Airbus Ship Detection Challenge

Gurpreet Singh

May 19, 2019

1 Summary

We participated in an inactive with late submission Kaggle competition, Airbus Ship Detection Challenge. Ship detection has a wide range of real life application, in the areas of maritime traffic, monitoring of marine pollution, border control, etc. The challenge is to detect ships in satellite images and generate mask images of ships as shown in Figure 1. The final model we choose is a transfer learning model for image classification to classify if an image contains a ship or not and Mask R-CNN, a deep convolutional neural network architecture, for detection and segmentation of ships. Mask R-CNN is a flexible framework developed for the purpose of object instance segmentation. This model is an implementation of this Mask R-CNN technique on Python and Keras. It generates bounding boxes and segmentation masks for each instance of an object in a given image. We implement the convolutional neural network using the Mask-RCNN provided by Keras and run the code on a Google Cloud Platform with four Intel i7 CPUs, 32 GB memory and 1 NVIDIA Tesla T4 GPU. In the public leaderboard, our score is 0.67534. In the private leaderboard, our score is 0.81560. There are over 900 submission, and we are around the 800th position in the private leaderboard and 500th position in the public leaderboard.

2 Problem Description

One of the most exciting applications of deep learning is the ability for machines to understand images. There are four main classes of problems in detection and segmentation: Semantic Segmen-

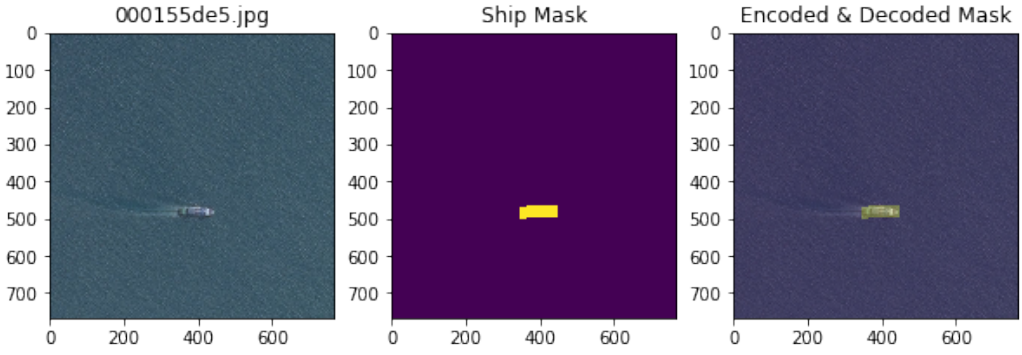

Figure 1: Ship Detection

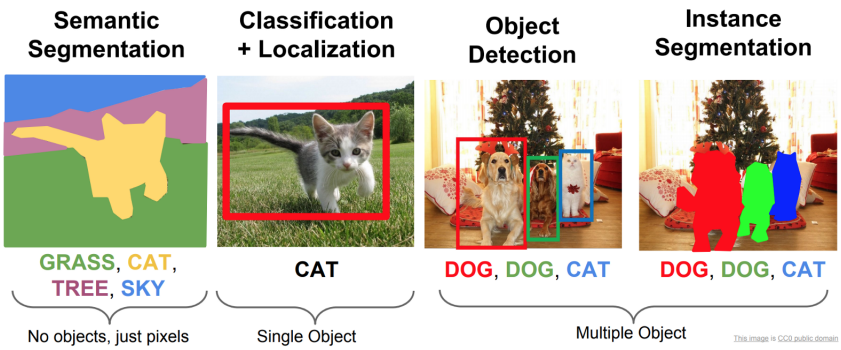

Figure 2: Fei-Fei Li Stanford Course — Detection And Segmentation

tation, Classification + Localization, Object Detection, and Instance Segmentation. The classes are shown in Figure 2. Ship detection in satellite images is an instance segmentation problem because we want to do pixel level segmentation. But we also need good object detection to do proper pixel segmentation.

Problem. The problem is to find ships from satellite images as quickly as possible. Airbus is looking into this problem to have a closer watch over the open seas because shipping traffic is growing fast. As more ships are in the sea, countries want to monitor the sea for illegal activities, such as illegal fishing, drug trafficking. This challenge looks to address the problem of automatic accurate and fast detection of ships in satellite images.

The first task is classify if an image has a ship or not and this is a binary classification problem. If an image has a ship, then the second task is to detect and segment the ship in the image. This is a detection and segmentation problem. The competition is at `https://www.kaggle.com/c/airbus-ship-detection/`.

Data. The dataset contains 208,162 768 satellite images with a total size exceeding 28GB. The dataset is split into 192,556 training images and 15,606 testing images.

Challenges. There are a number of challenges with detecting ships from satellite images.

- Data is imbalanced. 78% of the images in the training set do not have ships in them.
- Generating a pixel level mask and a bounding boxes for each ship.
- Clouds, haze or other atmospheric phenomena that may hide ships.
- Ships may be partially visible, variation in ship sizes (very small to large), multiple ships.

3 Solution

Model. We do transfer Learning with ResNet50 for image classification on if an image contains ships or not. If an image contains a ship then we use another model for ship segmentation called

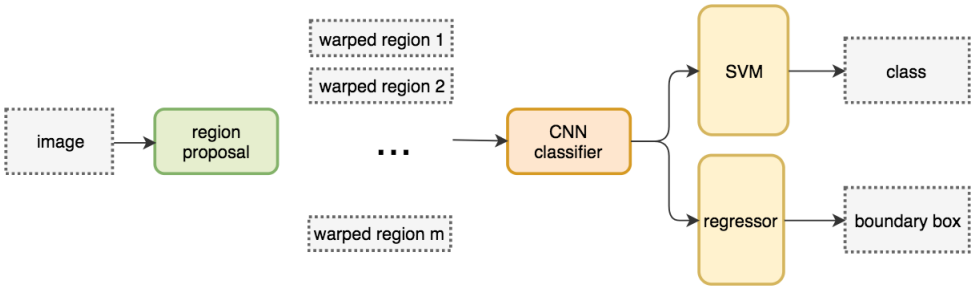

Figure 3: R-CNN

Mask R-CNN (Regions with CNN features). Mask R-CNN is selected for this project because it can generate high-quality segmentation mask for each object in an image. Mask R-CNN is extension on the Faster R-CNN for pixel level segmentation. Faster R-CNN is an extension of Fast R-CNN which is an extension on R-CNN. We briefly describe the methods in the next few paragraphs.

The simple approach to solve the segmentation problem would be to take different regions of interest (ROI) from the image, and use a CNN to classify the objects within that region. The problem with this approach is that the objects of interest might have different locations within the image and the objects may be different size. In 2014, [1] came up with R-CNN, a method where we use selective search to extract 2000 regions called region proposals from the image. Once the region proposals are selected, R-CNN warps the region to a standard square size and passes it through a CNN to compute features. On the final layer of the R-CNN, a Support Vector Machine (SVM) is used t classify if the region is an object, and if so which object is it. Figure 3 shows the R-CNN method. R-CNN is slow in training and inference.

The drawbacks of R-CNN was addressed by the same author in [2] to build a faster object detection algorithm called Fast R-CNN. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a feature map. From the feature map, we select the region of proposals using selective search, apply RoI pooling to warp them into fixed dimension and then feed them into a fully connected layer. Then we use a softmax layer to predict the class of the proposed region. Fast R-CNN is faster than R-CNN is because you don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it. Fast R-CNN is significantly faster in training and testing over R-CNN.

R-CNN & Fast R-CNN both uses selective search to find out the region proposals. Selective search is a slow and affects the performance of the network. [3] proposed Faster R-CNN, where instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. Figure 5 shows Faster R-CNN.

Mask R-CNN was proposed by [4] and it extends Faster R-CNN. In Mask R-CNN, a fully convolutional network (FCN) is added on top of the CNN features of Faster R-CNN to generate a mask or a segmentation output. Figure 6 shows Mask R-CNN.

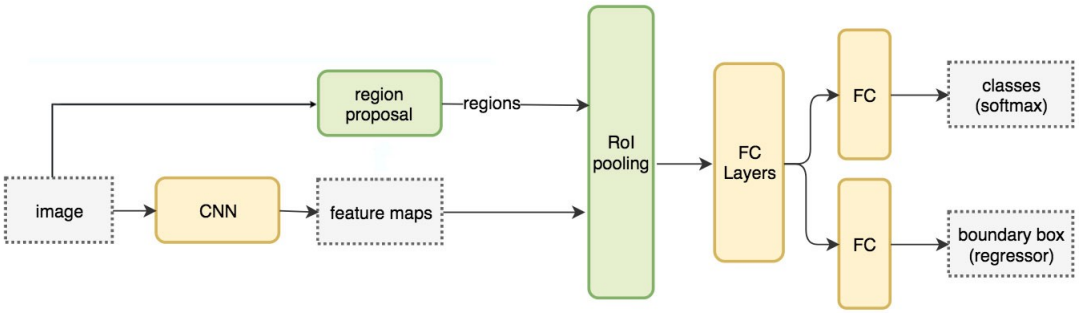

Figure 4: Fast R-CNN

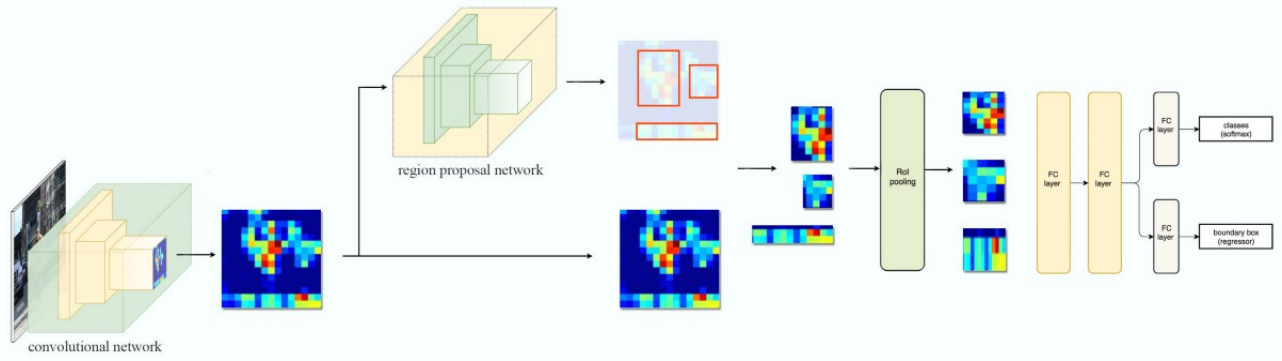

Figure 5: Faster R-CNN

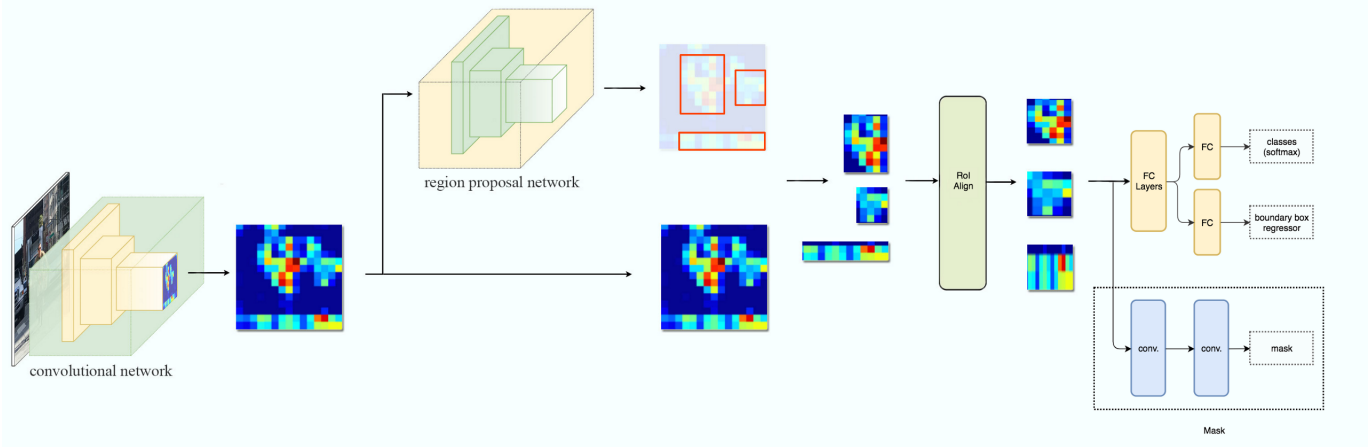

Figure 6: Mask R-CNN

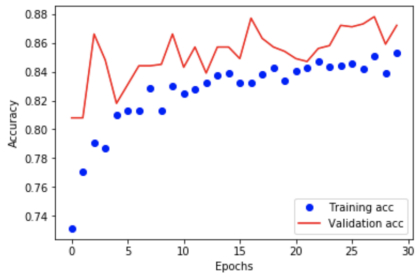

Figure 7: Mask R-CNN

Implementation. We directly use the Mask R-CNN model provided by [5] using Keras with TensorFlow as the backend. Our code is available at `https://github.com/gopysingh/CS585-Project`. We run the code on a Google Cloud Platform with four Intel i7 CPU, 32 GB memory, and 1 NVIDIA Tesla T4 GPU. It takes 1 hour to train the classifier model and it takes 3.3 hours to train the detector model.

Settings. We created two models: classifier and detector. The classifier used ResNet50 model for training. We used the softmax in the final layer to give probability if the input image contains ships or not. The classifier used a SGD optimizer. The loss function is categorical cross-entropy. The learning rate is 0.0005. The epochs is 50. The batch size is 32.

The detector used the Mask R-CNN Keras implementation from [5]. The are several settings for the Mask R-CNN. We used most of the default settings, and they can be seen in the ShipDetection.ipynb notebook. We used 50 epochs with 300 steps per epoch. Mask R-CNN used ResNet101 as the backbone. The learning rate is the 0.001. Most other settings were left as default.

Advanced tricks. We used pre-trained weights for Microsoft COCO for a better starting point for training of the detector. COCO is a large-scale object detection, segmentation, and captioning dataset.

Cross-validation. For the classifier, we partition the training data to 80%-20% for hyperparameter tuning. Figure plots the the convergence curves on 80% training data and 20% validation data. The training accuracy is close to the validation accuracy however the validation accuracy looks better. We tried various different values for hyper parameters but this is the best accuracy that we achieved. Our classifier model is slightly an under fit of the validation data. The graph is shown in Figure 7.

For the detector, we also partition the training data to 80%-20%. The training and validation losses are shown in Figure 8. It looks like we under fit the training data. We expected the output to see the curves decrease for all the graphs but they look they unstable. This may be due to the hyperparameters we choose.

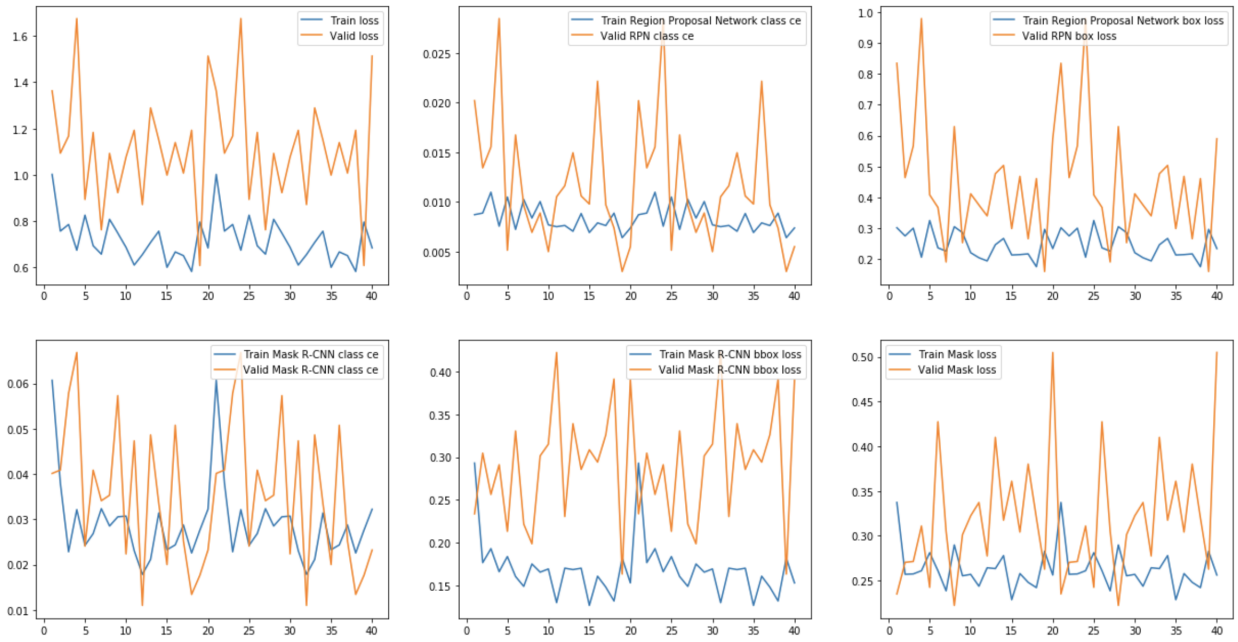

Figure 8: Mask R-CNN

4 Compared Methods

At first, we implemented a simple CNN with several hidden layers for image classification. The accuracy that we achieved with the simple CNN was around 80%. To achieve better accuracy we looked into image classification with transfer learning. Using transfer learning, we achieved 87% accuracy on our test and 89% accuracy on the validation set.

We only implemented the Mask R-CNN method for the detection and segmentation part for this project. However, many other participators in this competition used a U-Net and inception model for the detection and segmentation. U-Net is another convolutional neural network originally designed to perform medical image segmentation but it works well on a wide variety of tasks, including detecting ships in satellite images. We did not implement the U-Net but some high ranking participants achieved great results with using the U-Net along with the inception model.

5 Outcome

We participated in an inactive competition. In the public leaderboard, our score is 0.67534. In the private leaderboard, our score is 0.81560. Our name does not show in the leaderboard because the competition is inaction. There are over 900 submission, and from inspection of the leaderboard we are around the 800th position in the private leaderboard and 500th position in the public leaderboard. The screenshots of our scores are shown in Figure 10.

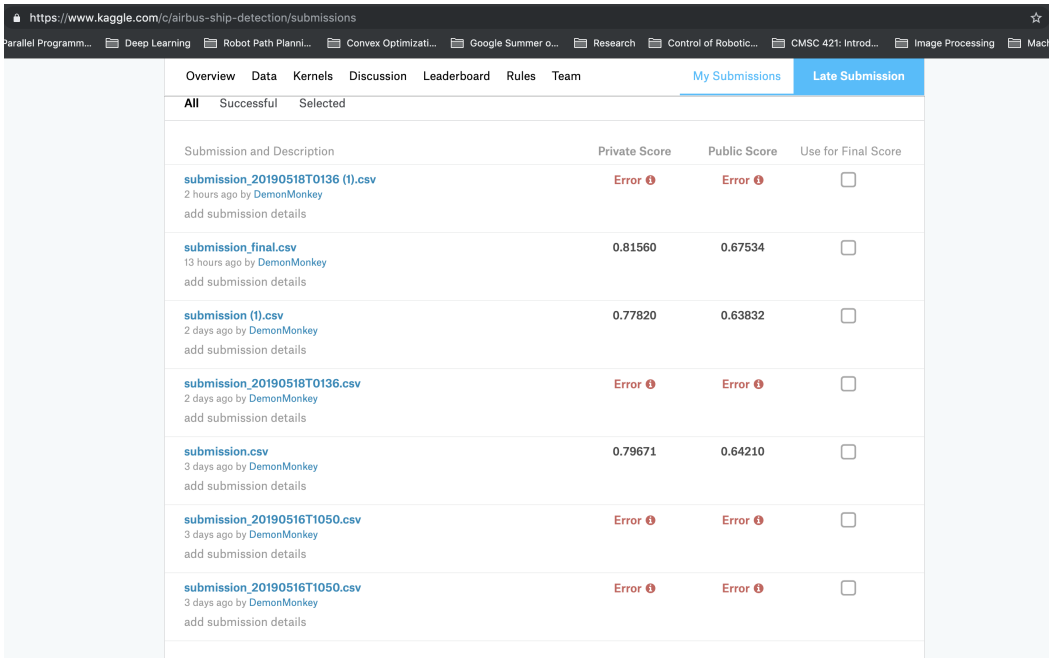

Figure 9: Leaderboard

Figure 10: Sample Output 1

Figure 11: Sample Output 2

Figure 12: Sample Output 3