




인공위성 영상에서 객체 탐지 하기

SIA - A IFFEL 해커톤 최종 발표



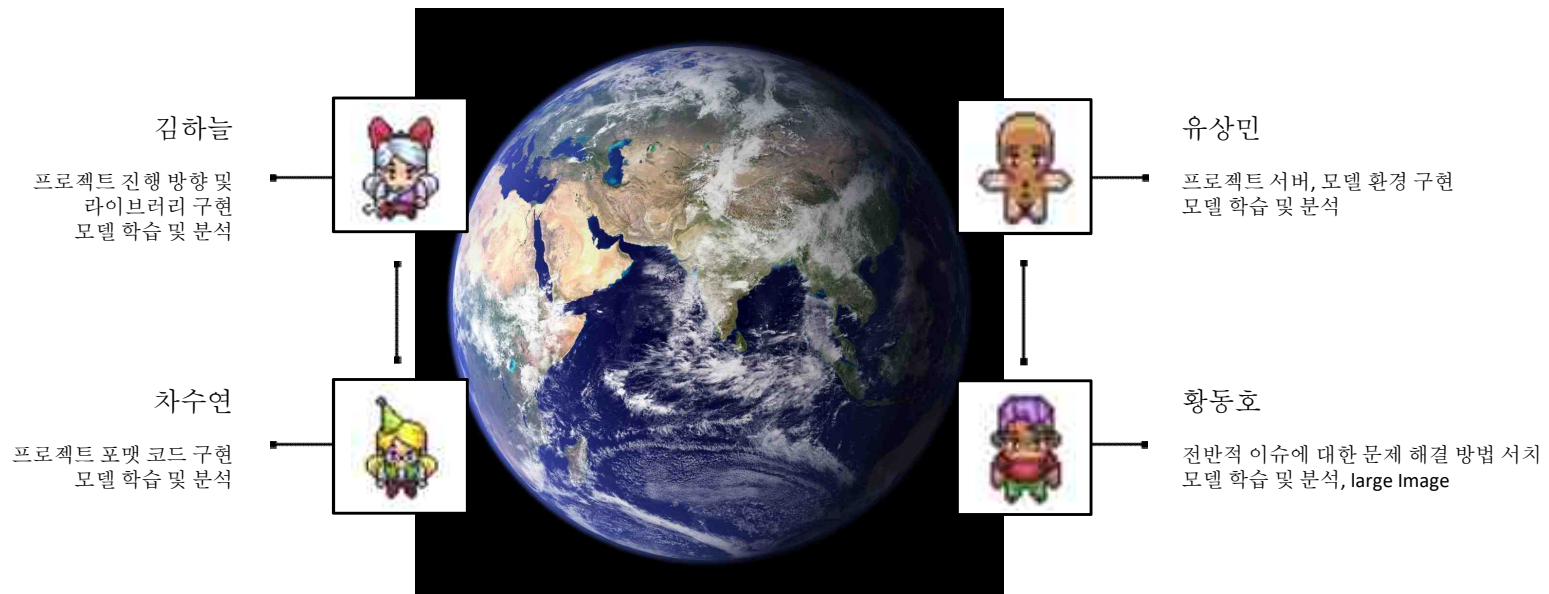
팀명: I Feel Earth

팀원: 김하늘, 유상민, 차수연, 황동호

2021.12.12

MEMBERS

I FEEL EARTH 팀 구성원입니다.



CONTENTS



01

프로젝트 배경

02

프로젝트 진행 순서

03

프로젝트 진행 과정 및 결과

04

프로젝트 회고

05

객체 탐지 기대 효과

06

Q&A

CHAPTER 01

프로젝트 배경

01

기업 목표

위성 영상에서 목표 객체
들을 탐지한다.

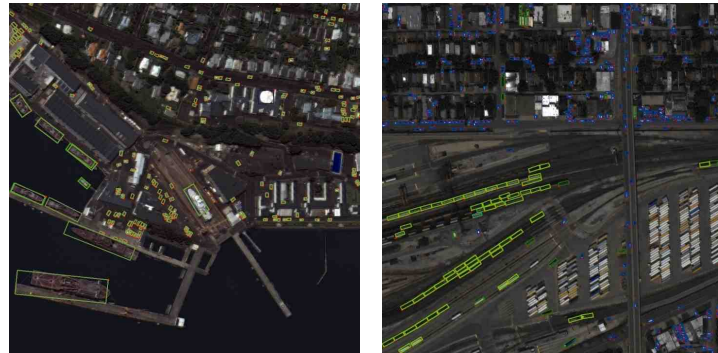
Object Detection (Rotated
Bounding Box)



02

팀 목표

위성 영상에서
Object Detection 수행
Rotated Bounding Box 출력



C1

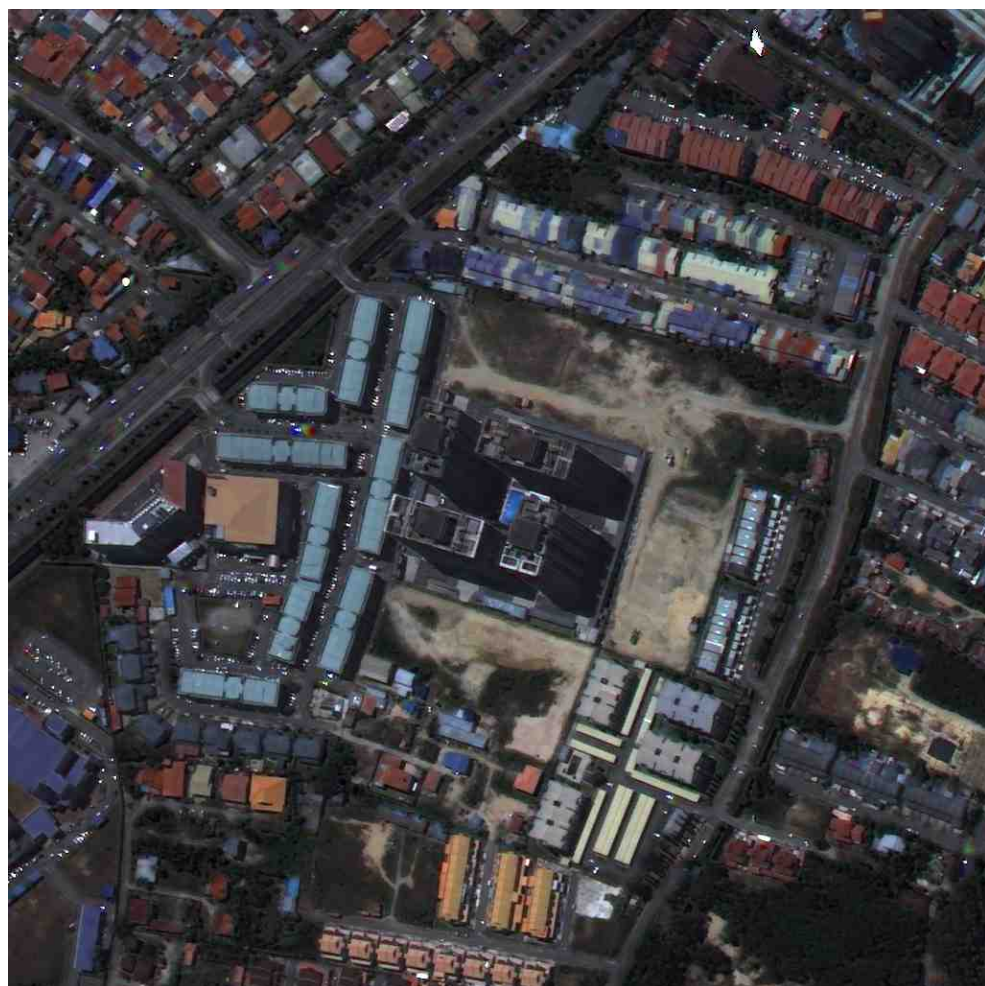
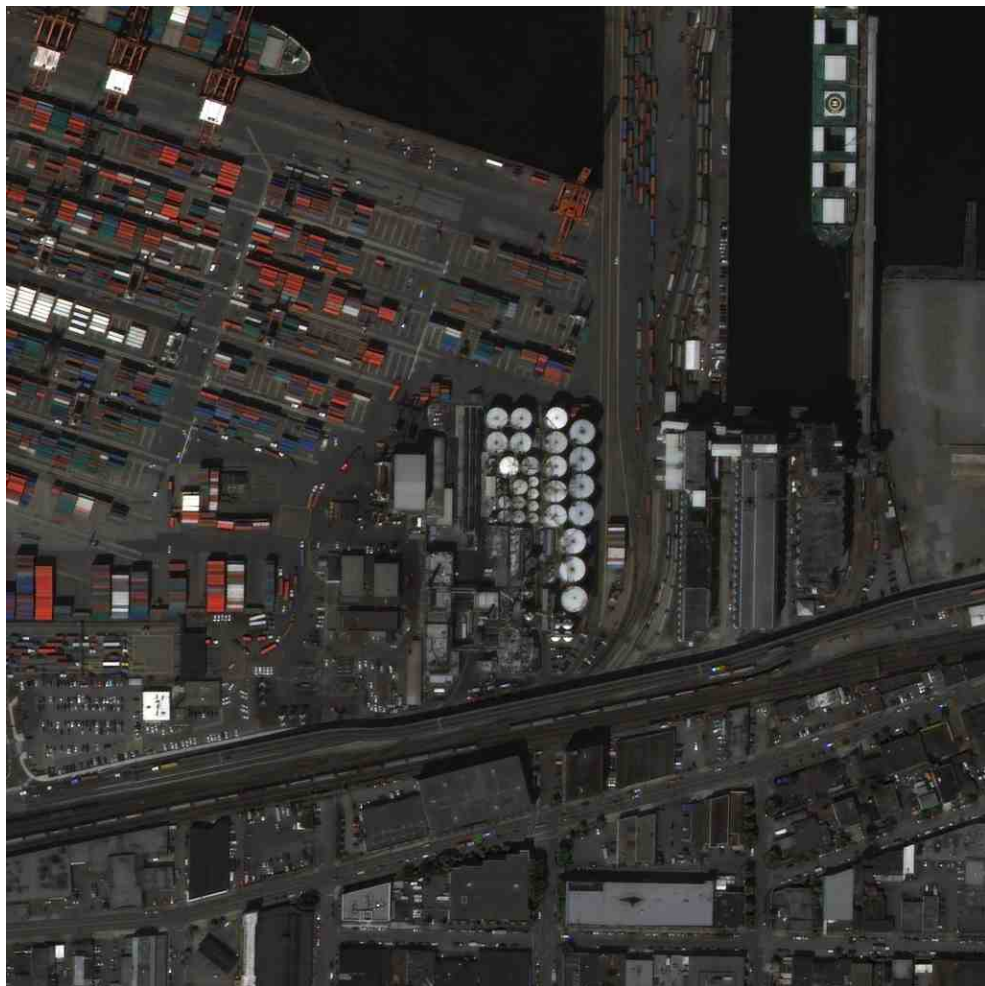
DATASET 소개

- 데이터셋 다운로드
 - AI 허브 (<https://aihub.or.kr/aidata/7982>)
- 데이터셋 (Image)
 - 아리랑 3A호(KOMPSAT-3A) 에서 취득한 위성영상 패치
 - Scene 형태의 영상을 AI를 위해서 1024 크기의 패치로 자른 형태
 - R,G,B 3채널로 생성된 영상 활용
 - GSD(Ground Sample Distance): 0.55m
 - 21종의 다양한 객체들 (소형차, 트럭, 기차, 운동경기장 등)
- 데이터 구성
 - RGB 형태 png 및 tif 파일
 - kml (위성 정보)
 - json (어노테이션 정보 - 객체 ID/이름, 객체 길이, 위경도 좌표 등)



DATASET 소개

01



CHAPTER 02

프로젝트 진행 순서

프로젝트 진행 순서

U2

Step1

탐색적 데이터 분석(EDA)
논문 탐색

~10/31

Step3

새로운 Rotated
모델 및 라이브러리

12/1~

Step5

큰 영상에서 객체 탐지

12/8~

Step2

베이스 모델 구현

Step4

모델 성능 평가 및 정리

CHAPTER 03

프로젝트 진행 과정 및 결과

STEP 01

탐색적 데이터 분석(EDA)
논문 탐색

JSON 파일

Step 01

Json 파일을 통해서 어떤 정보를 얻을 수 있을까?

Name	Value
type_name	"dam"
type_id	"12"
road_imcoords	"EMPTY"
object_imcoords	"577.5335936447841,644.9753585661617,426.2586173885984,412.9800659741642,497.070506704797,366.8064270382977,648.3454829609827,598.8017196302953"
object_angle	2.1486187195514876
ingest_time	"2020-11-04T06:53:36.815709Z"
image_id	"OBJ07450_PS3_K3A_NIA0834.png"
building_imcoords	"EMPTY"

Object 구성

Object Class: 21개



Small ship



Large ship



Civilian aircraft



Military aircraft



Small car



Bus



Truck



Train



Crane



Bridge



Oil tank



Dam



Indoor playground



Outdoor playground



Helipad



Roundabout



Helicopter



Individual container



Grouped container



Swimming pool



Etc

Step 01

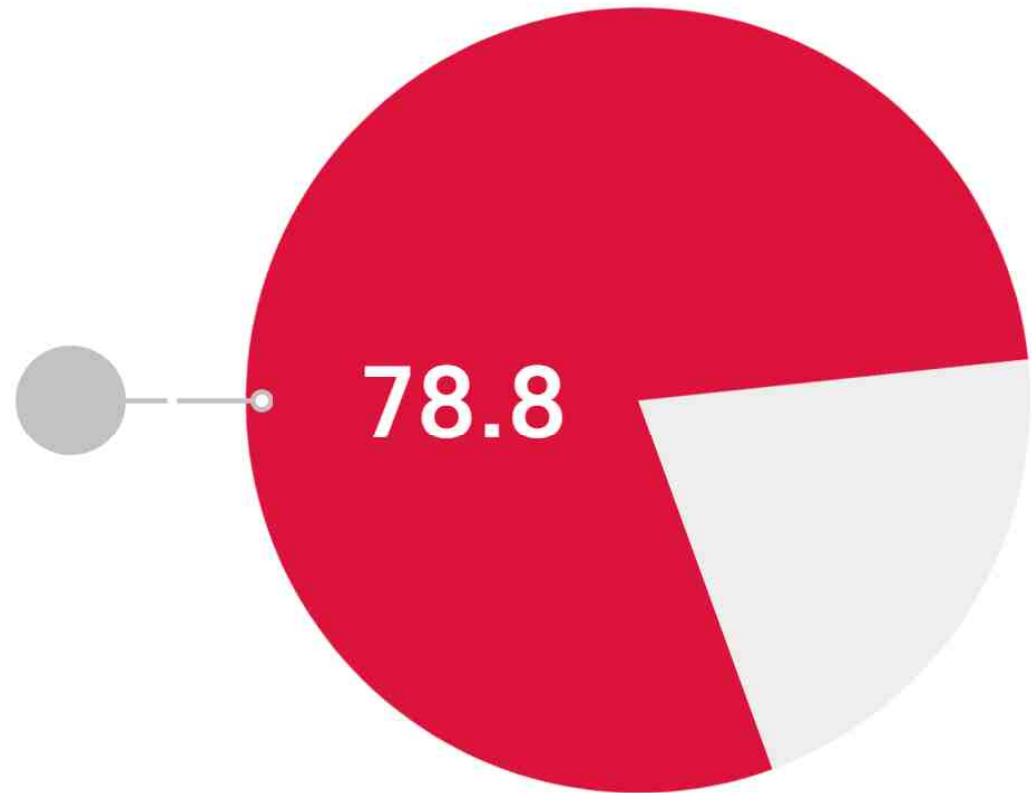
Object 구성

Step 01

Object 개수: 423,750개

Object 분포 비율

Small car
334,199개

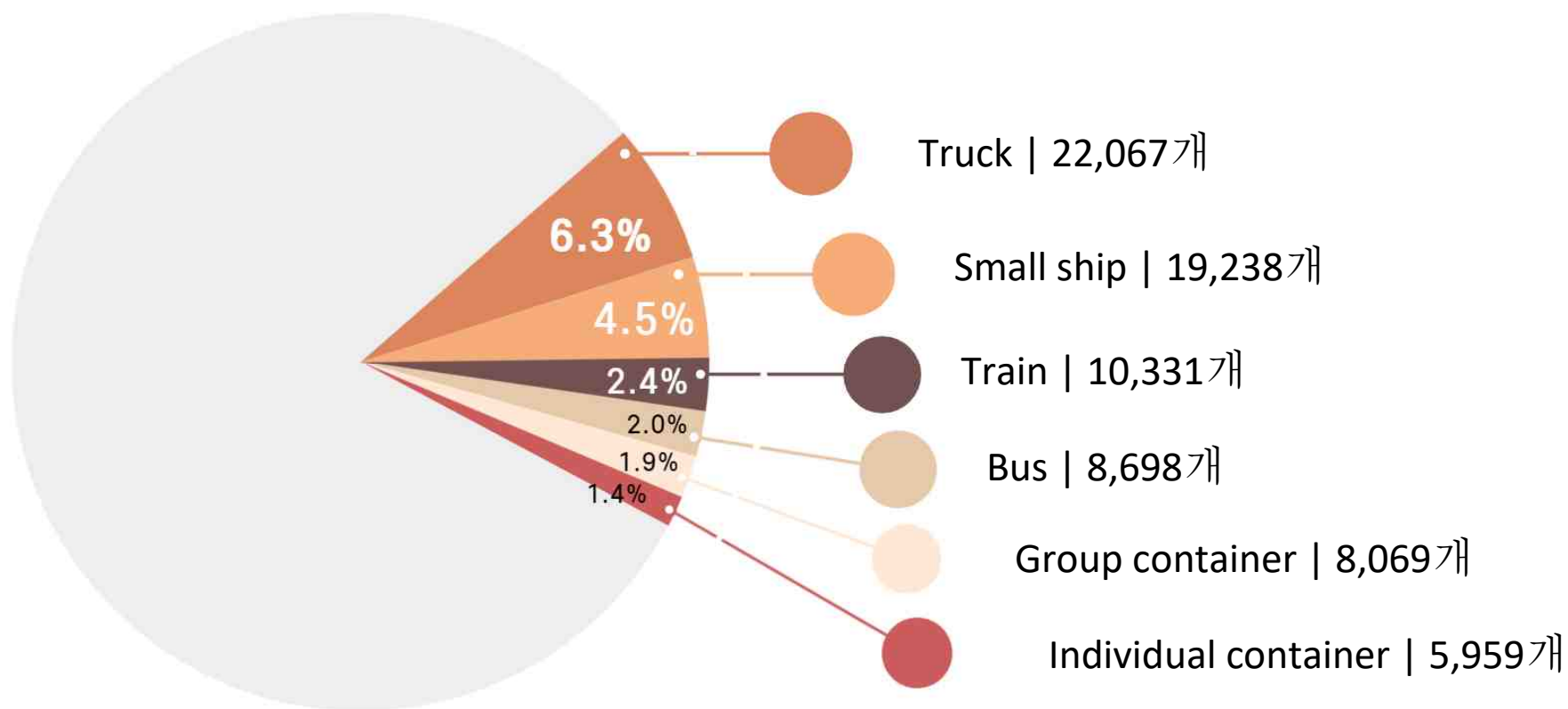


Object 구성

Step 01

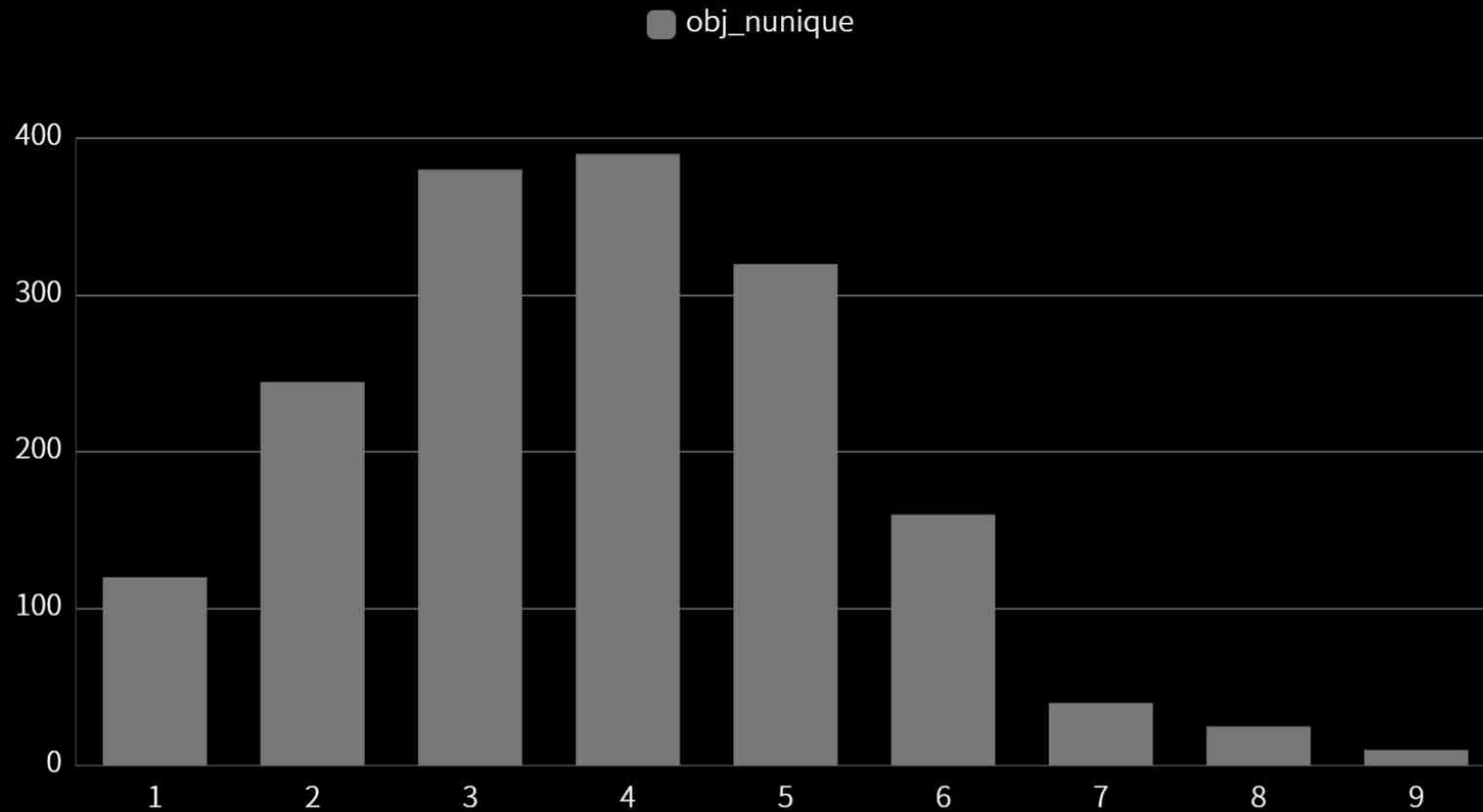
나머지 21%의 구성은?

7개 이외의 14개 객체의 비율은 1% 미만



Object 구성

각 영상당 포함하는 객체 개수



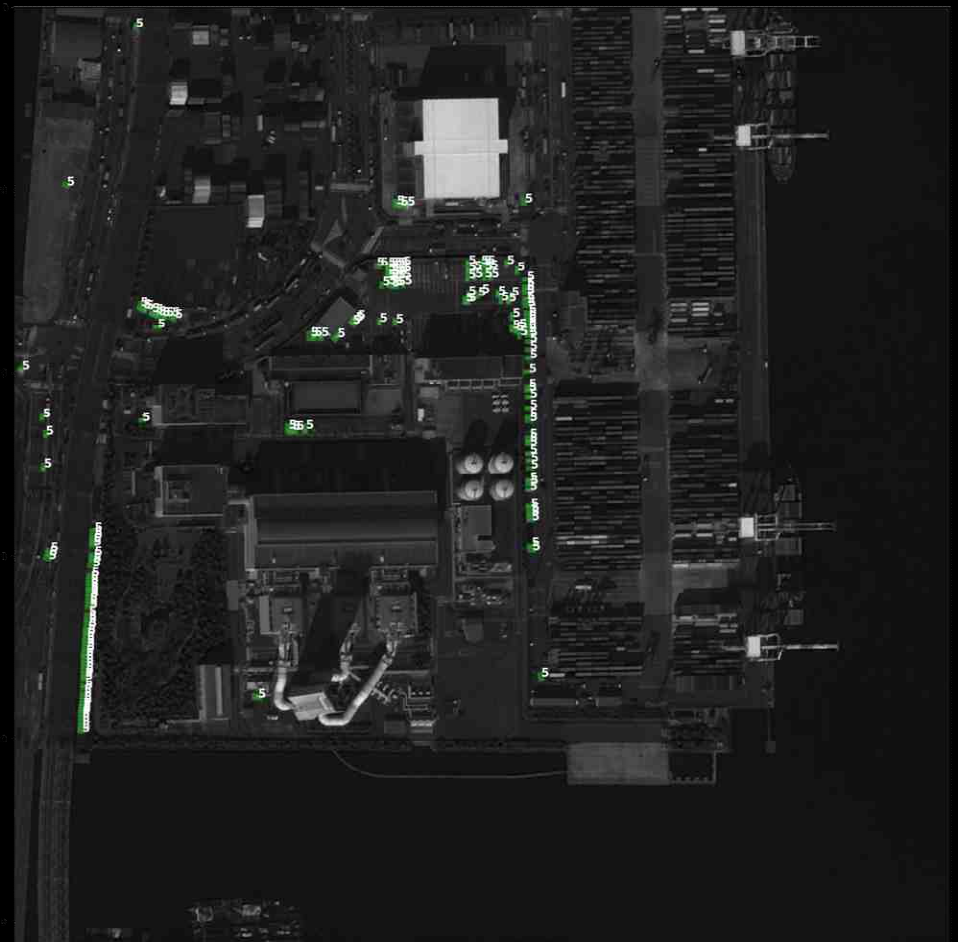
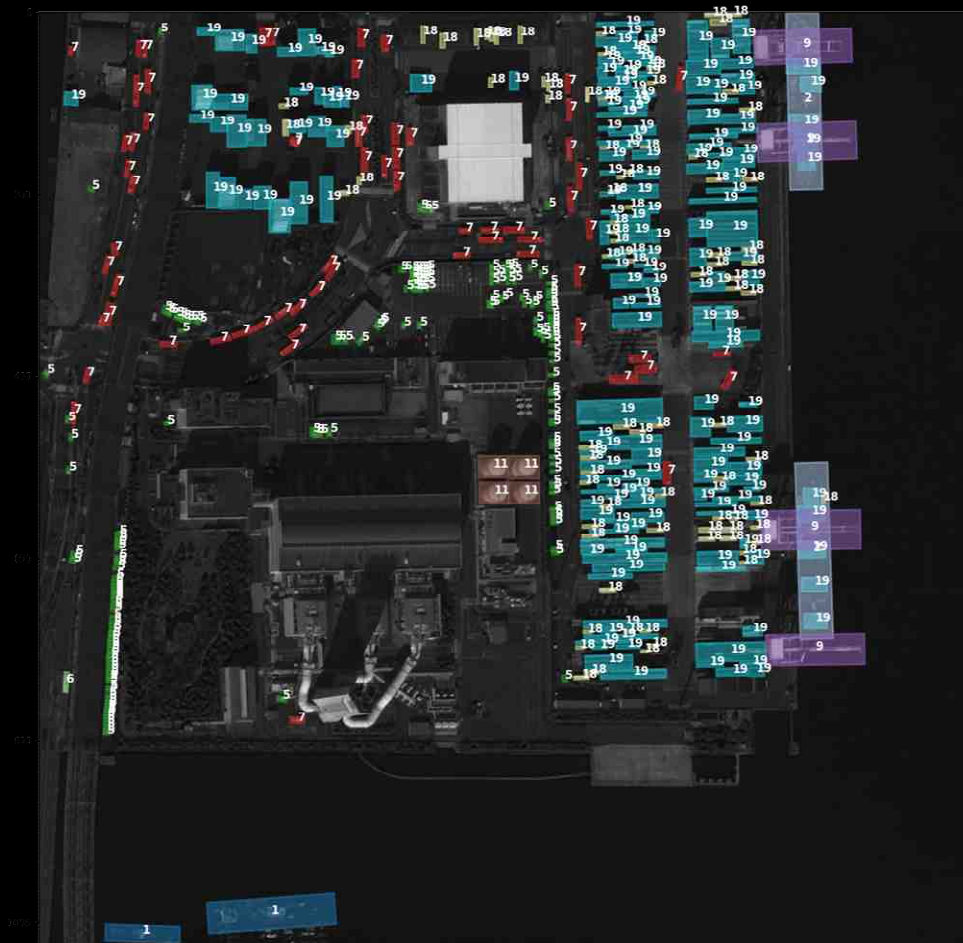
Object 구성

- 각 객체별 개수가 불균형하다.
→ 불균형 데이터를 학습해야하는 문제

Object	Count
total	423750
small car	334199
truck	27067
small ship	19238
train	10331
bus	8698
grouped container	8069
individual container	5959
swimming pool	2188
oil tank	1631
etc	1542
military aircraft	1298
civilian aircraft	875
large ship	546
crane	496
roundabout	371
helipad	309
outdoor playground	276
bridge	266
helicopter	184
dam	164
individual playground	43

Bounding Box 시각화

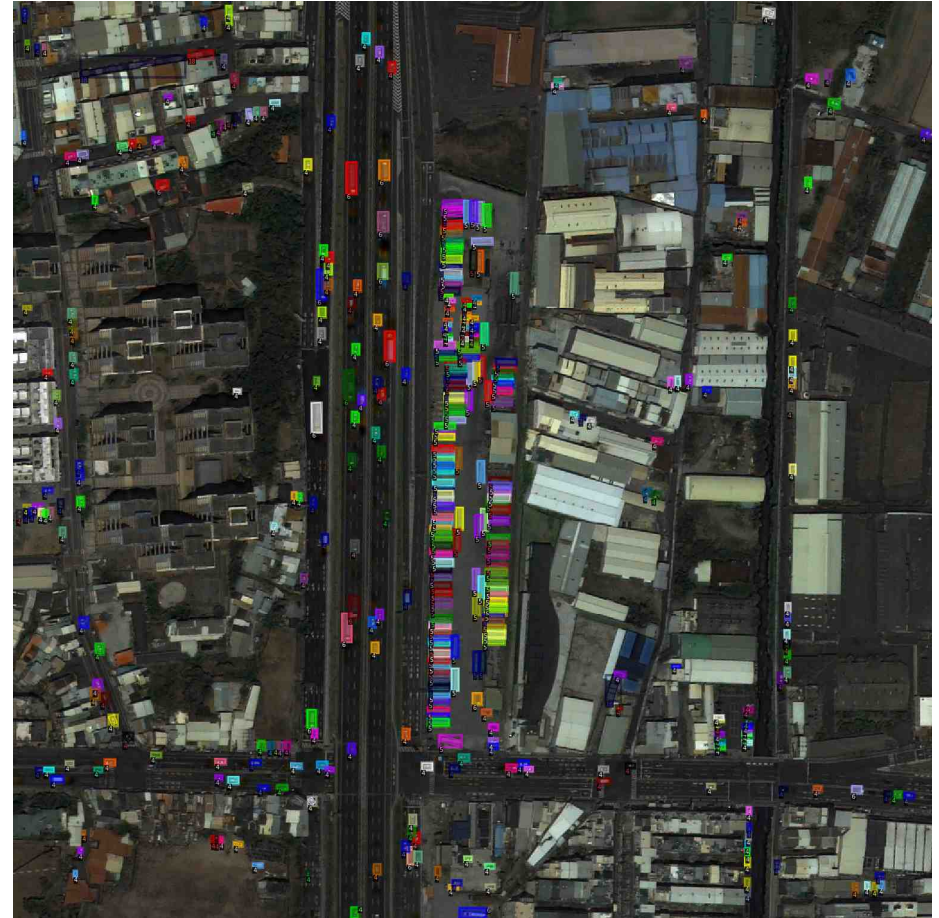
Step 01



인공위성 영상 DATASET 특징

Step 01

- 객체의 크기가 작다.
- 객체가 밀집되어 있다.



인공위성 영상 DATASET 특징

Step 01

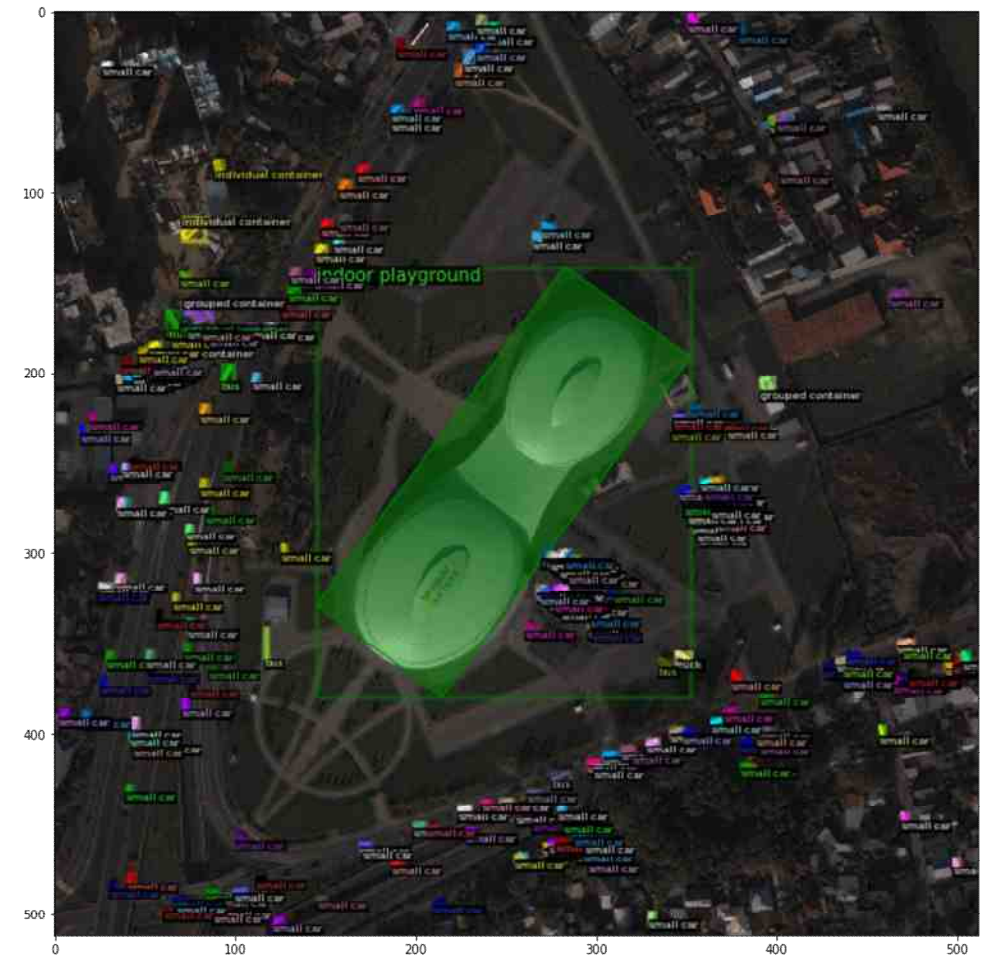
- 많은 객체들이 종횡비가 다르다.
 - 데이터 크기의 불균형
 - 종횡비가 큰 객체의 위치는 정확하게 찾아내기 어려움



인공위성 영상 DATASET 특징

Step 01

- 항공 뷰의 특성상,
객체가 회전된 경우가 많다.
→ 각도를 고려한 탐지가 필요하다.



문제 정의

Step 01

- Object Detection
 - Classification + Localization
 - (a) Horizontal Bounding Box(HBB)
 - (b) Oriented Bounding Box(OBB)
 - Multiple Object 검출하는 모델 구현
 - Rotated Bounding Box 구현



(a) HBB



(b) OBB

STEP 02

베이스 모델 구현

Simplified RBox CNN

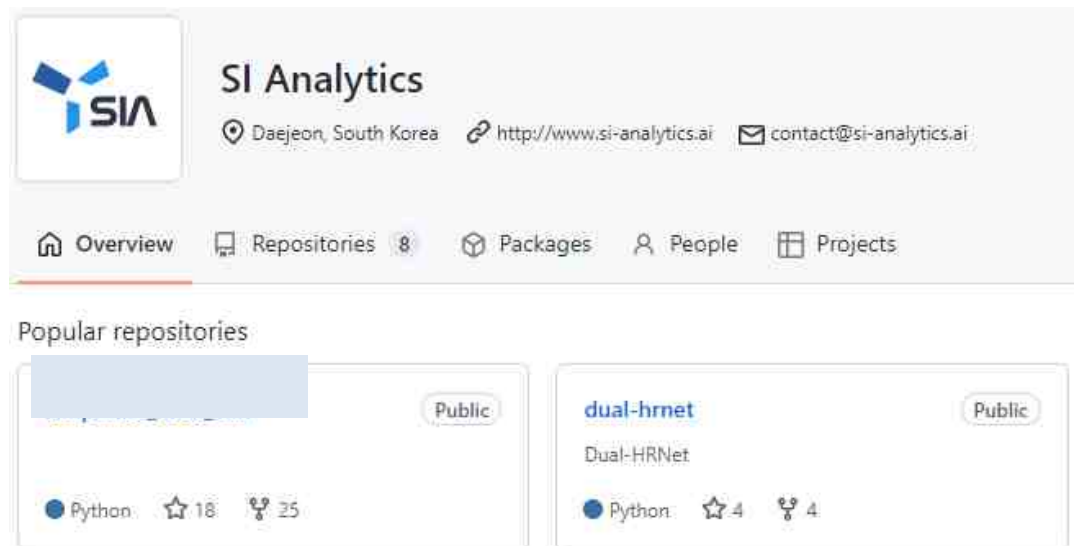
Step 02

RBox-CNN을 단순화한 코드

Faster R-CNN 모델 기반

- 논문: [RBox-CNN: rotated bounding box based CNN for ship detection in remote sensing image](#)

Tensorflow Object Detection API를 기반으로 만들어짐



The screenshot shows the GitHub profile of SI Analytics. The profile header includes the SI Analytics logo, the name "SI Analytics", and location "Daejeon, South Korea". Below the header are navigation tabs: Overview, Repositories (8), Packages, People, and Projects. The "Popular repositories" section displays two repositories: a redacted repository (Public) with 18 stars and 25 forks, and "dual-hrnet" (Public) with 4 stars and 4 forks. Both repositories are Python-based.

Repository Name	Stars	Forks	Language	Visibility
[Redacted]	18	25	Python	Public
dual-hrnet	4	4	Python	Public

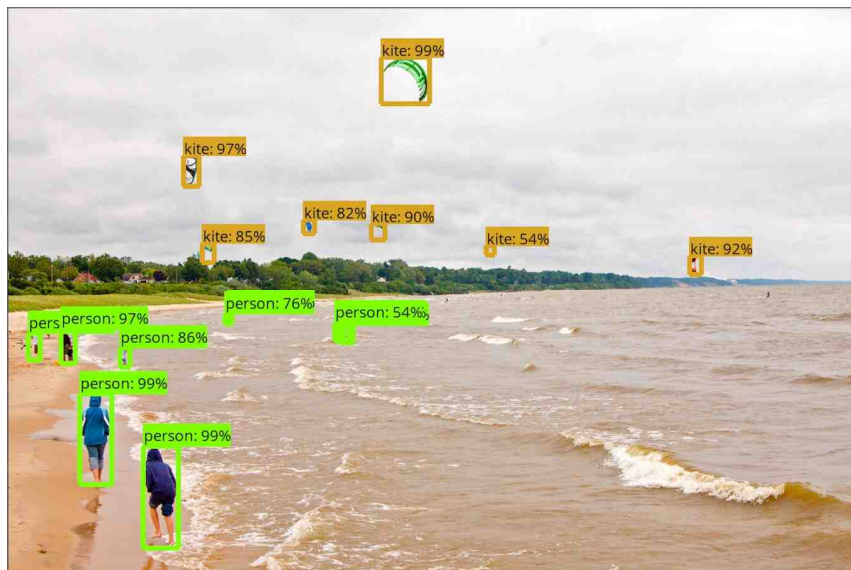
Tensorflow Object Detection API

Step 02

TensorFlow Object Detection API

TensorFlow 2.2 TensorFlow 1.15 Python 3.6

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.



TensorFlow를 기반으로 구축된

오픈 소스 프레임워크

이를 통해 객체 감지 모델을 구현할 수 있다.

시도한 것

Step 02

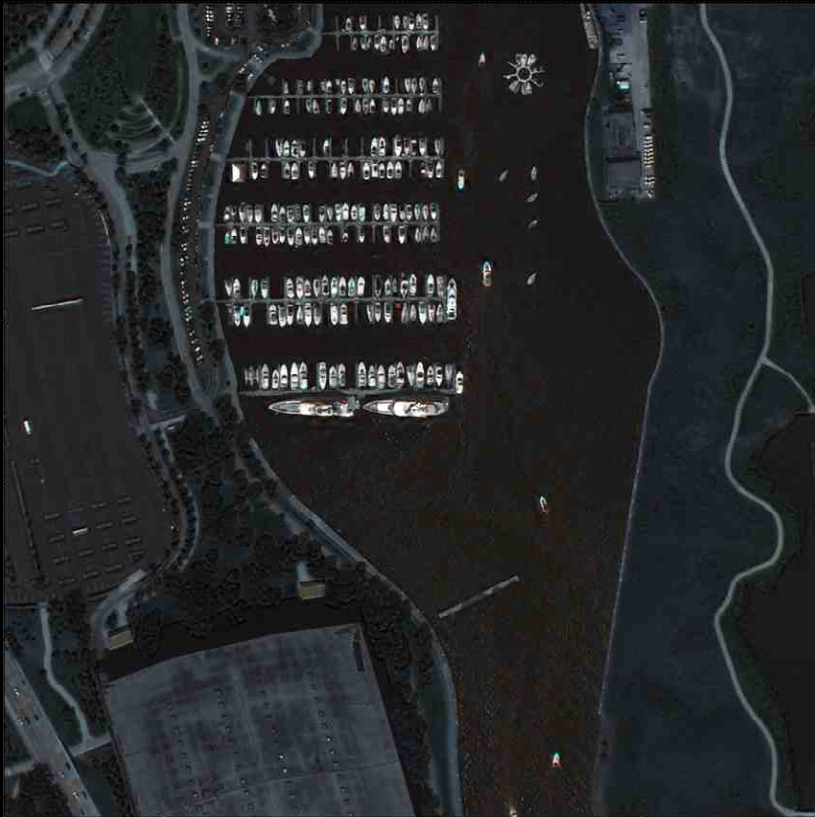
TFRecord File 생성(Tensorflow에서 학습시키기 위한 데이터 저장 형태) 후 학습 진행

모델 학습 및 평가 진행

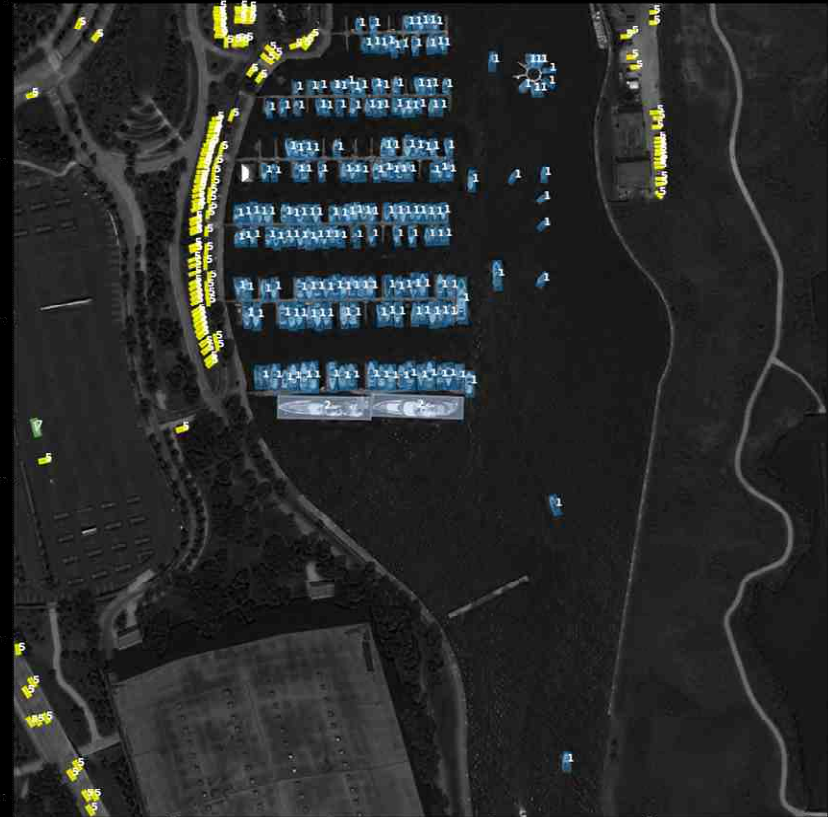
- 학습 속도와 성능 문제로 하이퍼 파라미터 튜닝 시도
 - Learning Rate, Batch Size 변경하여 학습해보고 각각 수치의 의미 추론
 - Patch Size 줄여서 학습
- Category 순서를 잘못 설정하는 실수
- Category 분류 학습
 - ✓ 'small car'만 뽑아서 학습
 - ✓ 'small car'만 제외하고 학습

결과와 문제점

Step 02



Real Image



Ground Truth

결과와 문제점

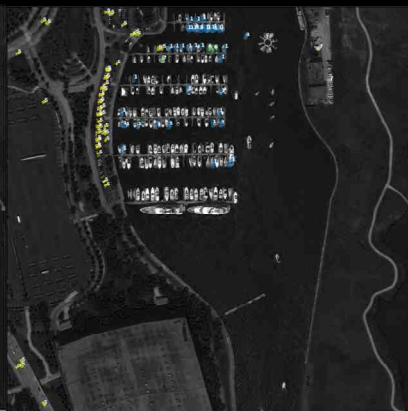
Step 02



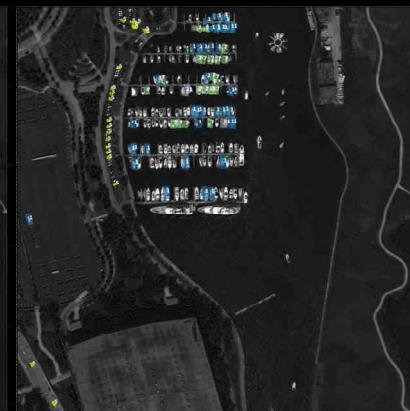
10,000번



30,000번



40,000번



50,000번



60,000번

- 학습량이 늘수록 검출한 객체수가 늘었지만 객체를 제대로 찾진 못했음
- IoU나 MaP은 좋은 수치는 아니지만 조금씩 좋아지고 있고, loss는 떨어지지 않고 있음
- GCP에서 학습 시키고 추가 모델을 선정하여 진행하려 했으나,
CUDA version, GPU 등의 설정 문제로 진행하지 못함

STEP 03

새로운 Rotated 모델 및 라이브러리

모델 및 라이브러리 변경

Step 03

약 3주 동안 진행하며 깨달은 것들

- 더 활성화 된 최신 라이브러리를 사용하는 것의 장점
 - 위성 영상 특징에 맞는 Backbone Network 선택지가 넓어짐
 - 전이 학습이 가능

학습이 제대로 이뤄지지 않은 이유 분석

1. Dataset 접근

- a. OverSampling & UnderSampling
- b. 비슷한 class 병합
- c. 특정 Class Resampling
- d. Data Augmentation

2. Weight Balancing

- a. Focal Loss

모델 및 라이브러리 변경

Step 03

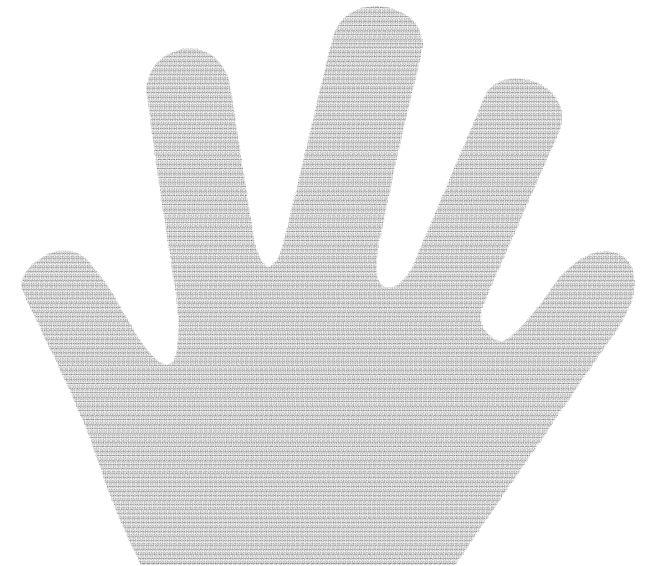
Bbox가 Rotated 되는 모델과 활용할 수 있는 라이브러리를 찾고 시도

1. 라이브러리 구현 시도

- Tensorflow Object Detection API
- MMDetection
- Detectron2




2. 모델 구현 시도

- Faster R-CNN
- Mask R-CNN
- YOLOv3
- RetinaNet(ResNet+FPN backbone)



라이브러리를 제대로 사용하기까지 어려웠던 점

라이브러리 구성과 기능을 이해하고 이용하는데까지...

	tools	Engine을 사용한 학습 파일
	configs	하이퍼 파라미터가 정의된 파일
	model_zoo	다양한 Baseline 결과를 가진 Pretrain Model 제공

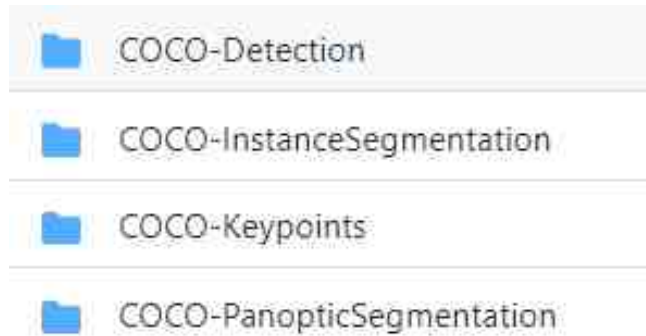
... 등과

처음 해보는 생소한 파이토치  PYTORCH

COCO format 변경

Backbone Network?

Step 03



후보

- Faster R-CNN
- RetinaNet
- RPN & Fast R-CNN

RetinaNet 으로 결정!

STEP 04

모델 성능 평가 및 정리

결과

Step 04

학습 후 확인한 것들

1. AP (Average Precision)

- AP, AP50, AP75, Aps, Apm, API
 - AP는 Average Precision을 의미
 - 숫자는 IoU 임계 값이 각각 50%, 75%인 AP를 의미
 - 알파벳은 객체 크기별(small, medium, large) AP를 의미
- Category별 AP

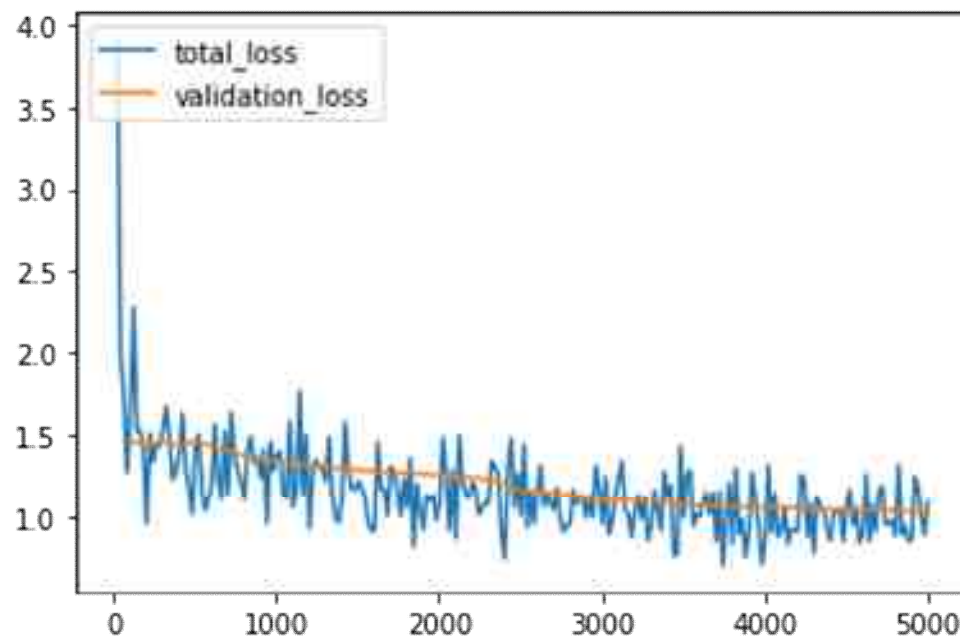
2. total_loss / validation loss

결과

Step 04

total_loss와 validation loss를 확인해가며 학습 진행

- Overfitting 확인 등



성능 평가

Step 04

Object Detection(물체 검출) 모델의 성능 평가를 Average Precision(AP)로 평가

Training Step	AP	AP50	AP75
10,000	8.104	16.200	8.642
25,000	11.280	22.66	10.910
50,000	12.321	23.320	11.770
150,000	13.140	23.475	13.262
250,000	13.964	24.217	13.123

성능 비교

Step 04

Detectron2에서 COCO model로 비교한 성능 지표

RetinaNet:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50	1x	0.205	0.041	4.1	37.4	190397773	model metrics
R50	3x	0.205	0.041	4.1	38.7	190397829	model metrics
R101	3x	0.291	0.054	5.2	40.4	190397697	model metrics

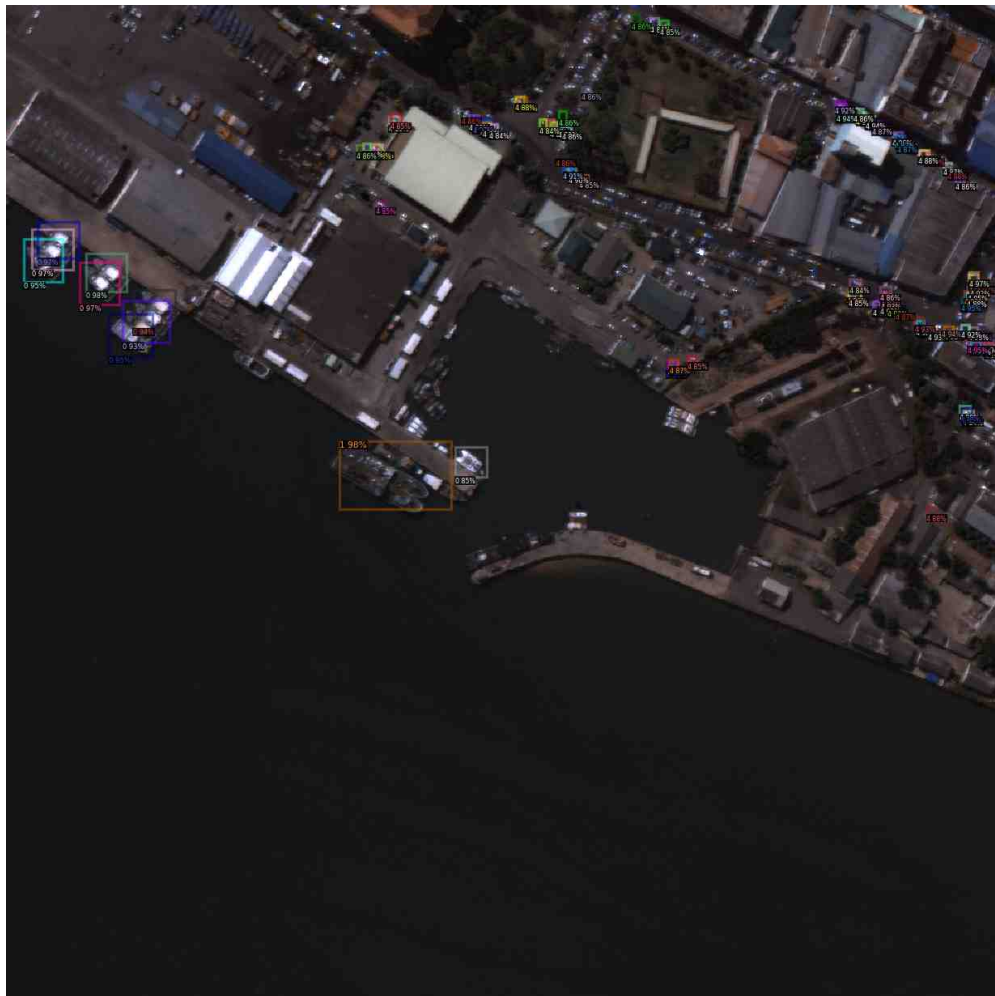
성능 비교

Step 04

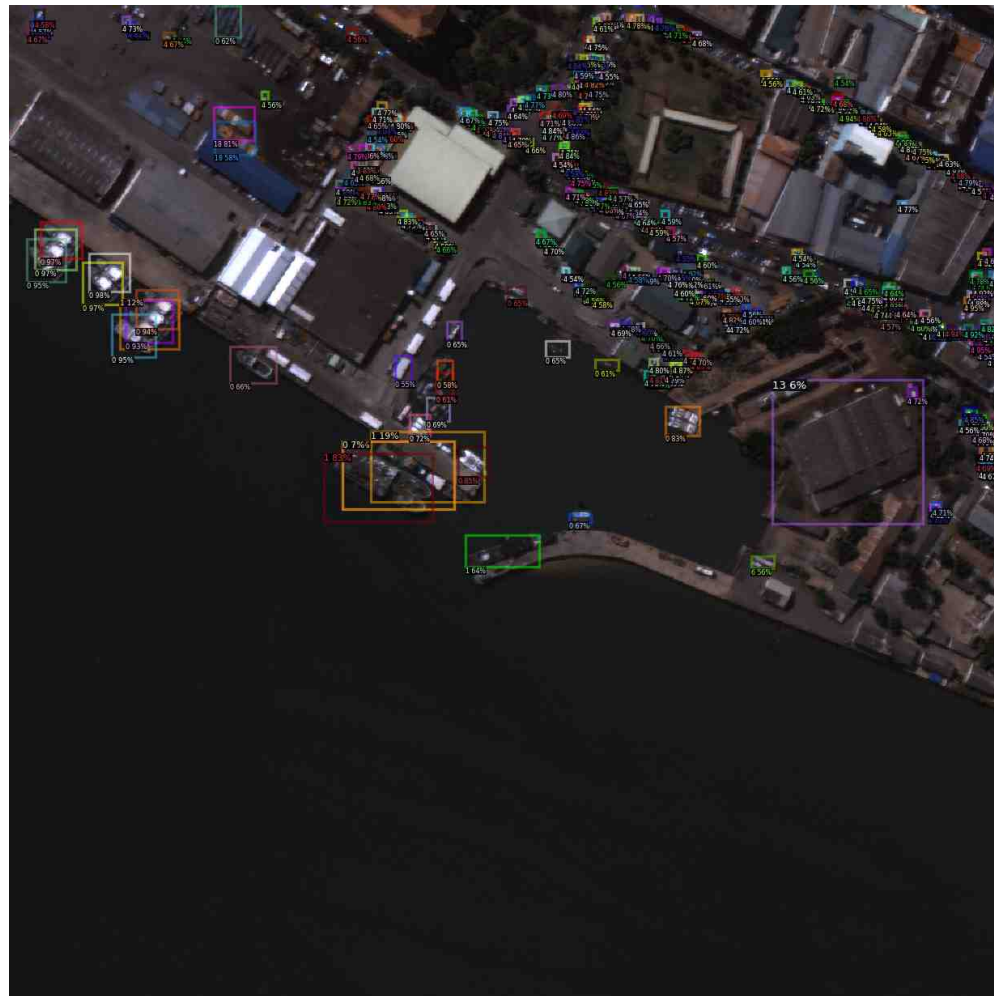
RetinaNet 논문에서 발표한 모델 성능

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

RESULT



Step 05



STEP 05

Real Image Detection

큰 영상에서 객체 탐지

Gdal로
Large image
불러오기

Step1

Step 2

Image Crop

Crop Image로
Prediction

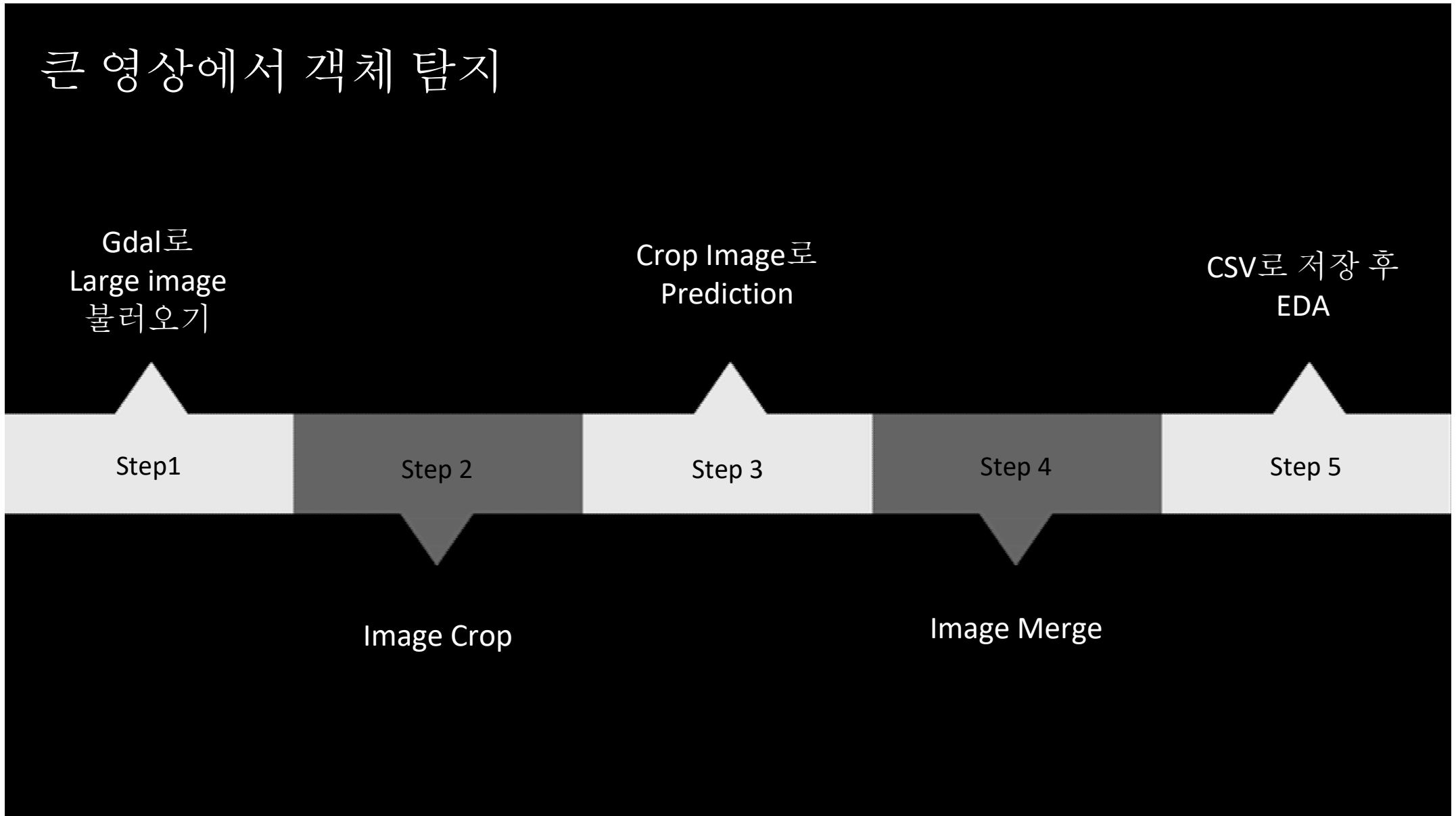
Step 3

Image Merge

Step 4

CSV로 저장 후
EDA

Step 5



Large Image

Step 05

<잘라낸 스펙>

파일 크기 : 802MB

pixel 사이즈: (w.h) = (12362,11344)

<원본 스펙>

파일크기 : 6.47GB

pixel 사이즈: (w,h) = (33480, 34600)



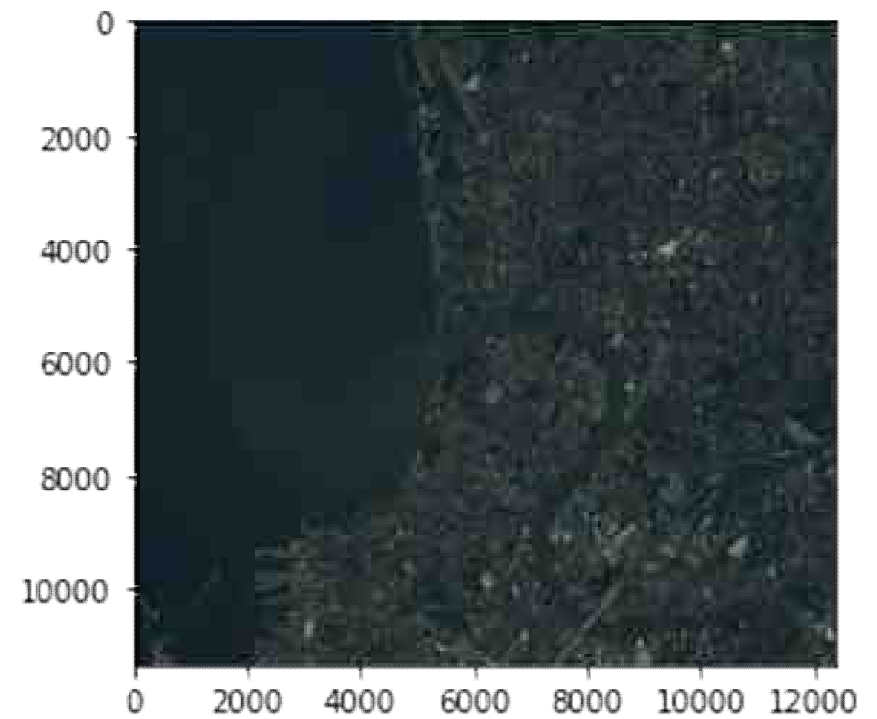
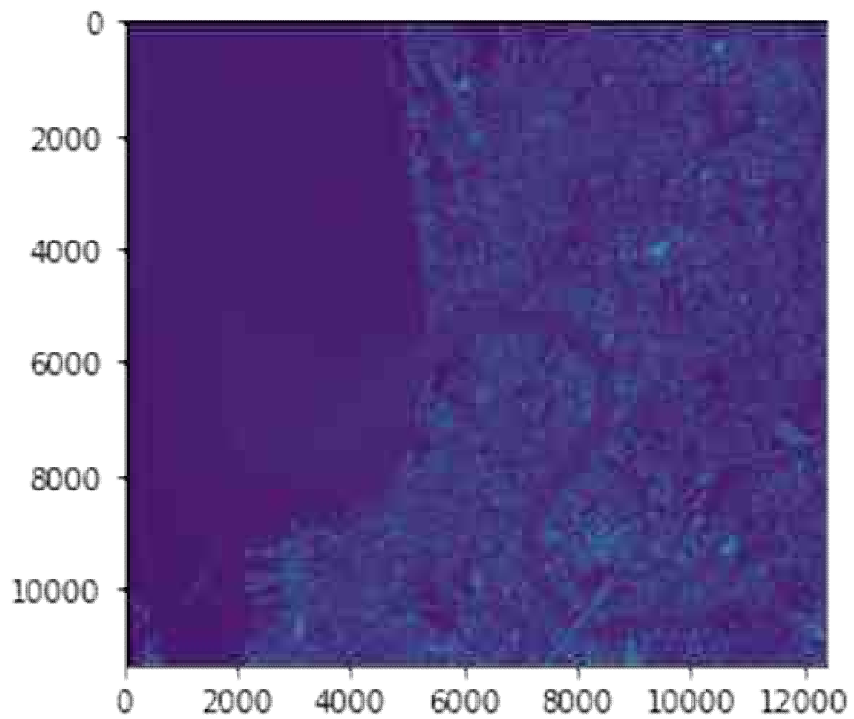
QGIS® 로 확대한 영상



Large Image 전처리

Step 05

- RasterBand로 이뤄진 데이터 → array 변환 → 하나의 이미지로 merge
- uint16 → uint8



Large Image를 그대로 사용하기에 학습 성능이 떨어짐

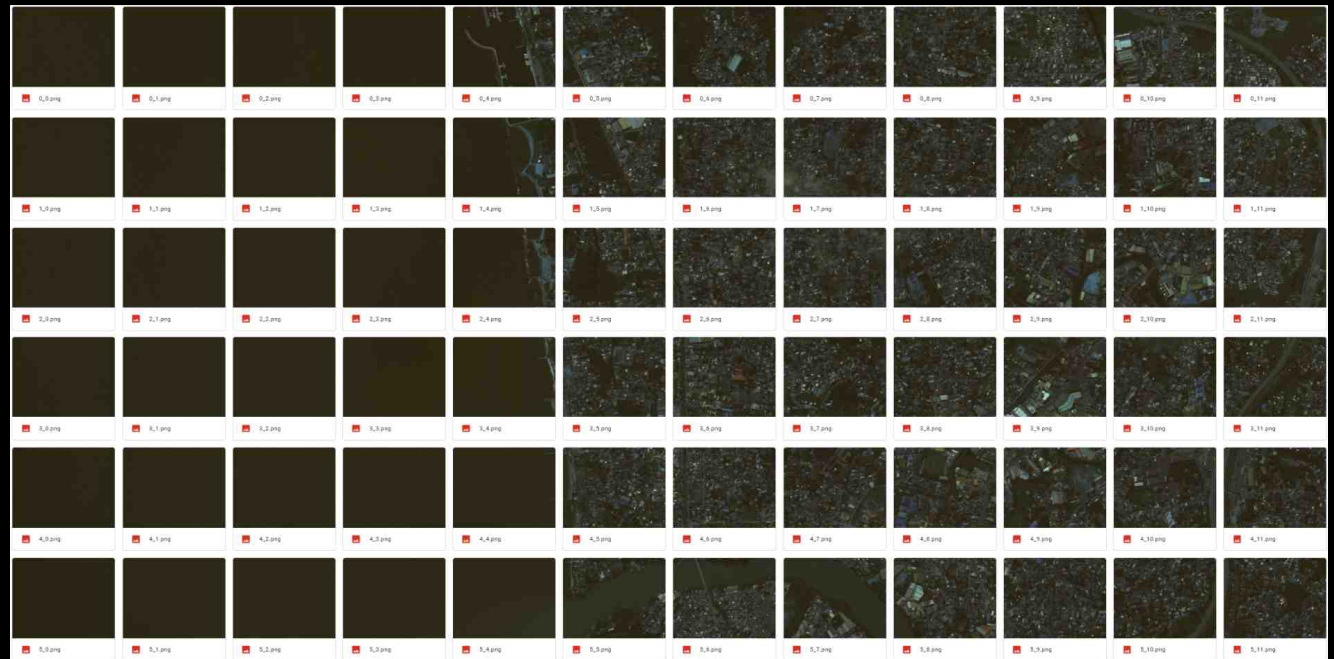
→ Crop을 통해 작은 Patch로 잘라 예측 시도

1. Patch Size를 1,024로 지정해 Large Image를 여러개의 Patch로 나누었음
2. Patch Size대로 이미지를 나눴을 때 남는 부분이 생기는 문제가 발생
3. Padding을 사용하거나 남는 부분은 제외해도 된다고 해서
이미지 상단과 좌측 (80, 74) 만큼을 제외하고 Image Crop 진행

PREDICT

Step 05

Crop한 이미지로 Predict 진행 후 저장



Predicted Result



CHAPTER 4

객체 탐지 기대 효과

객체 탐지 기대 효과

위성 영상과 인공지능 기술을 통해
관심 객체의 위치와 종류를 판단함으로써
다양한 목적으로 활용될 수 있다.

→ 변화 탐지 가능

예시

- 광역 교통량 모니터링
 - 경제 지표 분석
- 선박 탐지를 통한 해안 관리
 - 국방 안보의 목적



CHAPTER 05

프로젝트 회고

프로젝트 후 아쉬운 점

- Rotated Bounding Box로 시각화하지 못한 점
- Class Imbalance 문제를 해결하지 못해 개수가 적은 객체는 잘 예측하지 못했다는 점
- GCP 자원을 충분히 활용하지 못한 점
- WandB라는 MLops tool을 활용하지 못한 점

Large Image의 경우

- uint16 → uint8 로 변환하는 과정에서 정보손실로 인해 이미지가 어둡게 나오는 문제를 해결하지 못한 점
- Image Stretching 등의 기법을 통해 8bit 이미지로 변환할 때 원본 이미지 정보를 그대로 사용한다면 성능을 더 향상시킬 수 있을 것 같다.
- (더 추가할 예정)

어떻게 보완할 수 있을까?

- 객체 별로 다른 오버피팅의 정도를 해결하기 위해, 카테고리 별로 나눠서 분석 시도
- Data Imbalance 문제를 해결하여 예측 성능 향상하는 것
 - OverSampling & UnderSampling, 특정 Class Resampling, Data Augmentation 등
- 시간이 더 있었다면, 모델 튜닝 시도

새롭고 흥미로웠던 점

- 위성영상이라는 경험해보지 못한 Data
- 위성영상에서 Rbox가 왜 적합한지? 일반 Detection에서 쓰이는 Bbox와의 차이
- 정성적 평가뿐만 아니라 정량적으로도 평가를 했다는 점
- 군사적, 경제적, 사회적 목적으로 사용될 수 있는 기술을 경험했다는 점