In the Name of God

University of Tehran

Faculty of Electrical and Computer Engineering

Neural Networks and Deep Learning

Assignment 3

c

Assignment Details

Student Name	Mohammad Taha Majlesi
Student ID	810101504

Contents

List of Figures

1 Oriented R-CNN for Object Detection

1.1 Theoretical Foundations

1.1.1 Conceptual Understanding

a) The Core Problem and Proposed Solution As stated in the paper, the best-performing two-stage oriented object detection models generate oriented proposals for potential object regions. This process, however, is often computationally expensive and time-consuming, creating a bottleneck in the system. This paper introduces a model that claims to be superior to previous models in both performance and speed.

Over the past decade, there has been significant progress in the field of object detection. Due to numerous applications, a sub-field focusing on oriented object detection has gained considerable attention. Traditional object detection methods rely on axis-aligned bounding boxes, which do not fully cover oriented objects and lead to inaccurate localization. To address this, various approaches have been developed.

One of the earliest methods for generating oriented proposals was the Rotated RPN, which used 54 anchor boxes with various angles, sizes, and aspect ratios. While this method improved performance, the large number of anchor boxes led to high computational and memory costs. To tackle this, the RoI Transformer model was introduced, which learns oriented proposals from horizontal regions of interest. This reduced the number of anchor boxes and improved performance, but the network became heavy and complex, increasing computational costs.

Thus, a model that can efficiently generate oriented proposals is key to resolving the computational bottleneck in oriented object detection. This paper introduces a simple two-stage framework for oriented object detection, named Oriented R-CNN, which achieves better performance than other models while also being competitive in speed with single-stage models.

b) The Midpoint Offset Representation The paper introduces a novel representation for oriented objects called "midpoint offset." In this method, each oriented bounding box is represented by six parameters: $(x, y, w, h, \Delta\alpha, \Delta\beta)$. These parameters allow for the calculation of the coordinates of the four vertices of the proposed box (v_1, v_2, v_3, v_4).

- (x, y) are the coordinates of the center of the bounding box.
- (w, h) are the width and height of the box.
- $\Delta\alpha$ is the offset of vertex v_1 from the midpoint of the top edge of the bounding box. Due to symmetry, $-\Delta\alpha$ represents the offset of vertex v_3 from the midpoint of the bottom edge.
- Similarly, $\Delta\beta$ and its negative represent the offsets of vertices v_2 and v_4 from the midpoints of the right and left edges, respectively.

The coordinates of the four vertices are derived as follows:

$$
\begin{aligned}
v_1 &= (x, y - h/2) + (\Delta\alpha, 0) \\
v_2 &= (x + w/2, y) + (0, \Delta\beta) \\
v_3 &= (x, y + h/2) + (-\Delta\alpha, 0) \\
v_4 &= (x - w/2, y) + (0, -\Delta\beta)
\end{aligned}
$$

$$\begin{cases} v1 = \left(x, y - \frac{h}{2}\right) + (\Delta\alpha, 0) \\ v2 = \left(x + \frac{w}{2}, y\right) + (0, \Delta\beta) \\ v3 = \left(x, y + \frac{h}{2}\right) + (-\Delta\alpha, 0) \\ v4 = \left(x - \frac{w}{2}, y\right) + (0, -\Delta\beta) \end{cases}$$

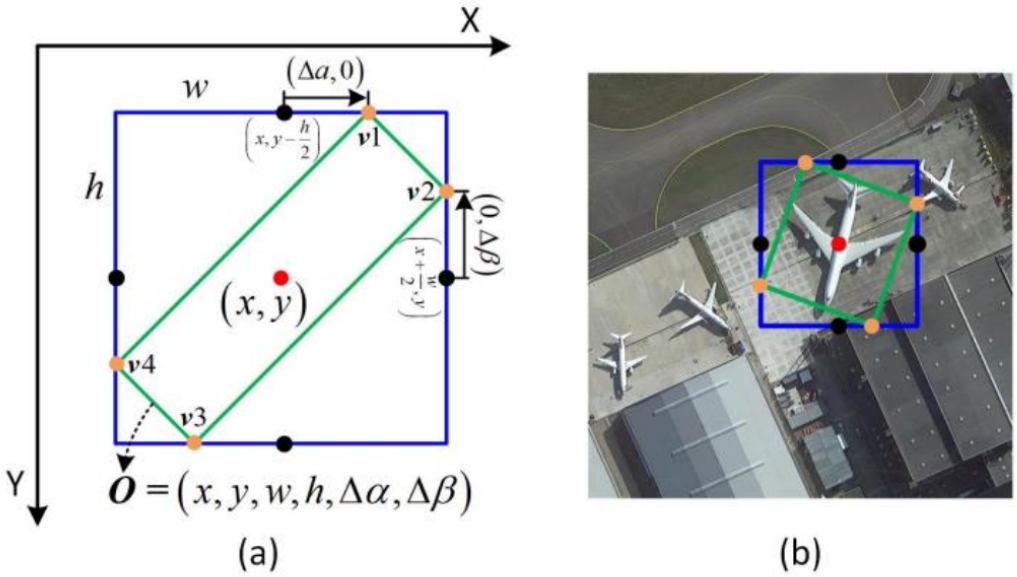

Figure 1: Illustration of the midpoint offset representation from Figure 3 of the paper. (a) shows the parameters, and (b) shows an example on an aerial image.

This representation, as shown in Figure 1, is very simple and, unlike traditional methods that often use an angle parameter, it avoids the use of angles. An oriented bounding box is defined using 6 parameters, all of which are length units. A key problem with using angles is their periodicity (e.g., 0 and 360 degrees are the same), which can create issues in loss calculation and learning. By using the midpoint offset parameters, this problem is entirely avoided.

1.1.2 Model Components

a) Overall Architecture The proposed model consists of two main parts:

1. **Oriented RPN:** Responsible for generating high-quality proposals with low computational cost.
2. **Oriented R-CNN Head:** Used for classifying and refining the proposals.

The process begins with a ResNet model, a popular and powerful backbone in computer vision, pre-trained on the ImageNet dataset. Features are extracted using a Feature Pyramid Network (FPN). The FPN is designed to handle objects at multiple scales by creating a pyramid of features with strong semantics at all levels. It works by taking the feature maps from different stages of the ResNet backbone, upsampling the deeper, more semantic-rich layers, and combining them with shallower, more

spatially-precise layers through lateral connections. This results in a set of feature maps $\{P_2, P_3, P_4, P_5\}$ that are rich in both detail and meaning.

The initial part of the model takes these five feature pyramid levels as input. It passes them through a 3x3 convolutional layer to generate new features. These new feature maps are then fed into two parallel 1x1 convolutional layers. For each pixel in the feature map, three anchor boxes with aspect ratios of 1:2, 1:1, and 2:1 are considered. One of the convolutional branches, the regression branch, predicts six offset values $(\delta_x, \delta_y, \delta_w, \delta_h, \delta_\alpha, \delta_\beta)$ for each anchor. These offsets are then used to transform the anchor box into an oriented proposal using the midpoint offset representation. The other convolutional branch, the classification branch, outputs a score for each anchor, indicating the probability of it containing an object.

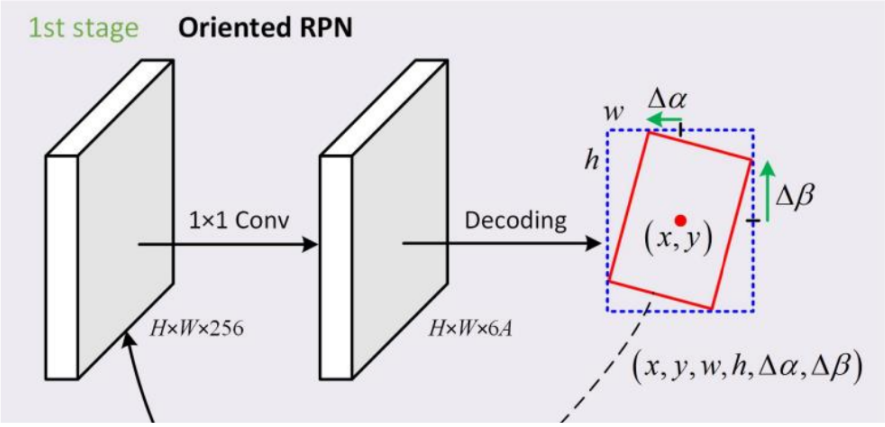

Figure 2: The Oriented RPN module from Figure 2 of the paper.

b) Training the RPN To train the RPN, positive and negative samples must be identified. A proposal is considered positive if its Intersection over Union (IoU) with a ground-truth box is at least 0.7, or if it has the highest IoU with a ground-truth box, provided that IoU is at least 0.3. Proposals with an IoU less than 0.3 with all ground-truth boxes are considered negative. The loss function for the RPN is then the sum of a classification loss and a regression loss:

$$L_{RPN} = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

where L_{cls} is the cross-entropy loss for classification, and L_{reg} is the Smooth L1 loss for regression, which is robust to outliers compared to L2 loss.

1.1.3 Rotated RoI Align

a) The Need for RoI Align The proposals generated by the RPN are parallelograms of varying sizes. However, for the subsequent fully connected layers in the R-CNN head, fixed-size feature maps are required. The purpose of an RoI (Region of Interest) pooling/aligning layer is to convert these variable-sized regions into a fixed-size representation.

Standard RoI Align is designed for axis-aligned boxes. It works by dividing the RoI into a grid of fixed size (e.g., 7x7) and then, for each grid cell, sampling a fixed number

of points. The feature values at these points are computed using bilinear interpolation from the feature map, and then max-pooling is applied over the sampled points in each cell. This avoids the harsh quantization of RoI Pooling and leads to better accuracy.

However, this standard method is not directly applicable to the parallelogram-shaped proposals from the Oriented RPN.

b) The Rotated RoI Align Mechanism The Rotated RoI Align module must first transform the parallelogram proposal into a rectangular one. This is achieved by identifying the shorter diagonal and extending it to match the length of the longer diagonal, effectively creating a rectangle. Then, for each feature layer, the dimensions of this rectangle are scaled by the layer's stride. The scaled rectangle is then mapped onto the feature map.

A 7x7 grid is superimposed on this mapped rectangle. For each cell in the grid, the feature values are sampled and aggregated (e.g., through averaging or max-pooling). This process results in a fixed-size 7x7 feature map for each proposal, regardless of its initial size or orientation. This fixed-size map can then be fed into the subsequent R-CNN head.

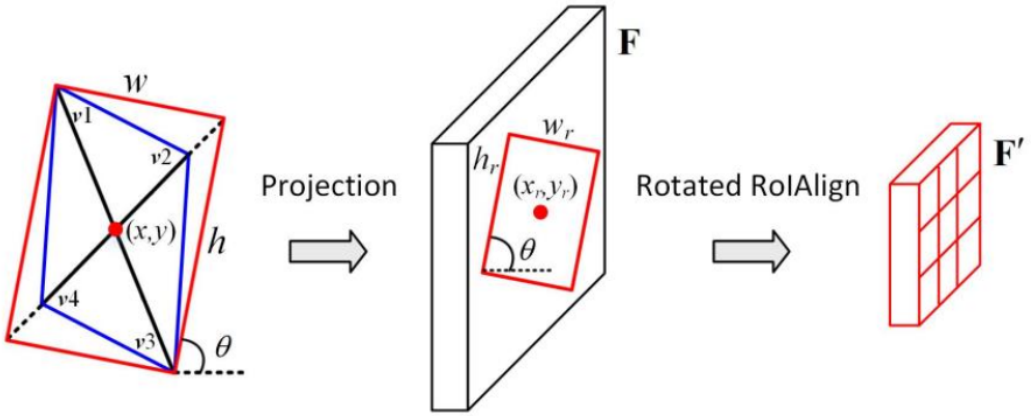

Figure 3: The mechanism of Rotated RoI Align from Figure 5 of the paper.

This process effectively "un-rotates" and scales the features within each proposal, allowing the network to process them in a standardized way.

1.1.4 Performance and Efficiency

a) Model Training and Inference The model is trained end-to-end, with both the RPN and R-CNN head being optimized simultaneously. During inference, the proposals generated can have high overlap. To handle this, only the top 2000 proposals from each feature layer are kept. Non-Maximum Suppression (NMS) is then applied to eliminate redundant boxes. To speed up this process, horizontal NMS with a high IoU threshold (0.8) is used first, followed by oriented NMS in the second stage for each class with a low IoU threshold (0.1) for proposals with a class probability greater than 0.05.

b) Comparison with Other Models The paper compares its model with two other major two-stage oriented detectors:

1. **Rotated RPN:** This model uses 54 anchors per location, resulting in high memory and computational costs. The proposed model uses only 3 anchors, making it significantly more efficient.
2. **RoI Transformer:** This model learns to transform horizontal RoIs into oriented ones. While it reduces the number of anchors, the transformation process itself is computationally complex and heavy.

The proposed Oriented R-CNN is more efficient due to several key design choices:

- **Fewer Anchors:** Using only 3 anchors per location drastically reduces computation compared to the 54 used in Rotated RPN.
- **Simple Representation:** The midpoint offset representation is simple, uses only 6 parameters, and avoids complex geometric transformations needed for angle-based methods.
- **Efficient RoI Align:** The Rotated RoI Align, while more complex than standard RoI Align, is more efficient than the full transformation network used in RoI Transformer.
- **Lightweight Head:** The number of layers in the R-CNN head is minimal, reducing computational overhead.

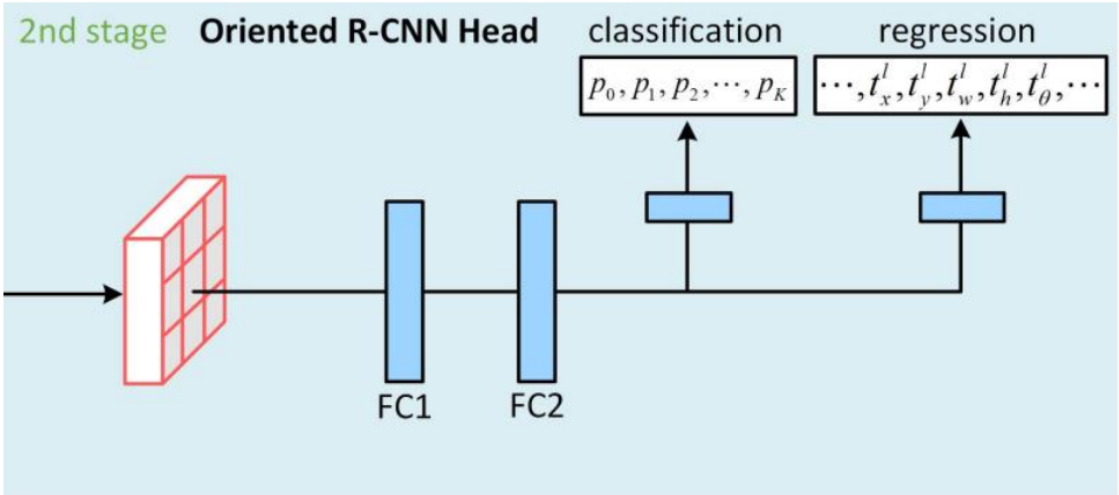

Figure 4: Architectures of three oriented object detection models from Figure 1 of the paper: (a) Rotated RPN, (b) RoI Transformer, (c) The proposed Oriented RPN.

As shown in Figure 4, the proposed model has significantly fewer parameters in its proposal generation stage compared to the other two models, leading to better memory usage and speed.

1.2 Practical Implementation

1.2.1 Environment and Dataset Setup

The dataset used is the HRSC2016 (High-Resolution Ship Collection 2016), which contains aerial images of ships from Google Earth. After downloading the images, we

prepare to load them. For this, we create a custom PyTorch 'Dataset' class that loads the images and their corresponding annotations. The dataset inspection reveals that the images are in '.bmp' format and the annotations are in '.xml' format. Our custom dataset class will handle the conversion of annotations to the required midpoint offset format.

```python
import torch
from torch.utils.data import Dataset
from PIL import Image
import os
import numpy as np
import xml.etree.ElementTree as ET

class HRSCDataset(Dataset):
    def __init__(self, image_dir, ann_dir, transform=None):
        self.image_dir = image_dir
        self.ann_dir = ann_dir
        self.transform = transform
        self.image_files = [f.split('.')[0] for f in os.listdir(image_dir)]

    def __len__(self):
        return len(self.image_files)

    def _parse_annotation(self, ann_path):
        # ... (Code to parse XML and convert to midpoint offset)
        # This part involves geometric calculations to convert from
        # center, w, h, angle to the 6-parameter midpoint offset format.
        pass

    def __getitem__(self, index):
        img_name = self.image_files[index]
        img_path = os.path.join(self.image_dir, img_name + '.bmp')
        image = Image.open(img_path).convert('RGB')

        ann_path = os.path.join(self.ann_dir, img_name + '.xml')
        boxes = self._parse_annotation(ann_path)
        boxes = torch.as_tensor(boxes, dtype=torch.float32)

        target = {}
        target['boxes'] = boxes
        target['image_id'] = torch.tensor([index])

        if self.transform:
            image = self.transform(image)

        return image, target
```

Listing 1: Custom PyTorch Dataset for HRSC2016

After creating the data loaders, we can visualize some samples to ensure the annotations are correctly parsed and displayed.

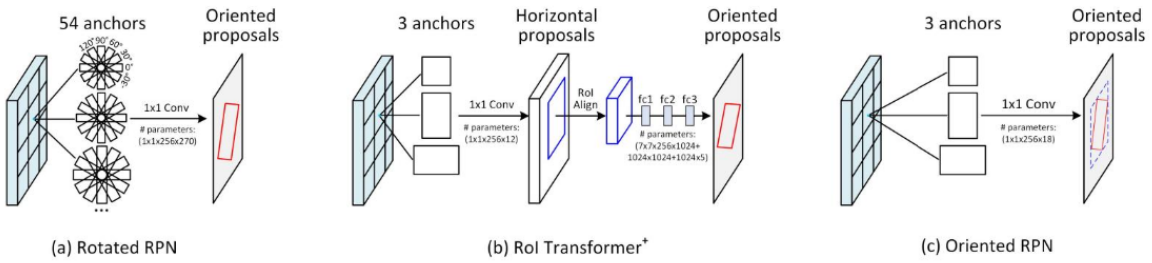

Figure 5: Several sample images with their corresponding oriented bounding boxes.

1.2.2 Model Implementation

The model is built by connecting several modules.

Feature Pyramid Network (FPN) This module acts as the backbone for feature extraction. It takes the multi-scale feature maps from a pre-trained ResNet and combines them to produce a pyramid of features that are rich in semantics at all scales. The implementation involves a bottom-up pathway (the ResNet itself), a top-down pathway to upsample and propagate semantic information, and lateral connections to merge the feature maps.

```
def draw_oriented_box(ax, box, color='r'):
  x,y,w,h,a,b = box
  v = [(x+a,y+h/2),(x+w/2,y+b),(x-a,y-h/2),(x-w/2,y-b)]
  rect = patches.Polygon(v, linewidth=2.5, edgecolor=color,
facecolor='none')
  ax.add_patch(rect)
```

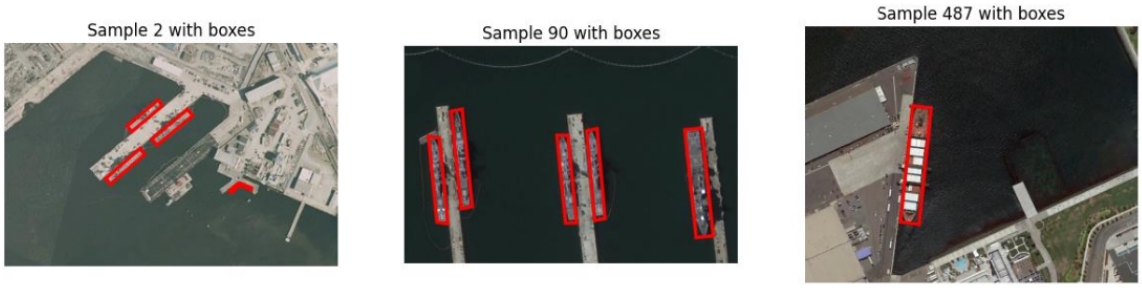

Figure 6: The structure of a Feature Pyramid Network (FPN), showing the bottom-up and top-down pathways with lateral connections. Source: FPN paper.

Oriented RPN This module takes the feature maps from the FPN and generates oriented proposals. It consists of a 3x3 convolution followed by two parallel 1x1 convolutions for classification (object vs. background) and regression (predicting the 6 midpoint offset parameters).

Rotated RoI Align This is perhaps the most complex module to implement. It takes the parallelogram proposals from the RPN and the feature maps from the FPN. For

each proposal, it first transforms it into a rectangle, then maps it onto the corresponding feature map, and finally samples a fixed-size grid (7x7) of features using bilinear interpolation.

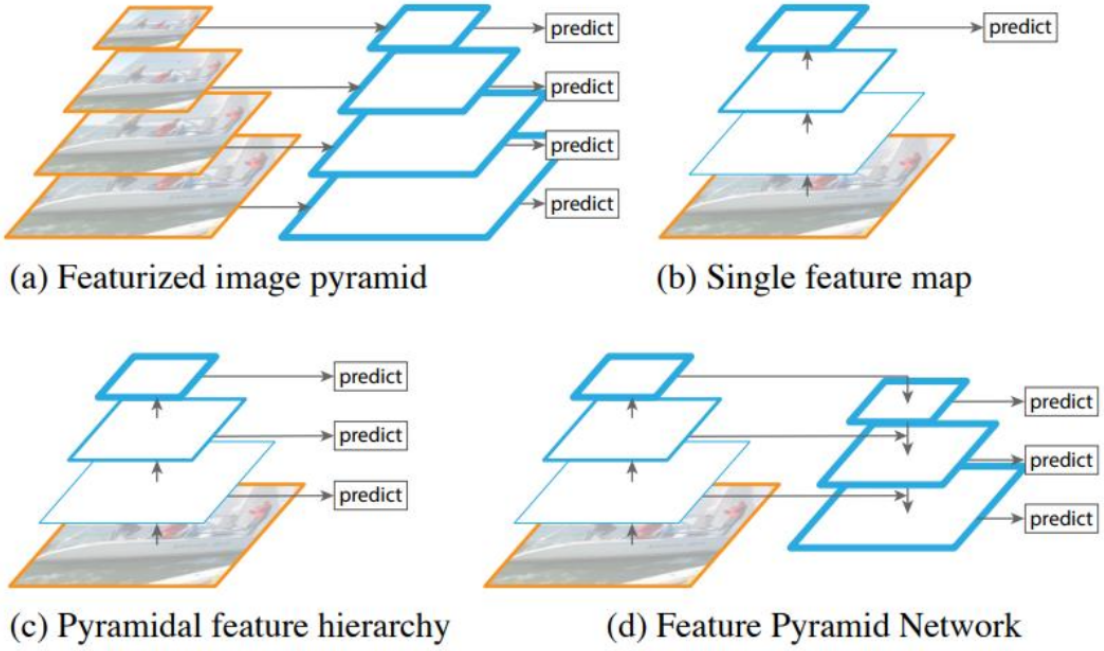

Figure 7: Several examples of transforming a parallelogram (input proposal) into an oriented rectangle for the RoI Align step.

Oriented R-CNN Head This final module takes the 7x7 feature maps from the RoI Align layer. It flattens them and passes them through two fully connected layers. The output then splits into two branches: a classification branch that predicts the object class (e.g., ship vs. background) and a regression branch that refines the 6 parameters of the bounding box.

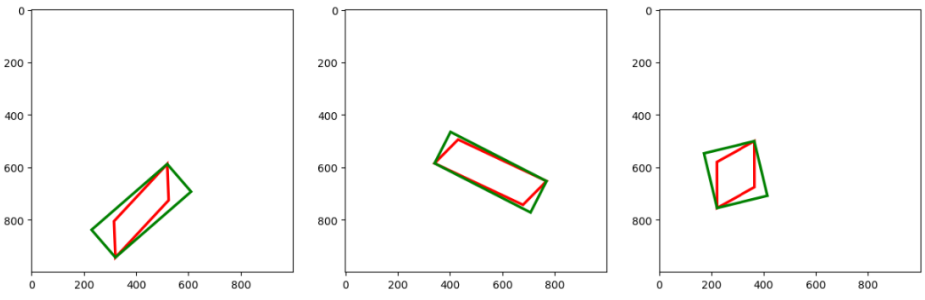

Figure 8: The Oriented R-CNN Head module from Figure 2 of the paper.

1.3 Conclusion

In this exercise, after studying and reviewing the Oriented R-CNN paper and similar papers like Faster R-CNN, we attempted to build the Oriented R-CNN model for detecting oriented bounding boxes of objects.

By writing a custom 'Dataset' class, we read the data and drew their oriented bounding boxes. We implemented the various modules of the model. However, we ultimately faced model errors and were unable to train it.

In our opinion, the main reason for the lack of success was the high complexity and detail of the model. Various mathematical matrix transformations are used in the RoI section and other parts, and most of the modules are not available in the PyTorch library and we had to implement them ourselves. This resulted in a final model with nearly 500 lines of code and a large number of long and complex modules, making debugging very difficult. The original implementation also uses the MMDetection library, which simplifies the model implementation, but due to the inability to use this tool and implementing the modules from scratch, the probability of errors was very high. However, from a theoretical and even practical standpoint, we became well-acquainted with the constituent modules of the model and learned how the model works and what different details are needed for designing the model to increase its speed and performance.