MPCS 53112 Advanced Data Analytics

Project Report

Airbus Ship Detection Challenge

Jingxuan Wen | 12249862 | jingxuanw@uchicago.edu

---

1. Abstract

Airbus Ship Detection Challenge is a Kaggle challenge proposed by a company providing maritime monitoring services. The goal of this challenge is to detect ship in satellite images. In this project, I trained a Mask R-CNN network to solve both detection and segmentation problem. The training and test process is conducted on Detectron2, a powerful deep learning platform proposed by Facebook research team. My best model succeeds in lowering the loss to 0.185 (±0.02) in the segmentation problem after the 19-hour training process, which ranks around the 18th on the Kaggle Leaderboard.

2. Problem

I solved the ship detection challenge with a Mask R-CNN model in the project. To be specific, the problem is to detect ship and segment the detected ship out from satellite images, which are taken from high angle shot and cover various whether conditions like cloud and haze. I have two main missions: detection and segmentation.

The detection problem is to detect whether there is one or more ship in the input image and to get the position of each of them. The result returned by detection process is represented by bounding box, which is a rectangular outline containing the detected ship for each ship.

The segmentation problem is to predict a set of pixels for each ship in the input image that all pixels in the set belong to the the ship and all pixels not in the set don't belong to the ship. This set of pixels is called a mask. Intuitively, a mask indicates the shape and the position of a ship.

For each sample, the input is a 768*768 colorful image of ocean which may or may not contain any number of ships. The expected output is the same image but adding a bounding box and a mask for each ship. All data are provided by Kaggle. The training dataset contains 193k images and corresponding annotations. The test dataset contains 15.5k images. The evaluation method is the F2-Score under different IoU thresholds. The IoU stands for Intersection over Union. If the segmentation mask produced by the model has an intersection with the ground truth mask over a certain threshold, then the prediction is correct.

Input image

Output prediction. The vertical rectangle outline is the bounding box predicted by the detection process. The highlighted area of the ship is the mask generated by the segmentation process.

The problem is of high value both practically and theoretically. Practically, it provide a solution for maritime monitoring. As shipping traffic is growing fast, it also 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. Accurate detection and segmentation of ships is the base for more diverse applications.

Theoretically, object detection and image segmentation are important subjects in computer vision. They usually work together to simplify an image and more efficiently analyze it. They have been researched by many scholars in a long time and attracted the enthusiasm in related fields again recently because of the innovative application of deep neural network. This project is based on Mask R-CNN, one of the most significant contribution in this field, and shows how a well trained model can level up the precision of the solution to these challenging problems.

Speaking of the traditional subject in computer vision field, the problem in this project has some unique features different from the original background of detection and segmentation problems. The most significant characteristic of this practical project is that the satellite images have uncertain orientations. The ship may face to any direction because the image is taken from high angle shot. So the variety of the orientation in the training dataset is necessary for the generalization of the model. I address this difficulty by augmenting the data during the training phase. There will be a detailed explanation of data processing in the later chapters as well as a contrast test showing how much the data augmentation affects the performance of the system.

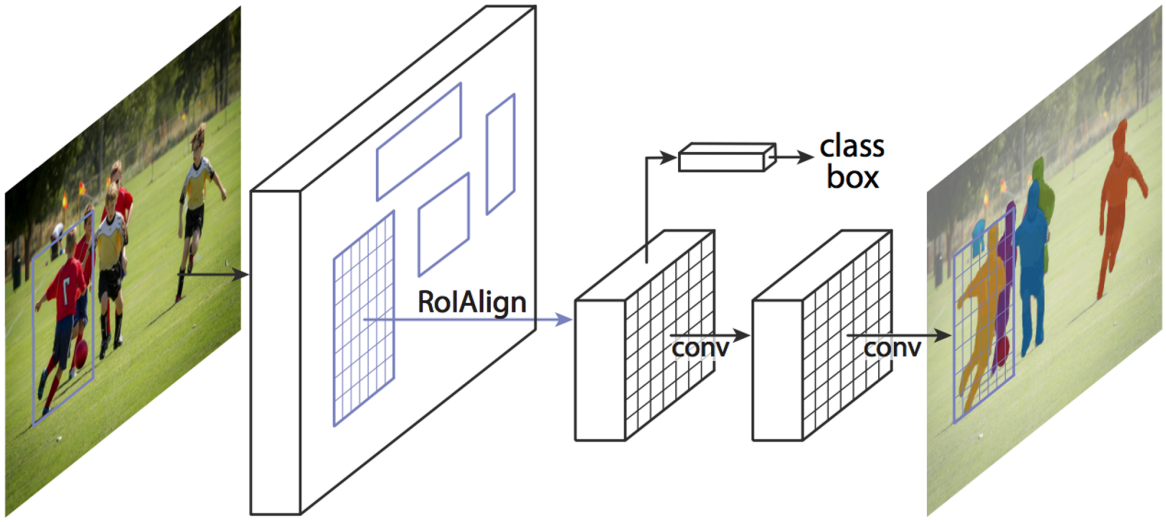
3. Related Work

As this challenge is held in the last year. There are already a few researches on this specific problem. However, after reading several of them. I find them quite similar to each other and most of them mainly focus on detailed parameter settings only functions under their training environments. So I decided to dig into the model itself and find my own solution. I read the original paper of Mask R-CNN by Kaiming He et al. very carefully and based my own solution on their state-of-art model. This is the most related work to my solution. I put the formal introduction of Mask R-CNN in the next chapter, together with an overview of the whole pipeline of my system. Other implementation details are places in the chapter after the next.

4. System

I applied Mask R-CNN as my model and constructed several other tools to provide a complete pipeline from data processing to result analysis. The system is in classic deep learning style and consists of following components: data processing, model constructing, model training, model testing, and result analysis. I'll describe the structure of the core model, Mask R-CNN, in this chapter and explain other implementations and settings in the later chapters.

Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision. It's the state of art model proposed by Kaiming He in 2017. It detects objects in the input image, generates segmentation mask for each object, and classify the detected objects. In this project, I only need its first two functions. But it's actually capable of more.



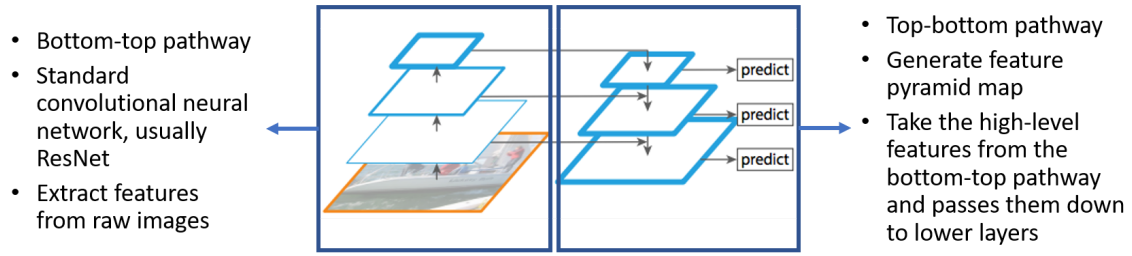
Mask R-CNN framework

The backbone of Mask R-CNN is a Feature Pyramid Network (FPN) style deep neural network. The Feature Pyramid Network structure consists of a bottom-top pathway, a top-bottom pathway and inter-connections between these two pathways.

The bottom-top pathway can be any standard convolutional neural network. In our case, it's a residual neural network (ResNet), a popular convolutional neural network model proposed by Microsoft Research team. It extracts features from raw images. Different layers will get features of different resolutions. For example, the bottom of the network may get low-level features like points and lines while the top layers may extract the contour of ships.

The top-bottom pathway takes the high-level feature map extracted by the bottom-top pathway as the input and generates the feature pyramid map by fusing the high-level features with features extracted by each layer of the network. Each feature map in the feature pyramid has the same size as their corresponding layer in the bottom-top pathway.

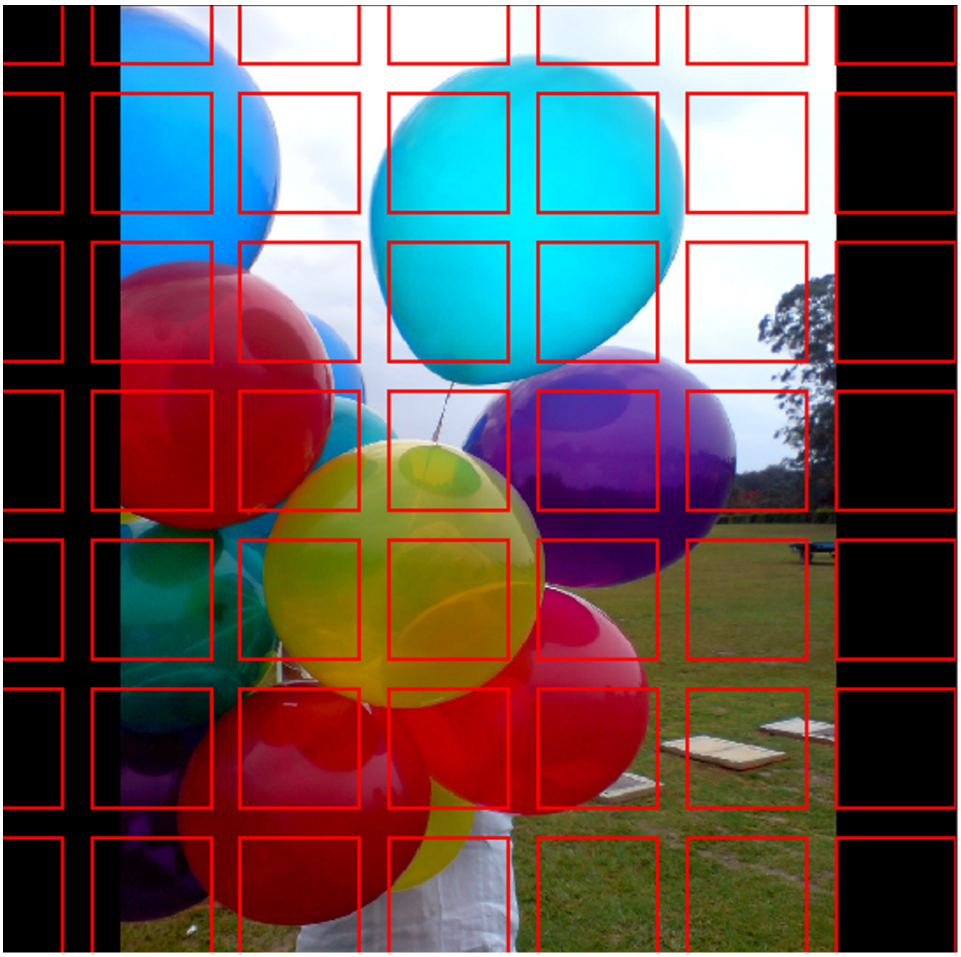
FPN outperforms other single convolutional neural networks because it maintains strong semantically features at various resolution scales, and it allows layers at every level in the convolutional neural network to have access to both lower-level and higher-level features.



FPN framework

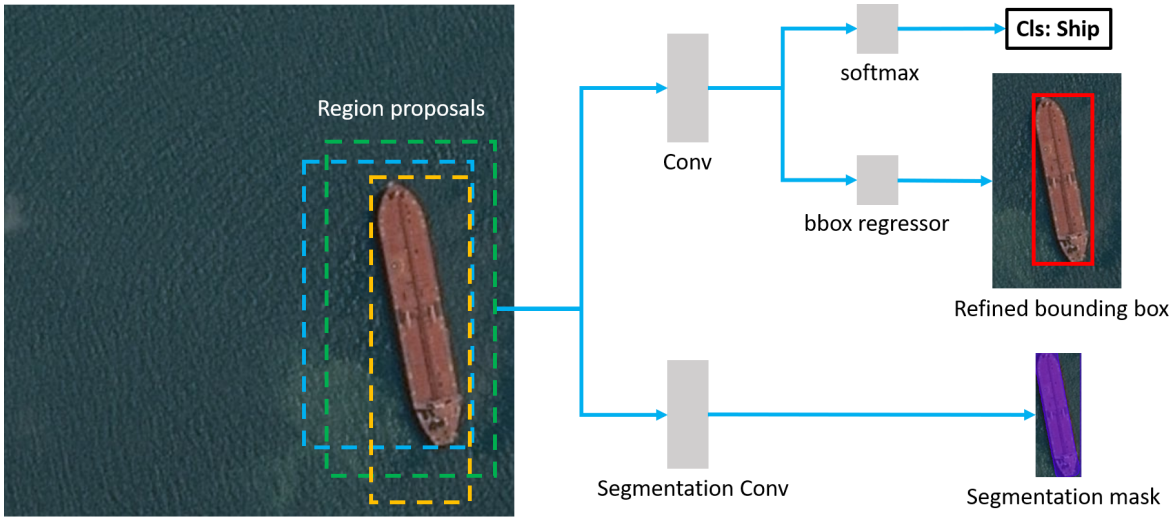
Mask RCNN conducts its functions in two steps. First, it generates proposals about the regions likely to contain an object on the input image. Second, it predicts the class of the object, refines the bounding box and generates the segmentation mask of the object based on the first stage proposal.

The first step is completed by a network module called Region Proposal Network (RPN). It is a lightweight neural network that scans all feature maps of top-bottom pathway in a sliding-window fashion and finds areas that possibly contain objects. The regions that the RPN scans over are called anchors, which are boxes of different sizes and aspect ratios distributed over the feature map. They overlap each other to cover as much of the feature map as possible.



Simplified illustration of anchor boxes (red boxes)

The second step is another neural network that takes the regions proposed in the first step and assigns them to several specific areas of a feature map. Then it scans these areas, and finally generates object classes, bounding boxes and segmentation masks. In this project, we only have one class for objects, which is ship. So only the refined bounding boxes and the segmentation masks contribute to the final goal.



Step 2 framework

Hereto we have the knowledge of Mask R-CNN framework, what problem it solves, and how it solves them. In the next chapter I will introduce the implementation of other components in the system.

5. Implementation

(1) Data Processing

The first step in the whole pipeline is data processing. The way I pre-process the data in this project can be divided into two parts, data augmentation and data formatting.

First is data augmentation. As mentioned at the end of the Chapter 2, there is a problem when dealing with satellite images. That is, the ships may face to any direction because the satellite images have uncertain orientations. Therefore, we need to augment the training data to improve the orientation robustness of the model. Besides, there are various weather conditions in satellite images, so the model need to tolerate shifty brightness and contrast ratio. This is also addressed in data augmentation.

I augment the data in several manners. First is rotating images by a random multiple of 90 degrees. The second is horizontally flip the image. By this two way I can theoretically achieve plenty of orientations for ships. I also adjusted the brightness and the contrast ratio of the image to improve the model's robustness against different whether conditions in images. All these four augmenting methods are applied on each image in the training dataset with some certain probabilities to further introduce randomness into the system.

Augmented data of a given sample. On the left is an image in the training dataset and on the right is the augmented result by four methods

After augmenting the data, I convert it from run-length-decoding to COCO format for the convenience in training process. COCO dataset is a dataset introduced by Microsoft Research team for object detection and segmentation. It is a very common-used dataset and its annotation format is hereby widely adopted by popular deep learning research platforms.

I also clean the dataset during the converting. First, I remove the images without ship. This is because according to related research, this kind of images contributes little in the training phase while having a negative effect on the convergence of the model. Second, I remove images with low quality of annotation for the similar reason. The quality of the annotation is mainly decided by the size of the mask in an image. It can be concluded from former researches in image segmentation that if the object is too small in some training samples, then these samples can hardly help with any feature extraction but apply a long-lasting penalty in evaluation phase because it's very difficult for the model to detect objects far away from the center of its cluster in the feature map. If the training goes to extreme to cover all samples, even these outliers, then the model could be terribly biased and inevitably over-fits the training dataset. These removed samples will be put back in the late phase of training to refine the model but not strikingly shift its weights.


```
{
    "id": 8623,
    "image_id": 4526,
    "category_id": 1,
    "iscrowd": 0,
    "area": 104,
    "bbox": [
        257.0,
        433.0,
        14.0,
        8.0
    ],
    "segmentation": [
        [
            269.0,
            440.5,
            257.0,
            440.5,
            257.5,
            433.0,
            270.0,
            432.5,
            269.0,
            440.5
        ]
    ],
    "width": 768,
    "height": 768
},
```


Annotation for a segmentation mask in COCO format

Above steps complete the data processing.

(2) Model Constructing

I encountered some issues when attempting to construct and train the model at the beginning. The biggest problem is that my computing device is RTX 2060, which demands CUDA 11.0 or above to conducting tensor computing. However, CUDA 11.0 is too new for most deep learning toolkits and frameworks, so for some early tests, I have to train the model with only my CPU, which is extremely time-consuming. This problem is finally addressed as I find out that Detectron2 provides APIs for CUDA 11.0 in its last version while maintaining a good support for Mask R-CNN. So I adopt Detectron2 as my model constructing and training platform.

Detectron2 is Facebook AI Research team's last generation software system that implements state-of-the-art object detection algorithms. It is a ground-up rewrite of the previous version, Detectron, and it originates from maskrcnn-benchmark, another algorithm system project by the same team.

Though Detectron2 provided a series of powerful tools to help model training and result analysis, I only use it to rapidly construct models of different configurations and use some of its APIs to make RTX 2060 contribute in the model training.

I select several models and training strategies with their configurations varying around three main factors: the backend network used in FPN, the number of region proposals in the first stage of Mask R-CNN and how the input data is augmented. The contrast experiments among these models prove some of my thoughts and introduce some interesting new findings that deepen my understanding of Mask R-CNN.

The configurations I select and test are as following:

- ResNeXt-101-32x8d as backend, non-augmented data, 512 region proposals per image
- ResNeXt-101-32x8d as backend, augmented data, 512 region proposals per image
- ResNet-50 as backend, augmented data, 512 region proposals per image
- ResNet-50 as backend, non-augmented data, 512 region proposals per image
- ResNet-50 as backend, augmented data, 128 region proposals per image

(3) Model Training

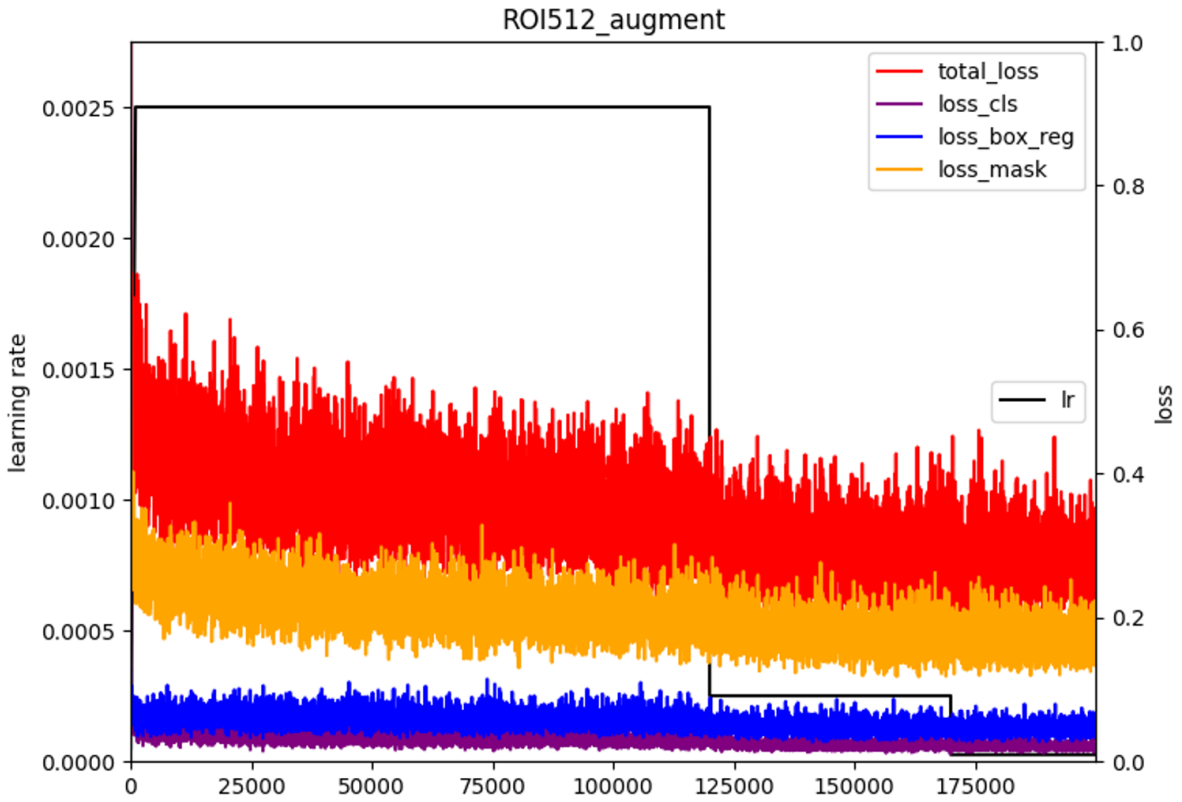
All models are initialized with weights pre-trained on ImageNet, an image dataset widely used for pre-training in detection and segmentation research area. All models are first trained for 20000 iterations to provide an initiatory performance feedback. This early-phase result is used for two parallel purposes: to conduct contrast experiments later and to select models with more potential for further training.

After the preliminary screening, I continue to train with two configurations:

- ResNeXt-101-32x8d as backend, non-augmented data, 512 region proposals per image
- ResNet-50 as backend, augmented data, 512 region proposals per image

The later configuration provide the best model.

As for the best model, I trained it for 200000 iterations with augmented data and succeeded to lower the loss of segmentation to 0.185 (±0.02). I use the loss of segmentation as the main indicator of model performance because segmentation is the most difficult task for the system and makes up the most penalty when evaluating the model. The whole process lasted around 19 hours and monitoring log is visualized as below:



The training process of the best model. The total_loss is an approximation to the sum of all other losses. The loss_cls is the loss of classification task, which it irrelevant in this project. The loss_box_reg is the loss of bounding box, which indicates the performance in detection task. The loss_mask is the loss of generated mask, which indicates the performance of segmentation task.

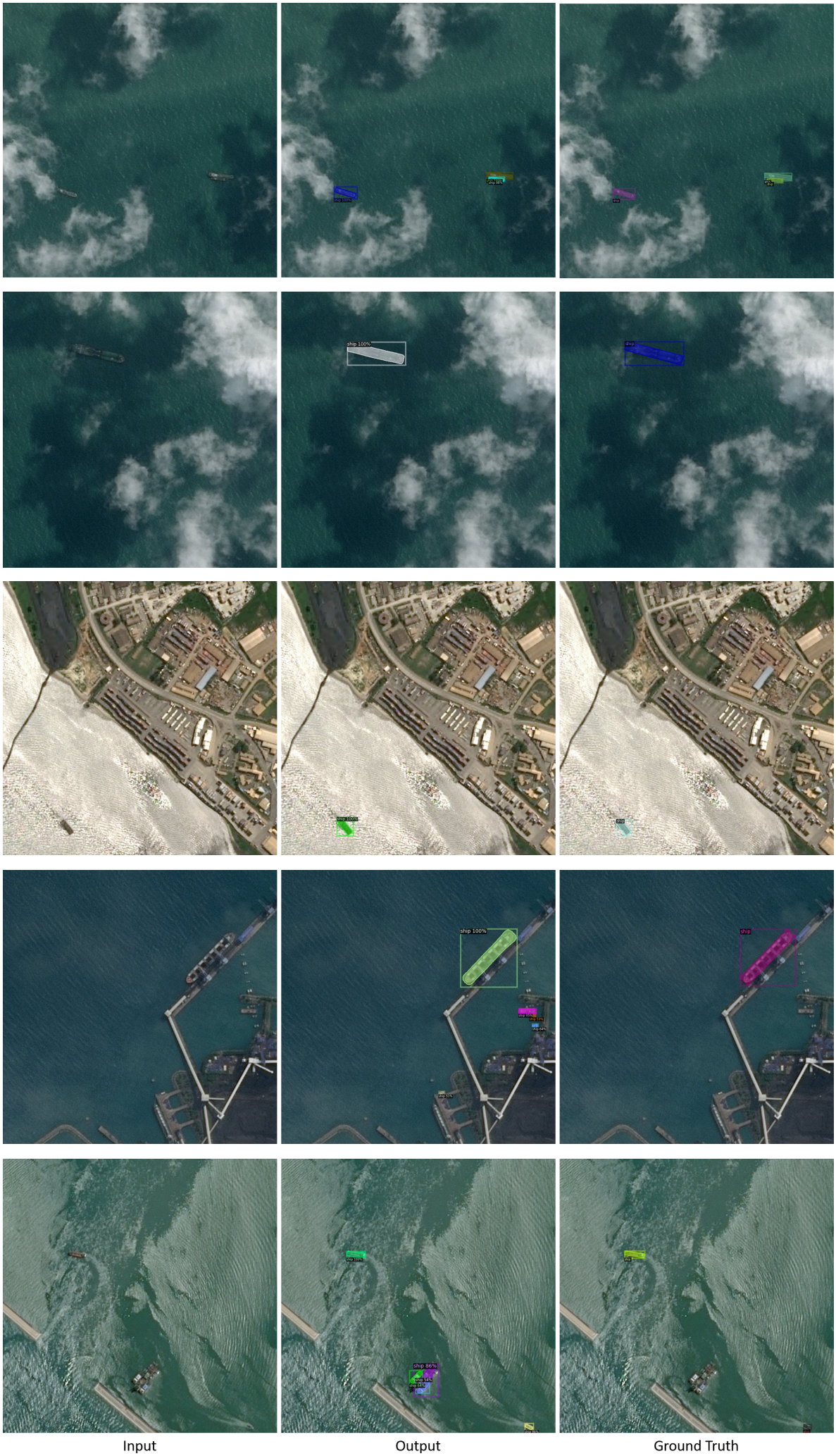
As we can see the detection task is easier than the segmentation task in a general way. However, the loss of these two task shows covariation, suggesting that this two process cooperate with each other. In other works, if the bounding box prediction is fine enough, then the most of wrong pixels are ruled out before the segmentation process starts and the generated mask is therefore more accurate. This observation is consistent with our expectation.

6. Results and Analysis

(1) Result

As the testing dataset is only for Kaggle evaluation and has no ground truth, I use part of the training dataset as the validation dataset. The samples in the validation dataset is not used in the training phase.

The best model has a relative high performance in all circumstances in general. However, different circumstances lead to difference level of difficulties. Below are some of the most challenging samples that gets my attention when testing the model:



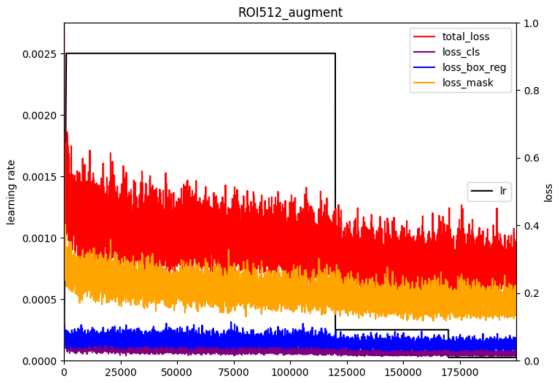
Input Output Ground Truth

These examples includes bad whether conditions like cloud and haze, multiple ships in the same image and images where ships are close to the coasts. The last condition, where ships are close to lands, is the most challenging kind because the in this condition, ship is not the only man-made construct. In most cases, the model is about to distinguish ships from buildings. But there are still some cases that buildings are recognized as ships, as you can see in the last two examples shown above.

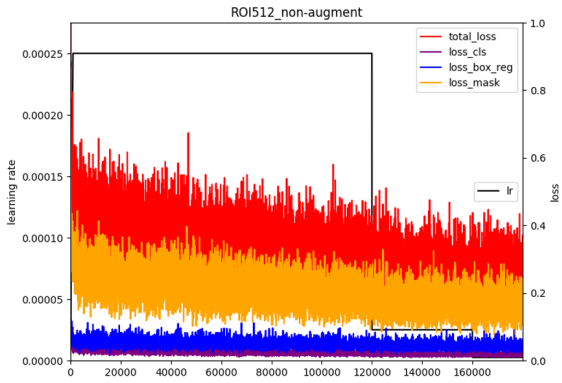
(2) Experiments and Analysis

Besides the model testing result for the best model, I also compare the different configurations mentioned above quantitatively in their performance and their tendency during the training phase. Some results of these contrast experiments are interesting and instructive.

I first tested how data augmentation affects the model performance. By logic, the model need to be trained on augmented dataset, especially containing images with various orientations. The first experiments proves this hypothesis. I compare the best two configurations. The only difference of them is that one is trained on augmented data while the other is trained on the original data. The visualization of their monitoring log show a huge difference with regard to the segmentation task.

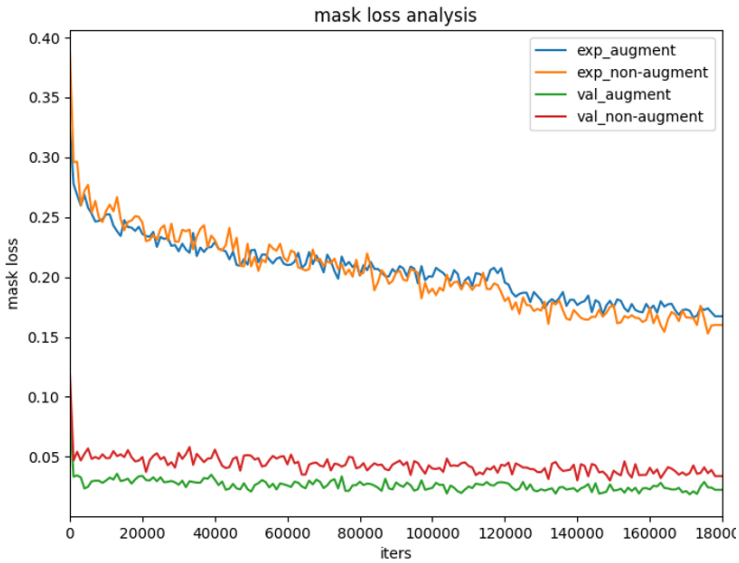


Trained with augmented data



Trained with non-augmented data

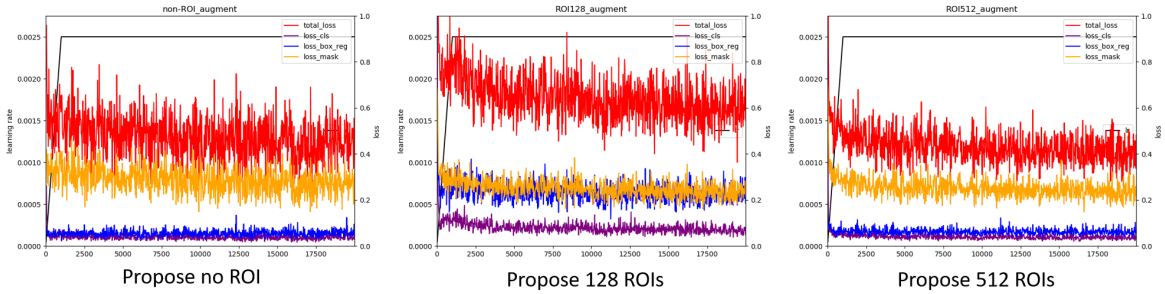
The loss_mask of the model on the left, which is trained using augmented data, has clearly less volatility then the other model. This suggests that data augmentation improve the model's robustness so it performance is more stable. The chart below shows this feature more clearly:



Quantitative comparison for loss_mask

This chart is generated by grouping each 1000 iterations together to compute the mean value and the variance. The mean value suggests that both model can achieve the similar final accuracy. However, the variance shows that the model trained on non-augmented data has a volatility about 1.5 times the other, which introduces instability to the prediction. The above experiment validates my intuition and tells that the positive effect of data augmentation is significant.

The second experiment is about the setting of RPN. I compared three conditions: propose no region of interest (ROI) in the first step of Mask R-CNN, propose 128 regions of interest, and propose 512 regions of interest. I trained them for 20000 iterations and an interesting phenomenon appears.



Propose no ROI | Propose 128 ROIs | Propose 512 ROIs

When using no ROI, the performance is quite unstable and loss_mask is high, but the loss_bbox_reg is low. When using 128 ROIs, the performance is even worse, with loss_mask slightly more stabilized but loss_bbos_reg increasing rapidly. This is counterintuitive. Why using ROI leads to a worse performance? The answer is that RPN is a lightweight network, whose role in the detection task can be implicit replaced by interlayers of the network. So the detection performance is good in the first model. However, by not providing any region proposals for the later steps, the segmentation performance suffers because it has to start with completely zero knowledge of the input image. This explains why using 128 ROIs improves the segmentation performance slightly when compared to using no ROI. And why the detection performance drops when using 128 ROIs? That's because the number of ROI proposals is insufficient and some of the object is not included in any ROI, while the bounding box prediction network only consider area covered by ROI proposed in the first step when using any number of ROI. This leads to the unusual performance in the second model. The last one works fine in any aspect, suggesting that 512 is a proper number for the region proposal setting.

7. Future work

For the future work. I would like to explore another solution to this problem using U-Net, a simpler network model designed for segmentation and detection. As we mentioned before, Mask R-CNN can detect and segment different kinds of objects and classify them, which is a function we don't need in this project. U-Net doesn't have this function, and therefore has a more light-weight structure. I would like to see its performance when dealing with this problem and whether all the experiences and findings I obtained from this project can still help.

8. Effort and Learning

My work in this project can be divided into several main portions listed below in the order of the system pipeline.

- Data Processing (30% code, 15% time)
    - Convert the raw data to COCO format
    - Implement augmentation functions
- Install Training Environment (10% code, 10% time)

    - Detectron2 with CUDA 11
    - Nvidia-docker
    - API toolkits (pycocotools etc.)
- Model Training (35% code, 50% time)
    - Experimental training to select configurations
    - Formal training (200000 iters, 19 hours)
- Analysis and Comparison (25% code, 25% time)
    - Test different configurations
    - Implement common tools
    - Data visualization and analysis

I learned a lot by doing this project. First I deepened my understanding to Mask R-CNN, a state-of-art model for object classification, detection and segmentation by reading its original paper carefully. This enable me to solve similar problems more effectively. Second, I got familiar with a range of common tools and platforms for deep learning research. I used COCO standard, Detectron2 and many other tools in this project, which improved my practical ability. Finally, by training the model and comparing them, I obtained more training skills for deep learning tasks and got a better vision of deep neural network in general. The most satisfying part to me during this project is that after solving multiple issues, I achieved a model ranking pretty high on the leaderboard. I feel my effort paid off rather than succeeding by luck. This encouraged my a lot.

9. Supplementary

My project is open-sourced at my github.

10. Bibliography

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 2961-2969. 2017.