Airbus Ship Detection

Aytekin YILDIZHAN
N18147923
CMP 719 Computer Vision Project Report
Computer Engineering Department, Hacettepe University
Ankara, Turkey
aytekinyildizhan@hacettepe.edu.tr

***Abstract*—Deep Learning (DL) is truly helpful techniques for image detection, segmentation and classification etc. In this project, we analyze the Airbus Ship dataset and detect the ships in the satellite images. Firstly, we do some preprocessing the data. There is no bounding box information in our data set. Only mask information is included. We convert masks into the bounding boxes. Then, we estimate anchor boxes and we train the data with YOLOv2 object detector by using these anchor boxes. We use six different networks which are ResNet-50, GoogleNet, MobileNet-V2, VGG16, VGG19, and InceptionResNet-V2. Only InceptionResNet-V2 could not detect any ships on the test set. Secondly, we use three pre-trained network systems trained on PASCAL VOC2007 dataset. These are Fast-RCNN, Faster-RCNN and Single Shot MultiBox Detector (SSM). We download and install MatConvNet's library and we just test the single image. Finally, we show the results and discussion about future work.**

***Index Terms*—computer vision, deep learning, convolutional neural network, ship detection**

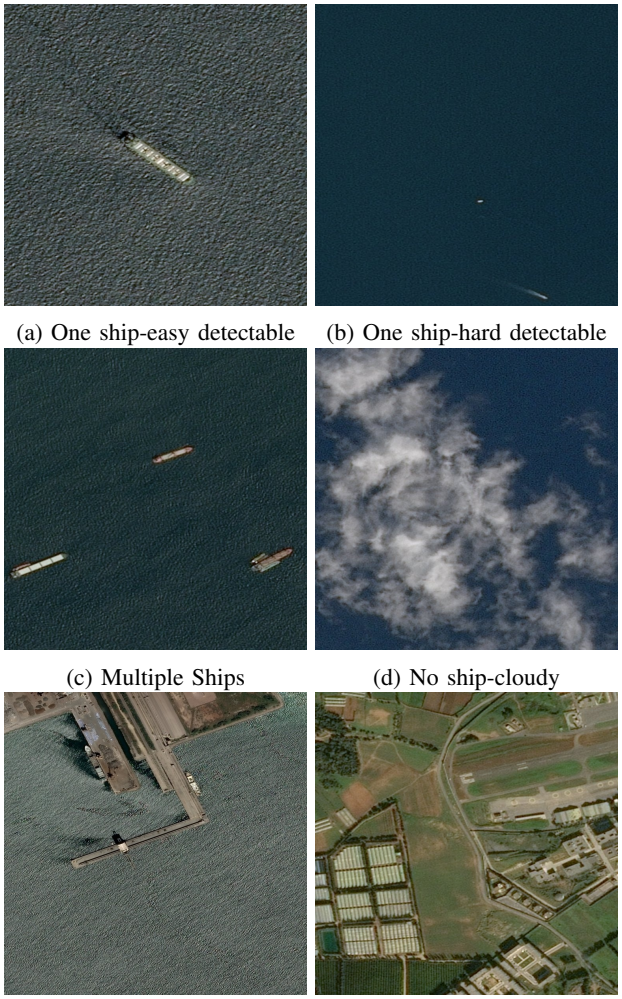
I. Introduction

This project aims to detecting ships in the satellite images. This problem is highly challenging because ships are really small in the satellite images. Various scenes including open water, wharf, haze, buildings and clouds appear in the dataset. These obstacles increases the difficulty of detection and most of the images do not include any ships. Figure 1 shows the some example ship images.

The dataset is Airbus Ship Detection Challenge and obtained Kaggle from in this link [1]. The dataset contains 15606 test images and 231723 train images with size $768 \times 768 \times 3$ and a total size 28.53 Gb. (.rar file). There are no ships in 150000 images. We do not use all of these images due to computational limits. We take random samples from the training image file. In progress report, we selected 162 images as training set and 108 images as test set. In final report, we increase these numbers. We selected 4999 images as training set, 2142 images as test set (%70 training and %30 testing). Also while creating the training and test set, we take images containing only one ship.

Our dataset has two features. One is that 'ImageId' and the other one is 'EncodedPixel'. ImageId column is the image name and EncodedPixel column indicates that where the ships located in pixelwise. A prediction of "no ship in image" should have a blank value in the EncodedPixel column. EncodedPixel uses run-length encoding (RLE) on the pixel values. The pixels are one-indexed and numbered from top to bottom, then left to right. E.g. 1 3 implies starting at pixel 1 and running a total of 3 pixels (1,2,3). For example ImageId is 00023d5fc.jpg and EncodedPixel is 1 3 10 5. That implies pixels 1,2,3,10,11,12,13,14 are to be included in the ship. These coordinates represent segmentation boxes of ships.

This project is implemented with MATLAB 2019b and NVIDIA cuDNN library (v.10.2) is used for accelerated train-

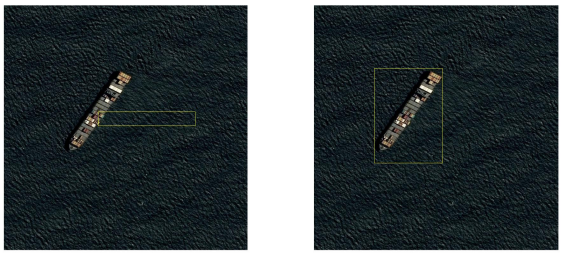


(a) One ship-easy detectable (b) One ship-hard detectable

(c) Multiple Ships (d) No ship-cloudy

(e) Two ship-wharf (f) No ship-buildings

Fig. 1: Examples of Ship Image



(a) Wrong Bounding Box (b) Correct Bounding Box

Fig. 2: Examples of Bounding Boxes

ing and detection. Our GPU is GeForce GTX 1650. Our CNN models are trained in Microsoft Azure Cloud with GPUs. We detect bounding boxes from RLE (EncodedPixel is written by RLE). In progress report, we didn't find exactly bounding boxes. These bounding boxes are misplaced. However in final report, we detect bounding boxes exactly. Figure 2 indicates that wrongly and correctly placed bounding boxes.

Afterwards, we estimate anchor boxes and we train the dataset with YOLOv2 Object Detection Network. We use ResNet-50 in progress report. In final report besides ResNet-50, we also use GoogleNet, MobileNet-V2, VGG16, VGG19, and InceptionResNet-V2. We cannot detect any image with InceptionResNet-V2. We used 5 anchor boxes in progress report but in final report, we use 3 anchor boxes. After training, we calculate and show the result of ResNet-50, GoogleNet and MobileNet-V2. We cannot get the detection result of VGG16 and VGG19 due to the out of memory.

Finally, we load and implement pre-trained models. In progress report, we load yolov2VehicleDetector.mat from MATLAB file. In final report, we use these models that are Fast-RCNN, Faster-RCNN and Single Shot MultiBox Detector [2]. Their training data is PASCAL VOC2007 dataset. Then, we show the some image score of these networks.

Our report is constructed as follows: (2) report on related work that has been done in the past, (3) proposed methods and results from our experiments, (4) showed an example of detected ship, (5) showed an example of pre-trained MatConvNet Models and (6) conclusion and future works.

II. Related Work

There are many work dealing with the topic of ship detection. Xiao et. al [3] proposed a novel ship-detection method based on CNN images with complex backgrounds with rotated bounding boxes. They create paired semantic segmentation network to predict the four parts (i.e., top left, bottom-right, top-right, bottom-left parts) of each ship and by combining paired top-left and bottom-right parts (or top-right and bottomleft parts), they can take the minimum bounding box of these two parts as the rotated anchor. Zhang et. al [4] proposed a novel high-speed SAR ship detection approach by mainly using depthwise separable convolution neural network (DS-CNN). They used DS-CNN, which consists of a depthwise convolution (D-Conv2D) and a pointwise convolution (PConv2D), to substitute for the conventional convolution neural network (C-CNN). In this way, the number of network parameters gets obviously decreased, and the ship detection speed gets dramatically improved. Chen et. al [5] classify the images 'ship' and 'no ship' using the their project dataset. However, they do not detect exact location of ships. They compare CNN networks and machine learning techniques.

III. Proposed Methods And Experimental Results

A. Data Preparation

First of all, we have to find bounding boxes from RLE. The encodedPixel column looks like this: start, length, start, length, ... , where each pair of (start, length) draws a line of length pixels starting from position start. We have to find the start position as a (x, y) coordinate. The solution is that the startPixel modulo 768 is equal to x coordinate and startPixel divided by 768 is equal to y coordinate. For example, EncodedPixel is 264661, 17. Our startPixel is 264661 and length is 17. $264661 \pmod{17} = 469$ and $264661 \mid 17 = 344$. Therefore we have (x,y) coordinate of this EncodedPixel $(469, 344)$. When all EncodedPixel are implemented, we have (x, y) coordinates each row. Then, we determine maximum and minimum of x value and y value. Then, we got $x_{min}, x_{max}, y_{min}$ and y_{max}. We calculate $width = x_{max} - x_{min}$ and $height = y_{max} - y_{min}$. Finally we got the bounding boxes as a format of $(x_{min}, y_{min}, width, height)$.

We re-sized the original image to the size of $256 \times 256 \times 3$ then we start to analyze the data. Also we use data augmentation techniques to improve our results. In progress report, we could not find exact bounding boxes. However in final report, we solve this problem in Figure 2.

B. Estimation Anchor Boxes

In this part, we estimate the anchor boxes. We investigated the number of anchor boxes. We choose the number of anchors as a 3 in final report. Our anchor boxes are calculated in Equation 1.

$$\begin{bmatrix} 37 & 24 \\ 21 & 7 \\ 43 & 58 \end{bmatrix} \tag{1}$$

C. Yolov2 Object Detector using Anchor Boxes

In this part, we train the data with YOLOv2 object detector by using the anchor boxes that we found in section B. Our hyperparameters are shown in Table I. All networks type use the same hyperparameters. We use ResNet-50, GoogleNet, MobileNet-V2, VGG16, VGG19, and InceptionResNet-V2 in the final report.

1) ResNet-50: In this part, we use ResNet-50 for training data. This experiment takes 59 minutes and iteration number is 4368. Training loss curve is shown in Figure 3. In Figure 3, our training curve is decreasing and final RMSE loss is 0.3206. In ResNet-50, 586 of 2142 images are detected (%27 are detected). In Figure 4, average precision is shown.

TYPE OF PARAMETER	VALUE
Mini batch size	8
Learning Rate	0.001
Max. Epochs	7
Optimizer	sgdm
Shuffle	never
VerboseFrequency	30
Verbose	true
CheckpointPath	tempdir

TABLE I: Hyperparameters of Our Model

2) GoogleNet: In this part, we use GoogleNet for training data. This experiment takes 19 minutes and iteration number is 4368. Training loss curve is shown in Figure 5. In Figure 5, our training curve is decreasing and final RMSE loss is 0.6036. In GoogleNet, 1359 of 2142 images are detected (%63 are detected). In Figure 6, average precision is shown.

3) MobileNet-V2: In this part, we use MobileNet-V2 for training data. This experiment takes 12 minutes and iteration number is 1869. Training loss curve is shown in Figure 7. In Figure 7, our training curve is decreasing and final RMSE loss is 0.8573. In MobileNet-V2, 1447 of 2142 images are detected (%67 are detected). In Figure 8, average precision is shown.

4) VGG16 and VGG19: In this part, we use VGG16 and VGG19 for training data. Training loss curve of VGG16 is shown in Figure 9 and VGG16 is shown in Figure 10. In Figure 9, our training curve is decreasing and final RMSE loss is 0.4869 and in Figure 10, our training curve is decreasing and final RMSE loss is 0.2379. Due to the out of memory, VGG16 and VGG19 cannot implement on 2142 test image but we can show some detected ships on the image.

D. Comparison of The Results

In this part, we compare the results of networks that we use. Table II shows this comparison.

As can be seen from Table II, VGG19 has better loss than other methods. MobileNet-V2 has higher detection ratio than other networks. GoogleNet is the highest average precision among them.

NETWORK	DETECTION RATIO	AV. PRECISION	RMSE LOSS
ResNet-50	%27	0.12	0.3206
GoogleNet	%63	0.33	0.6036
MobileNet-V2	%67	0.17	0.8573
VGG16	–	–	0.4869
VGG19	–	–	0.2379

TABLE II: Comparison of Networks

IV. Example of Visualition of Networks

In this part, one of test image are implemented and showed the results. This sample image is not from the training and test set. Figure 11 shows the example image and detected a ship on this image.

As you can see in Figure 11, VGG19 got the highest score but MobileNet-V2 detected two ships but there is a ship on this image.

V. Pre-Trained MatConvNet Models

MatConvNet is a useful library that Convolutional Neural Networks (CNNs) for MATLAB computer vision applications. It can be downloaded from this link [6]. To install and use MatConvNet, we need a compiler supported by MATLAB. In this experiment, we use Microsoft Visual C++ 2019 (C) and install MatConvNet.

In this part, we use Fast-RCNN, Faster-RCNN and Single Shot MultiBox Detector [2]. Also we downloaded pre-trained network from this link [7]. We use fast-rcnn-vgg16-dagnn.mat, faster-rcnn-vggvd-pascal.mat and ssd-mcn-pascal-vggvd-300.mat. All of models are pre-trained models from PASCAL VOC2017. Figure 12 shows one of the example image.

As can be seen from Figure 12, SSD and Faster RCNN detect the ship but classify this ship wrongly. Fast RCNN cannot detect any ship due to the bounding box format.

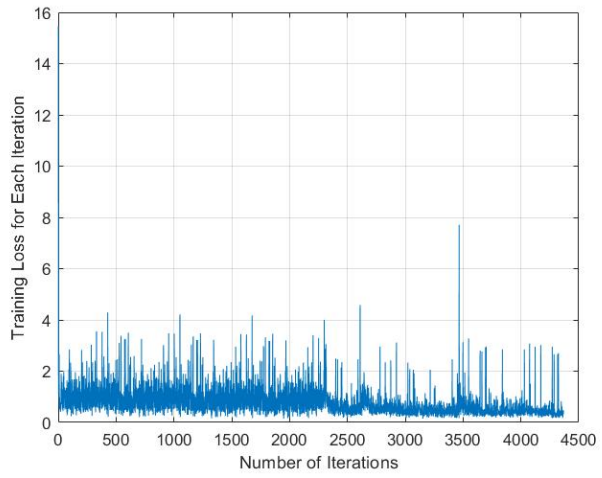


Fig. 3: Number of Iteration vs. Training Loss of ResNet-50

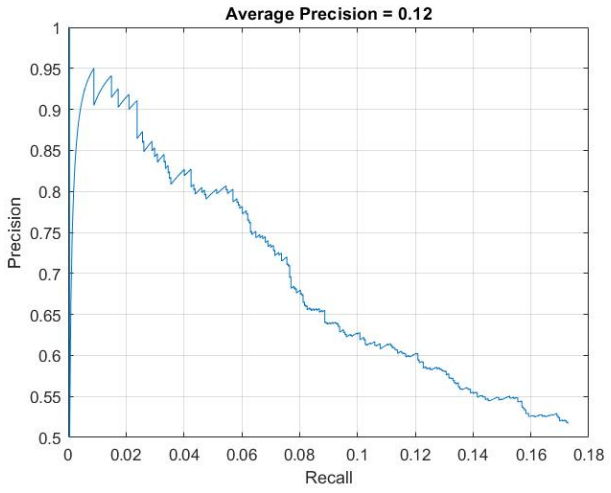


Fig. 4: Average Presicion of ResNet-50 = 0.12

VI. Conclusion and Future Works

This project goals detecting ships in the satellite images. Among all experimenal methods, VGG19 gave the best result (0.2379 RMSE loss) and MobileNet-V2 gave the best detection ratio (%67). GoogleNet is the highest average precision (0.33) but not good enough. We used only 8 percent of our data set. It will be more useful to increase the number of data in future. Also we will try implementing different deeper networks (e.g., DenseNet201).

In deep learning, GPU is very important component. Therefore, if we want to more data to implement, we have to increase GPU process power. Also in MATLAB, Parallel Computing Toolbox and MATLAB Parallel Server on the Cloud will increase our implementation speed. We use NVIDIA cuDNN and Azure Cloud with GPUs in this experiment. As you can see in Table II, VGG16 and VGG19 cannot be applied on 2142 test image due to out of memory. When mini batch size is equal to 16, also our system cannot work. Our hyperparameter can be changed on future works. However, unlike the progress report, the experiments in this report were more successful and faster. In training set, we have one-ship on image. We can increase our performance by adding images with multiple ships in our training and test set. In this experiment, we use PASCAL VOC2017 pre-trained dataset. We can also use different dataset (e.g., MSCOCO and ILSVRC) and we can improve our result.

Finally, in the future, Python is used instead of MATLAB. The usage of Python are increasing day by day since Python has more advantages than MATLAB. PyTorch Mask R-CNN implementation and DetectronV2 will be more useful than MATLAB.

References

[1] https://www.kaggle.com/c/airbus-ship-detection/data.

[2] Liu W.; Anguelov, D.; Erhan, D.; Szegedy, C; Reed, S.; Fu, C.Y.; and Berg A.C. SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science. 2016, pp. 21-37

[3] Xiao, X.; Zhou, Z.; Wang, B.; Li, L.; Miao, L. Ship Detection under Complex Backgrounds Based on Accurate Rotated Anchor Boxes from Paired Semantic Segmentation. Remote Sens. 2019, 11, 2506.

[4] Zhang, T.; Zhang, X.; Shi, J.; Wei, S. Depthwise Separable Convolution Neural Network for High-Speed SAR Ship Detection. Remote Sens. 2019, 11, 2483.

[5] http://cs229.stanford.edu/proj2018.

[6] https://www.vlfeat.org/matconvnet/download/matconvnet-1.0-beta25.tar.gz.

[7] http://www.robots.ox.ac.uk/ albanie/mcn-models.html.

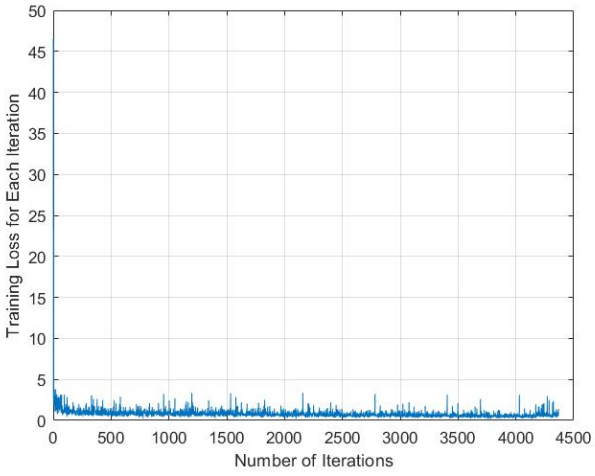


Fig. 5: Number of Iteration vs. Training Loss of GoogleNet

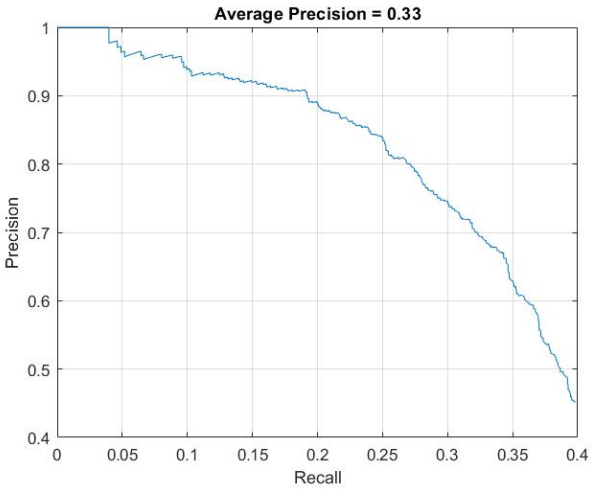


Fig. 6: Average Presicion of GoogleNet = 0.33

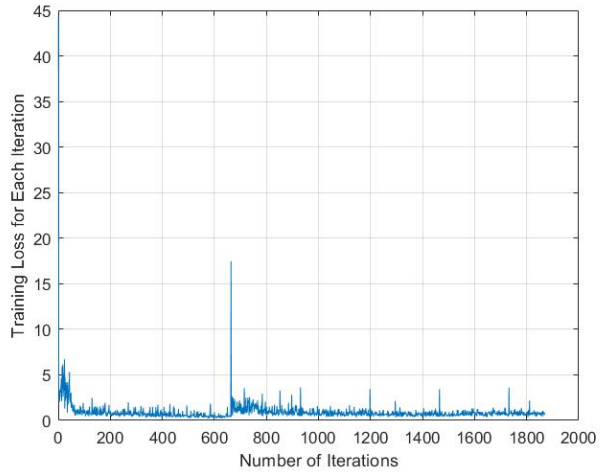


Fig. 7: Number of Iteration vs. Training Loss of MobileNet-V2

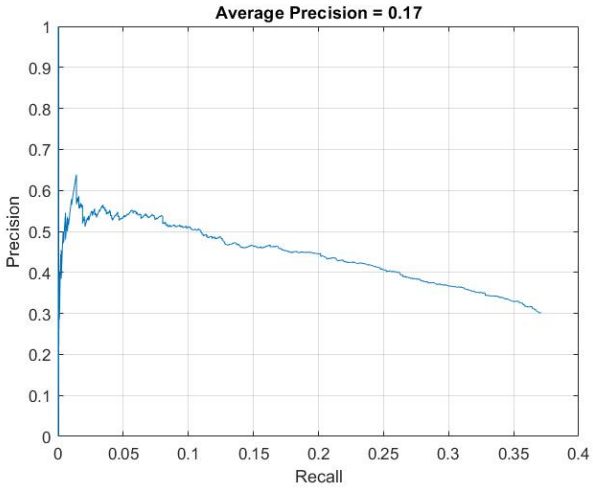


Fig. 8: Average Presicion of MobileNet-V2 = 0.17

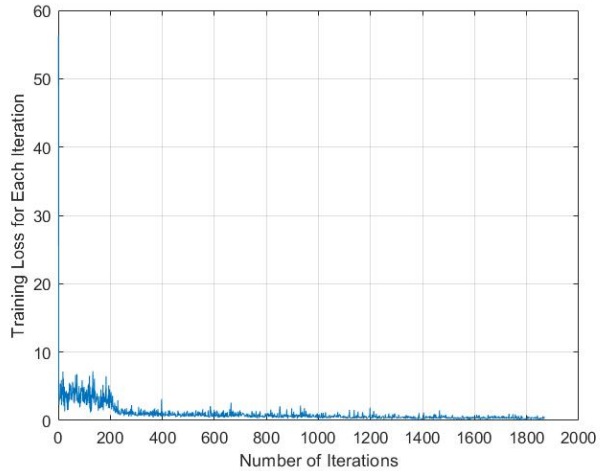


Fig. 9: Number of Iteration vs. Training Loss of VGG16

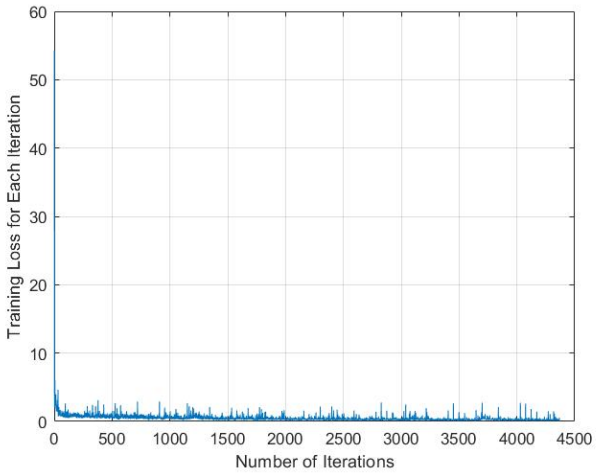
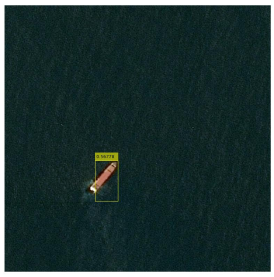
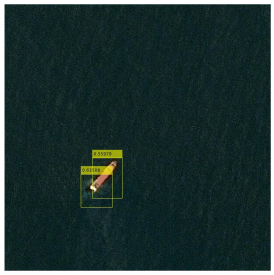


Fig. 10: Number of Iteration vs. Training Loss of VGG19

(a) Example Image



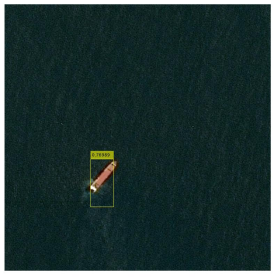
(b) ResNet-50, Score:0.56778



(c) MobileNet-V2, Score: 0.63508 ,0.55979

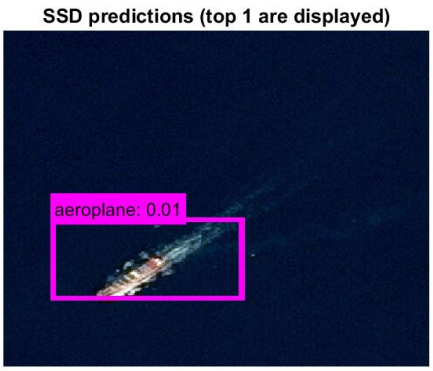
(d) GoogleNet, Score: 0.77426

(e) VGG16, Score: 0.76989

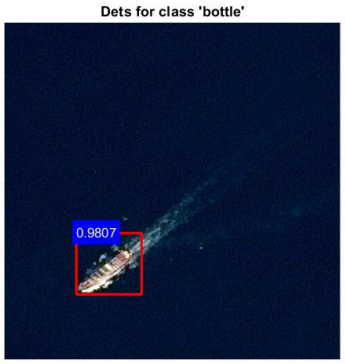


(f) VGG19, Score: 0.82047

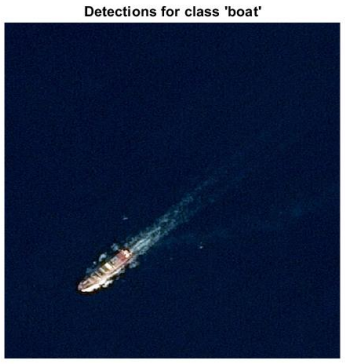
Fig. 11: Example of Detection of Ship



(a) SSDl



(b) Faster RCNN



(c) Fast RCNN

Fig. 12: Pre-Trained MatConvNet Model Example