SI ANALYTICS

베이스라인 제출해보기

위성 이미지 객체 검출 경진대회 베이스라인 제출해보기

Jamyoung Koo
(Research Engineer, SI Analytics)

위성 이미지 객체 검출 경진대회

Date: 10th Jan, 2010

Contents

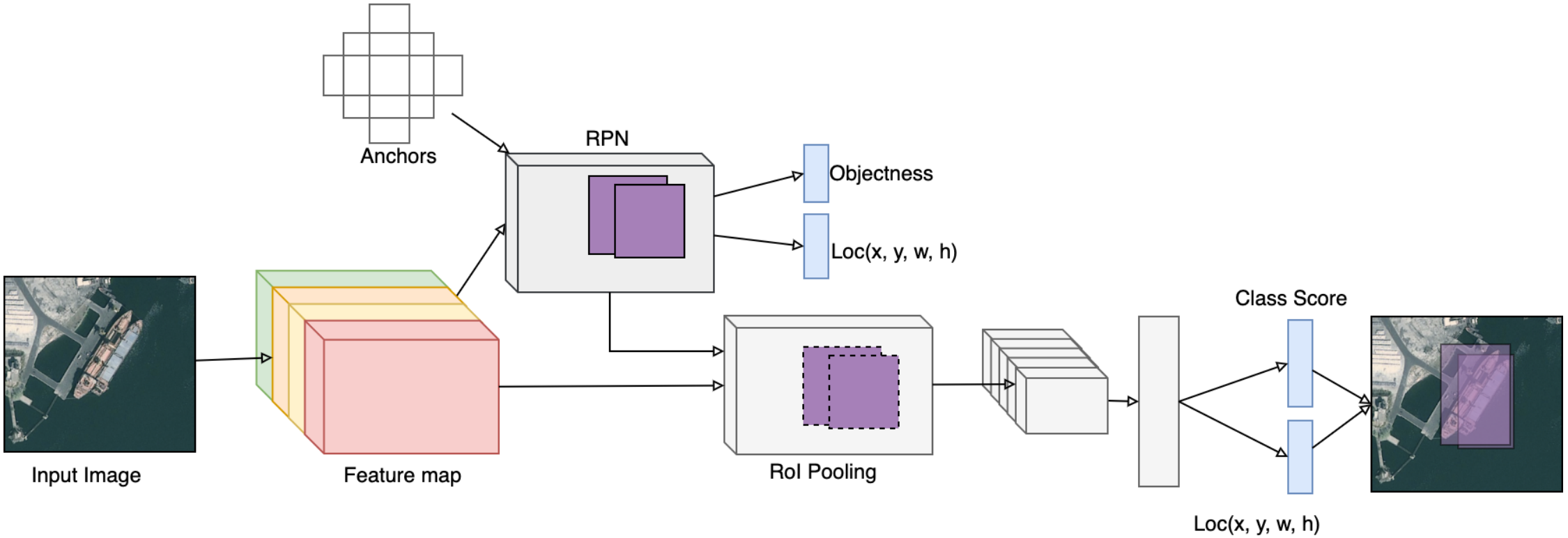
01 RBox-CNN 이해하기

RBox-CNN 이해하기: 배경 지식

- Faster R-CNN
  - Paper: Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015
  - 1차 밋업에서 Faster R-CNN에 대해 hands-on 진행
  - 2차 밋업에서 Faster R-CNN에 알고 있다는 가정으로 진행

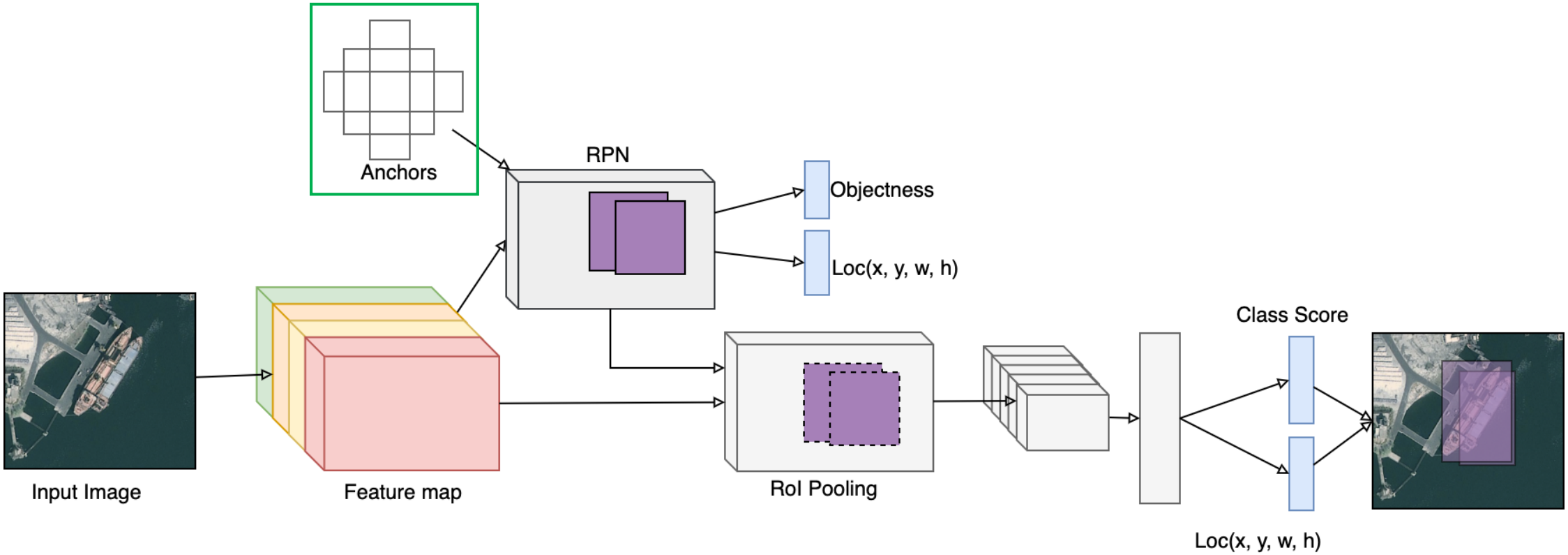
RBox-CNN 이해하기: 배경 지식

- Faster R-CNN: Architecture



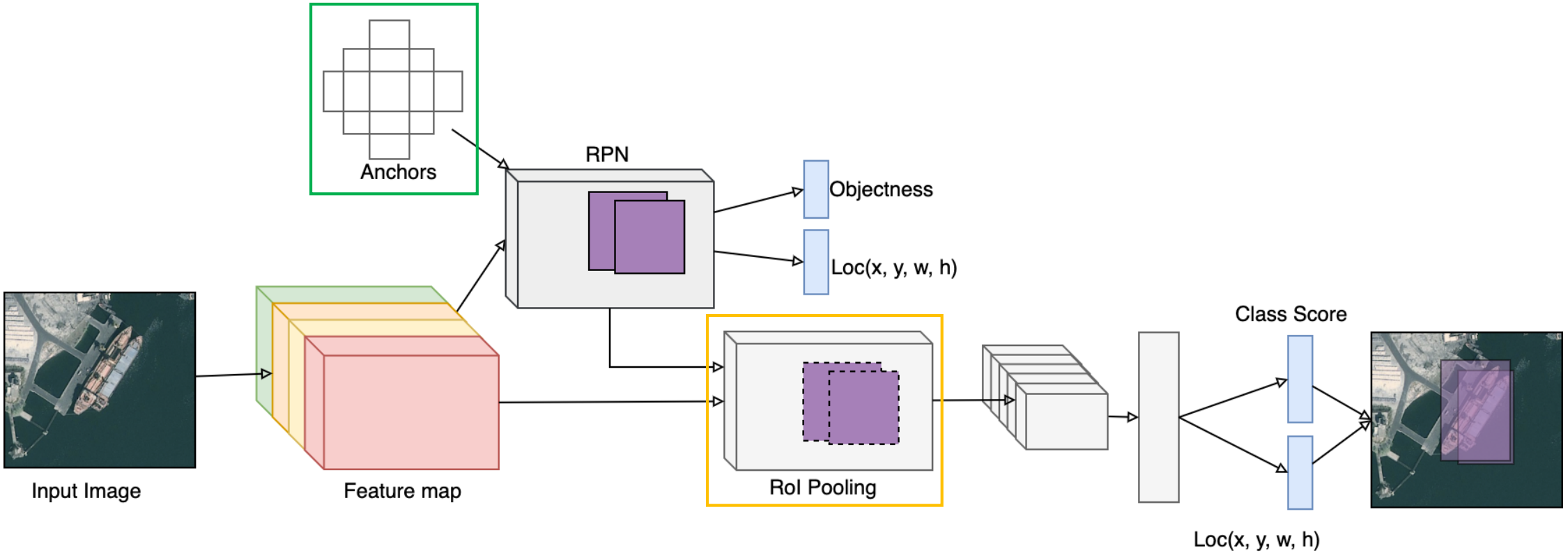
RBox-CNN 이해하기: 배경 지식

- Faster R-CNN: Architecture
  - Anchor



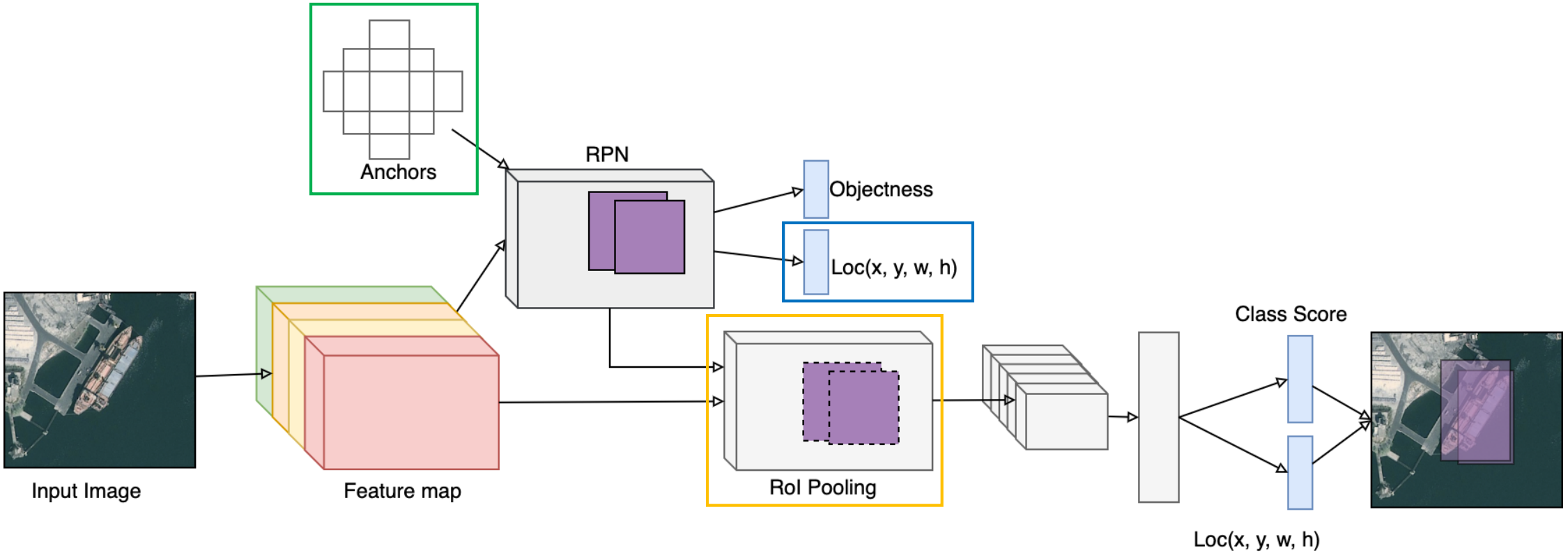
RBox-CNN 이해하기: 배경 지식

- Faster R-CNN: Architecture
  - Anchor, RoI Pooling



RBox-CNN 이해하기: 배경 지식

- Faster R-CNN: Architecture
  - Anchor, RoI Pooling, Box Regression



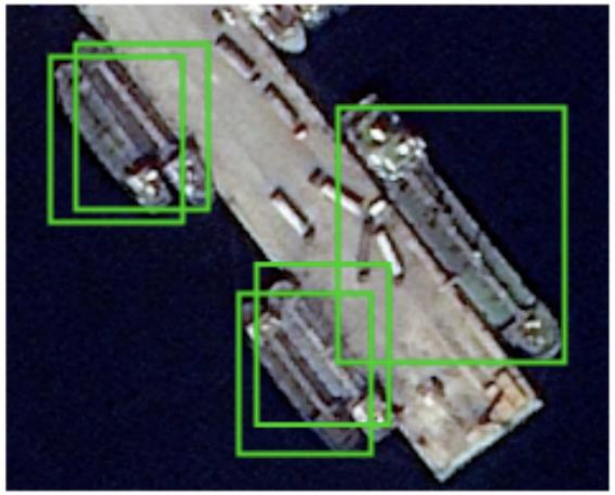
RBox-CNN 이해하기: 배경 지식

- Rotated Bounding Box V.S. Bounding Box

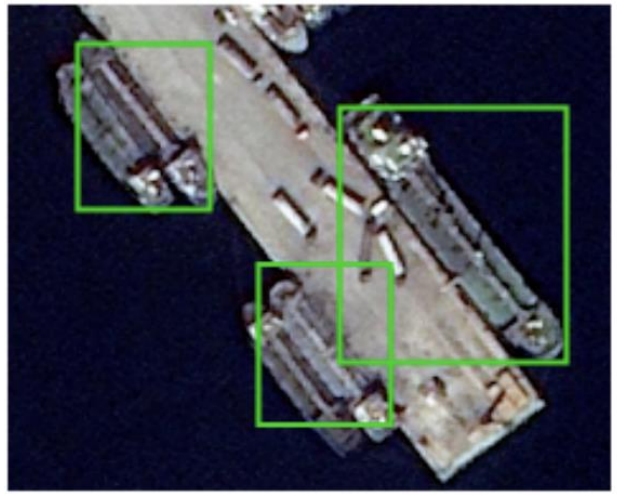
	Bounding Box	Rotated Bounding Box
길이	X	O
폭	X	O
각도	X	O
선수	X	O
장점	쉽고 빠르게 표현 가능	정확한 길이 및 부가정보 제공
단점	정확한 길이 및 정보 제공 못함	데이터 제작시 많은 가공, 고난이도 예측
예시		

RBox-CNN 이해하기: 배경 지식

- Rotated Bounding Box V.S. Bounding Box
  - Non-Maximum Supression(NMS) 적용 시 문제 발생



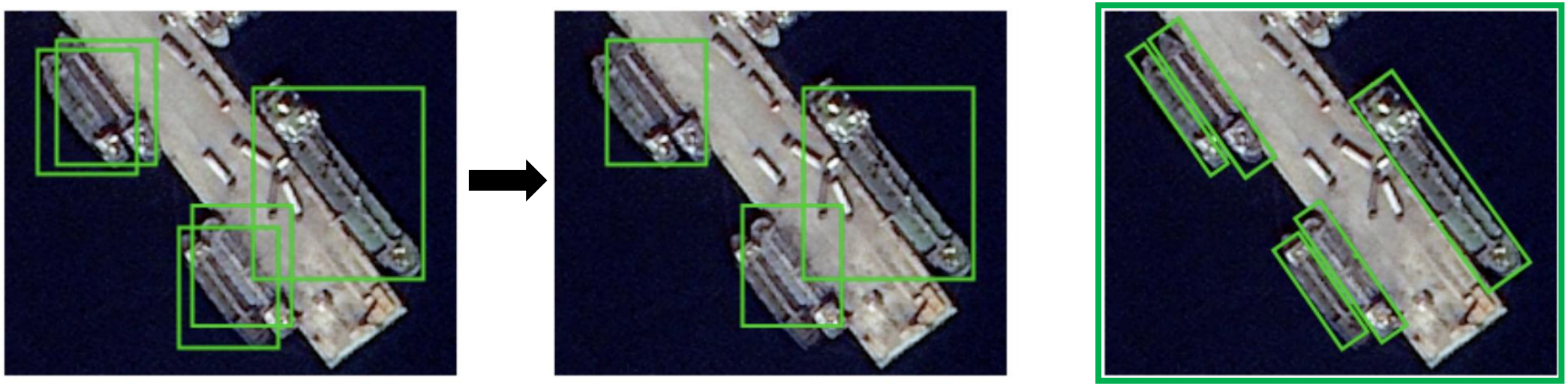
BBox 검출



NMS 적용

RBox-CNN 이해하기: 배경 지식

- Rotated Bounding Box V.S. Bounding Box
  - Non-Maximum Supression(NMS) 적용 시 문제 발생



BBox 검출 NMS 적용 RBox 검출 후 NMS 적용

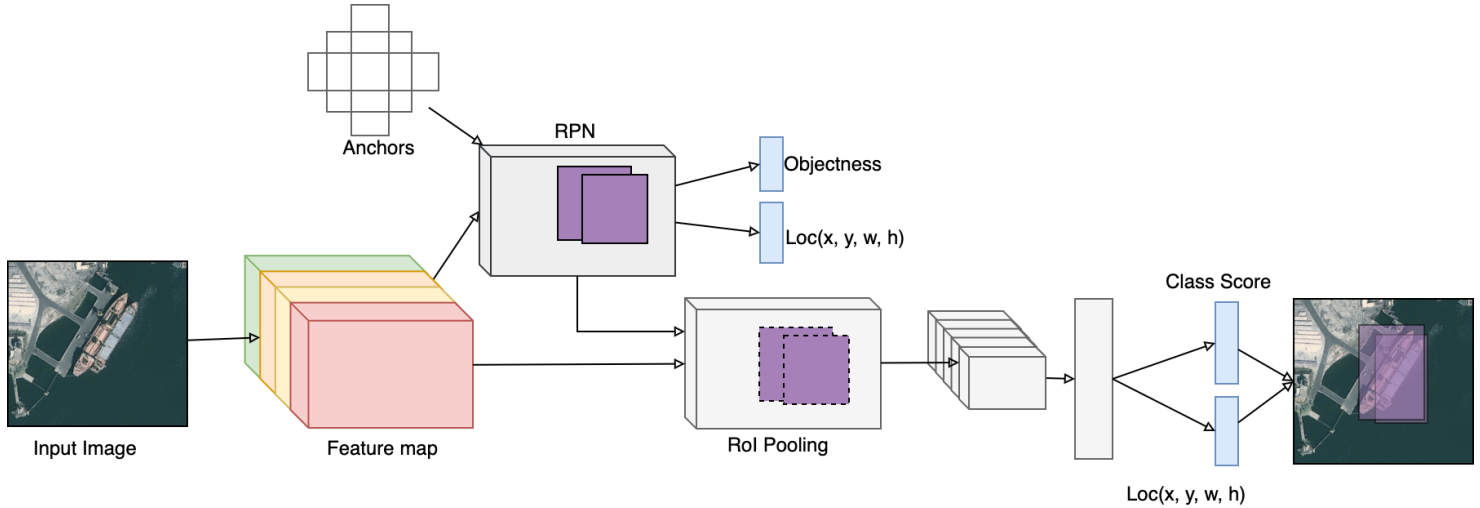
RBox-CNN 이해하기: Faster R-CNN 비교

- RBox-CNN
  - Paper: Koo, Jamyoung, et al. "RBox-CNN: rotated bounding box based CNN for ship detection in remote sensing image." *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2018.

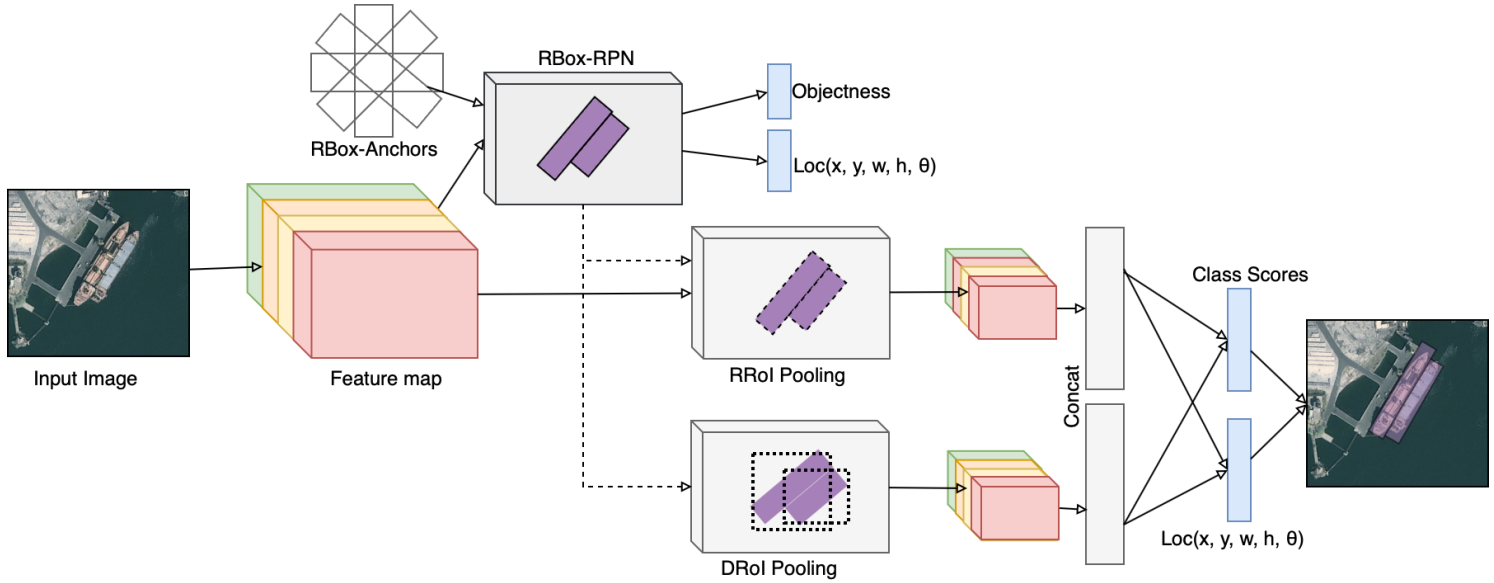
RBox-CNN 이해하기: Faster R-CNN 비교

- RBox-CNN

Faster R-CNN



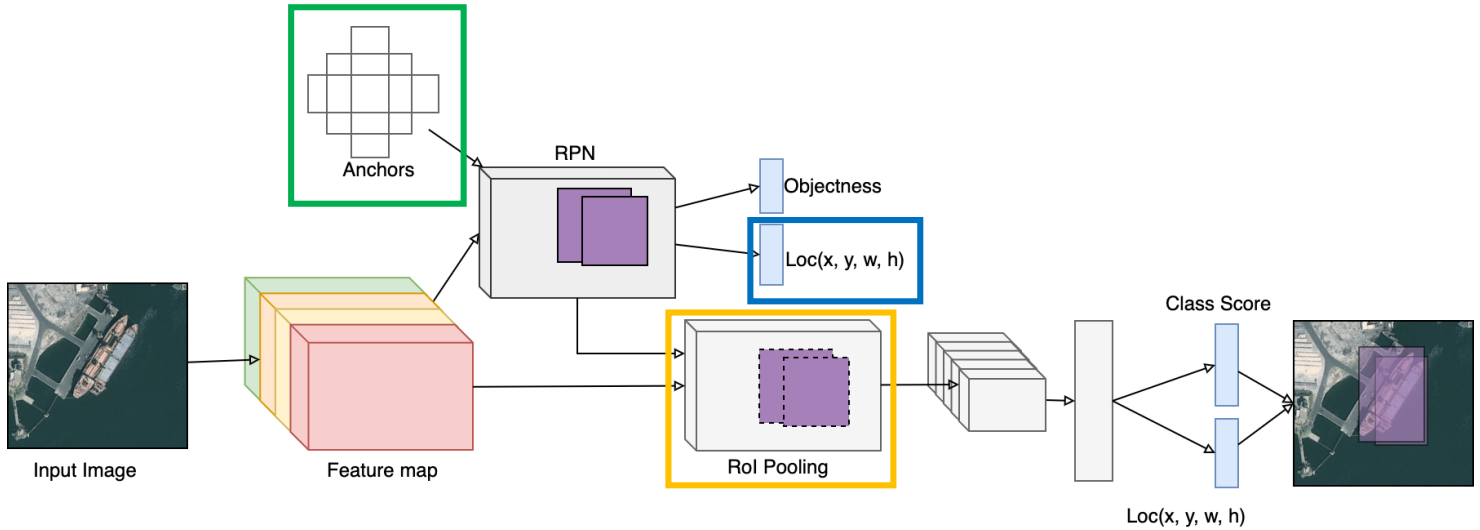
RBox-CNN



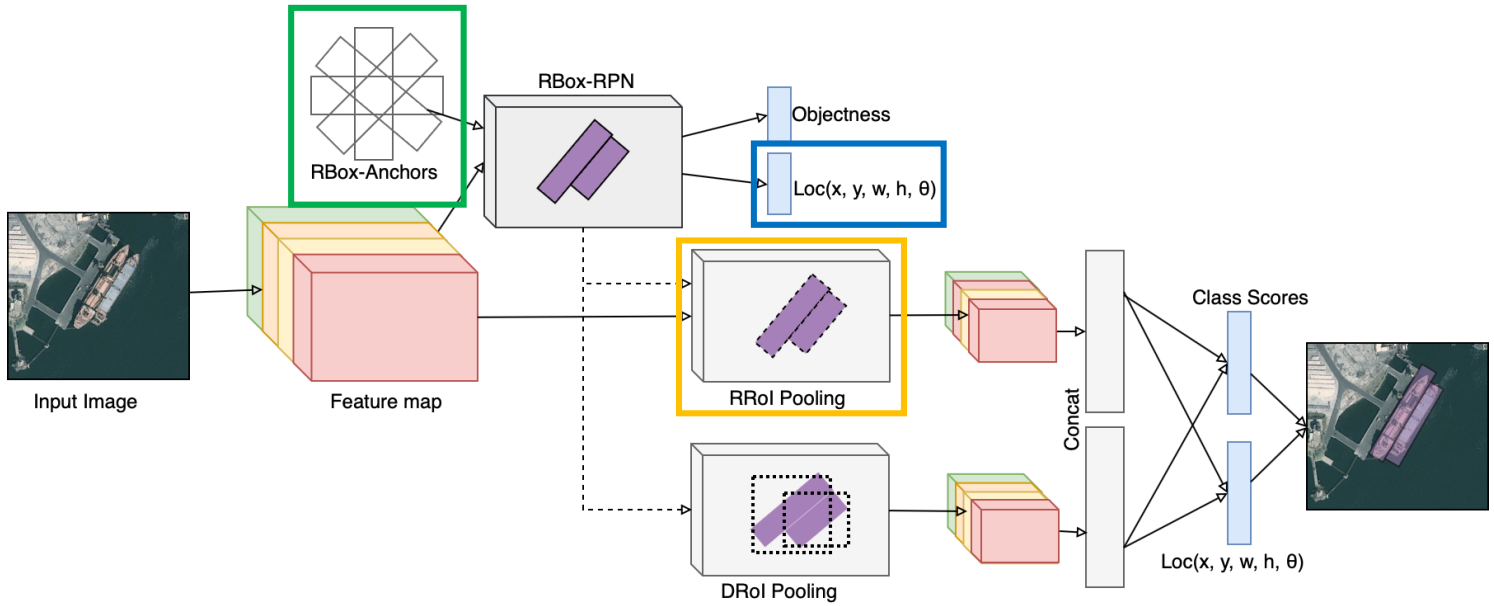
RBox-CNN 이해하기: Faster R-CNN 비교

- RBox-CNN

Faster R-CNN

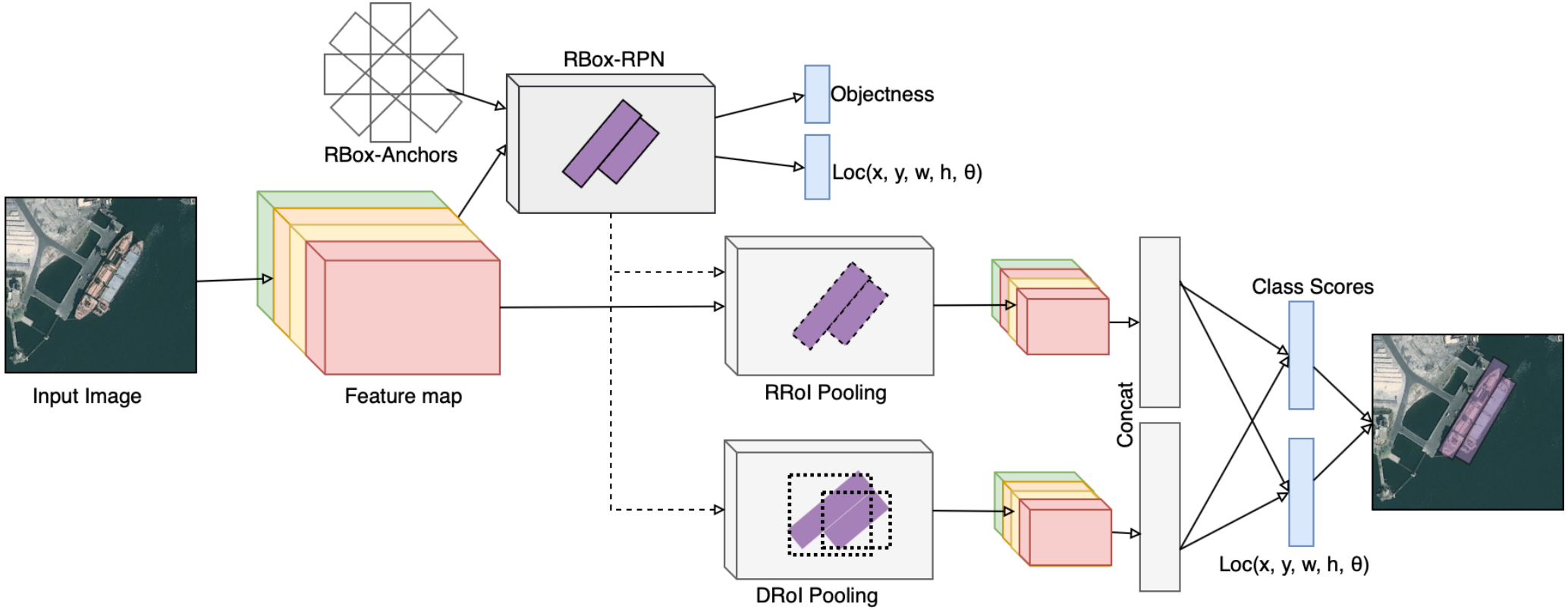


RBox-CNN



RBox-CNN 이해하기: Architecture

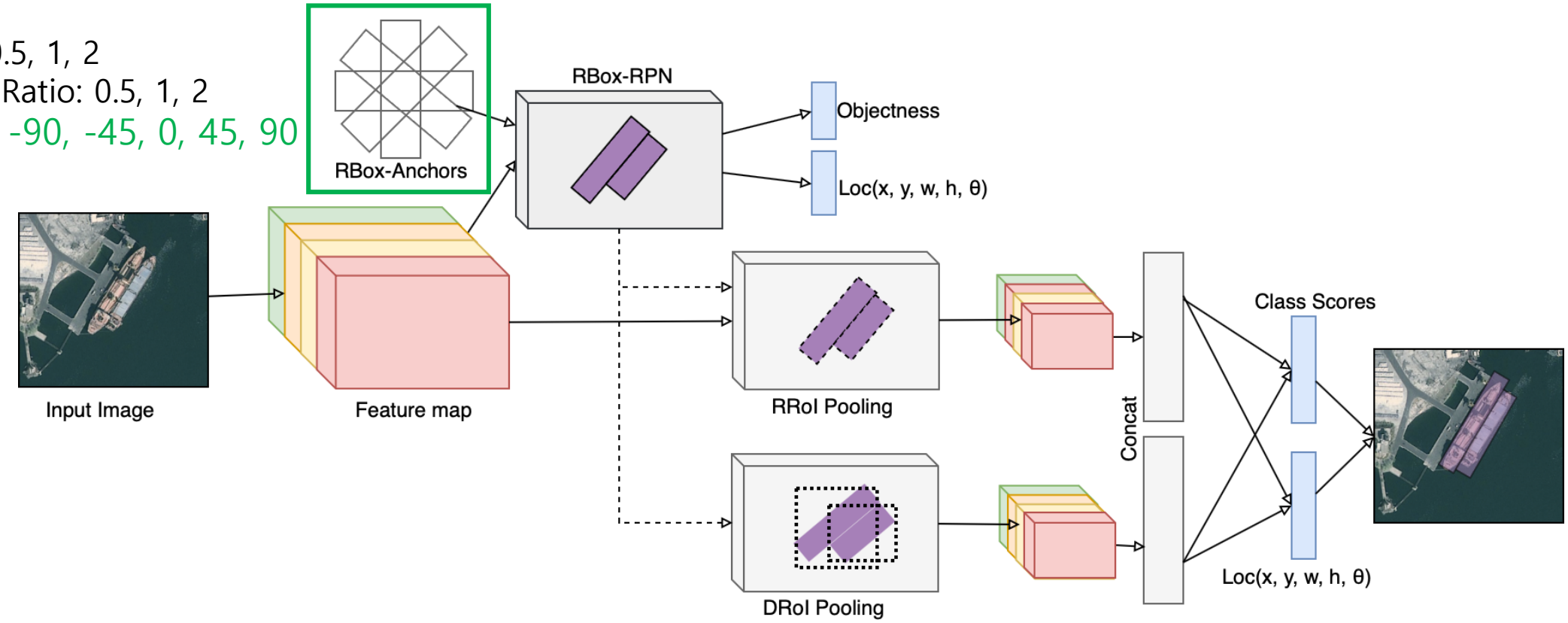
- RBox-CNN: Architecture



RBox-CNN 이해하기: RBox-Anchor

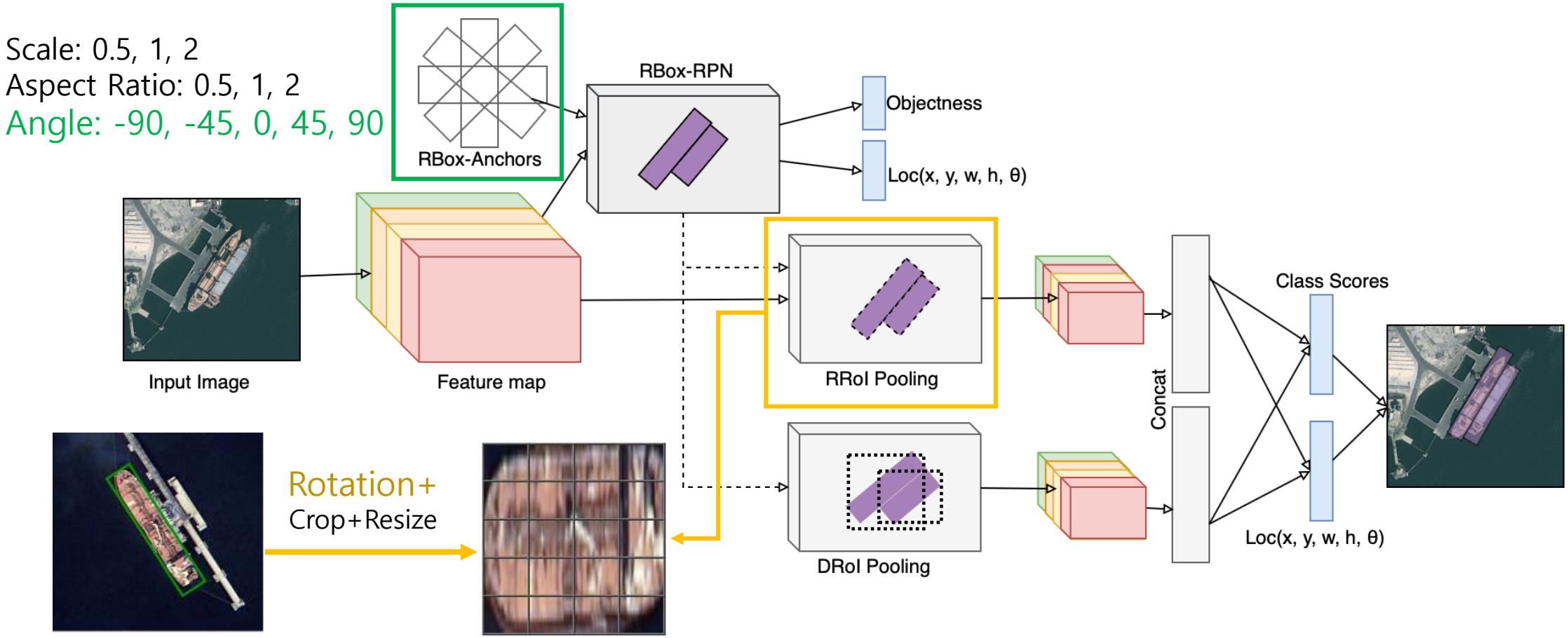
- RBox-CNN: RBox-Anchor

Scale: 0.5, 1, 2
Aspect Ratio: 0.5, 1, 2
Angle: -90, -45, 0, 45, 90



RBox-CNN 이해하기: Rotated RoI Pooling

- RBox-CNN: Rotated RoI Pooling(RRoI Pooling)

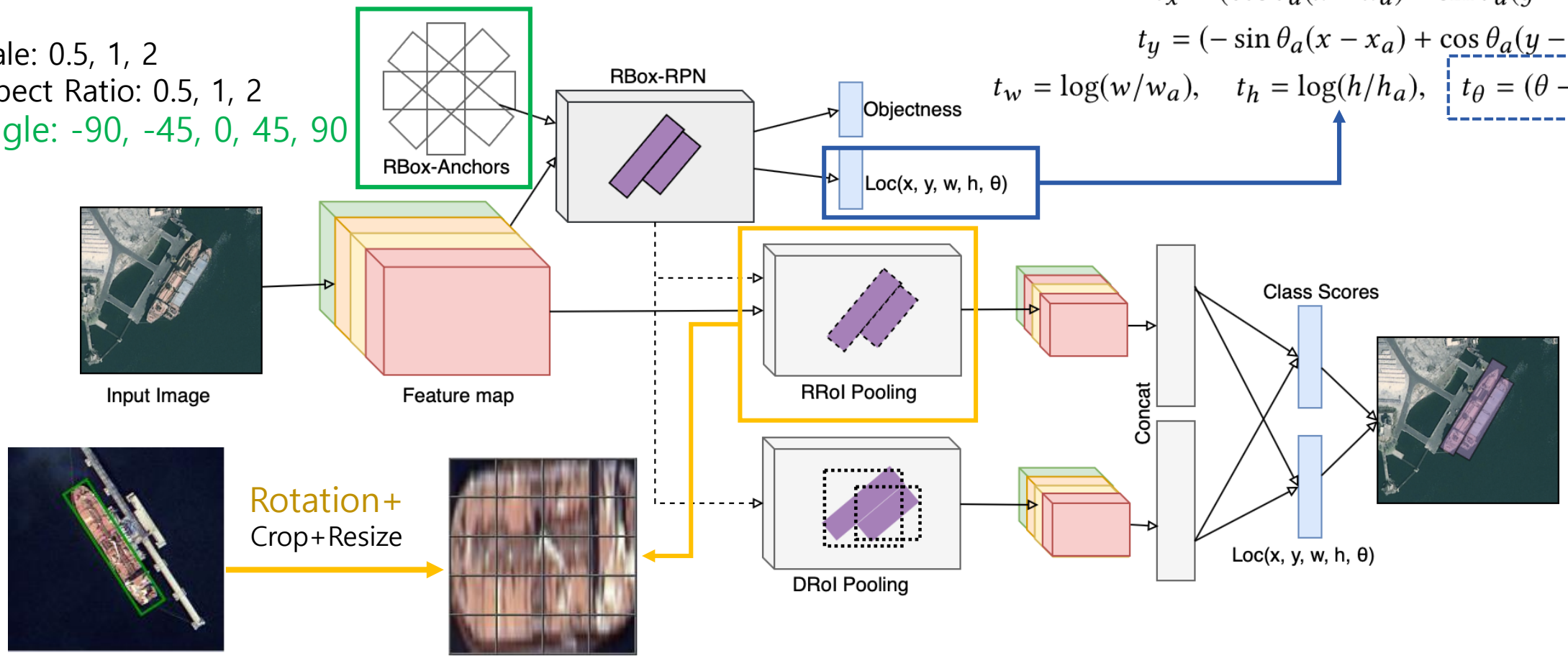


RBox-CNN 이해하기: Rotated RoI Pooling

- RBox-CNN: RBox Regression

$$t_x = (\cos\theta_a(x - x_a) + \sin\theta_a(y - y_a))/w_a$$
$$t_y = (-\sin\theta_a(x - x_a) + \cos\theta_a(y - y_a))/h_a$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \quad t_\theta = (\theta - \theta_a)\pi/2$$

Scale: 0.5, 1, 2
Aspect Ratio: 0.5, 1, 2
Angle: -90, -45, 0, 45, 90



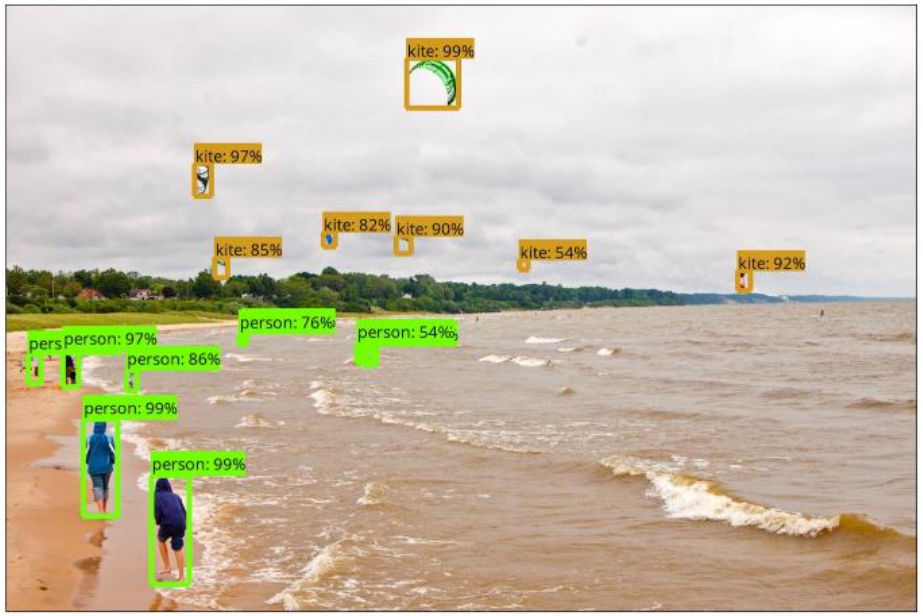
02 RBox-CNN 구현 살펴보기

RBox-CNN 구현: 배경지식

- Tensorflow Object Detection API
  - Github: https://github.com/tensorflow/models/tree/master/research/object_detection
  - 구현된 모델
    - ✓ SSD
    - ✓ Faster R-CNN
    - ✓ R-FCN

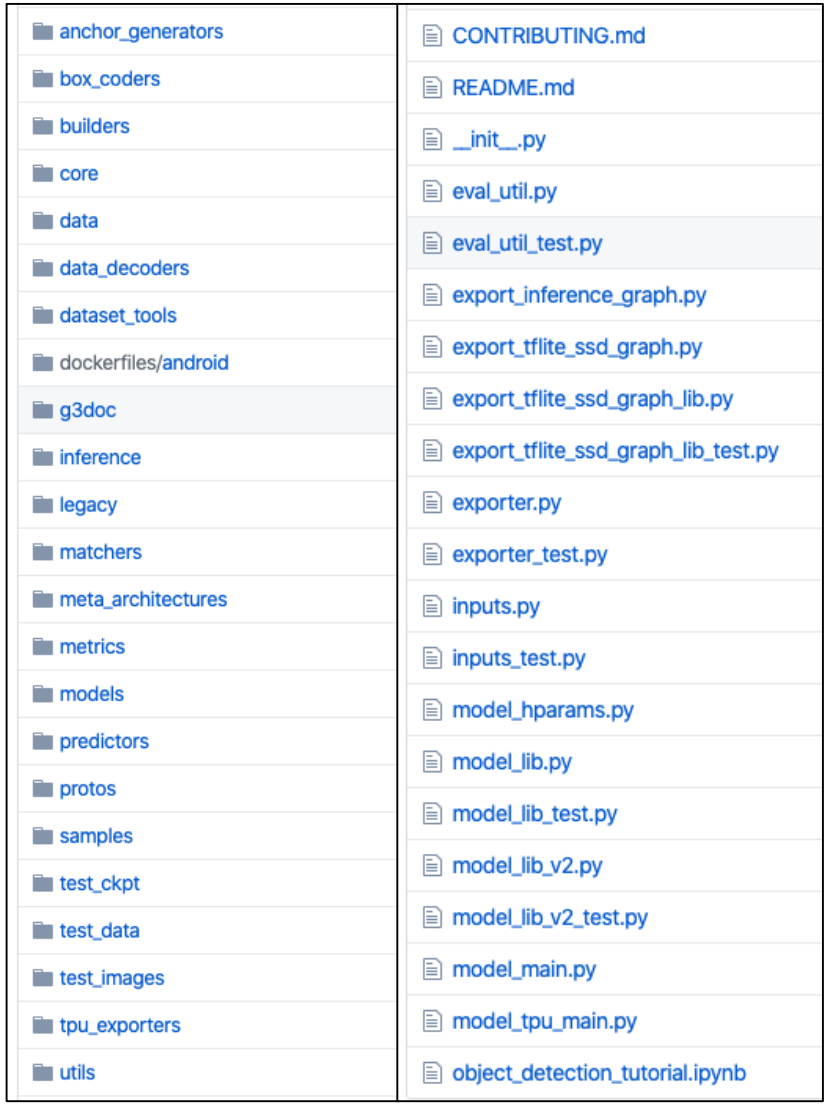
Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.



RBox-CNN 구현: 배경지식

- Tensorflow Object Detection API
  - 장점: 모듈화가 잘 되어서 아이디어를 넣고 빼기가 좋음
  - 단점: 코드 복잡도가 높아 처음에 진입장벽이 높음
    - ✓ 파일수만 대략 500개
  - 1차 밋업자료인 Faster R-CNN에 대해 hands-on 참고
    - ✓ https://github.com/SIAnalytics/Hands-on-OD

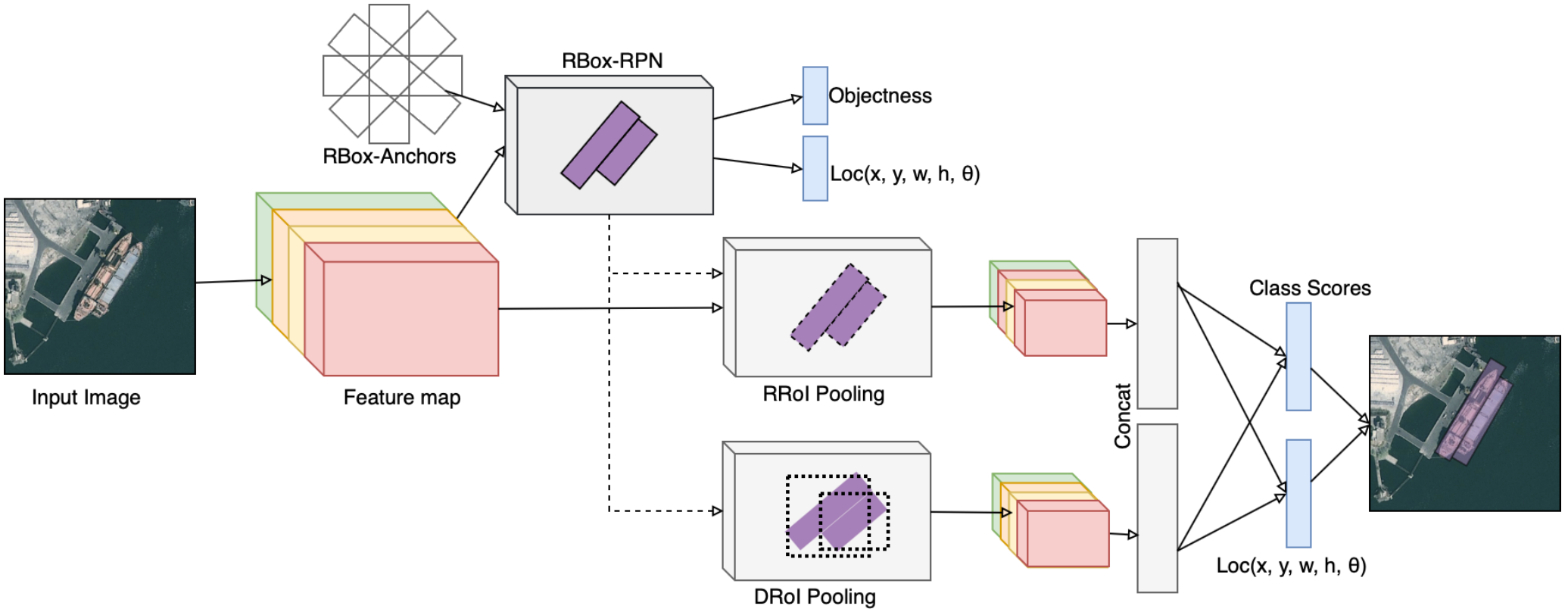


RBox-CNN 구현: 배경지식

- RBox-CNN basd on Tensorflow Object Detection API
  - Github: https://github.com/SIAnalytics/simplified_rbox_cnn
  - Code Version of Tensorflow Object Detection API
    - ✓ 대략 2017년 10월 버전
  - 환경 요구사항
    - ✓ python 3.5+
    - ✓ tensorflow 1.6

RBox-CNN 구현: RBox-Anchor

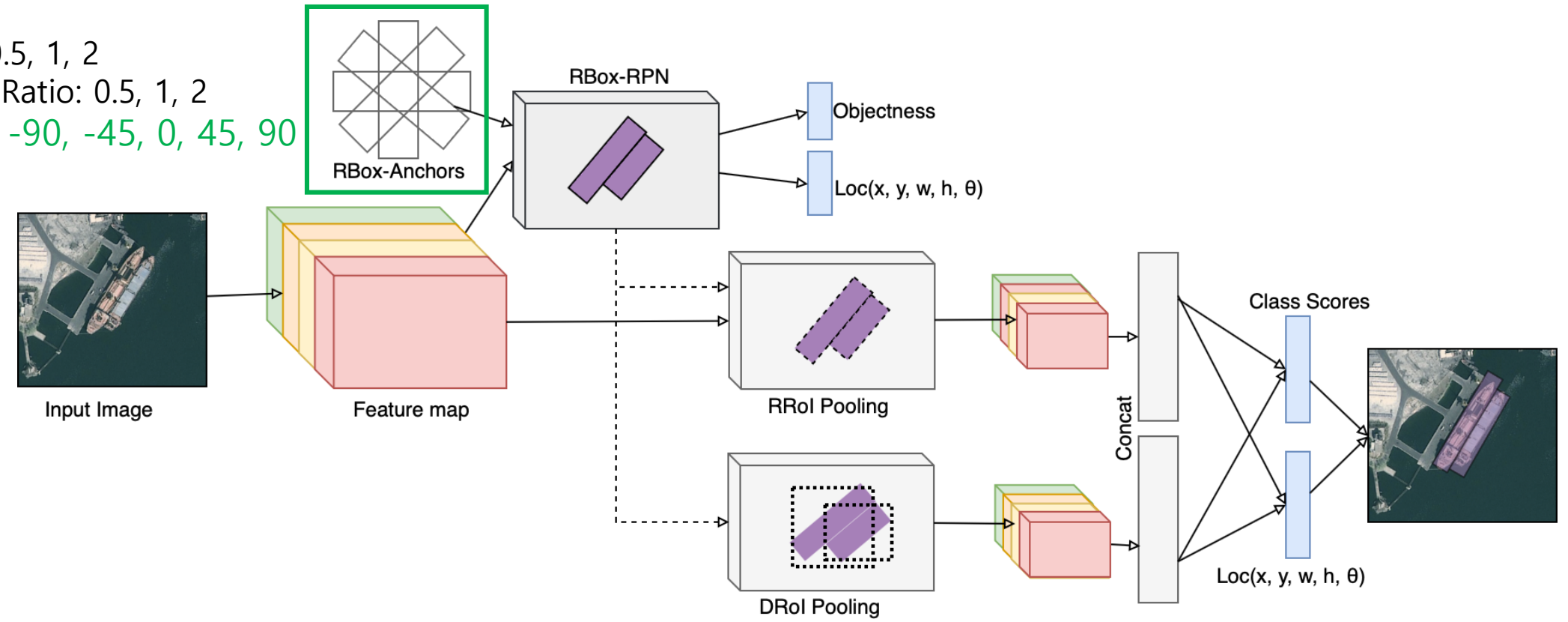
- RBox-CNN



RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor

Scale: 0.5, 1, 2
Aspect Ratio: 0.5, 1, 2
Angle: -90, -45, 0, 45, 90



RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 코드 파일: anchor_generators/grid_rbbox_anchor_generator.py
  - 주요 함수: tile_anchors(…)

```
# Get a grid of box centers
y_centers = tf.to_float(tf.range(grid_height))
y_centers = y_centers * anchor_stride[0] + anchor_offset[0]
x_centers = tf.to_float(tf.range(grid_width))
x_centers = x_centers * anchor_stride[1] + anchor_offset[1]
x_centers, y_centers = ops.meshgrid(x_centers, y_centers)

widths_grid, x_centers_grid = ops.meshgrid(widths, x_centers)
heights_grid, y_centers_grid = ops.meshgrid(heights, y_centers)
bbox_centers = tf.stack([y_centers_grid, x_centers_grid], axis=3)
bbox_sizes = tf.stack([heights_grid, widths_grid], axis=3)
bbox_centers = tf.reshape(bbox_centers, [-1, 2])
bbox_sizes = tf.reshape(bbox_sizes, [-1, 2])

# Add angles
n_angles = len(angles)
n_boxes = shape_utils.combined_static_and_dynamic_shape(bbox_centers)[0]
bbox_angles = tf.transpose(tf.convert_to_tensor([angles]))
bbox_angles = tf.tile(bbox_angles, [n_boxes, 1])
bbox_angles = tf.reshape(bbox_angles, [-1, 1])
```

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 코드 파일: anchor_generators/grid_rbbox_anchor_generator.py
  - 주요 함수: tile_anchors(…)

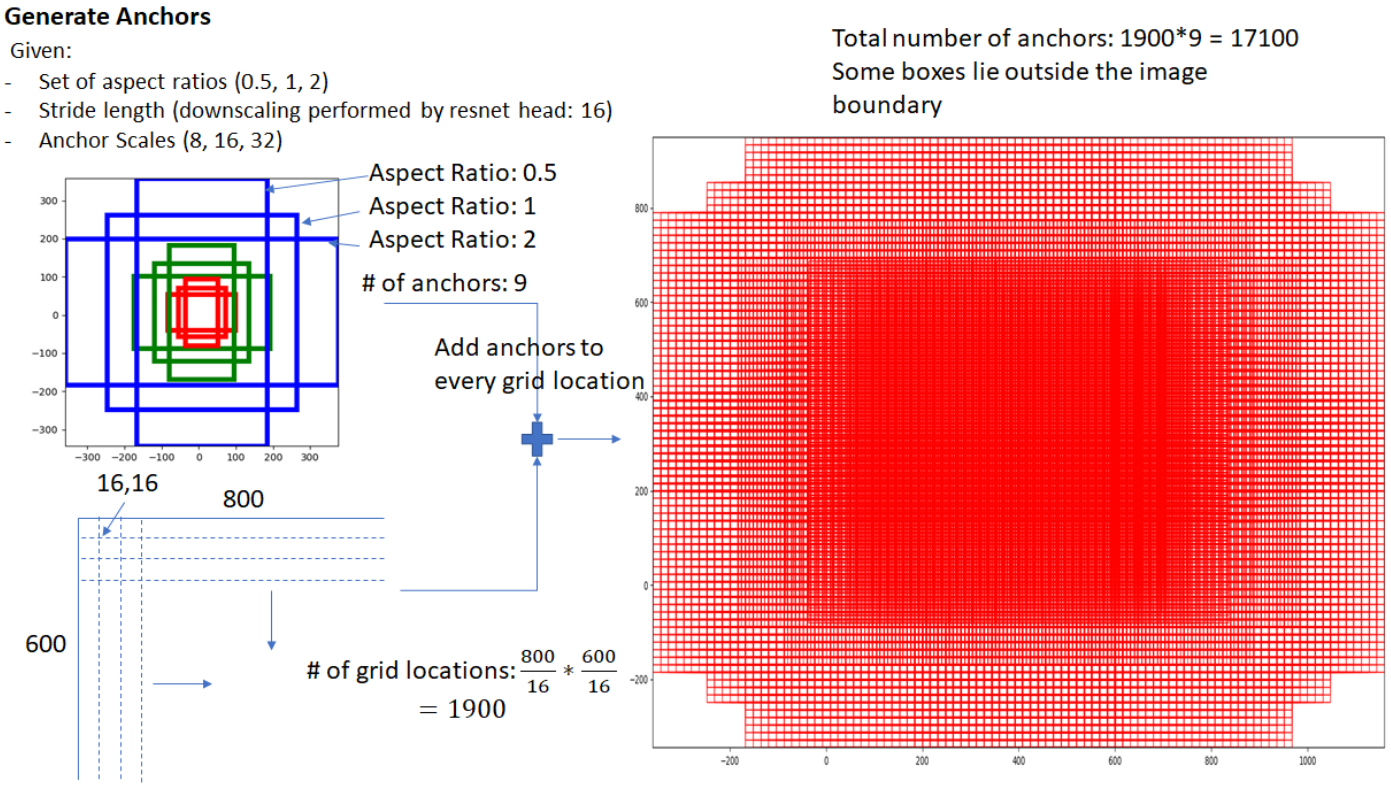
기존 BBox Anchors 생성 코드
= Scale + Aspect Ratio

```
# Get a grid of box centers
y_centers = tf.to_float(tf.range(grid_height))
y_centers = y_centers * anchor_stride[0] + anchor_offset[0]
x_centers = tf.to_float(tf.range(grid_width))
x_centers = x_centers * anchor_stride[1] + anchor_offset[1]
x_centers, y_centers = ops.meshgrid(x_centers, y_centers)

widths_grid, x_centers_grid = ops.meshgrid(widths, x_centers)
heights_grid, y_centers_grid = ops.meshgrid(heights, y_centers)
bbox_centers = tf.stack([y_centers_grid, x_centers_grid], axis=3)
bbox_sizes = tf.stack([heights_grid, widths_grid], axis=3)
bbox_centers = tf.reshape(bbox_centers, [-1, 2])
bbox_sizes = tf.reshape(bbox_sizes, [-1, 2])

# Add angles
n_angles = len(angles)
n_boxes = shape_utils.combined_static_and_dynamic_shape(bbox_centers)[0]
bbox_angles = tf.transpose(tf.convert_to_tensor([angles]))
bbox_angles = tf.tile(bbox_angles, [n_boxes, 1])
bbox_angles = tf.reshape(bbox_angles, [-1, 1])
```

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 코드 파일: anchor_generators/grid_rbbox_anchor_generator.py
  - 주요 함수: tile_anchors(…)

기존 BBox Anchors 생성 코드
= Scale + Aspect Ratio



from http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 코드 파일: anchor_generators/grid_rbbox_anchor_generator.py
  - 주요 함수: tile_anchors(…)

기존 BBox Anchors 생성 코드
= Scale + Aspect Ratio

+ Angle

```
# Get a grid of box centers
y_centers = tf.to_float(tf.range(grid_height))
y_centers = y_centers * anchor_stride[0] + anchor_offset[0]
x_centers = tf.to_float(tf.range(grid_width))
x_centers = x_centers * anchor_stride[1] + anchor_offset[1]
x_centers, y_centers = ops.meshgrid(x_centers, y_centers)

widths_grid, x_centers_grid = ops.meshgrid(widths, x_centers)
heights_grid, y_centers_grid = ops.meshgrid(heights, y_centers)
bbox_centers = tf.stack([y_centers_grid, x_centers_grid], axis=3)
bbox_sizes = tf.stack([heights_grid, widths_grid], axis=3)
bbox_centers = tf.reshape(bbox_centers, [-1, 2])
bbox_sizes = tf.reshape(bbox_sizes, [-1, 2])

# Add angles
n_angles = len(angles)
n_boxes = shape_utils.combined_static_and_dynamic_shape(bbox_centers)[0]
bbox_angles = tf.transpose(tf.convert_to_tensor([angles]))
bbox_angles = tf.tile(bbox_angles, [n_boxes, 1])
bbox_angles = tf.reshape(bbox_angles, [-1, 1])
```

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 테스트 코드 파일: anchor_generators/grid_rbbox_anchor_generator_test.py

```
scales = [1.0, 2.0]
aspect_ratios = [1.0]
angles = [0, 0.1]

exp_anchor_corners = [[0., 0., 10., 10., 0.],
                      [0., 0., 10., 10., 0.1],
                      [0., 0., 20., 20., 0.],
                      [0., 0., 20., 20., 0.1],

                      [0., 19., 10., 10., 0.],
                      [0., 19., 10., 10., 0.1],
                      [0., 19., 20., 20., 0.],
                      [0., 19., 20., 20., 0.1],

                      [19., 0., 10., 10., 0.],
                      [19., 0., 10., 10., 0.1],
                      [19., 0., 20., 20., 0.],
                      [19., 0., 20., 20., 0.1],

                      [19., 19., 10., 10., 0.],
                      [19., 19., 10., 10., 0.1],
                      [19., 19., 20., 20., 0.],
                      [19., 19., 20., 20., 0.1]]
```

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 테스트 코드 파일: anchor_generators/grid_rbbox_anchor_generator_test.py

Size of Feature Map=(2, 2)
→ 2 x 2 x (2 x 2) = 16

```
scales = [1.0, 2.0]
aspect_ratios = [1.0]
angles = [0, 0.1]

exp_anchor_corners = [[0., 0., 10., 10., 0.],
                      [0., 0., 10., 10., 0.1],
                      [0., 0., 20., 20., 0.],
                      [0., 0., 20., 20., 0.1],

                      [0., 19., 10., 10., 0.],
                      [0., 19., 10., 10., 0.1],
                      [0., 19., 20., 20., 0.],
                      [0., 19., 20., 20., 0.1],

                      [19., 0., 10., 10., 0.],
                      [19., 0., 10., 10., 0.1],
                      [19., 0., 20., 20., 0.],
                      [19., 0., 20., 20., 0.1],

                      [19., 19., 10., 10., 0.],
                      [19., 19., 10., 10., 0.1],
                      [19., 19., 20., 20., 0.],
                      [19., 19., 20., 20., 0.1]]
```

RBox-CNN 구현: RBox-Anchor

- RBox-CNN: RBox-Anchor
  - 테스트 코드 파일: anchor_generators/grid_rbbox_anchor_generator_test.py

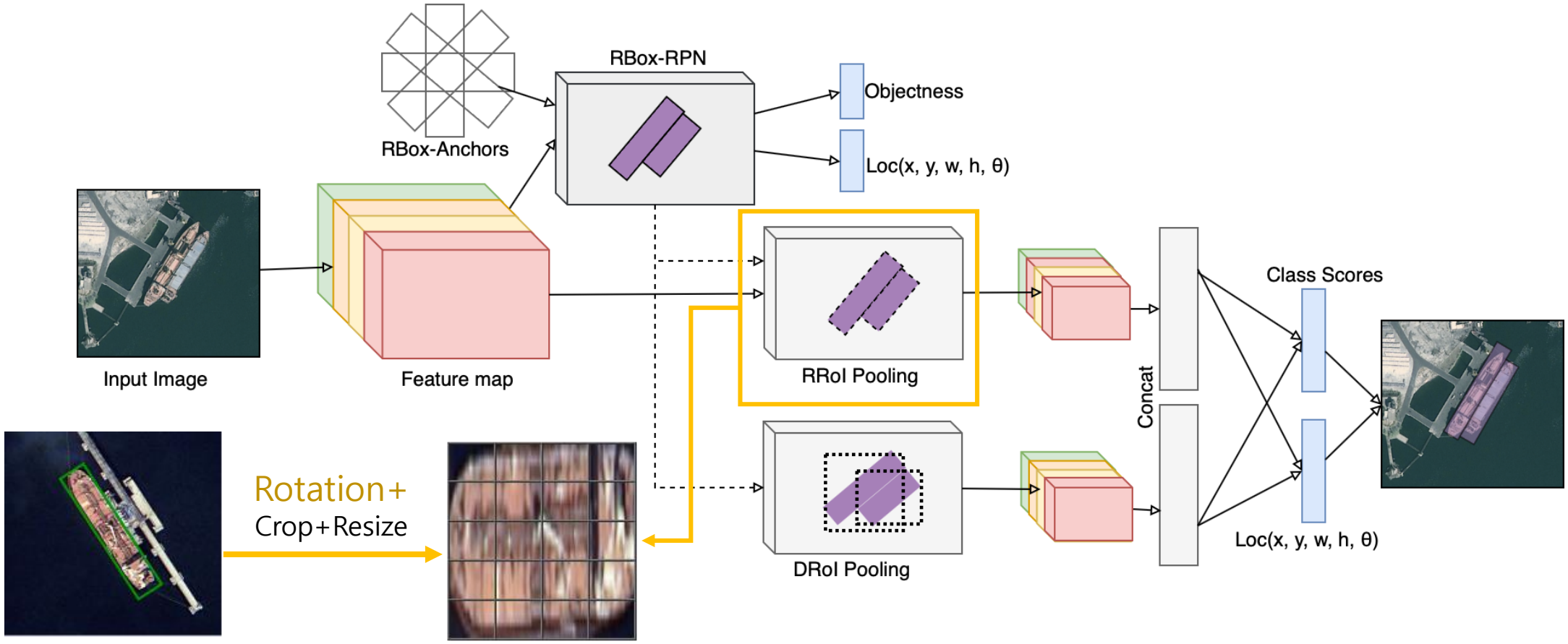
Size of Feature Map=(2, 2)
→ 2 x 2 x (2 x 2) = 16

Total: 16 RBox-Anchors
(cx, cy, w, h, theta)

```
scales = [1.0, 2.0]
aspect_ratios = [1.0]
angles = [0, 0.1]

exp_anchor_corners = [[0., 0., 10., 10., 0.],
                      [0., 0., 10., 10., 0.1],
                      [0., 0., 20., 20., 0.],
                      [0., 0., 20., 20., 0.1],

                      [0., 19., 10., 10., 0.],
                      [0., 19., 10., 10., 0.1],
                      [0., 19., 20., 20., 0.],
                      [0., 19., 20., 20., 0.1],

                      [19., 0., 10., 10., 0.],
                      [19., 0., 10., 10., 0.1],
                      [19., 0., 20., 20., 0.],
                      [19., 0., 20., 20., 0.1],

                      [19., 19., 10., 10., 0.],
                      [19., 19., 10., 10., 0.1],
                      [19., 19., 20., 20., 0.],
                      [19., 19., 20., 20., 0.1]]
```

RBox-CNN 구현: RRoI Pooling

- RBox-CNN: RRoI Pooling



RBox-CNN 구현: RRoI Pooling

- RBox-CNN: RRoI Pooling
  - 코드 파일: core/rotated_subsampling.py
  - 주요 함수: rotated_subsampling(…)

```
translation_matrix = tf.concat([
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 1, None] * original_size[1] - tf.cast(original_size[1] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 0, None] * original_size[0] - tf.cast(original_size[0] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 2), tf.float32),
],
    axis=1)

rotation_matrix = tf.contrib.image.angles_to_projective_transforms(boxes[:, 4],
                                                                   tf.cast(size_max, tf.float32),
                                                                   tf.cast(size_max, tf.float32)
                                                                   )

composed_matrix = tf.contrib.image.compose_transforms(translation_matrix, rotation_matrix)

transformed_images = tf.contrib.image.transform(transformed_images, composed_matrix,
                                                interpolation='BILINEAR')

height_ratio = (boxes[:, 2] * original_size[0]) / tf.cast(size_max, tf.float32)
width_ratio = (boxes[:, 3] * original_size[1]) / tf.cast(size_max, tf.float32)
transformed_images = tf.image.crop_and_resize(transformed_images,
```

RBox-CNN 구현: RRoI Pooling

- RBox-CNN: RRoI Pooling
  - 코드 파일: core/rotated_subsampling.py
  - 주요 함수: rotated_subsampling(…)

0도로 회전

```
translation_matrix = tf.concat([
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 1, None] * original_size[1] - tf.cast(original_size[1] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 0, None] * original_size[0] - tf.cast(original_size[0] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 2), tf.float32),
],
    axis=1)

rotation_matrix = tf.contrib.image.angles_to_projective_transforms(boxes[:, 4],
                                                                   tf.cast(size_max, tf.float32),
                                                                   tf.cast(size_max, tf.float32)
                                                                   )

composed_matrix = tf.contrib.image.compose_transforms(translation_matrix, rotation_matrix)

transformed_images = tf.contrib.image.transform(transformed_images, composed_matrix,
                                                interpolation='BILINEAR')

height_ratio = (boxes[:, 2] * original_size[0]) / tf.cast(size_max, tf.float32)
width_ratio = (boxes[:, 3] * original_size[1]) / tf.cast(size_max, tf.float32)
transformed_images = tf.image.crop_and_resize(transformed_images,
```

RBox-CNN 구현: RRoI Pooling

- RBox-CNN: RRoI Pooling
  - 코드 파일: core/rotated_subsampling.py
  - 주요 함수: rotated_subsampling(…)

0도로 회전

Translate

$$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
translation_matrix = tf.concat([
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 1, None] * original_size[1] - tf.cast(original_size[1] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 0, None] * original_size[0] - tf.cast(original_size[0] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 2), tf.float32),
],
    axis=1)

rotation_matrix = tf.contrib.image.angles_to_projective_transforms(boxes[:, 4],
                                                                   tf.cast(size_max, tf.float32),
                                                                   tf.cast(size_max, tf.float32)
                                                                   )

composed_matrix = tf.contrib.image.compose_transforms(translation_matrix, rotation_matrix)

transformed_images = tf.contrib.image.transform(transformed_images, composed_matrix,
                                                interpolation='BILINEAR')

height_ratio = (boxes[:, 2] * original_size[0]) / tf.cast(size_max, tf.float32)
width_ratio = (boxes[:, 3] * original_size[1]) / tf.cast(size_max, tf.float32)
transformed_images = tf.image.crop_and_resize(transformed_images,
```

RBox-CNN 구현: RRoI Pooling

- RBox-CNN: RRoI Pooling
  - 코드 파일: core/rotated_subsampling.py
  - 주요 함수: rotated_subsampling(…)

0도로 회전

Crop & Resize

Translate

$$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

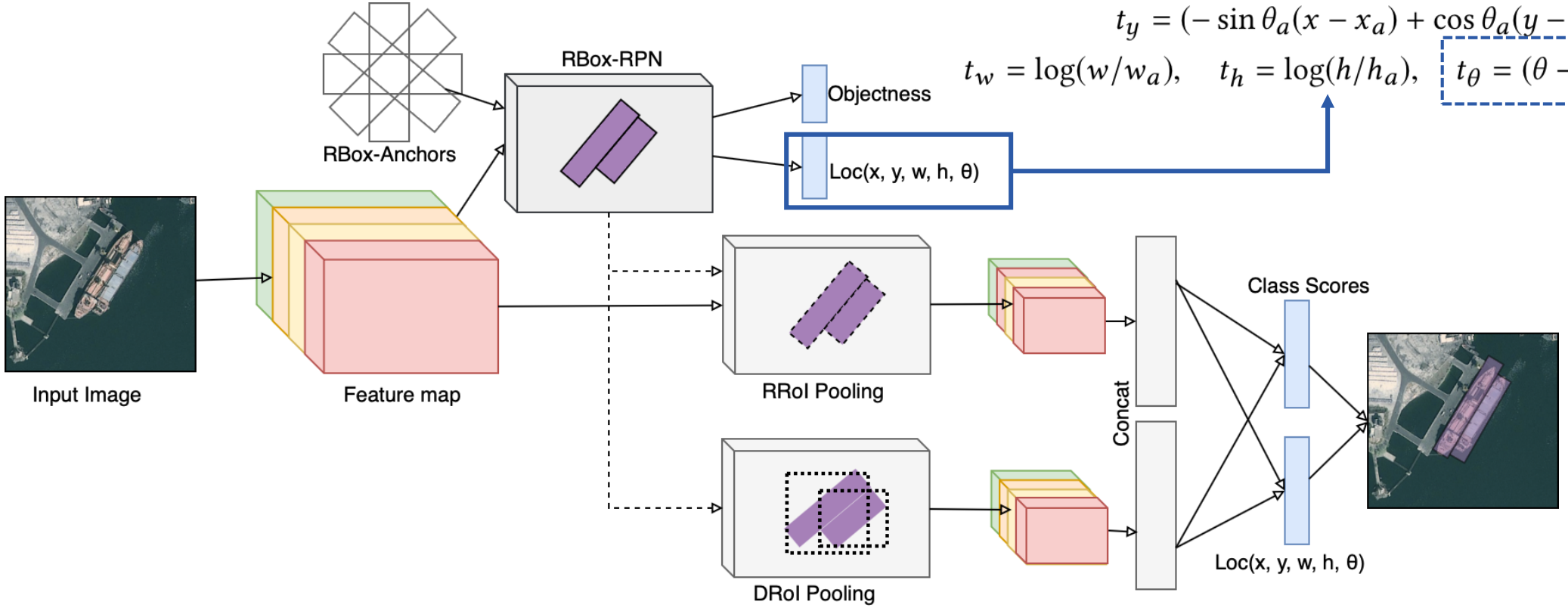
```
translation_matrix = tf.concat([
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 1, None] * original_size[1] - tf.cast(original_size[1] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 1), tf.float32),
    tf.ones((tf.shape(boxes)[0], 1), tf.float32),
    boxes[:, 0, None] * original_size[0] - tf.cast(original_size[0] / 2, tf.float32),
    tf.zeros((tf.shape(boxes)[0], 2), tf.float32),
],
    axis=1)

rotation_matrix = tf.contrib.image.angles_to_projective_transforms(boxes[:, 4],
                                                                   tf.cast(size_max, tf.float32),
                                                                   tf.cast(size_max, tf.float32)
                                                                   )

composed_matrix = tf.contrib.image.compose_transforms(translation_matrix, rotation_matrix)

transformed_images = tf.contrib.image.transform(transformed_images, composed_matrix,
                                                interpolation='BILINEAR')
```

```
height_ratio = (boxes[:, 2] * original_size[0]) / tf.cast(size_max, tf.float32)
width_ratio = (boxes[:, 3] * original_size[1]) / tf.cast(size_max, tf.float32)
transformed_images = tf.image.crop_and_resize(transformed_images,
```

RBox-CNN 구현: RBox Regression

- RBox-CNN: RBox Regression

$$t_x = (\cos\theta_a(x - x_a) + \sin\theta_a(y - y_a))/w_a$$
$$t_y = (-\sin\theta_a(x - x_a) + \cos\theta_a(y - y_a))/h_a$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \quad t_\theta = (\theta - \theta_a)\pi/2$$



RBox-CNN 구현: RBox Regression

- RBox-CNN: RBox Regression
  - 코드 파일: box_coders/faster_rcnn_rbox_coder.py
  - 주요 클래스: FasterRcnnRBoxCoder

```
tx = (tf.cos(ang_a) * (xcenter - xcenter_a) + tf.sin(ang_a) * (ycenter - ycenter_a)) / wa
ty = (-tf.sin(ang_a) * (xcenter - xcenter_a) + tf.cos(ang_a) * (ycenter - ycenter_a)) / ha
tw = tf.log(w / wa)
th = tf.log(h / ha)
ta = (ang-ang_a) / (math.pi / 2)
```

RBox-CNN 구현: RBox Regression

- RBox-CNN: RBox Regression
  - 코드 파일: box_coders/faster_rcnn_rbox_coder.py
  - 주요 클래스: FasterRcnnRBoxCoder

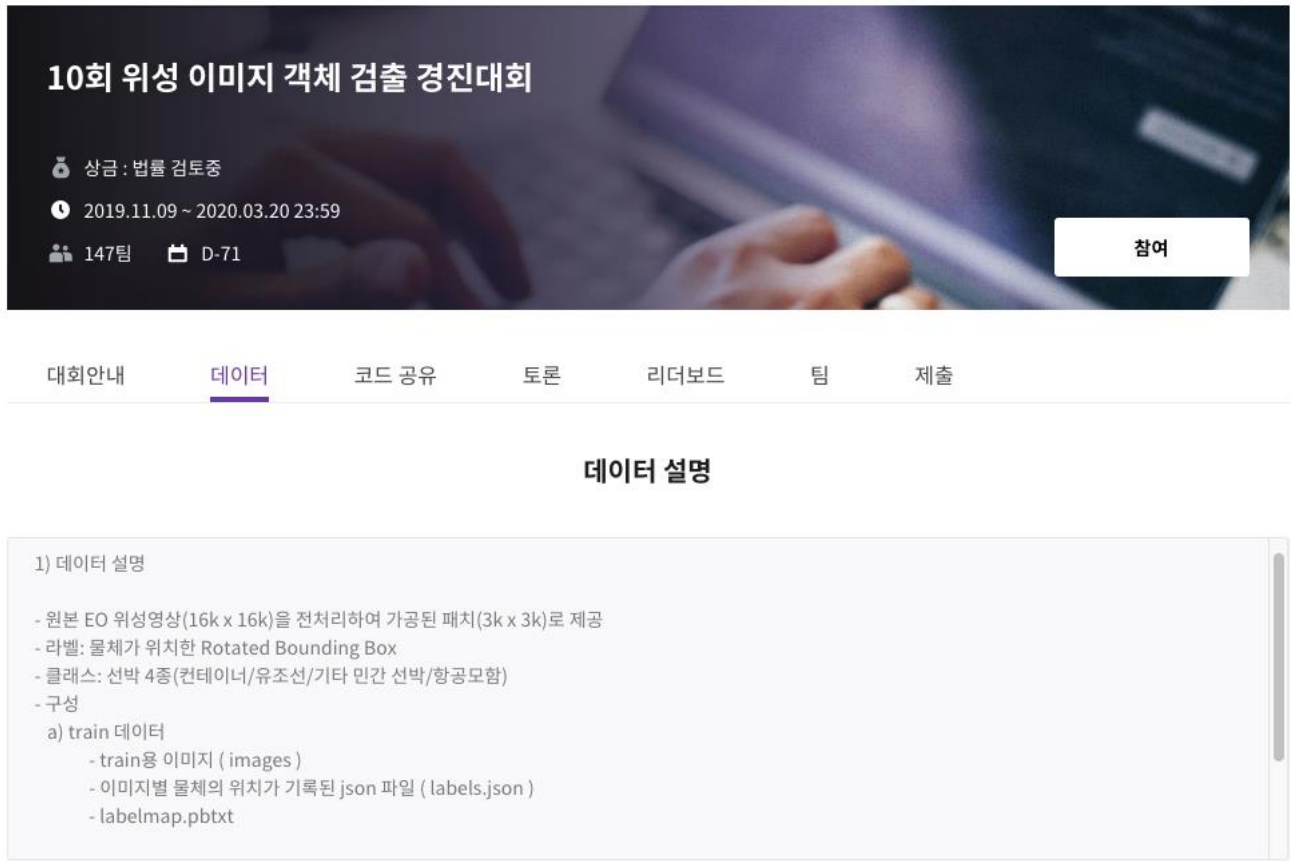
```
tx = (tf.cos(ang_a) * (xcenter - xcenter_a) + tf.sin(ang_a) * (ycenter - ycenter_a)) / wa
ty = (-tf.sin(ang_a) * (xcenter - xcenter_a) + tf.cos(ang_a) * (ycenter - ycenter_a)) / ha
tw = tf.log(w / wa)
th = tf.log(h / ha)
ta = (ang-ang_a) / (math.pi / 2)
```

$$t_x = (\cos\theta_a(x - x_a) + \sin\theta_a(y - y_a))/w_a$$
$$t_y = (-\sin\theta_a(x - x_a) + \cos\theta_a(y - y_a))/h_a$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \quad t_\theta = (\theta - \theta_a)\pi/2$$

03 베이스라인 제출해보기

베이스라인 제출: 데이터셋 다운로드

- 데이터셋 다운로드
  - 데이콘 대회 페이지: https://newfront.dacon.io/competitions/official/235492/data?join=0
  - 영상 파일 수
    - ✓ 학습: 1,664개
    - ✓ 테스트: 1,237개
  - 용량
    - ✓ 학습: 약 21GB
    - ✓ 테스트: 약 15GB



베이스라인 제출: 데이터셋 다운로드

- 데이터셋
  - 라벨 포맷: GeoJSON
  - 라벨 내 선박 항목

항목	설명
geometry	네 모서리 점의 지리좌표
bounds_imcoords	네 모서리 점의 이미지좌표
edited_by	라벨 검토자
feature_id	고유 해쉬값
iamge_id	이미지 파일 이름
ingest_time	영상 촬영 시간
type_id	선박 종류 아이디
type_name	선박 종류 이름


```
"features": [
    {
        "geometry": {
            "coordinates": [
                [
                    [
                        -72.09348,
                        41.397123,
                        0.0
                    ],
                    [
                        -72.093269,
                        41.397229,
                        0.0
                    ],
                    [
                        -72.092903,
                        41.396501,
                        0.0
                    ],
                    [
                        -72.093115,
                        41.396395,
                        0.0
                    ]
                ]
            ],
            "type": "Polygon"
        },
        "properties": {
            "bounds_imcoords": "271.37967471076746,
            "edited_by": "Ilwon Lee @ SI Analytics"
            "feature_id": [
                "3cd627bc8bb5856948f4a00622666c7d12
            ],
            "image_id": "0.png",
            "ingest_time": "2019:01:14 20:40:08",
            "type_id": 4,
            "type_name": "maritime vessels"
        },
        "type": "Feature"
    },
```

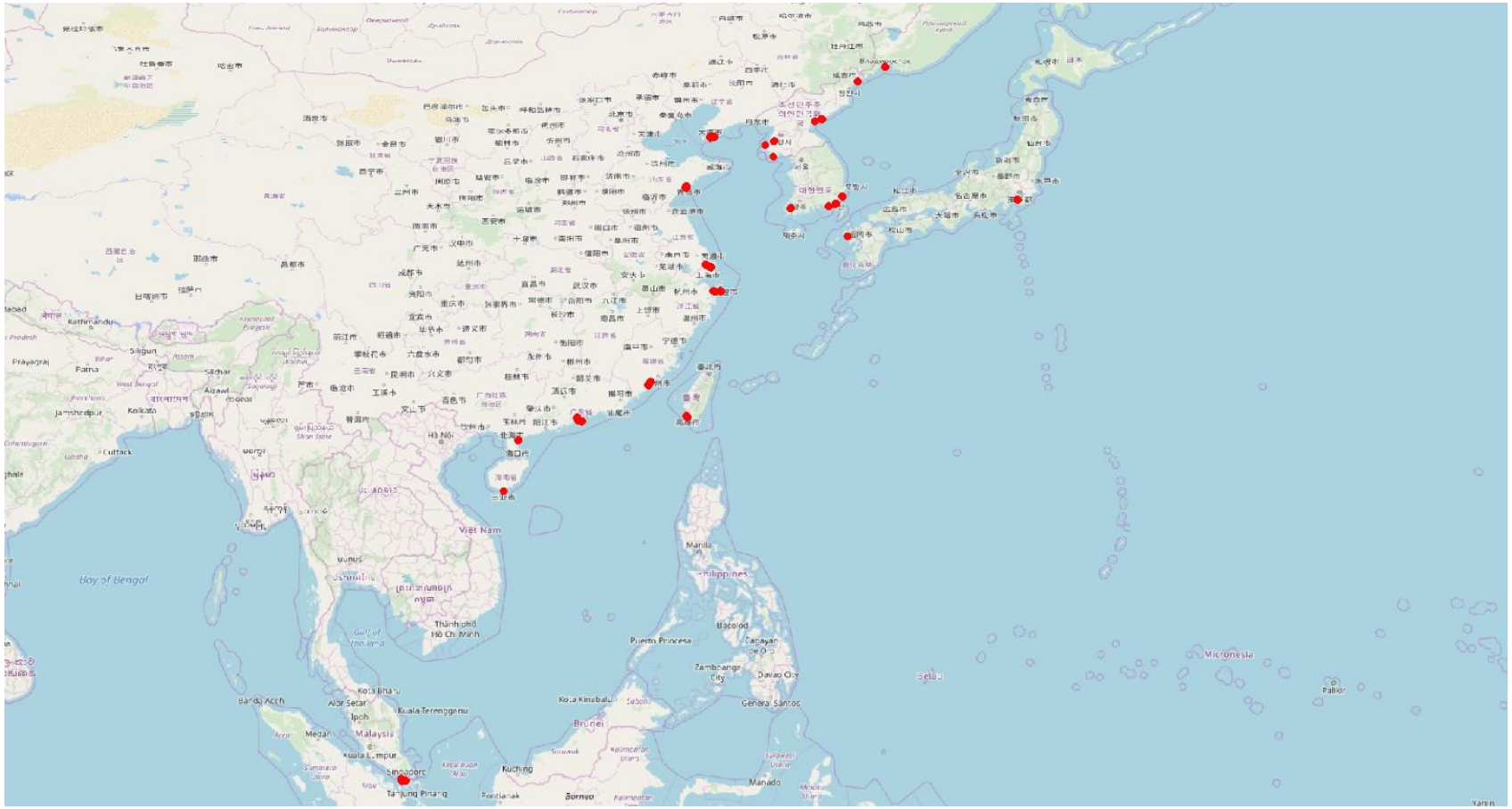
베이스라인 제출: 데이터셋 다운로드

- 데이터셋
  - 공개된 학습영상 내 선박 수: 약 10,000 개
  - 선박 종류

아이디	이름
1	container
2	oil tanker
3	aircraft carrier
4	maritime vessels

베이스라인 제출: 데이터셋 다운로드

- 데이터셋: 촬영지역

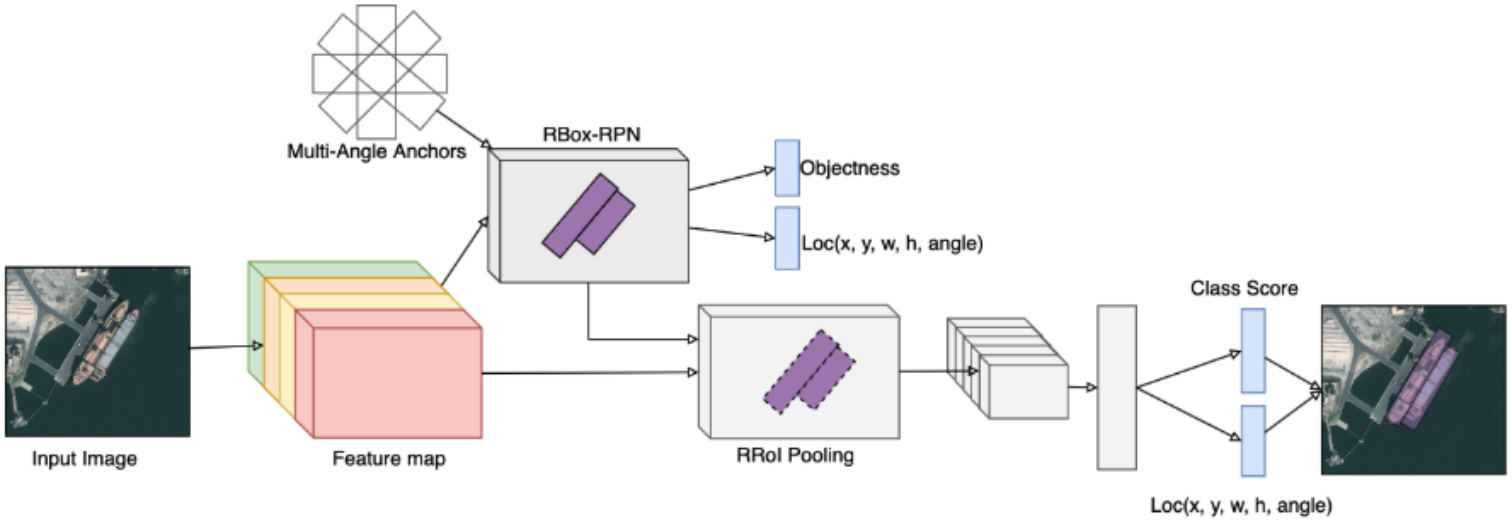


베이스라인 제출: 코드 다운로드

- 코드 다운로드
  - Git 주소: https://github.com/SIAnalytics/simplified_rbox_cnn

RBox-CNN

This is implementation of simplified version of RBox-CNN "RBox-CNN: rotated bounding box based CNN for ship detection in remote sensing image".
This code is based on Tensorflow Object Detection API and specially modified for the Object Detection Challenge in Satellite Image



베이스라인 제출: TFRecords 생성

- TFRecords
  - Tensorflow에서 사용 가능한 데이터 직렬화(Serialization) 파일 포맷
  - 많은 양의 데이터를 빠르게 읽을 수 있음

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py …

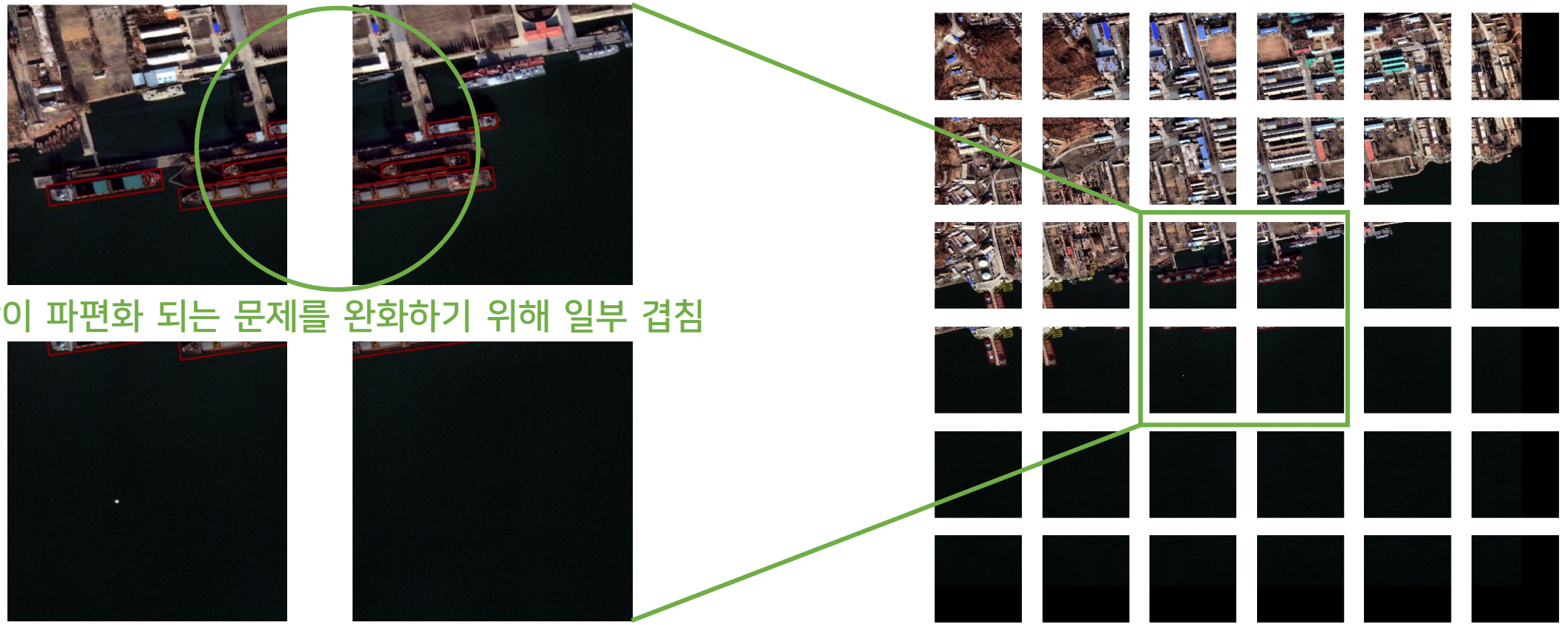
베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py … (먼저 내부에서 일어나는 일을 살펴보면)

작은 영상으로 조각
(일부분 겹침)

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py … (먼저 내부에서 일어나는 일 살펴보면)



선박이 파편화 되는 문제를 완화하기 위해 일부 겹침

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py …

```
$ python3 create_dataset.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Co
gument of issubdtype from `float` to `np.floating` is deprecated. In future,
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: create_dataset.py [-h] --src_dir DIR [--dst_path FILE]
                         [--patch_size PATCH_SIZE]
                         [--patch_overlay PATCH_OVERLAY]
                         [--object_fraction_thresh OBJECT_FRACTION_THRESH]
                         [--is_include_only_pos]

Create TF Records from geojson

optional arguments:
  -h, --help            show this help message and exit
  --src_dir DIR         Root directory to geojson and images
  --dst_path FILE       Path to save tfrecords
  --patch_size PATCH_SIZE
                        Patch size
  --patch_overlay PATCH_OVERLAY
                        Overlay size for patching
  --object_fraction_thresh OBJECT_FRACTION_THRESH
                        Threshold value for determining contained objects
  --is_include_only_pos
                        Whether or not to include only positive patch
                        image(containing at least one object)
```

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py …

```
$ python3 create_dataset.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Co
gument of issubdtype from `float` to `np.floating` is deprecated. In future,
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: create_dataset.py [-h] --src_dir DIR [--dst_path FILE]
                         [--patch_size PATCH_SIZE]
                         [--patch_overlay PATCH_OVERLAY]
                         [--object_fraction_thresh OBJECT_FRACTION_THRESH]
                         [--is_include_only_pos]

Create TF Records from geojson

optional arguments:
  -h, --help            show this help message and exit
  --src_dir DIR         Root directory to geojson and images
  --dst_path FILE       Path to save tfrecords
  --patch_size PATCH_SIZE
                        Patch size
  --patch_overlay PATCH_OVERLAY
                        Overlay size for patching
  --object_fraction_thresh OBJECT_FRACTION_THRESH
                        Threshold value for determining contained objects
  --is_include_only_pos
                        Whether or not to include only positive patch
                        image(containing at least one object)
```

패치 크기와 겹침 크기

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py …

```
$ python3 create_dataset.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Co
gument of issubdtype from `float` to `np.floating` is deprecated. In future,
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: create_dataset.py [-h] --src_dir DIR [--dst_path FILE]
                         [--patch_size PATCH_SIZE]
                         [--patch_overlay PATCH_OVERLAY]
                         [--object_fraction_thresh OBJECT_FRACTION_THRESH]
                         [--is_include_only_pos]

Create TF Records from geojson

optional arguments:
  -h, --help            show this help message and exit
  --src_dir DIR         Root directory to geojson and images
  --dst_path FILE       Path to save tfrecords
  --patch_size PATCH_SIZE
                        Patch size
  --patch_overlay PATCH_OVERLAY
                        Overlay size for patching
  --object_fraction_thresh OBJECT_FRACTION_THRESH
                        Threshold value for determining contained objects
  --is_include_only_pos
                        Whether or not to include only positive patch
                        image(containing at least one object)
```

패치 크기와 겹침 크기

객체가 잘린 경우 허용 비율

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py …

```
$ python3 create_dataset.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Co
gument of issubdtype from `float` to `np.floating` is deprecated. In future,
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: create_dataset.py [-h] --src_dir DIR [--dst_path FILE]
                         [--patch_size PATCH_SIZE]
                         [--patch_overlay PATCH_OVERLAY]
                         [--object_fraction_thresh OBJECT_FRACTION_THRESH]
                         [--is_include_only_pos]

Create TF Records from geojson

optional arguments:
  -h, --help            show this help message and exit
  --src_dir DIR         Root directory to geojson and images
  --dst_path FILE       Path to save tfrecords
  --patch_size PATCH_SIZE
                        Patch size
  --patch_overlay PATCH_OVERLAY
                        Overlay size for patching
  --object_fraction_thresh OBJECT_FRACTION_THRESH
                        Threshold value for determining contained objects
  --is_include_only_pos
                        Whether or not to include only positive patch
                        image(containing at least one object)
```

패치 크기와 겹침 크기

객체가 잘린 경우 허용 비율

객체가 있는 패치만 포함

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py ₩

    --src_dir=dataset/train/ ₩

    --dst_path=tfrecords/train_40p_w_neg_img_only_pos.tfrecords ₩

    --is_include_only_pos

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py ₩

    --src_dir=dataset/train/ ₩

    --dst_path=tfrecords/train_40p_w_neg_img_only_pos.tfrecords ₩

    --is_include_only_pos
  - 실행 중 화면

```
10%|#####9                                                    | 154/1593 [01:12<11:15,  2.13it/s]
```

베이스라인 제출: TFRecords 생성

- TFRecords 생성
  - 실행: python create_dataset.py ₩

    --src_dir=dataset/train/ ₩

    --dst_path=tfrecords/train_40p_w_neg_img_only_pos.tfrecords ₩

    --is_include_only_pos
  - 실행 중 화면

```
10%|#####9                                                    | 154/1593 [01:12<11:15,  2.13it/s]
```

  - 실행 완료

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

Anchor 설정

```
  scales: [0.25, 0.5, 0.75, 1]
  aspect_ratios: [3.0, 5.0, 7.0]
  angles: [-1.57079632679, -1.0471975512, -0.5235987756, 0.0, 0.5235987756, 1.0471975512, 1.57079632679]
```

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

Anchor 설정

```
 scales: [0.25, 0.5, 0.75, 1]
 aspect_ratios: [3.0, 5.0, 7.0]
 angles: [-1.57079632679, -1.0471975512, -0.5235987756, 0.0, 0.5235987756, 1.0471975512, 1.57079632679]
```

Pretrained Weight 경로

```
 fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
```

We recommend to set 'fine_tune_checkpoint_path' to the pretrained model(faster_rcnn_resnet101_coco_11_06_2017)

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

Anchor 설정

```
 scales: [0.25, 0.5, 0.75, 1]
 aspect_ratios: [3.0, 5.0, 7.0]
 angles: [-1.57079632679, -1.0471975512, -0.5235987756, 0.0, 0.5235987756, 1.0471975512, 1.57079632679]
```

Pretrained Weight 경로

```
 fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
```

We recommend to set 'fine_tune_checkpoint_path' to the pretrained model(faster_rcnn_resnet101_coco_11_06_2017)

Data Augmentation 설정

```
 data_augmentation_options {random_horizontal_flip_rbox {}}
 data_augmentation_options {random_vertical_flip_rbox {}}
 data_augmentation_options {random_rotate_rbox {by_90: true}}
```

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

Anchor 설정

```
 scales: [0.25, 0.5, 0.75, 1]
 aspect_ratios: [3.0, 5.0, 7.0]
 angles: [-1.57079632679, -1.0471975512, -0.5235987756, 0.0, 0.5235987756, 1.0471975512, 1.57079632679]
```

Pretrained Weight 경로

```
 fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
```

We recommend to set 'fine_tune_checkpoint_path' to the pretrained model(faster_rcnn_resnet101_coco_11_06_2017)

Data Augmentation 설정

```
 data_augmentation_options {random_horizontal_flip_rbox {}}
 data_augmentation_options {random_vertical_flip_rbox {}}
 data_augmentation_options {random_rotate_rbox {by_90: true}}
```

TFRecords 경로

```
 tf_record_input_reader {
   input_path: "PATH_TO_BE_CONFIGURED"
 }
```

베이스라인 제출: 학습

- 학습: 모델 설정
  - 설정 파일: configs/rbox_cnn_resnet101.config

RBox-CNN 모델 선택

```
model {
  rfaster_rcnn {
```

Anchor 설정

```
  scales: [0.25, 0.5, 0.75, 1]
  aspect_ratios: [3.0, 5.0, 7.0]
  angles: [-1.57079632679, -1.0471975512, -0.5235987756, 0.0, 0.5235987756, 1.0471975512, 1.57079632679]
```

Pretrained Weight 경로

```
  fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
```

We recommend to set 'fine_tune_checkpoint_path' to the pretrained model(faster_rcnn_resnet101_coco_11_06_2017)

Data Augmentation 설정

```
  data_augmentation_options {random_horizontal_flip_rbox {}}
  data_augmentation_options {random_vertical_flip_rbox {}}
  data_augmentation_options {random_rotate_rbox {by_90: true}}
```

TFRecords 경로

```
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED"
  }
```

Label map 경로

```
  label_map_path: "PATH_TO_BE_CONFIGURED"
```

label_map.pbtxt

```
item {
    id: 1
    name: 'container'
}
item {
    id: 2
    name: 'oil tanker'
}
item {
    id: 3
    name: 'aircraft carrier'
}
item {
    id: 4
    name: 'maritime vessels'
}
```

베이스라인 제출: 학습

- 학습: 학습 실행
  - 설정 파일: python train.py ...

```
$ python3 train.py --help
Training executable for detection models.

This executable is used to train DetectionModels.

Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --pipeline_config_path=pipeline_config.pbtxt

flags:

train.py:
  --log_every_n_steps: The frequency, in terms of global steps, that the loss and global step are
    logged.
    (default: '1')
    (an integer)
  --pipeline_config_path: Path to a pipeline_pb2.TrainEvalPipelineConfig config file. If provided,
    other configs are ignored
    (default: '')
  --save_interval_secs: Interval in seconds to save a check point file
    (default: '3600')
    (an integer)
  --train_dir: Directory to save the checkpoints and training summaries.
    (default: '')

Try --helpfull to get a list of all flags.
```

베이스라인 제출: 학습

- 학습: 학습 실행
  - 설정 파일: python train.py ...

모델 설정 파일 경로

```
$ python3 train.py --help
Training executable for detection models.

This executable is used to train DetectionModels.

Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --pipeline_config_path=pipeline_config.pbtxt

flags:

train.py:
  --log_every_n_steps: The frequency, in terms of global steps, that the loss and global step are
    logged.
    (default: '1')
    (an integer)
  --pipeline_config_path: Path to a pipeline_pb2.TrainEvalPipelineConfig config file. If provided,
    other configs are ignored
    (default: '')
  --save_interval_secs: Interval in seconds to save a check point file
    (default: '3600')
    (an integer)
  --train_dir: Directory to save the checkpoints and training summaries.
    (default: '')

Try --helpfull to get a list of all flags.
```

베이스라인 제출: 학습

- 학습: 학습 실행
  - 설정 파일: python train_n_eval.sh

```
############ Set Configurations#############################################################
gpu_no=1
pipeline_config_path='configs/rbox_cnn_resnet101.config'
dst_path='../ckpt/rbox_cnn_resnet101/'
log_every_n_steps=100
save_interval_secs=3600
############################################################################################


# train
CUDA_VISIBLE_DEVICES=$gpu_no python3 train.py \
    --logtostderr \
    --pipeline_config_path=$pipeline_config_path \
    --train_dir=$dst_path \
    --save_interval_secs=$save_interval_secs \
    --log_every_n_steps=$log_every_n_steps


# evaluation
CUDA_VISIBLE_DEVICES=$gpu_no python3 eval.py \
    --logtostderr \
    --pipeline_config_path=$pipeline_config_path \
    --checkpoint_dir=$dst_path \
    --eval_dir=$dst_path \
    --run_mode=all
```

학습 완료 후

학습 데이터 일부분을
검증용 TFRecords 만든 경우
저장된 모든 체크포인트에서

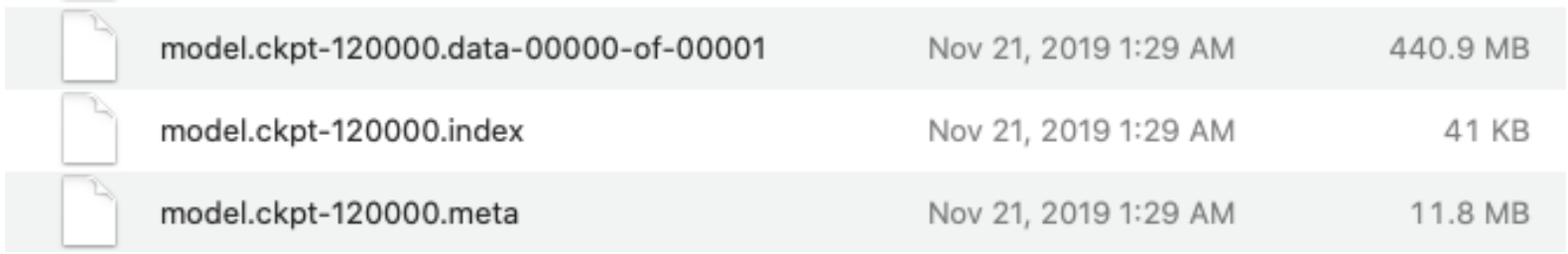
평가 수행

베이스라인 제출: 학습

- 학습: 학습 실행
  - 설정 파일: python train_n_eval.sh
  - 실행 중 화면

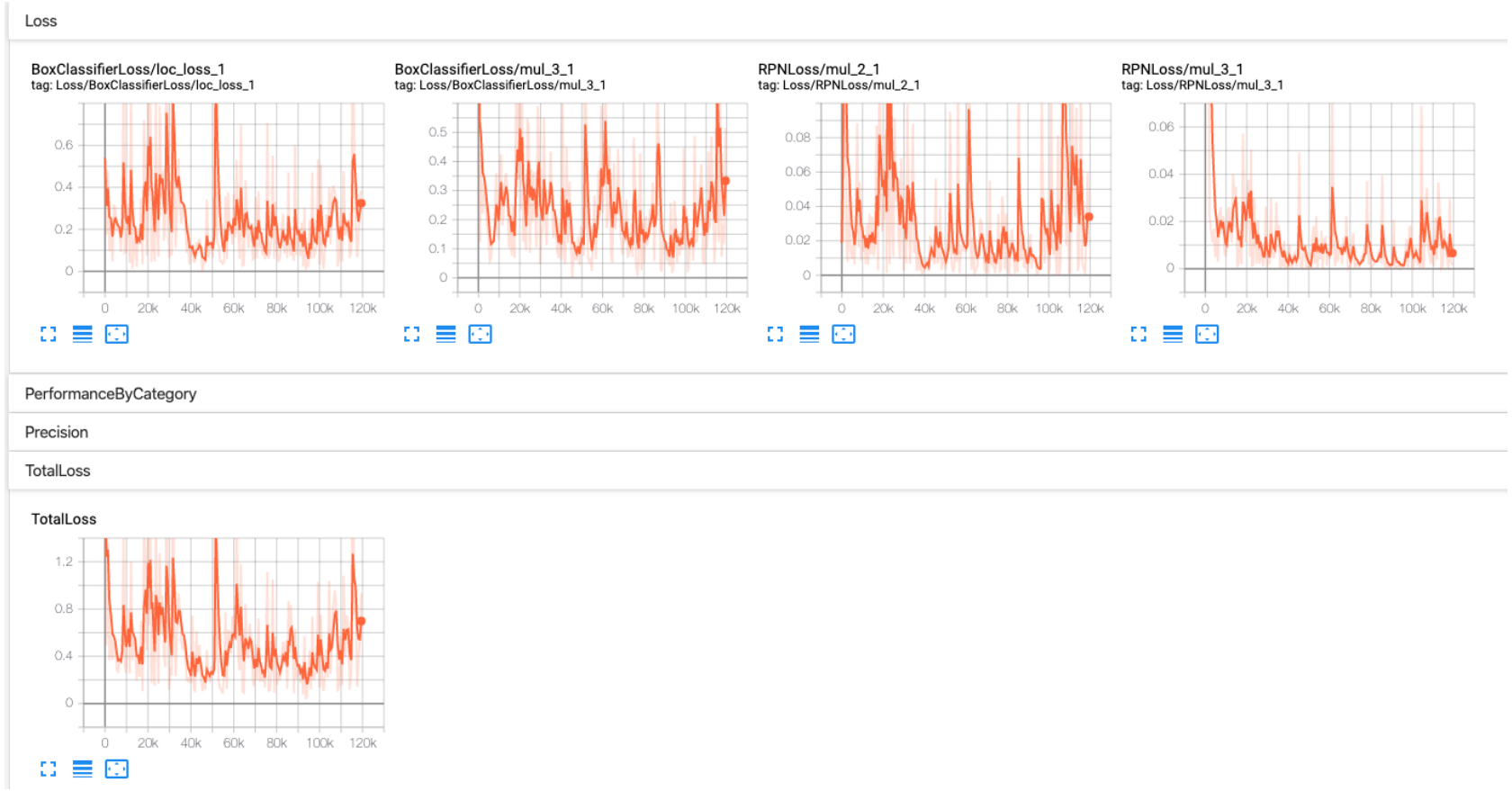
```
INFO:tensorflow:global step 100: loss = 0.2170 (0.291 sec/step)
INFO:tensorflow:global step 200: loss = 0.3361 (0.290 sec/step)
INFO:tensorflow:global step 300: loss = 0.6527 (0.292 sec/step)
INFO:tensorflow:global step 400: loss = 0.4046 (0.337 sec/step)
INFO:tensorflow:global step 500: loss = 1.4700 (0.290 sec/step)
INFO:tensorflow:global step 600: loss = 0.5721 (0.287 sec/step)
```

  - 실행 완료



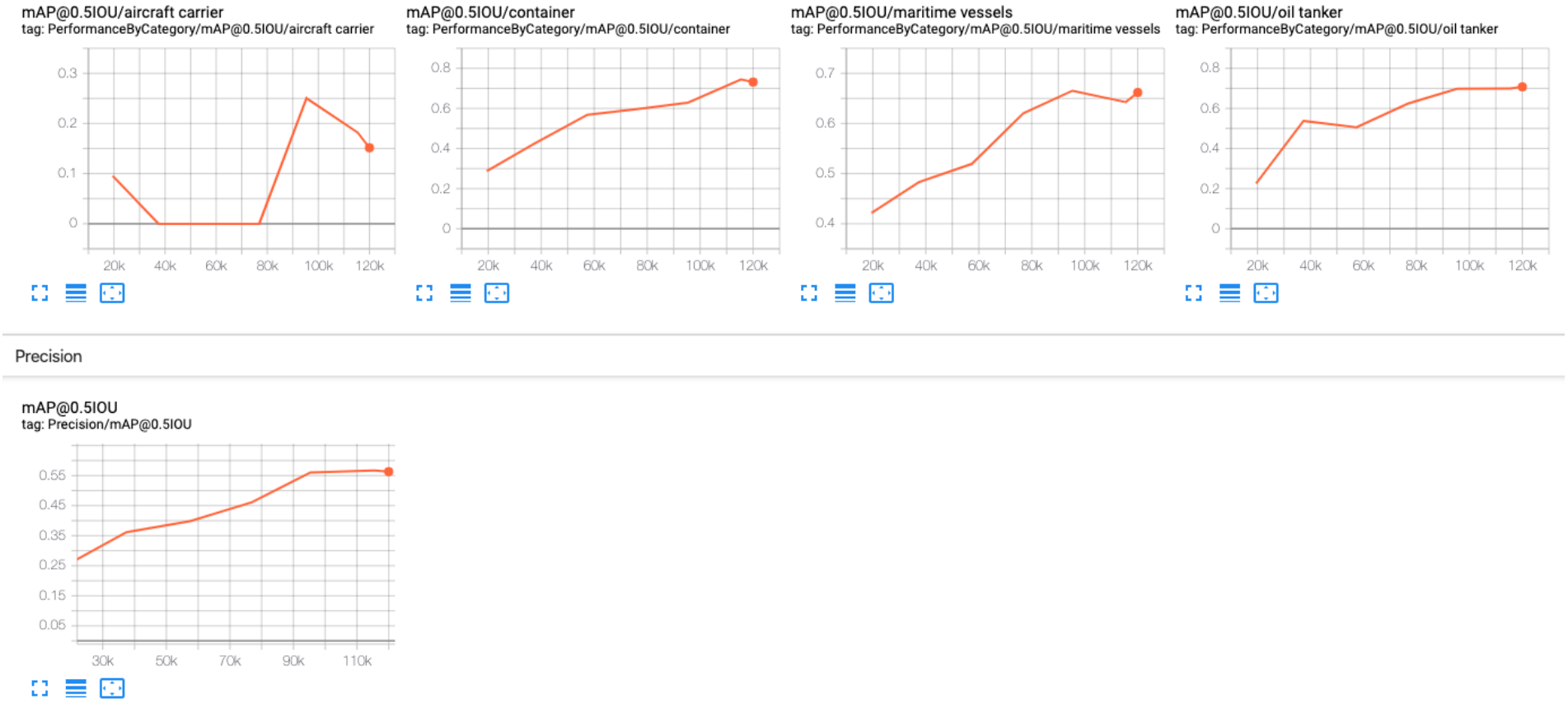
베이스라인 제출: 학습

- 학습: 텐서보드로 결과 확인
  - Loss



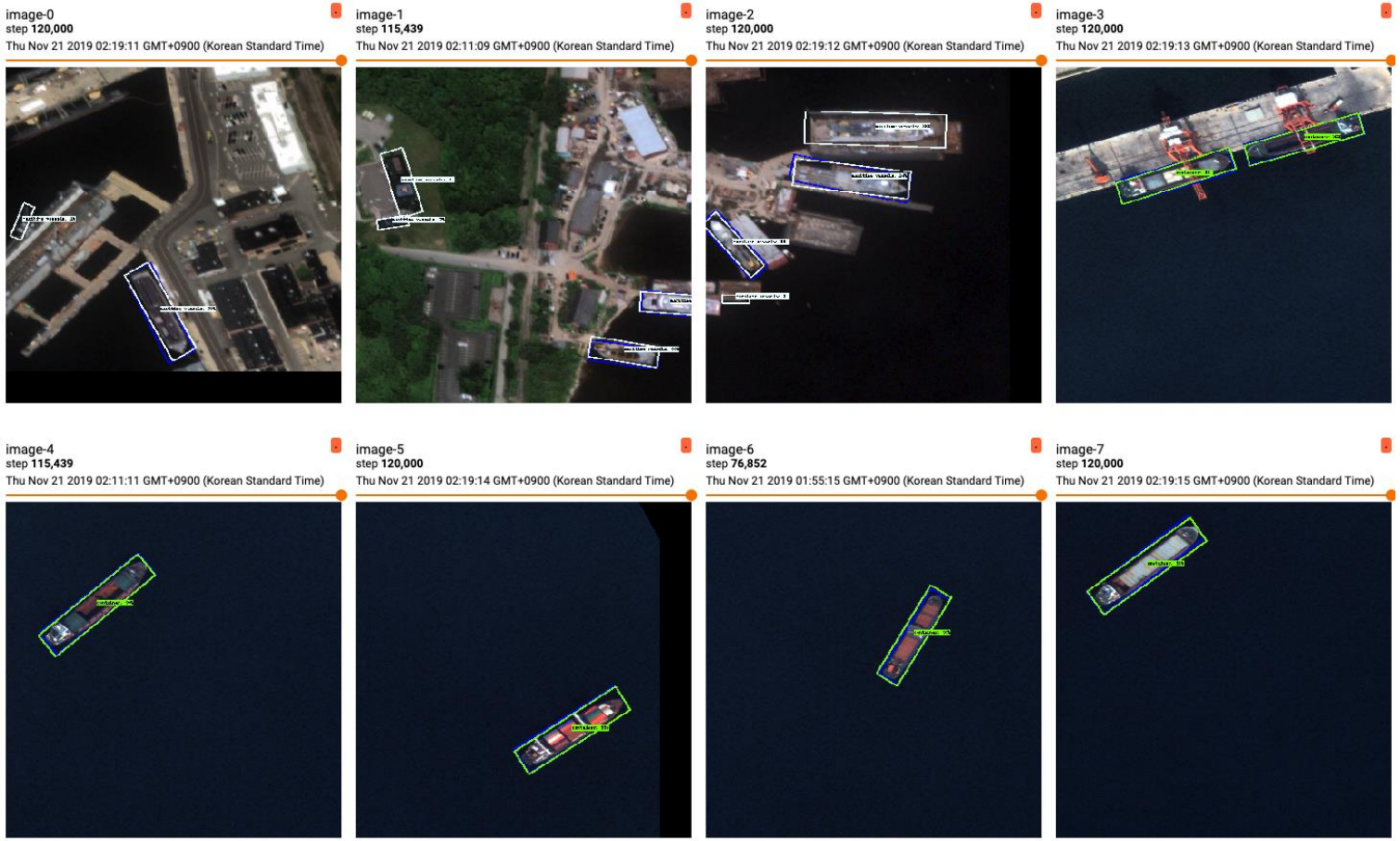
베이스라인 제출: 학습

- 학습: 텐서보드로 결과 확인
  - mAP(검증용 TFRecords를 만들어서 평가했다면 표시됨)



베이스라인 제출: 학습

- 학습: 텐서보드로 결과 확인
  - 검출 결과 예시 이미지(검증용 TFRecords를 만들어서 평가했다면 표시됨)



베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py …

```
$ python3 inference.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second ar
gument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: inference.py [-h] [--pipeline_config_path PIPELINE_CONFIG_PATH]
                    [--ckpt_path CKPT_PATH] [--image_dir IMAGE_DIR]
                    [--dst_path DST_PATH] [--patch_size PATCH_SIZE]
                    [--overlay_size OVERLAY_SIZE]

optional arguments:
  -h, --help            show this help message and exit
  --pipeline_config_path PIPELINE_CONFIG_PATH
                        Path to a pipeline_pb2.TrainEvalPipelineConfig config
                        file. (default: None)
  --ckpt_path CKPT_PATH
                        Path to trained checkpoint, typically of the form
                        path/to/model-{'type': 'str', 'nargs': None,
                        'required': False, 'help': 'Path to trained
                        checkpoint, typically of the form
                        path/to/model-%step.ckpt', 'dest': 'ckpt_path',
                        'const': None, 'default': None, 'choices': None,
                        'prog': 'inference.py', 'container':
                        <argparse._ArgumentGroup object at 0x7fea12a95da0>,
                        'metavar': None, 'option_strings': ['--
                        ckpt_path']}tep.ckpt (default: None)
  --image_dir IMAGE_DIR
                        Path to images to be inferred (default: None)
  --dst_path DST_PATH   Path to save detection output (default: None)
  --patch_size PATCH_SIZE
                        Patch size, width and height of patch is equal.
                        (default: 768)
  --overlay_size OVERLAY_SIZE
                        Overlay size for patching. (default: 256)
```

베이스라인 제출: 인퍼런스(Inference)

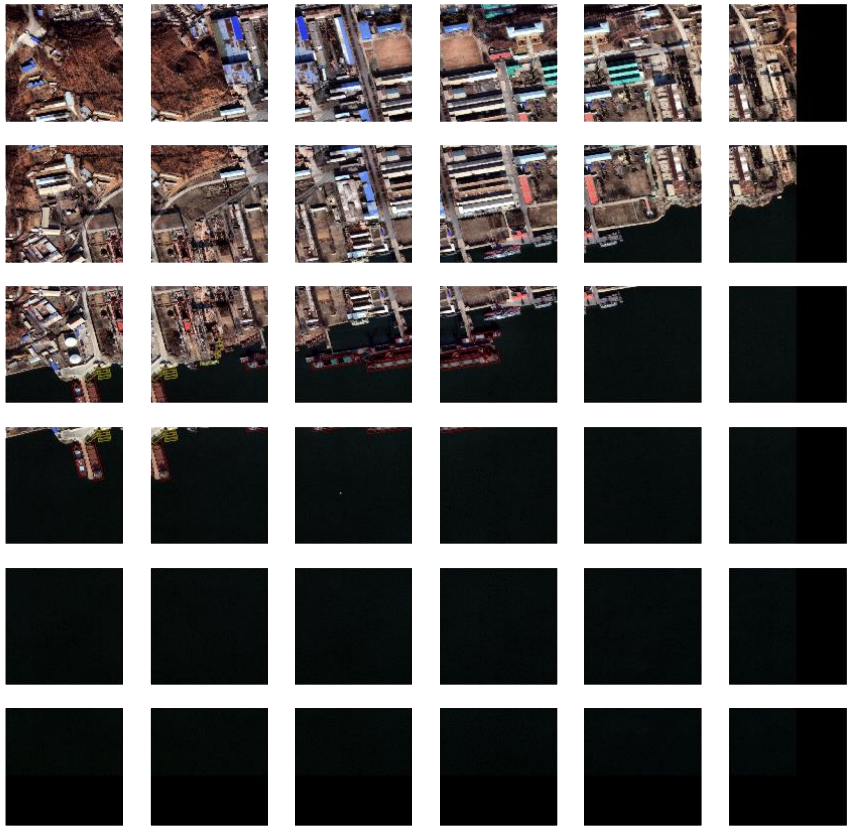
- 인퍼런스
  - 실행: python3 inference.py …

```
$ python3 inference.py --help
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second ar
gument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np
.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
usage: inference.py [-h] [--pipeline_config_path PIPELINE_CONFIG_PATH]
                    [--ckpt_path CKPT_PATH] [--image_dir IMAGE_DIR]
                    [--dst_path DST_PATH] [--patch_size PATCH_SIZE]
                    [--overlay_size OVERLAY_SIZE]

optional arguments:
  -h, --help            show this help message and exit
  --pipeline_config_path PIPELINE_CONFIG_PATH
                        Path to a pipeline_pb2.TrainEvalPipelineConfig config
                        file. (default: None)
  --ckpt_path CKPT_PATH
                        Path to trained checkpoint, typically of the form
                        path/to/model-{'type': 'str', 'nargs': None,
                        'required': False, 'help': 'Path to trained
                        checkpoint, typically of the form
                        path/to/model-%step.ckpt', 'dest': 'ckpt_path',
                        'const': None, 'default': None, 'choices': None,
                        'prog': 'inference.py', 'container':
                        <argparse._ArgumentGroup object at 0x7fea12a95da0>,
                        'metavar': None, 'option_strings': ['--
                        ckpt_path']}tep.ckpt (default: None)
  --image_dir IMAGE_DIR
                        Path to images to be inferred (default: None)
  --dst_path DST_PATH   Path to save detection output (default: None)
  --patch_size PATCH_SIZE
                        Patch size, width and height of patch is equal.
                        (default: 768)
  --overlay_size OVERLAY_SIZE
                        Overlay size for patching. (default: 256)
```

가능한 학습때와 같은 값으로 설정

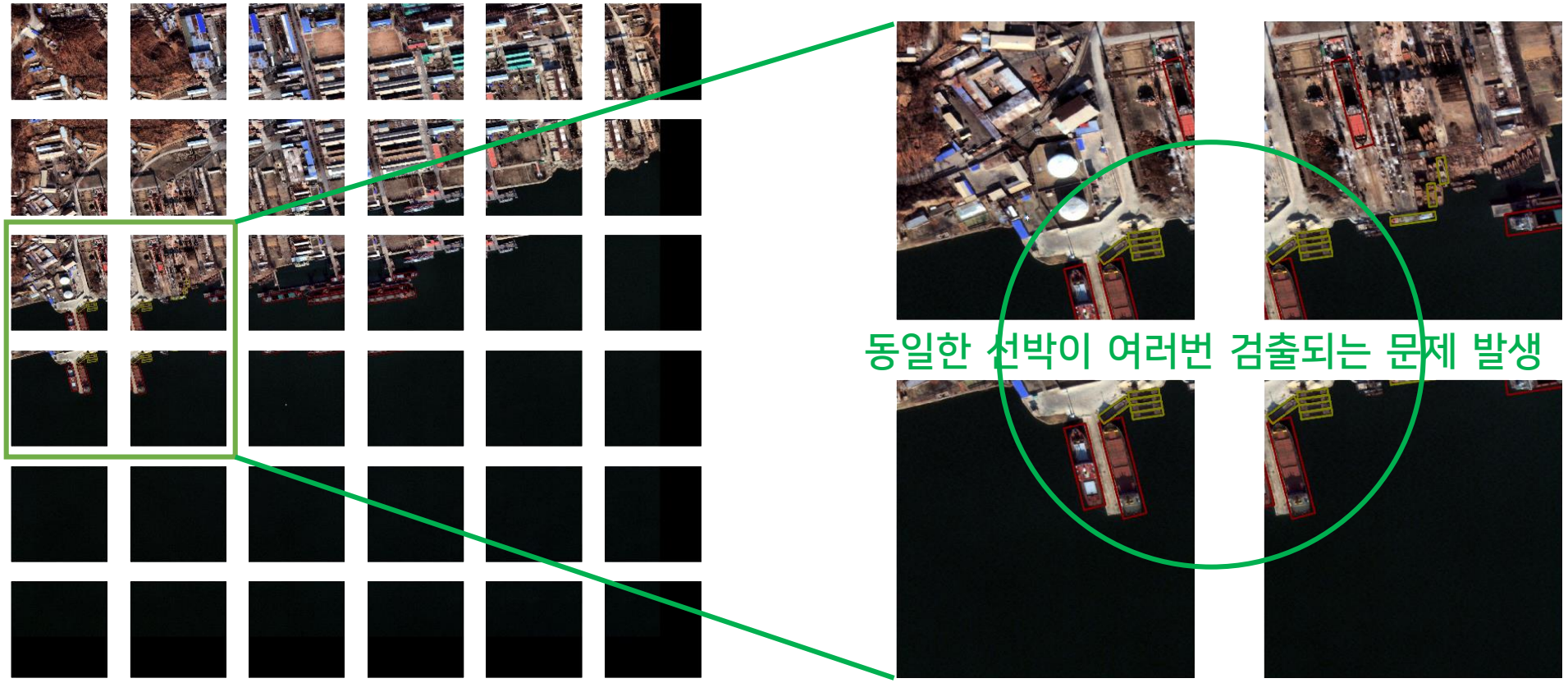
베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py … (내부에서 일어나는 일을 살펴보면)



베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py … (내부에서 일어나는 일을 살펴보면)



베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py … (내부에서 일어나는 일을 살펴보면)

베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py … (내부에서 일어나는 일을 살펴보면)

NMS 적용

베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py …

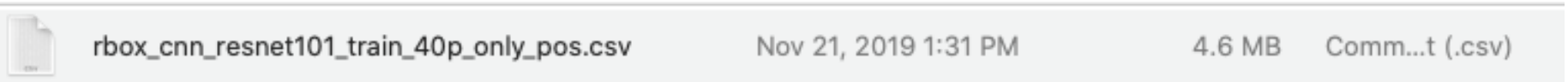
```
CUDA_VISIBLE_DEVICES='0'  python3 inference.py \
    --pipeline_config_path=configs/rbox_cnn_resnet101_train_40p_only_pos.config \
    --ckpt_path=ckpt/rbox_cnn_resnet101_train_40p_only_pos/model.ckpt-120000 \
    --image_dir=dataset/geojson/test/images \
    --dst_path=inference/rbox_cnn_resnet101_train_40p_only_pos.csv
```

베이스라인 제출: 인퍼런스(Inference)

- 인퍼런스
  - 실행: python3 inference.py …

```
CUDA_VISIBLE_DEVICES='0'  python3 inference.py \
    --pipeline_config_path=configs/rbox_cnn_resnet101_train_40p_only_pos.config \
    --ckpt_path=ckpt/rbox_cnn_resnet101_train_40p_only_pos/model.ckpt-120000 \
    --image_dir=dataset/geojson/test/images \
    --dst_path=inference/rbox_cnn_resnet101_train_40p_only_pos.csv
```

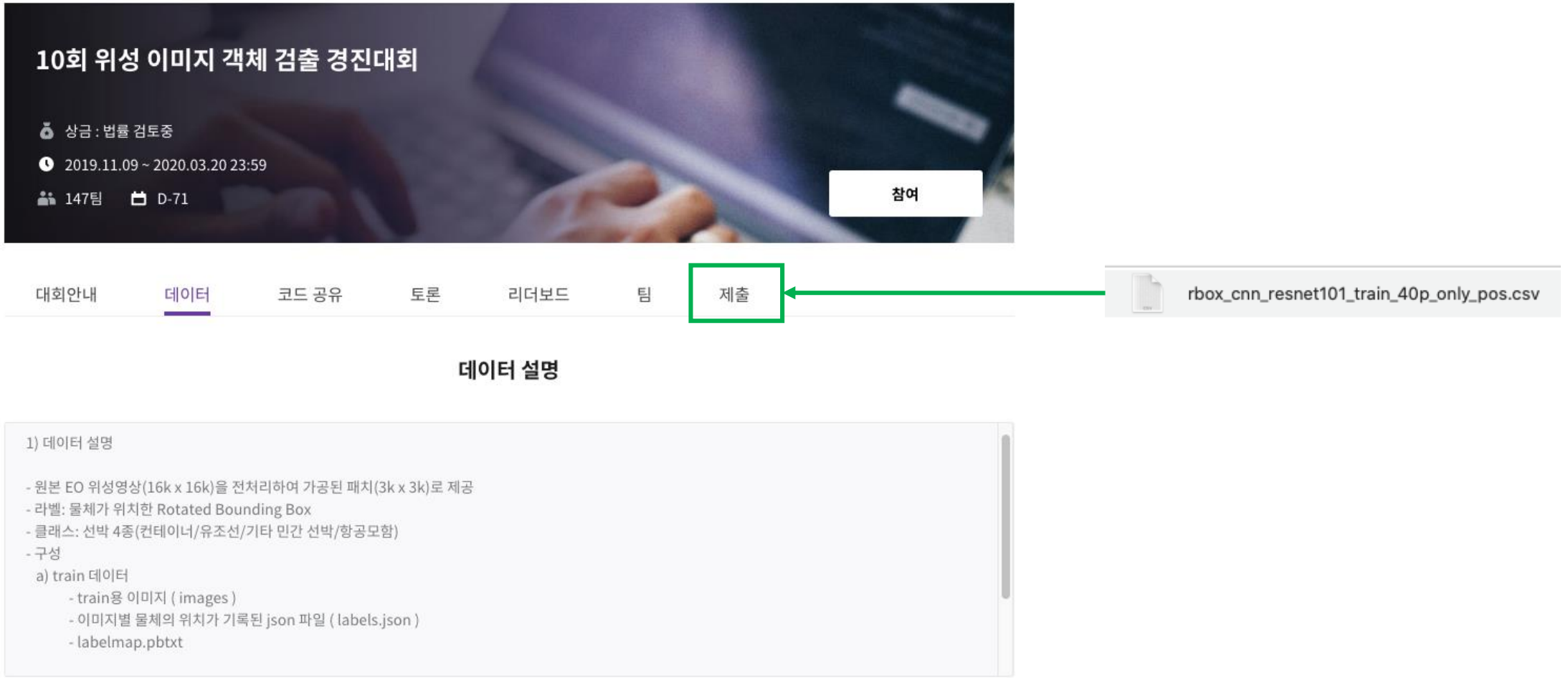
  - 실행 결과



file_name	class_id	confidence	point1_x	point1_y	point2_x	point2_y	point3_x	point3_y	point4_x	point4_y
480.png	1	0.0566545	1775.43931	25.7642225	1870.47532	-46.430483	1903.50601	-2.9493917	1808.46999	69.2453138
556.png	4	0.99049735	1545.19021	2382.47124	1633.3876	2440.35367	1613.60472	2470.49751	1525.40732	2412.61508
556.png	4	0.95283425	1435.83084	2313.44779	1530.30475	2374.08647	1512.43185	2401.9321	1417.95795	2341.29342
556.png	4	0.94760245	1638.79393	2450.45247	1739.64388	2517.05719	1722.45802	2543.07927	1621.60807	2476.47455
556.png	4	0.9302455	329.661959	1646.42487	401.450029	1715.45972	380.661039	1737.07781	308.872969	1668.04296
556.png	4	0.90371615	1329.70692	2364.64397	1411.11425	2424.10573	1396.43176	2444.2071	1315.02442	2384.74534
556.png	4	0.8669842	1001.53477	2026.50917	1124.87117	2096.87041	1111.90199	2119.60411	988.565598	2049.24287
556.png	4	0.8637537	65.6455314	1685.35683	133.614718	1752.6117	118.477424	1767.90977	50.5082378	1700.6549

베이스라인 제출: 제출

- 제출



베이스라인 제출: 제출

- 제출

주식회사 에스아이에이
SI ANALYTICS

Thank you for attention!

SI Analytics Co., Ltd. (Satrec Initiative Group)
441Expo-ro, Yuseong-gu, Daejeon, 34051, Korea

www.si-analytics.ai