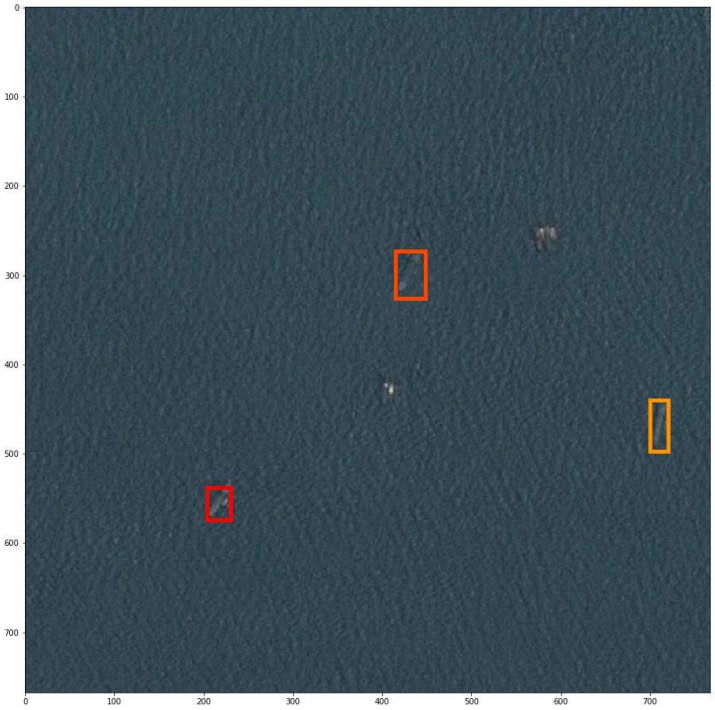
Examples

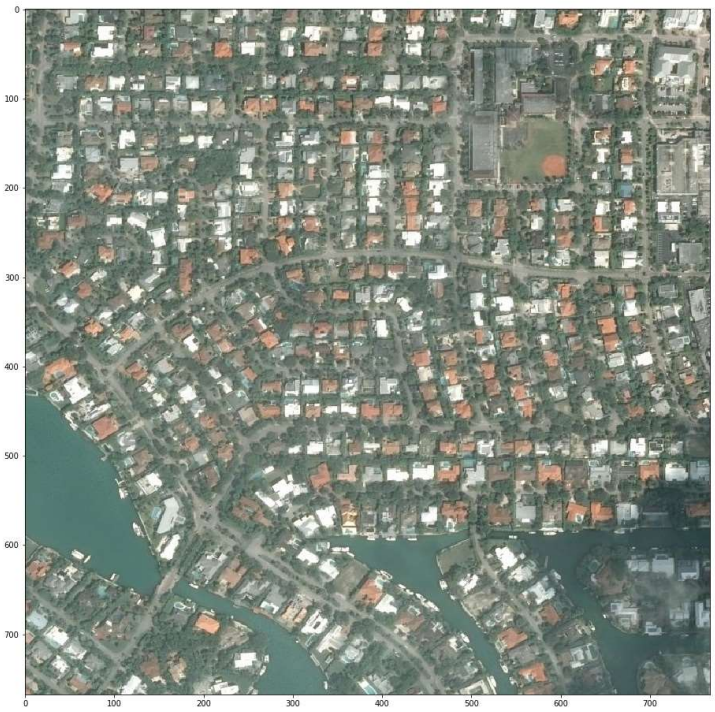
More examples are included in the *evaluation/* directory.

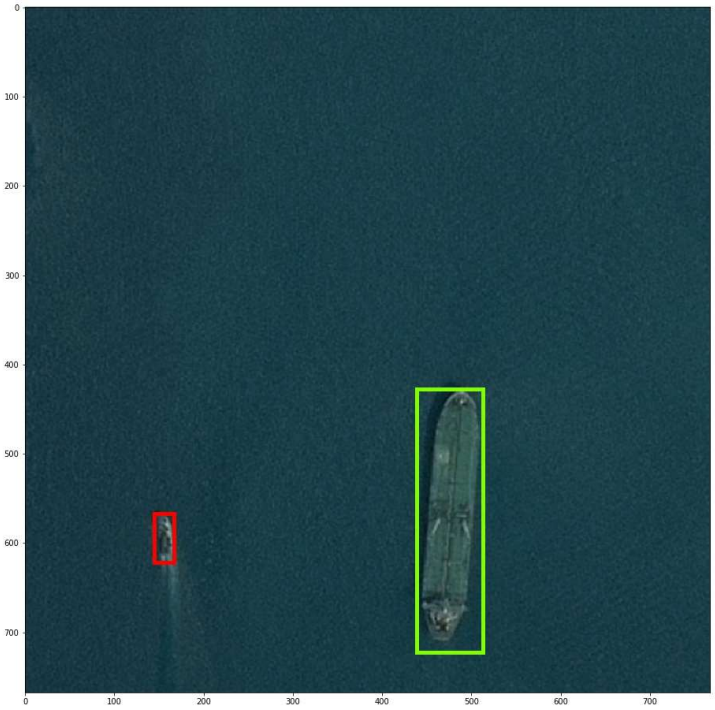


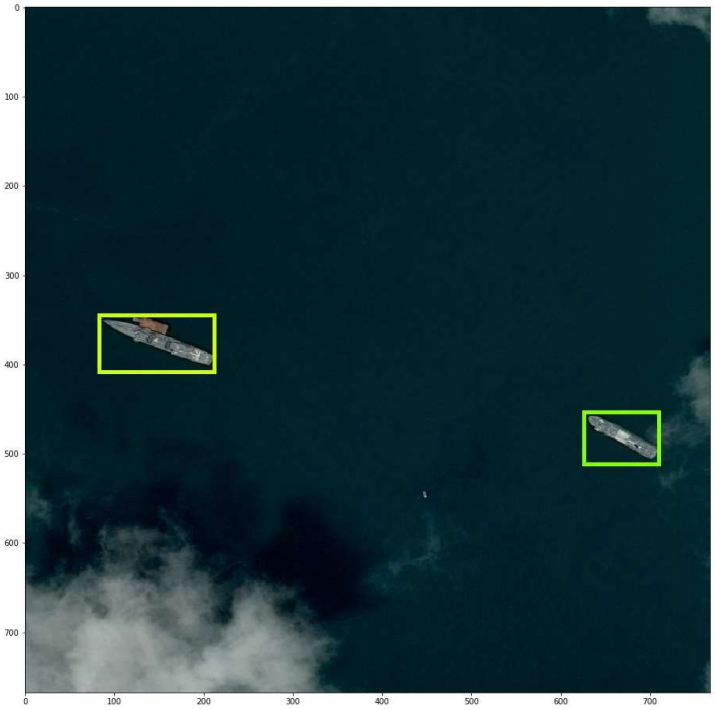

0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700

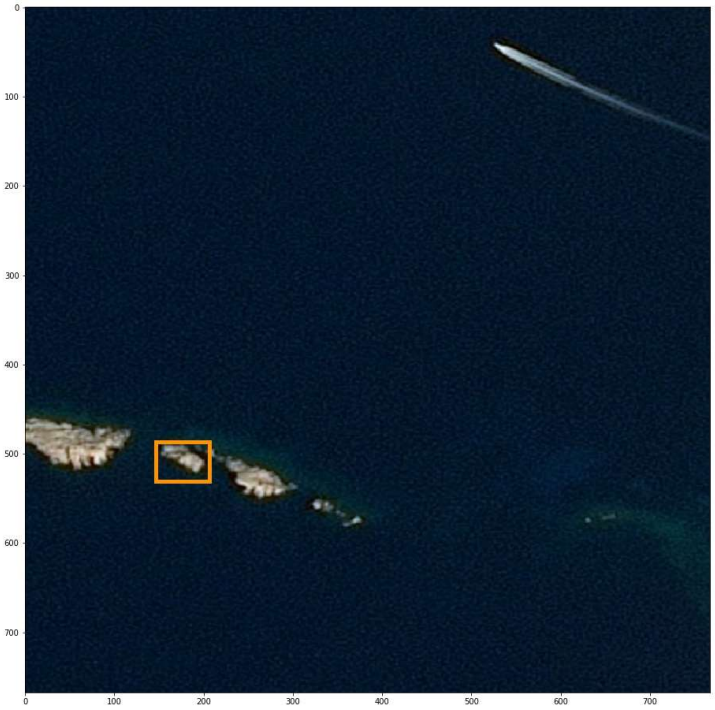

0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700

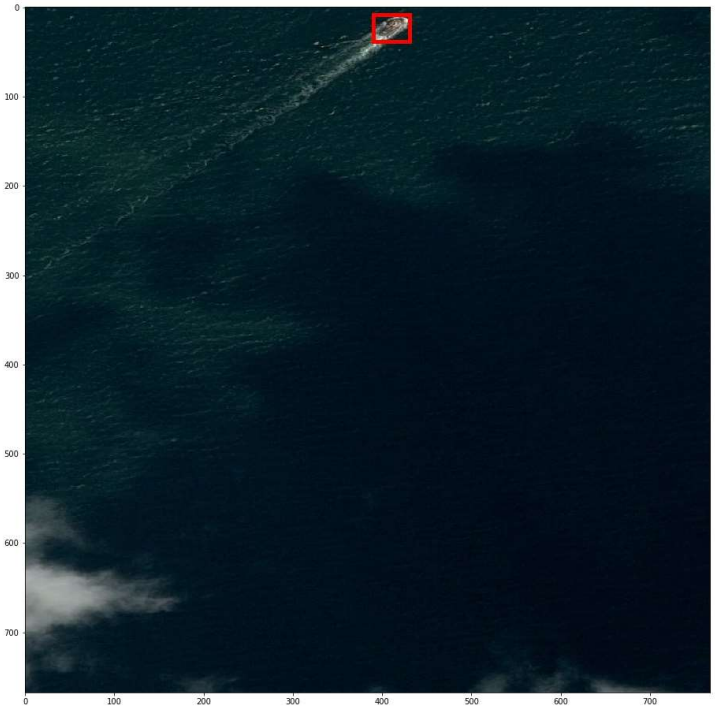

0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700