Airbus Ship Detection Challenge Preclassification, Mask R-CNN, U-NN

FIAS
Frankfurt Institute for Advanced Studies

Practicum Report

provided by:

Hevin Oezmen
Hesamedin Ghavami Kazzazi
Iurii Mozzhorin

Submission Date:

October 16, 2018

Supervisor: Prof. Dr. Visvanathan Ramesh

Version Explanation

Due to the varying preferences and different tools of choice for generating PDF files, our submission is divided into two parts/documents which are complementary to each other. This Documnet includes the Mask R-CNN section of our project and complements the other part that is submitted separately. It was generated with LaTeX.

Contents

List of Figures

Chapter 1

Introduction

The volume of shipping traffic is increasing rapidly. One of the side effects of this growth is that the number of ships in transit becomes larger. This 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. One of the active companies in this field is airbus. 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. Recently, Airbus has turned to Kagglers to increase the accuracy and speed of automatic ship detection[13].

1.1 Motivation

As it was mentioned in previous section, increasing the accuracy and speed of automatic ship detection is the aim of this challenge. To overcome this task, we decided to try the tools, which are widely utilized in this field. This brought us to Mask R-CNN and U-net. U-Net was originally developed for the biomedical data but has just demonstrated itself as liable for satellite images also. Mask R-CNN framework is an extended version of Faster R-CNN which was also an improvement to Fast R-CNN. The Mask Region-based Convolutional Neural Network has proven to be robust and performant in surmounting the image segmentation difficulties[4]. The complexity and results of Mask R-CNN has motivated us to study it further and utilized it for this project. The Mask R-CNN outputs bounding boxes and masks of the detected objects which are quite what we are seeking in the proposed challenge, to detect ships in an image and extract their masks.

Chapter 2

Mask R-CNN

In this chapter, the Mask R-CNN or Mask Region-based Convolutional Neural Network is introduced as a framework to be used to overcome the Airbus ship detection challenge. Based on the defined classes or objects, the system detects the objects and delivers their respective masks. The image segmentation property of this framework makes it a powerful tool for the ship detection challenge. In this section, the background of the model is explored first. Afterward, the model itself together with its components will be introduced. How this model handles the data, the model and its data delivery will be inspected subsequently. Finally, the evaluation and improvement suggestion will be provided.

2.1 Related Works

The computer vision section has improved rapidly in object detection and semantic segmentation thanks to the baseline systems, such as Fast R-CNN [1], Faster R-CNN [2] and Fully Convolutional Network (FCN) [3]. Compared to the mentioned computer vision tasks, the task of image segmentation has proven to be very challenging. Due to the challenging nature of instance segmentation, the development team of the Mask R-CNN has targeted a framework for this task, which is comparable to those mentioned earlier in performance, robustness and flexibility [4]. Before diving into the framework and its components, the image segmentation and R-CNN or Region-based CNN will be explored for setting the ground of the Mask R-CNN.

2.1.1 R-CNN and paving the way towards the Mask R-CNN

The Region-based Convolutional Neural Network approach generates category-independent region proposals, called Region of Interests or RoI, by using a region proposal module. Each region is then fed individually to a CNN (part of a large CNN) which acts as a feature extraction component. The CNN extracts a fixed-length vector from each region and

produces feature vectors as output, which is also fed to set of SVMs [1] for classification. In other words, it trains the backbone convolutional networks separately (and end-to-end) on every single RoI to classify each RoI into object(s) of interest or background. R-CNN acts mainly as a classifier and make no prediction about object bounds [6].

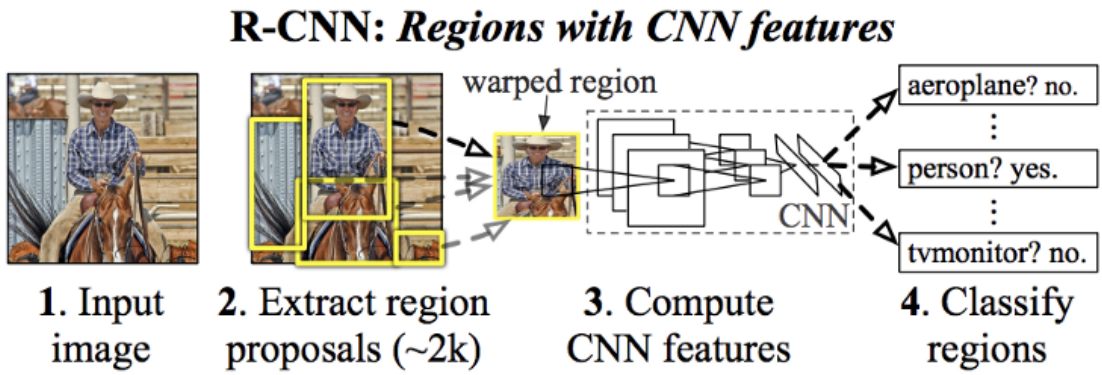

Figure 2-1: R-CNN Framwork [6]

This method was further improved to enable the attending of RoIs on feature maps by utilizing the RoI Pool, which led to a better performance and accuracy, hence the name Fast R-CNN. In Fast R-CNN, the input of the system is an entire image and a set of object proposals or RoIs. This input is processed by a large CNN together with a max pooling layer. This process pools each RoI into a fixed-sized (convolutional) feature map (containing only RoIs). By using a RoI pooling layer and a sequence of fully connected layers, each RoI on the feature map is then mapped onto a fixed-length feature vector. Finally, a softmax layer predict the class of proposed region and the offset values for the bounding box [1].

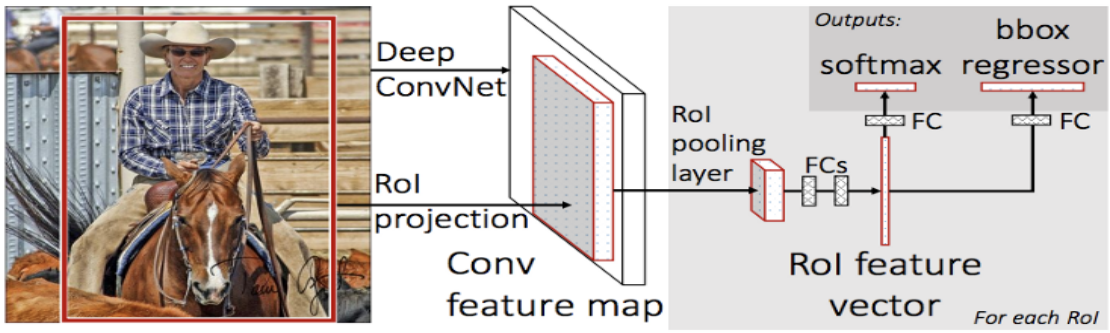

Figure 2-2: Fast R-CNN Framwork [1]

[1]Support Vector Machine

The both mentioned methods uses selective search to generate region proposals, which is a time-consuming task. The Faster R-CNN speeds up the whole process by omitting the selective search algorithm and letting the network learn the region proposal itself. In Faster R-CNN, the system is comprised of a deep fully convolutional network which serves as region proposal module and a Fast R-CNN detector. In this case, the RPN [1] module (utilizing the attention mechanism [7]) point the regions out to Fast R-CNN module. The RPN module uses sliding-window method and scans over the convolutional feature map, which is generated by the last shared convolutional layer. Multiple region proposals are predicted at the location of each sliding-window which are called anchors. Each anchor (region proposal) is provided with a score that estimates the probability of object or not object [2].

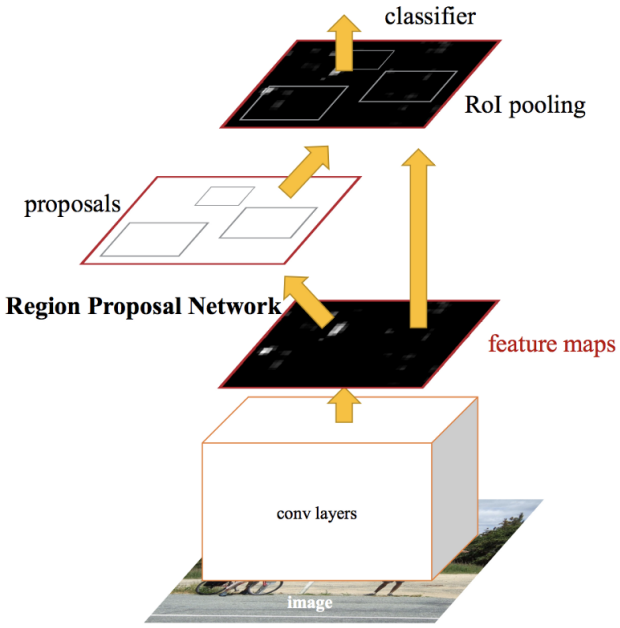

Figure 2-3: Faster R-CNN Framwork [2]

2.1.2 Image Segmentation

Image segmentation is the process of dividing a digital image into multiple regions or segments, which correspond to different objects or parts of objects. Every pixel in the image is assigned a label which correspond to an object or a class. As a result, a set of pixels, which are labeled identically, shares a specific character. This provides a mean to represent a digital image into a much more meaningful and easier to analyze object [5].

[1] Region-based Proposal Network

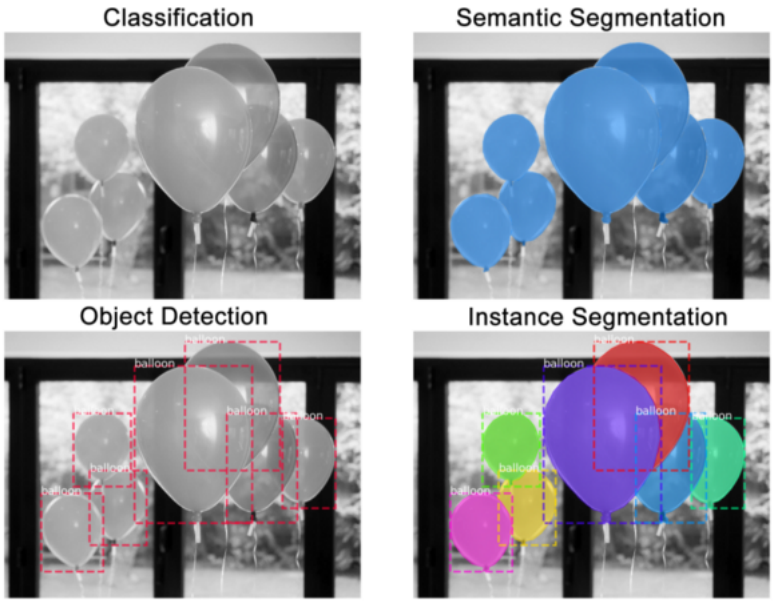

Figure 2-4: Tasks of Computer Vision [11]

2.2 Mask R-CNN Details

The Mask Region-based Convolutional Neural Network adopts the two-stages procedure of Fast R-CNN. It keeps the RPN stage untouched but adds a branch in parallel to the class and boundary box prediction in second stage, which outputs a binary mask for each RoI[4]. The Mask R-CNN framework, which is used for ship detection during this project, was introduced by Kaiming et al., 2017 in Mask R-CNN paper.

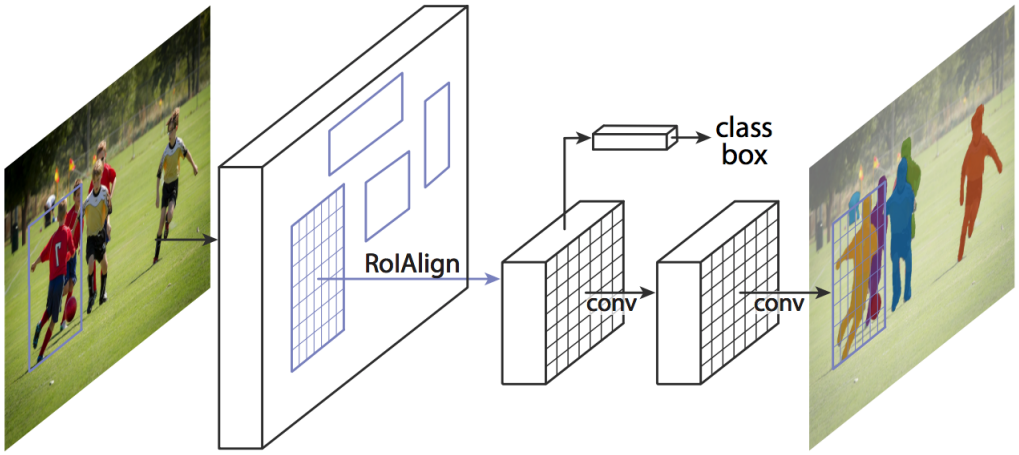

Figure 2-5: Mask R-CNN Framwork [4]

The implementation of the model is provided by Waleed A. via his Github page [8]. The model is generated by using the ***Model Class API*** of ***Keras***. This framework consists of the following components:

1. **Backbone** which is composed of a standard ResNet50 or Resnet101[1] and is configurable in code. This component serves as a feature extractor and is extended via Feature Pyramid Network(FPN). The backbone is implemented in ***Keras***. The ResNet has 5 stages. The first stage is a keras 2D convolution layer with output filters of 64, kernel size of (7,7), ReLU function as activation and max-pooling with a pool size of (3, 3). The second stage consists of a convolution block[2] and two identity blocks[3] each with kernel size 3. The third stage has one convolution block and three identity blocks each also with kernel size 3. The fourth stage makes the difference between ResNet50 and 101. In our case, namely ResNet50, this stage consists of a a convolution block and five identity blocks. The last stage is like the second stage but with diffrent output filters for convolution and identity blocks.

2. **Region Proposal Network** or **RPN** which slides a small neural network over the generated convolutional feature map and produces anchors at each sliding-window. This layer consists of 2D CNNs and uses ***soft max*** for activation.

3. **RoI Classifier** and **Bounding Box Regression** which runs over the RoIs generated by RPN and outputs two properties for each RoI, Class and Bounding Box Refinement. This layer is a fully connected layer(FC) and uses ReLU function for activation.

4. **RoI Pooling** which cropps and resizes the corresponding part of the convolutional feature map to a fixed size for feedind into the classifie.

5. **Segmentation Masks** which is a convolutional network and generates the binary mask for each (positive) RoI selected by RoI classifier

Before diving deeper in this project, it should be mentioned that the code to this project is accessible under our GitHub Repository under the URL: https://github.com/mozzhorin/ML-PR18

2.3 Data

the train data set consists of 104070 Satellite images with the resolution of 768 x 768 pixels[4]. The training images include either ships or no ships which is indicated by masks via Run-Length-Coding . A CSV file was provided by Kaggle which contains the corresponding masks for each image. An Example of this file is shown in figure 2-6. This data set

[1] Residual Network which employes deep residual learning to overcome the saturation of accuracy in much deeper networks when depth of network increases[10]

[2] A residual block with shortcut which has a convolutional layer at the shortcut since input and output have different dimensions

[3] A residual block whose shortcut can be directly used because input and output are of the same dimension

[4] the data set used for training was provided prior to changes due to the data leak[12]

was divide into 90% or 93663 images for training and 10% (10407 images) for evaluation during the training phase. Hence the number of steps per epochs and evaluation. The test data set has 15606 images with the same resolution as training data.

	ImageId	EncodedPixels
	192556 unique values	[null] 65% 43801 1 44567 4 ... 0% Other (81721) 35%
24	00052ed46.jpg	
25	000532683.jpg	458957 14 459725 14 460493 14 461261 14 462029 14 462797 14 463565 14

Figure 2-6: Example of Run-Length-Coding mask

2.4 Hardware Configuration

For the Maks R-CNN section of the project, a system with following configuration was employed:

- CPU: 12 Cores (24 Threads), each core running at max clock of 4 Ghz (AMD Threadripper 1920x)
- RAM: 32 GB DDR4 @ 3200 MHz
- GPU: Nvidia GTX 1080ti with 11 GB of GDDR5X video memory (Pascal with 3584 CUDA cores and boost clock of 1950 MHz)
- OS: Ubuntu 18.04
- Development Environment: Inside a Docker container generated via official TensorFlow docker image

2.5 Framework Configuration

As backbone, a RestNet50 was employed in Model. As a side note, it should be mentioned that we have implemented a ResNet20, a residual Network with 20-layer residual, for gaining a better performance in training. In comparison to ResNet50, we have improved the training performance per epoch on the configuration which was mentioned in previous section(2.4), but not significantly. Prior to these changes, the model with all its layer was already trained for 6 epochs with relative good results. Due to the insignificant

improvement in performance and advancement in training, we decided to continue with the RestNet50. Arguably, a residual network is more suitable for detection of more complex objects and more number of classes. For this project, a much simpler neural network could be also utilized with even better performance. But this was also a motivation to observe if sacrificing some performance is an acceptable trade-off for gaining more accurate detection. Further configurations of the framework are as follow:

- Backbone: ResNet50
- Batch size: 1 (1 image per GPU because of the resolution of the images)
- Number of epochs: 30 (Early stoping after 6 epochs)
- Number of steps per epoch: 93663 (due to the batch size and 90% of training data set for training)
- Number of validation steps: 10407 (due to the batch size and 10% of training data set for evaluation)
- Number of RoIs per image: 200
- Positive RoI ratio: 0.33
- Learning rate: 0.001
- Learning momentum: 0.9
- Weight decay: 0.0001
- Detection minimum confidence: 0.90
- Mask resized: from 768x768 to 384x384
- Image augmentation: not employed

2.6 Data Preparation

Initially, the images are loaded together with their corresponding masks.

Figure 2-7: Image with mask

The bounding boxes are then computed from provided masks. To improve the training performance, the mask pixels inside the computed bounding box is stored rather than the mask of entire image. This mask is further resized to smaller size, in our case half of the original resolution. Choosing smaller masks leads to losing smaller objects. This is governed by ***mini_mask*** in config class.

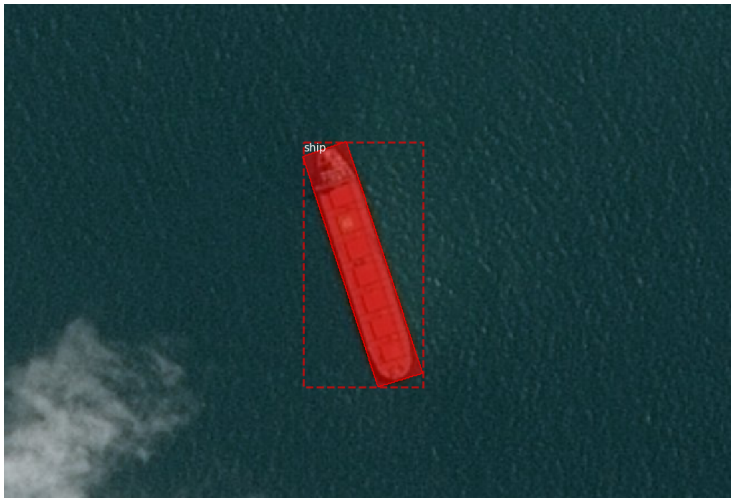

Figure 2-8: Iimage with bounding box

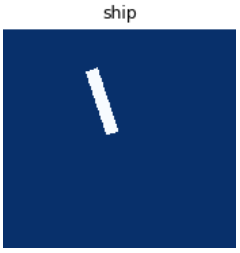

Figure 2-9: The respective Mask

During the training, the method ***data_generator*** of the class model generates the following data for each instance of an object, in our case ship, on the image:

- The RPN generates the anchors using the sliding-window method. Initialized by method ***build_rpn_model***, governed by ***RPN_ANCHOR_SCALES*** and ***RPN_ANCHOR_RATIOS*** of config class. The RoIs are generated(Fig. 2-11) and consequently the positive anchors and their refinement(Fig. 2-10).
- Based on the positive RoIs, the classifier calculates the bounding box of an object (dotted box) and applies refinement (box with solid lines) to it(Fig. 2-8). The classifier is initiated via the method ***fpn_classifier_graph()***
- The mask branch takes the positive regions selected by the RoI classifier and generates masks respectively(Fig. 2-9). This branch is a convolutional network and is initialized by the method ***build_fpn_mask_graph()***

Figure 2-10: Image with positive anchors

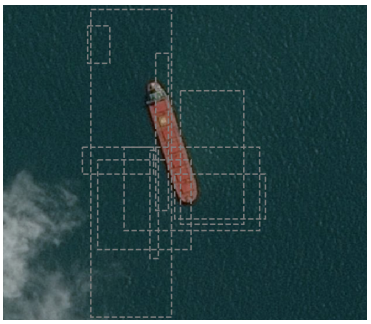

Figure 2-11: Image with RoIs(showing 10 random RoIs out of 200)

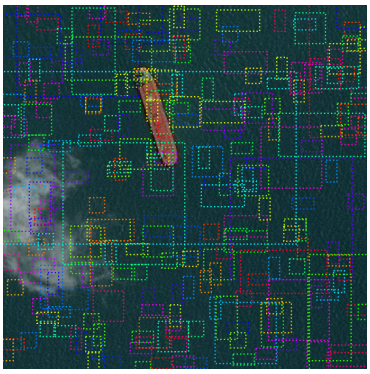

Figure 2-12: Image with negative anchors

2.7 Training

With the described configuration in earlier sections, each epoch took about 9 hours to complete. Due to the time constraint and relative good results, the training was stoped early after 6 epochs.

2.8 Detection

The detection method of the class model predicts/detects the ships and outputs the masks and bounding boxes respectively. Some examples of generated masks are shown in the following images:

Figure 2-13: Grey Image

Figure 2-14: Mask of image

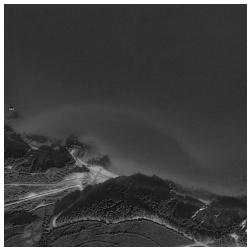

Figure 2-15: Image with small ship

Figure 2-16: Mask of smaller ships

2.9 Evaluation

2.9.1 Loss

The losses of various layers of the model were recorded during training and validation and are shown in the following graphs.

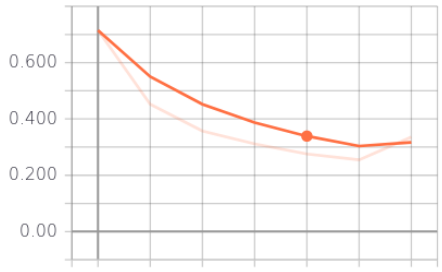

Figure 2-17: Train loss(Cross-Entropy)

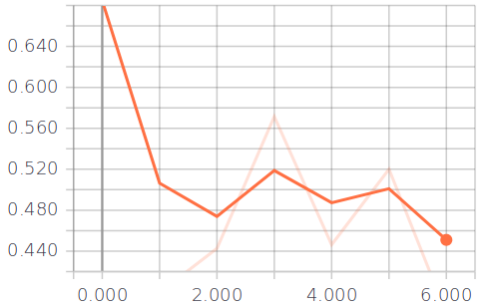

Figure 2-18: Evalutaion loss

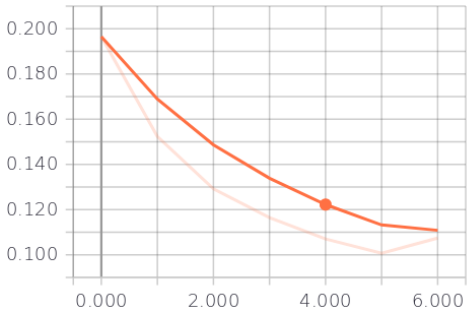

Figure 2-19: Mask loss (Cross-Entropy) training

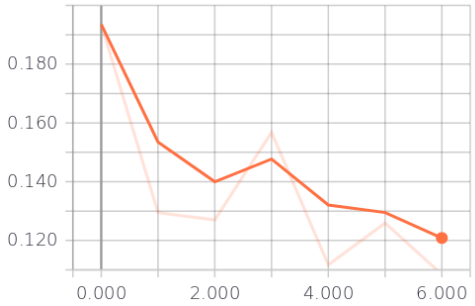

Figure 2-20: Mask loss evaluation

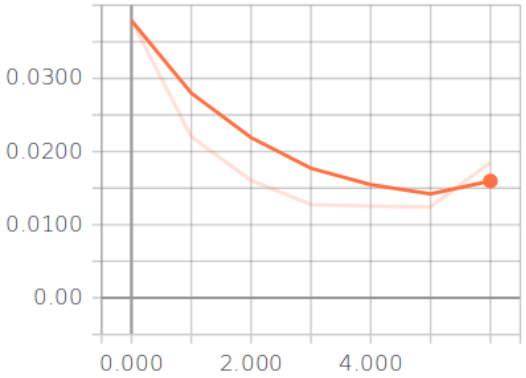

Figure 2-21: Class loss (Cross-Entropy) training

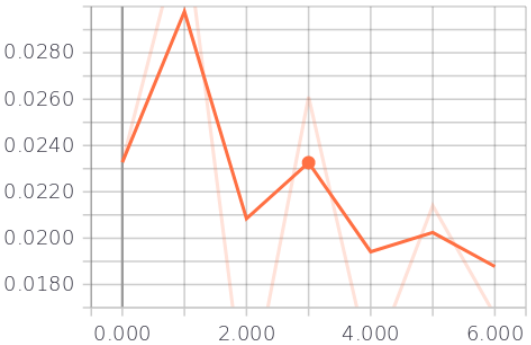

Figure 2-22: Class loss evaluation

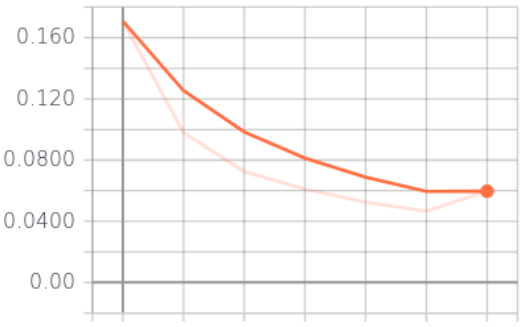

Figure 2-23: Bounding box loss training

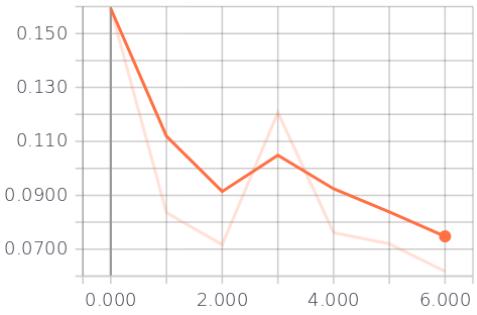

Figure 2-24: Bounding box loss evaluation

2.9.2 Kaggle Score

The predicted masks were encoded to Run-Length values and those values were saved in a csv file in the form of 2-6. The predictions were generated for 3 sets of test data and uploaded. The data sets and their respective results are as follow:

1. The entire test data, namly 15606 test images. The prediction/detection took about 1822 Seconds. The prediction generated the score 0.671 and positioned us at number 117

2. the pre-classified test data with threshold 0.4 consisting of 10164 images. The prediction process took 1217 Seconds and scored 0.674 which improved our ranking to 116

3. the pre-classified test data with threshold 0.5 consisting of 5414 images. The prediction process took 648 Seconds and generated the same score 0.674

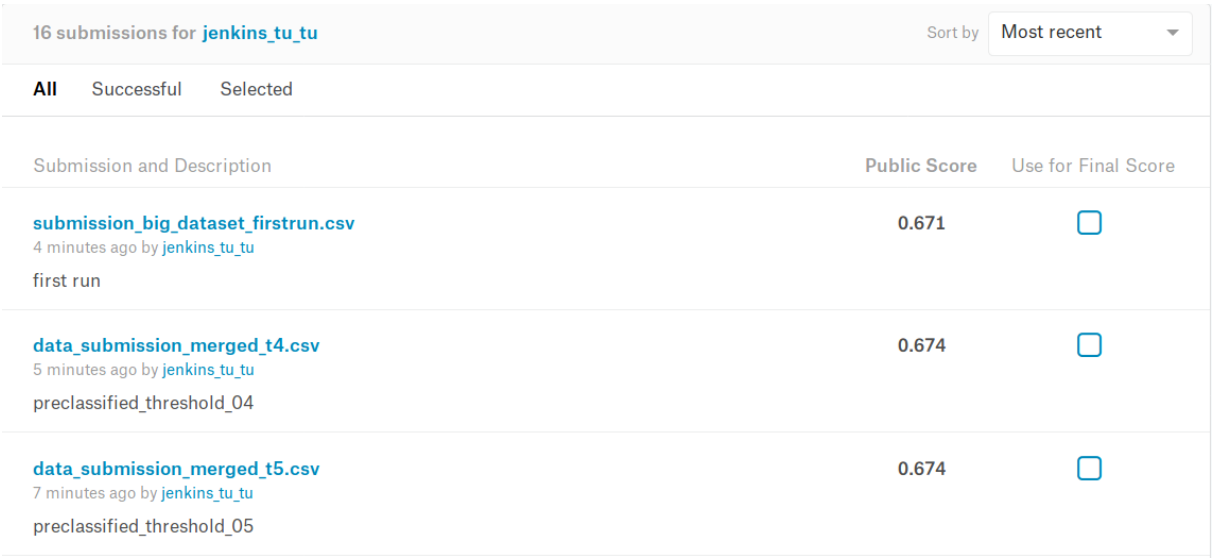

Figure 2-25: Submission Scores

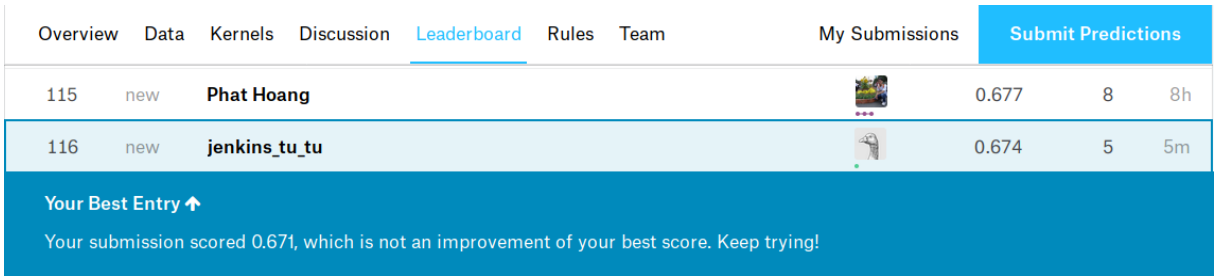

Figure 2-26: Kaggle Ranking

2.9.3 Summary

Based on the evaluation and loss values, it is obvious that this model can provide good results if it's trained adequately.After the 6th epoch, the values of evaluation losses have declined to a good extent, which indicates an improvment in learning the data.

References

[1] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[2] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[5] K.V. Kale, S.C. Mehrotra, R.R. Manza, and R.R. Manza. *Computer Vision and Information Technology: Advances and Applications*. I.K. International Publishing House Pvt. Limited, 2010.

[6] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[7] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015.

[8] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017.

[9] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Waleed Abdulla. Splash of color: Instance segmentation with mask r-cnn and tensorflow. https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow\-7c761e238b46. Last accessed 09/30/2018.

[12] Kaggle Inc. Data leak and next steps. https://www.kaggle.com/c/airbus-ship-detection/discussion/64388. Last accessed 10/15/2018.

[13] Kaggle Inc. Airbus ship detection challenge introduction. https://www.kaggle.com/c/airbus-ship-detection#description. Last accessed 09/03/2018.