

Computer Vision

중간고사 프로젝트 (버전 4)

Jaesung Lee

curseor@cau.ac.kr

2025. 04. 20.



중앙대학교
CHUNG-ANG UNIVERSITY



개요

- 포탈 공지문 : https://eclass3.cau.ac.kr/courses/120658/discussion_topics/394013
- 다음 슬라이드 설명에 리스트업된 10개 모델 중 최대 3개를 구현하여 데모 시작 전까지 깃허브(private)에 업로드 후 조교 collaborator 초대(14p 참조)
 - 초대용 메일 : tkdgur658@gmail.com
- 3개 모델은 다음 슬라이드에 설명되는 옴로(YOLOv8)의 오류 분석 결과를 토대로 선정 가능
- 데모 당일에는 구현된 모델을 하이퍼파라미터 튜닝 후 제출(모델의 메이저 구조는 변경 불가하나 앵커 박스, width, depth, 손실함수 가중치는 변경 가능)
- 제약 조건
 - 최종 제출시 모델 매개변수는 “4M 미만”이어야 함.
 - “훈련 최대 epoch은 20이하”여야 함.
 - “훈련 최대 배치사이즈는 16이하”여야 함.
 - “사전학습 모델 활용 불가”함.
 - “데이터 증강은 기존 데이터 사이즈의 두 배만 허용”됨.
- 평가지표는 예측 박스와 Ground Truth (GT) 박스 간의 IoU이며 이는 실험 코드에서 제공됨.
- 데모 당일 최종 제출물은 18p 참조.
- 데이터 정보
 - 테스트 : 항공기 감지
 - 이미지 크기 : 640 × 640 (RGB 채널)
 - 이미지 수 : train : val : test = 453 : 151 : 152

논문 리스트

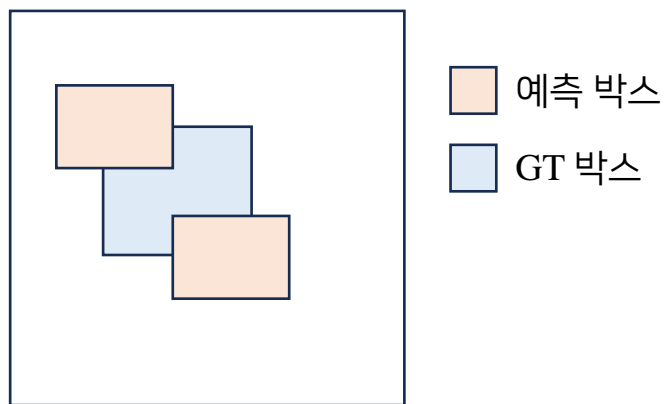
➤ 다음은 프로젝트 데모시 구현 수행 가능한 모델 10건임.



no.	Model name	Title	Year	Venue	# Parameters (M)	Input size	code
1	YOLOX-nano	Yolox: Exceeding yolo series in 2021	2022	ICSP	0.9	640×640	https://github.com/Megvii-BaseDetection/YOLOX
2	*YOLOv8-nano	A review on yolov8 and its advancements	2023	ICDICI	3.2	640×640	https://github.com/ultralytics/ultralytics
3	YOGA-n	YOGA: Deep object detection in the wild with lightweight feature learning and multiscale attention	2023	Pattern Recognition	1.9	640×640	https://github.com/LabSAINT/YOGA
4	Hyper-YOLO-T	Hyper-YOLO: When Visual Object Detection Meets Hypergraph Computation	2025	IEEE Transactions on Pattern Analysis and Machine Intelligence	3.1	640×640	https://github.com/imoonLab/Hyper-YOLO
5	YOLOv10-nano	Yolov10: Real-time end-to-end object detection	2024	NeurIPS	2.3	640×640	https://github.com/THU-MIG/yolov10
6	YOLOv11-nano	Yolov11: An overview of the key architectural enhancements	2024	Arxiv	2.6	640×640	https://github.com/ultralytics/ultralytics
7	LightweightOB	A lightweight remote sensing aircraft object detection network based on improved yolov5n	2024	Remote Sensing	0.4	640×640	https://github.com/wanxxiax/lightweightOB_2nd
8	YOLOv9-tiny	YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information	2024	ECCV	2.0	640×640	https://github.com/WongKinYiu/yolov9
9	FLDet-N	FLDet: Faster and Lighter Aerial Object Detector	2024	IEEE Transactions on Circuits and Systems for Video Technology	1.2	640×640	https://github.com/ws-y-yjs/FLDet/tree/main
10	YOLOv12-nano	YOLOv12: Attention-Centric Real-Time Object Detectors	2025	Arxiv	2.6	640×640	https://github.com/sun-smarterjie/yolov12

*baseline인 YOLOv8의 경우 공식 학술 논문이 출판되지 않았으므로 review 논문을 제시함. 아래 링크에서 해당 모델의 공식문서를 확인 가능함.
<https://docs.ultralytics.com/ko/models/yolov8/>

베이스라인 실험 결과 보고

- 실제 데모에 사용될 평가용 데이터셋에 대한 YOLOv8-nano, YOLOv6-nano, YOLOv5-nano의 10회 반복 실험 결과
 - 평가 지표는 Intersection over Union (IoU), Dice, Precision, Recall로 각 이미지 내 전체 예측 박스 영역과 전체 GT 박스 영역 간의 비교를 통해 계산됨



- P 는 예측 박스( 영역)의 픽셀 집합,
 G 는 GT 박스( 영역)의 픽셀 집합,
 $|\cdot|$ 는 집합의 크기(면적 또는 픽셀 수)를 의미할 때,

$$\begin{aligned} \text{IoU}(P, G) &= \frac{|P \cap G|}{|P \cup G|} = \frac{|P \cap G|}{|P| + |G| - |P \cap G|} & \text{Precision} &= \frac{\text{True Positives}}{\text{Predicted Positives}} = \frac{|P \cap G|}{|P|} \\ \text{Dice}(P, G) &= \frac{2|P \cap G|}{|P| + |G|} & \text{Recall} &= \frac{\text{True Positives}}{\text{Actual Positives}} = \frac{|P \cap G|}{|G|} \end{aligned}$$

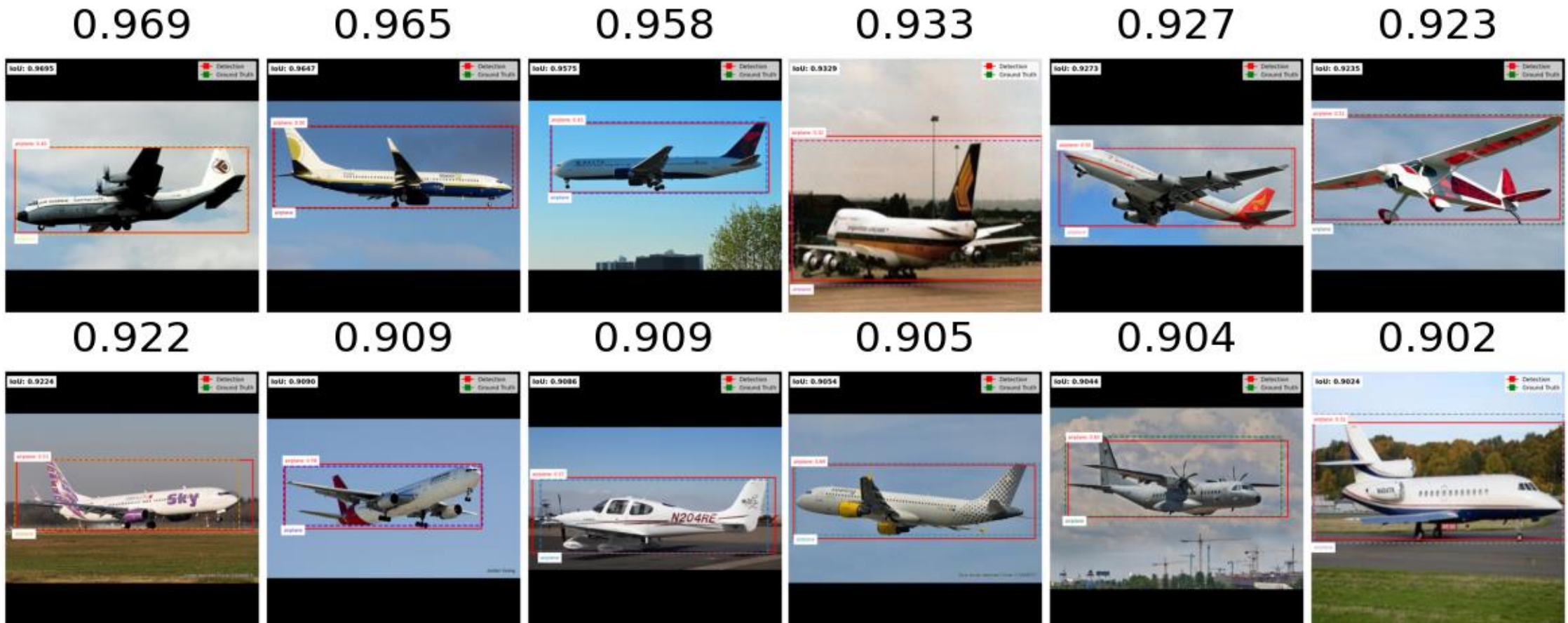
- 평가 결과는 다음과 같음. ▼1,2개는 각각 유의수준 0.10, 0.05에서 YOLOv8-nano와의 paired t-test 통과함을 나타냄.

Model	IoU	Dice	Precision	Recall
YOLOv8-nano	0.493 ± 0.037	0.562 ± 0.041	0.553 ± 0.045	0.607 ± 0.046
YOLOv6-nano	0.460 ± 0.051 ▼	0.526 ± 0.062 ▼	0.512 ± 0.055 ▼ ▼	0.586 ± 0.095
YOLOv5-nano	0.488 ± 0.036	0.563 ± 0.039	0.549 ± 0.040	0.622 ± 0.047

올로(YOLOv8)의 오류 분석 결과

➤ 성능 높은 케이스에 대한 샘플 특성 보고

- 몇몇 특성을 가지는 일부 데이터에 대해서 높은 IoU값을 보임.
- 예를 들어 흰색 항공기, 항공기 옆모습, 파란색-흰색 하늘 배경 등.

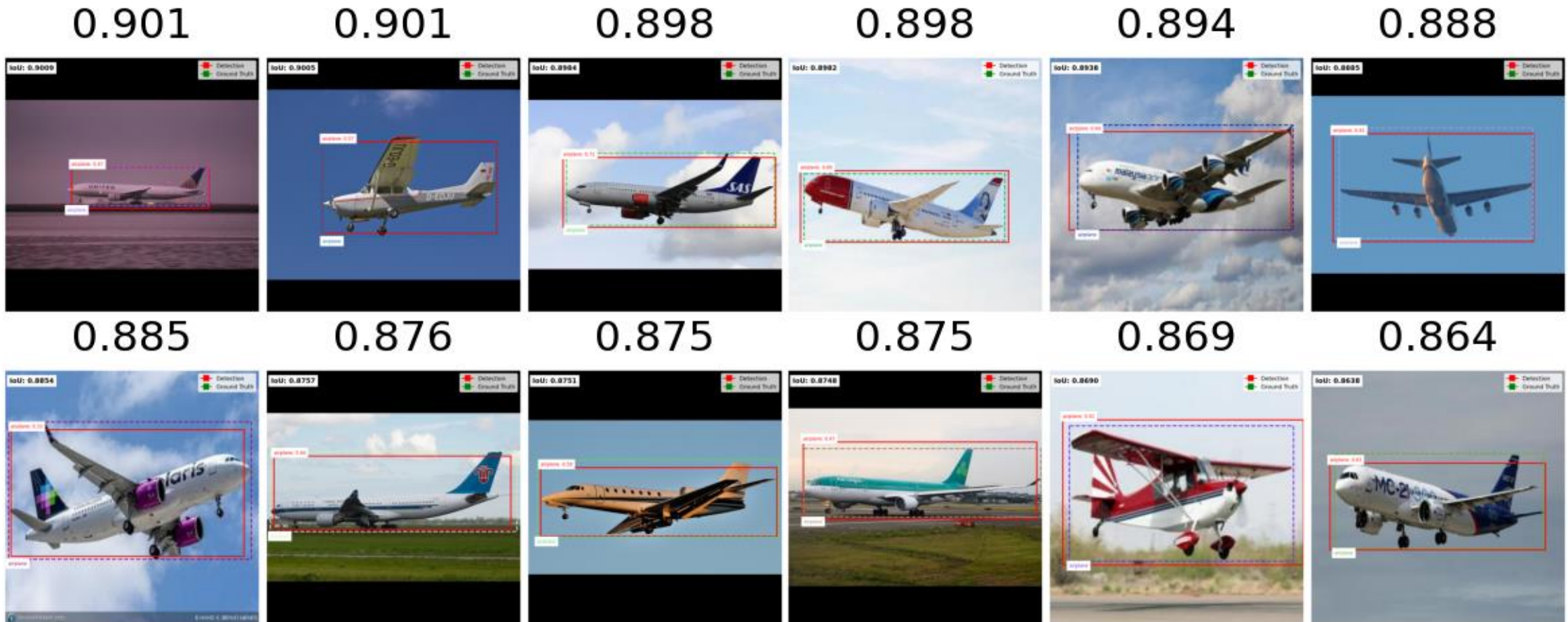


Competition 데이터 1번 시드에 대한 Top 1-12 시각화. 이미지 위 숫자는 전체 예측 지역과 GT 박스 간의 IoU를 나타냄.

올로(YOLOv8)의 오류 분석 결과

➤ 성능 높은 케이스에 대한 샘플 특성 보고

- 몇몇 특성을 가지는 일부 데이터에 대해서 높은 IoU값을 보임.
- 예를 들어 흰색 항공기, 항공기 옆모습, 파란색-흰색 하늘 배경 등.

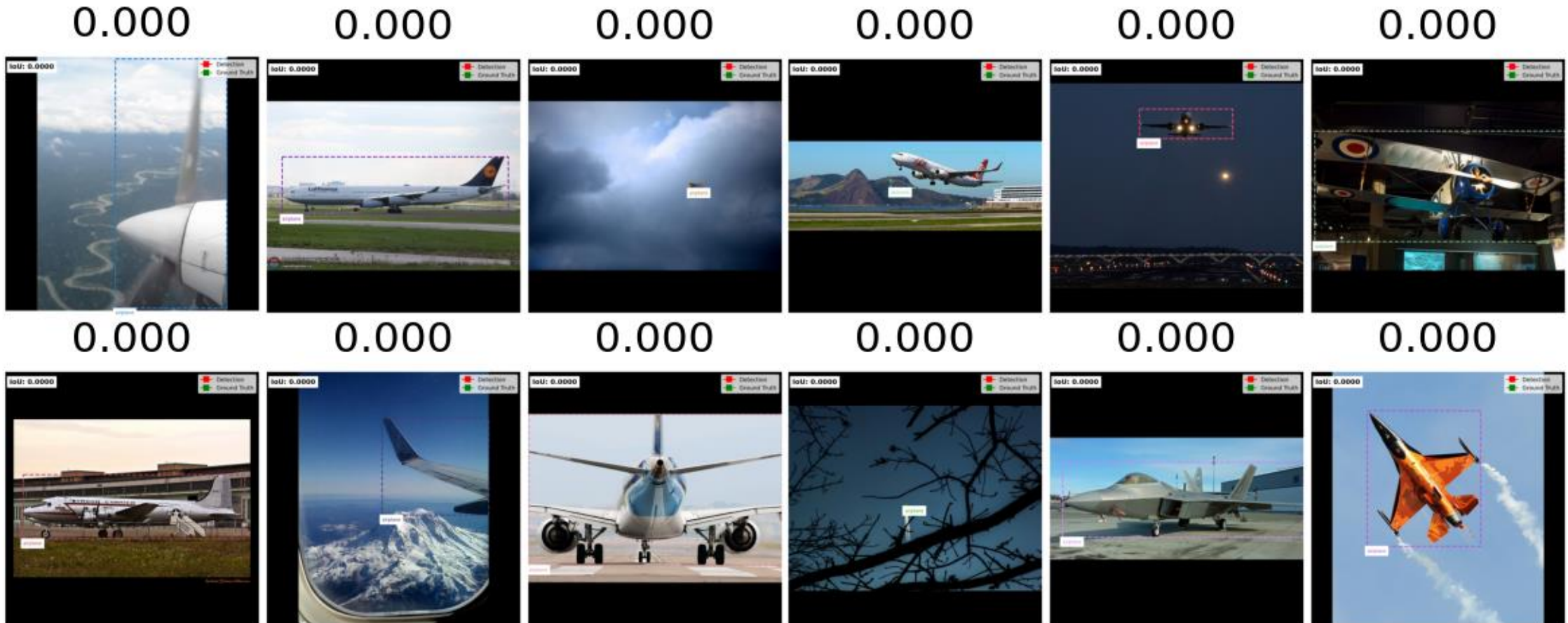


Competition 데이터 1번 시드에 대한 Top 13-24 시각화. 이미지 위 숫자는 전체 예측 지역과 GT 박스 간의 IoU를 나타냄.

올로(YOLOv8)의 오류 분석 결과

➤ 성능 낮은 케이스에 대한 샘플 특성 보고

- 반대로 그러한 특성을 갖지 않는 형태에 대해서 감지율이 매우 떨어짐.
- 예를 들어 항공기 앞뒷모습만 노출, 일부분만 노출, 다른 색상의 항공기, 어두운 배경, 작은 스케일의 항공기 등.

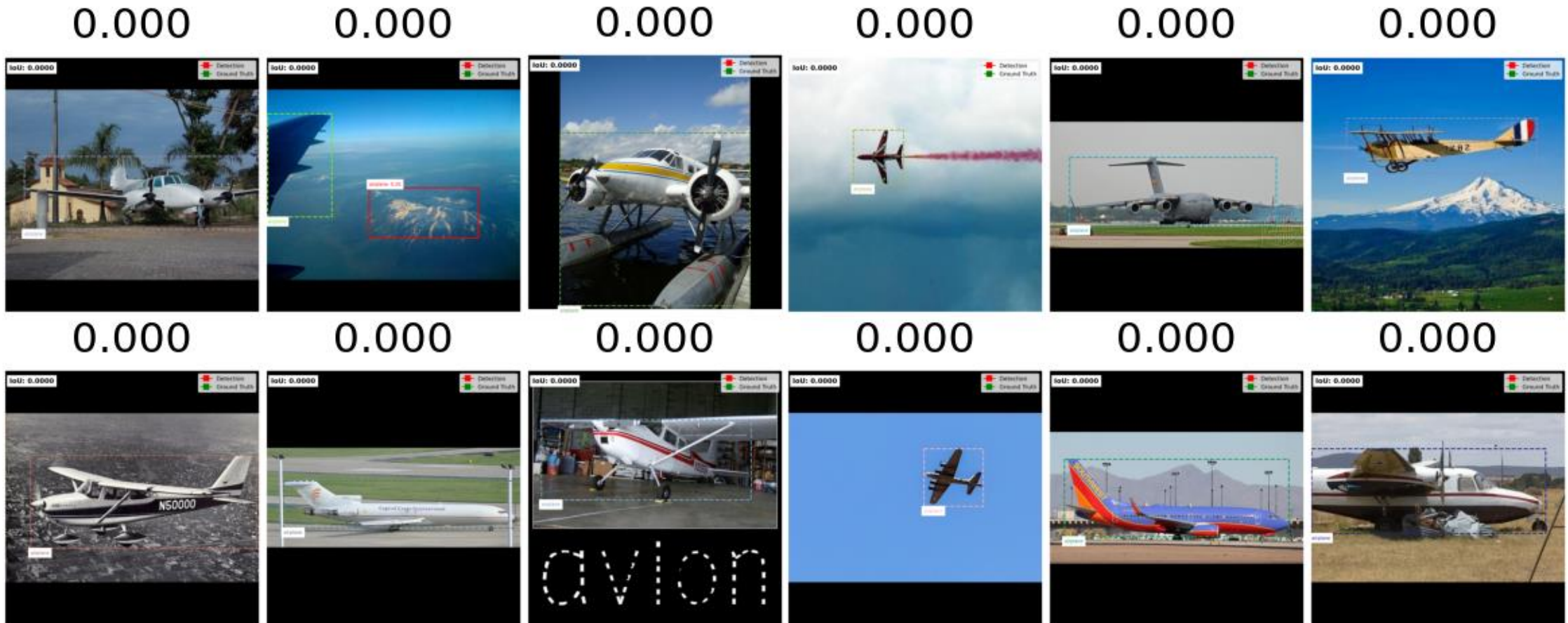


Competition 데이터 1번 시드에 대한 Worst 1-12 시각화. 이미지 위 숫자는 전체 예측 지역과 GT 박스 간의 IoU를 나타냄.

올로(YOLOv8)의 오류 분석 결과

➤ 성능 낮은 케이스에 대한 샘플 특성 보고

- 반대로 그러한 특성을 갖지 않는 형태에 대해서 감지율이 매우 떨어짐.
- 예를 들어 항공기 앞뒷모습만 노출, 일부분만 노출, 다른 색상의 항공기, 어두운 배경, 작은 스케일의 항공기 등.



Competition 데이터 1번 시드에 대한 Worst 13-24 시각화. 이미지 위 숫자는 전체 예측 지역과 GT 박스 간의 IoU를 나타냄.

오류 분석 정리 및 향후 해결방안 제시

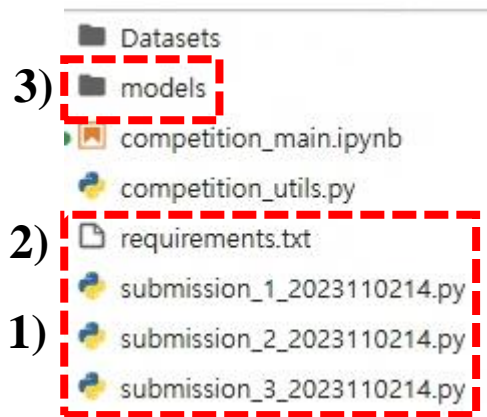
- 모델 예측이 흰색 항공기, 항공기 옆모습, 파란색-흰색 하늘 배경 등 일부 제한된 특징에 의존적으로 보임.
- 이로 인해 복잡한 배경, 다른 색상의 항공기, 확대 혹은 앞모습 등으로 나타난 이미지에 대해서는 인식률이 떨어짐.
- 간단한 향후 접근 방향 제시
 - 다양한 색상의 객체가 테스트셋에 등장할 수 있음을 가정하고 데이터 증강 기법을 적용
 - 드롭아웃과 같은 방법을 통해 특정 특징에 대한 의존성을 줄이며 어텐션 모듈 고안을 통해 복잡한 배경보다는 객체 학습에 집중할 수 있도록 설계
 - 확대 혹은 작은 스케일의 항공기 감지를 위해 멀티스케일 학습 방법을 포함

실험 코드 설명

- 배포되는 CV_midterm_competition_code_v2는 다음과 같은 파일을 포함하고 있음.

(깃허브 업로드를 통한) 의무 제출 파일

- 1) submission_1_{학번}.py, submission_2_{학번}.py, submission_3_{학번}.py (제출): 최종적으로 수정 후 제출해야할 파일.
train/val 데이터로 모델 학습 후 테스트셋에 대한 예측 json 파일을 저장하도록 동작해야 함.
- 2) requirements.txt (제출): 본 프로젝트를 실행하기 위한 라이브러리 모음집. 제출시 추가 라이브러리가 있다면 포함하도록 수정해야 함.
- 3) 이외에 위 submission 파일 동작을 위한 파일들(e.g., models)

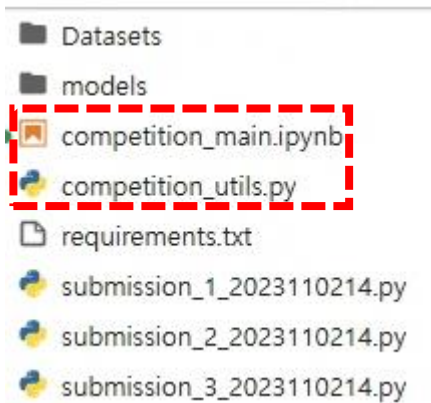


실험 코드 설명

- 배포되는 CV_midterm_competition_code_v2는 다음과 같은 파일을 포함하고 있음.

(옵션)

- 1) 나머지는 반복 실험 및 실험 결과 저장을 위해 v1에서 개선되었음.
- 2) 평가지표 코드 또한 앞선 슬라이드의 평가지표 설명에 맞추어 변동이 있으니 본 버전을 사용 권장



실험 코드 설명

➤ competition_main.ipynb: 실험 실행 코드

- 3개 submission 실험 및 반복 실험을 반영한 코드. 자세한 내용은 주석 참고.

```
from competition_utils import *

# 제출 함수를 리스트로 정의 (submission_N_학번)
# 깃허브에는 최대 3개 업로드 데모시 하이퍼파라미터 튜닝 후 1개만 제출
submission_functions = ['submission_1_2023110214', 'submission_2_2023110214', 'submission_3_2023110214']
for submission_function in submission_functions:
    exec(f"from {submission_function} import {submission_function}")

# 분석 방향에 따라 iteration 수 수정
# 평가시에는 데모와 다른 스플릿 시드를 만들어 [1, 10]로 진행 예정
iterations = [1, 10]

# 'Crown Detection'는 예제 데이터셋, 데모 및 평가시에는 'CV_Competition'
Dataset_Name = 'Crown Detection'

# 결과를 모아 하나의 csv 파일에 저장
results_df = pd.DataFrame(columns=[
    'Experiment Time', 'Iteration', 'Submission Function',
    'IoU', 'Dice', 'Precision', 'Recall', 'Output Json Path',
])
csv_filename = f"Evaluation_Results_{datetime.now().strftime('%y%m%d_%H%M%S')}.csv"

# 서로 다른 iteration, submission function으로 실험 진행
for iteration in range(iterations[0], iterations[1]+1):
    yaml_path = f'Datasets/{Dataset_Name}/data_iter_{iteration:02d}.yaml'
    for submission_function in submission_functions:
        ex_time = datetime.now().strftime('%y%m%d_%H%M%S')
        output_json_path = f"{ex_time}_{submission_function}_iter_{iteration}_detection_results.json"
        globals()[submission_function](yaml_path, output_json_path)
        labels_dir = f'Datasets/{Dataset_Name}/labels'
        vis_output_dir = f"{ex_time}_visualization_results"
        image_level_result_path = f"{ex_time}_{submission_function}_iter_{iteration}_image_level_results.json"
        stats = eval_and_vis(yaml_path, output_json_path, labels_dir, image_level_result_path, vis_output_dir, vis=False) # 분석 방향에 따라 vis=True 설정
        new_row = {
            'Experiment Time': ex_time,
            'Iteration': iteration,
            'Submission Function': submission_function,
            'IoU': stats['IoU']['avg'],
            'Dice': stats['Dice']['avg'],
            'Precision': stats['Precision']['avg'],
            'Recall': stats['Recall']['avg'],
            'Output Json Path': output_json_path,
        }
    results_df = pd.concat([results_df, pd.DataFrame([new_row])], ignore_index=True)
results_df.to_csv(csv_filename, index=False)
```

```
# 분석 방향에 따라 전체 결과를 모아 평균-표준편차-통계테스트 결과 저장

keep_columns = ['Iteration', 'Submission Function']
keep_measures = ['IoU', 'Dice', 'Precision', 'Recall']

reduction = 'Iteration'
row = 'Measure Type'
column = 'Submission Function'
reference_column = 'submission_1_2023110214'

custom_fmt_template = '{mean_fmt} ± {std_fmt} {significance}'
significance_levels = [0.1, 0.05, 0.01]
decimal_places = 3
transpose = True
make_tables(csv_filename, keep_columns, keep_measures, reduction, row, column,
            reference_column, custom_fmt_template, significance_levels, decimal_places, transpose)
```


실험 코드 설명

➤ submission_{n}_{학번}.py 파일

- submission_{n}_{학번}(yaml_path, output_json_path)가 수정되어야 하며 내부 동작과정은 자유롭게 수정 가능
- 아래 함수들은 모두 삭제 가능
- submission 내 주석과 같이 데모시 변경 가능한 하이퍼 파라미터는 앞부분에 한꺼번에 정의되어야하며 뒷부분은 데모 중 수정이 불가함

데모시 수정 가능한 부분.
세목은 1p 참조.

데모시 수정 불가한 부분

```
1 import os
2 import cv2
3 import yaml
4 import torch
5 import random
6 import numpy as np
7 from PIL import Image
8 from datetime import datetime
9 from models import YOLOv8n
10
11 def submission_1_2023110214(yaml_path, output_json_path):
12     ##### can be modified (Only Hyperparameters, which can be modified in demo) #####
13     data_config = load_yaml_config(yaml_path)
14     model_name = 'yolov8n'
15     ex_dict = {}
16     epochs = 20
17     batch_size = 16
18     optimizer = 'AdamW'
19     lr = 1e-3
20     momentum = 0.9
21     weight_decay = 1e-4
22
23     ##### can be modified (Only Models, which can't be modified in demo) #####
24     from ultralytics import YOLO
25     Experiments_Time = datetime.now().strftime("%y%m%d_%H%M%S")
26     ex_dict['Iteration'] = int(yaml_path.split('.')[0][-2:])
27     image_size = 640
28     output_dir = 'tmp'
29     optim_args = {'optimizer': optimizer, 'lr': lr, 'momentum': momentum, 'weight_decay': weight_decay}
30     devices = [0]
31     device = torch.device("cuda:"+str(devices[0])) if len(devices)>0 else torch.device("cpu")
32     ex_dict['Experiment Time'] = Experiments_Time; ex_dict['Epochs'] = epochs;
33     ex_dict['Batch Size'] = batch_size;
34     ex_dict['Device'] = device
35     ex_dict['Optimizer'] = optimizer;
36     ex_dict['LR']=optim_args['lr']; ex_dict['Weight Decay']=optim_args['weight_decay']; ex_dict['Momentum']=optim_args['momentum'];
37     ex_dict['Image Size'] = image_size
38     ex_dict['Output Dir'] = output_dir
39     Dataset_Name = yaml_path.split('/')[1]
40     ex_dict['Dataset Name'] = Dataset_Name; ex_dict['Data Config'] = yaml_path; ex_dict['Number of Classes'] = data_config['nc']
41     control_random_seed(42)
42     model = YOLO(f'{model_name}.yaml', verbose=False)
43     os.makedirs(output_dir, exist_ok=True)
44     ex_dict['Model Name'] = model_name; ex_dict['Model']=model;
45     ex_dict = YOLOv8n.train_model(ex_dict)
46     test_images = get_test_images(data_config)
47     results_dict = detect_and_save_bboxes(ex_dict['Model'], test_images)
48     save_results_to_file(results_dict, output_json_path)
```

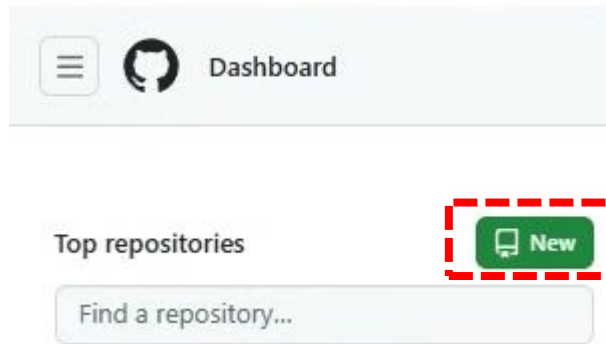
실험 코드 설명

➤ submission_{n}_{학번}.py 파일

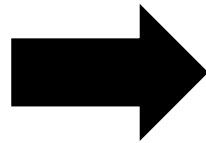
- 결과 json 파일은 아래와 같은 form으로 저장되어야 함.
- {image_path:[{'bbox':[x1, x2, y1, y2],
 'confidence': confidence,
 'class_id': 0,
 'class_name': 'airplane'}, ...]
 }
- 여기서 x1, y1은 각각 예측 박스의 좌상단 x좌표, y좌표(0~640)
- x2, y2은 각각 예측 박스의 우하단 x좌표, y좌표(0~640)
- confidence는 해당 박스에 객체가 존재 여부 점수(0~1)
- image_path는 해당 테스트 이미지 경로
- competition_main.ipynb 파일 실행시 예제 json을 확인 가능

```
{
  "Datasets/CV_Competition/images/3441df2.jpg": [
    {
      "bbox": [
        15.0374755859375,
        194.32615661621094,
        606.1367797851562,
        419.6409912109375
      ],
      "confidence": 0.5473001003265381,
      "class_id": 0,
      "class_name": "airplane"
    }
  ],
  "Datasets/CV_Competition/images/3441df2sada12.jpg": [
    {
      "bbox": [
        26.2508544921875,
        232.23922729492188,
        631.191162109375,
        441.5180969238281
      ],
      "confidence": 0.5675916075706482,
      "class_id": 0,
      "class_name": "airplane"
    }
  ],
  ...
}
```

소스코드 제출 요령



(1) 리파지토리 생성



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

tkdgur658

Repository name *

CV_submission

CV_submission is available.

Great repository names are short and memorable. Need inspiration? How about [sturdy-octo-meme](#)?

Description (optional)

☐ Public

Anyone on the internet can see this repository. You choose who can commit.

☒ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

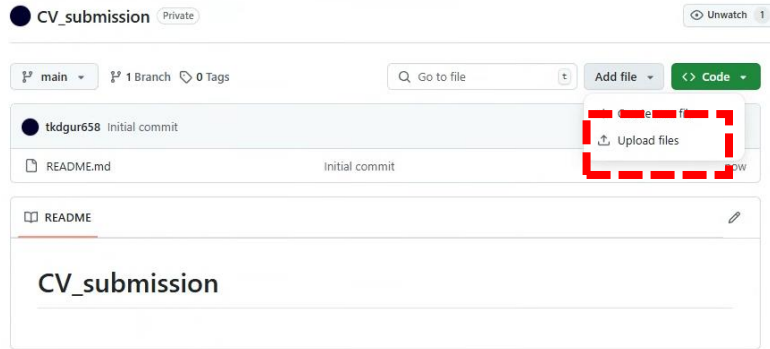
① You are creating a private repository in your personal account.

Create repository

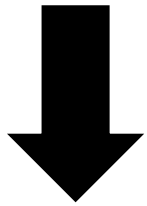
(3) private로 생성 (public XXX)
(4) .md 생성

(5)

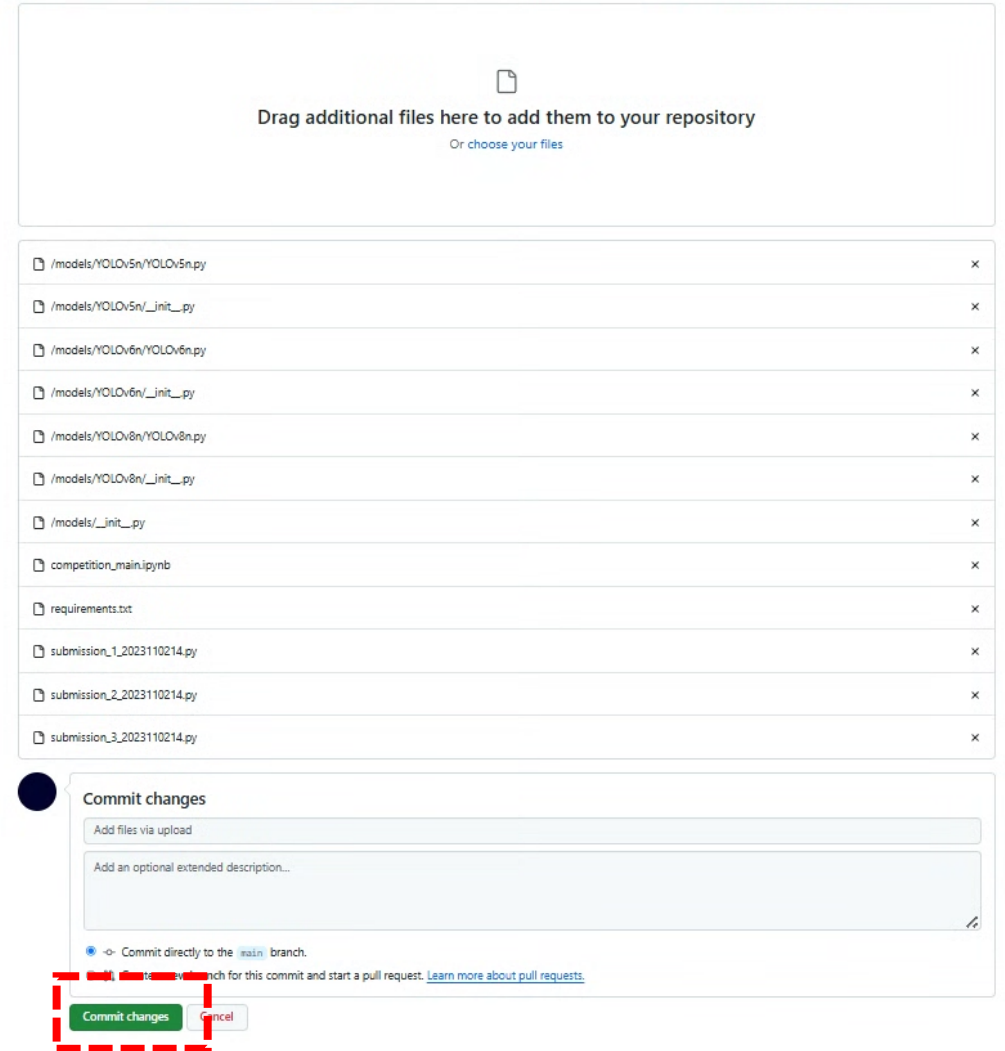
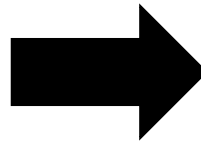
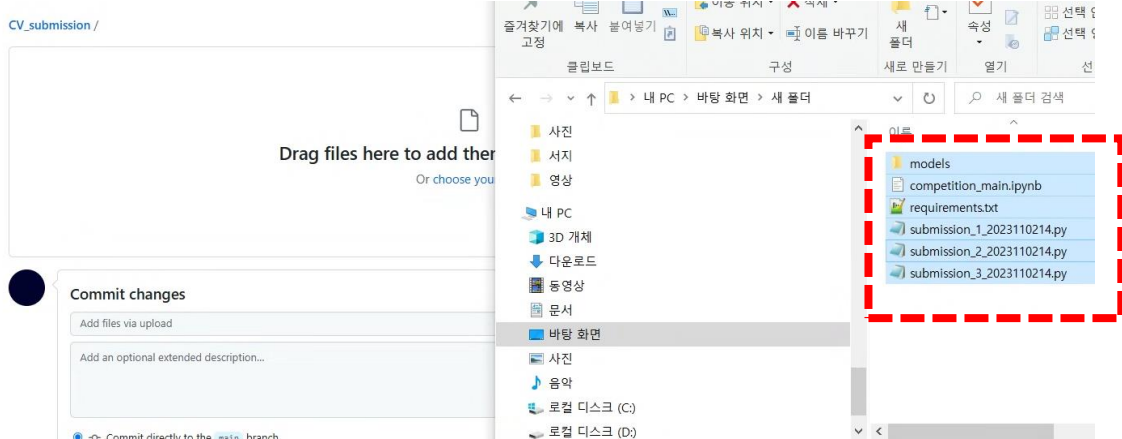
소스코드 제출 요령



(6) 파일 업로드

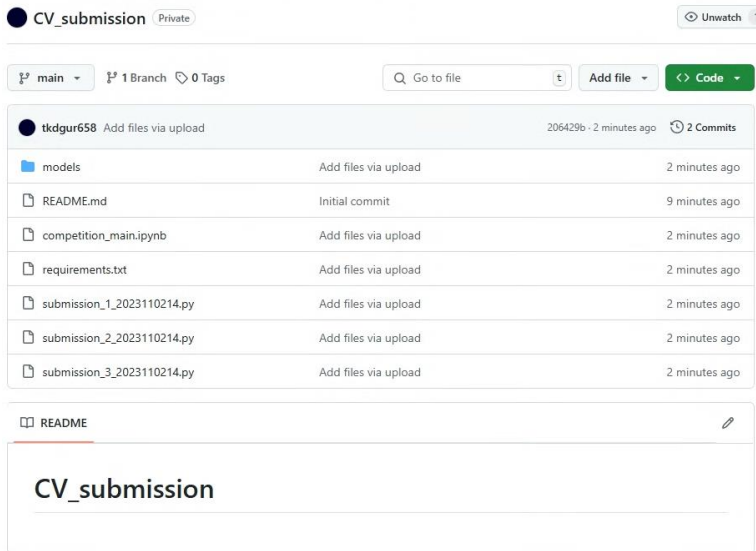


(7) 관련 코드를 모두 업로드



소스코드 제출 요령

(8) 업로드된 코드 확인

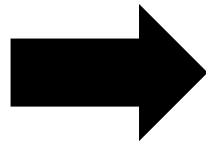


The screenshot shows a GitHub repository named 'CV_submission' with a 'main' branch. The file list includes 'models', 'README.md', 'competition_main.ipynb', 'requirements.txt', and three submission files. The README content is displayed below the file list.

File Name	Commit Message	Time
models	Add files via upload	2 minutes ago
README.md	Initial commit	9 minutes ago
competition_main.ipynb	Add files via upload	2 minutes ago
requirements.txt	Add files via upload	2 minutes ago
submission_1_2023110214.py	Add files via upload	2 minutes ago
submission_2_2023110214.py	Add files via upload	2 minutes ago
submission_3_2023110214.py	Add files via upload	2 minutes ago

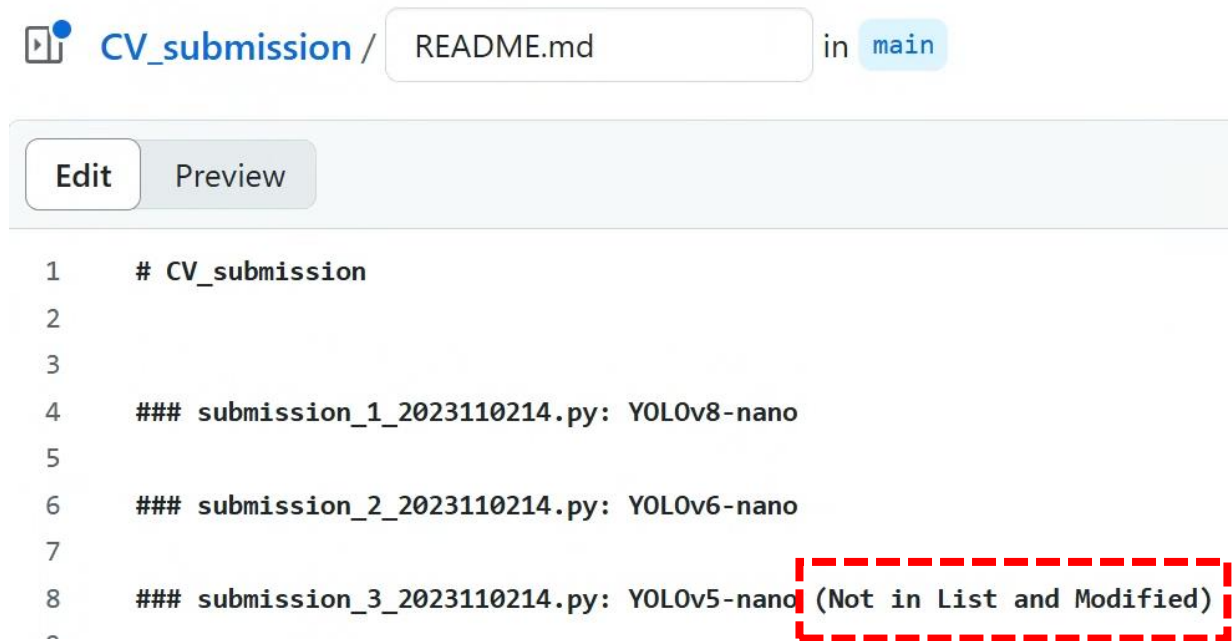
README content:

```
CV_submission
```



(9) README.md 파일 수정

- submission 파일과 모델명 매칭
- 10개 리스트에 포함되지 않은 모델 표기
- 커밋



The screenshot shows the 'README.md' file view in the 'CV_submission' repository. The file list includes 'models', 'README.md', 'competition_main.ipynb', 'requirements.txt', and three submission files. The README content is displayed below the file list.

File Name	Commit Message	Time
models	Add files via upload	2 minutes ago
README.md	Initial commit	9 minutes ago
competition_main.ipynb	Add files via upload	2 minutes ago
requirements.txt	Add files via upload	2 minutes ago
submission_1_2023110214.py	Add files via upload	2 minutes ago
submission_2_2023110214.py	Add files via upload	2 minutes ago
submission_3_2023110214.py	Add files via upload	2 minutes ago

README content:

```
# CV_submission

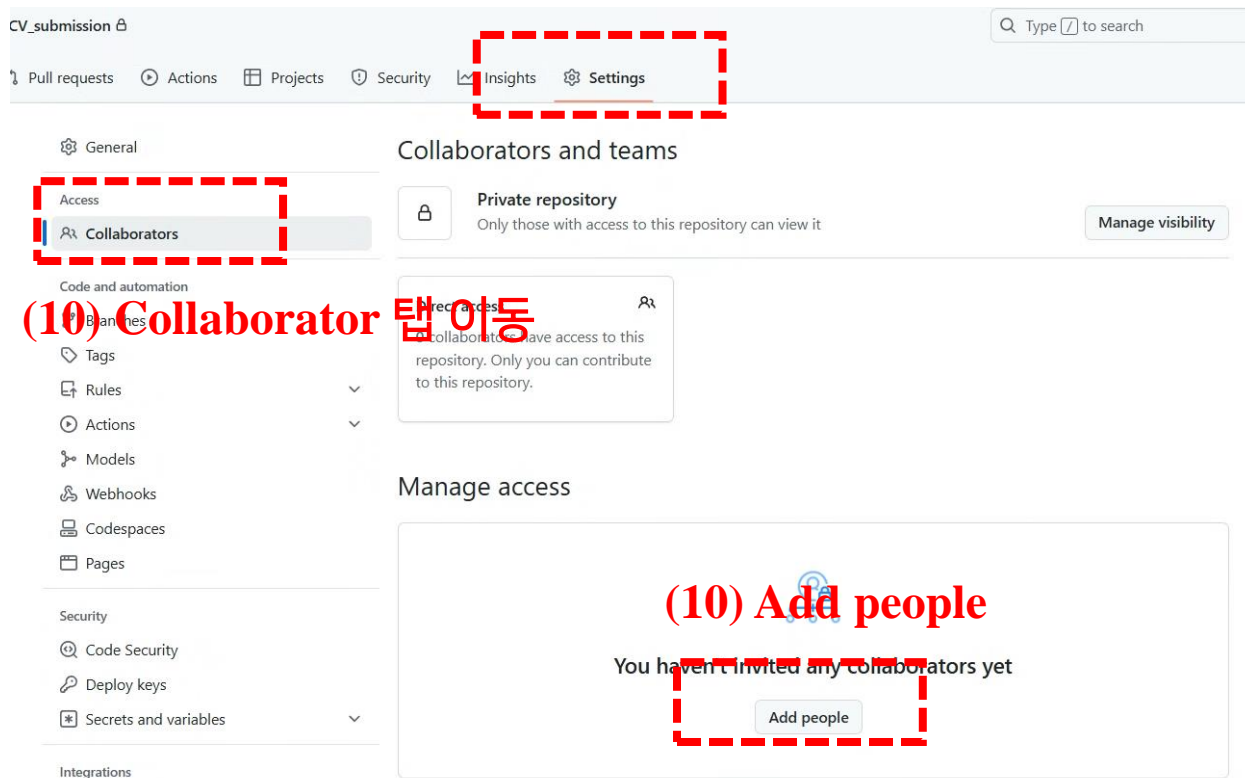
### submission_1_2023110214.py: YOLOv8-nano

### submission_2_2023110214.py: YOLOv6-nano

### submission_3_2023110214.py: YOLOv5-nano (Not in List and Modified)
```

소스코드 제출 요령

(10) 세팅 탭 이동

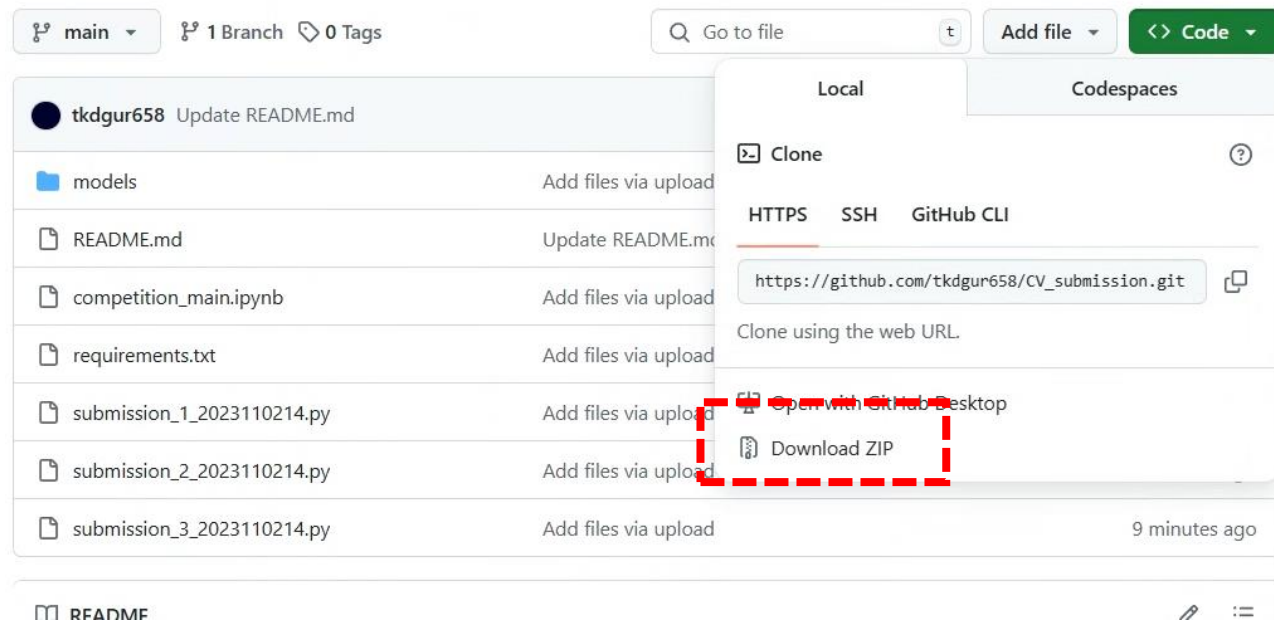


(10) Collaborator 탭 이동

(10) Add people

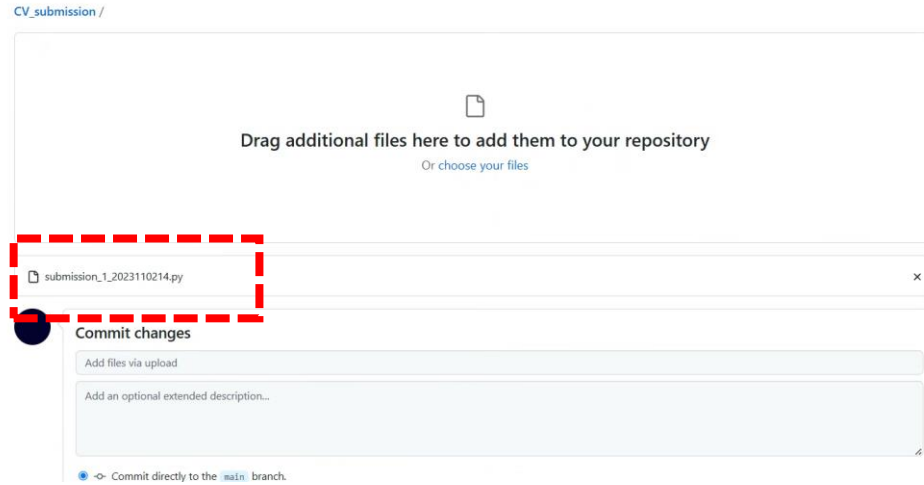
➔ tkdgur658@gmail.com 초대

데모 당일

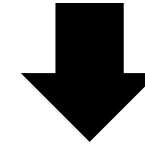


- 자리에 착석 후 본인 리파지토리 clone
- 데이터셋은 따로 제공 예정
- 모두가 서버에 소스코드와 데이터셋이 준비되면 동시에 하이퍼파라미터 튜닝 시작
- 1시간 동안 진행
- 1시간 뒤 서버 다운이니 중간 파일 지속적으로 다운 권장

데모 당일

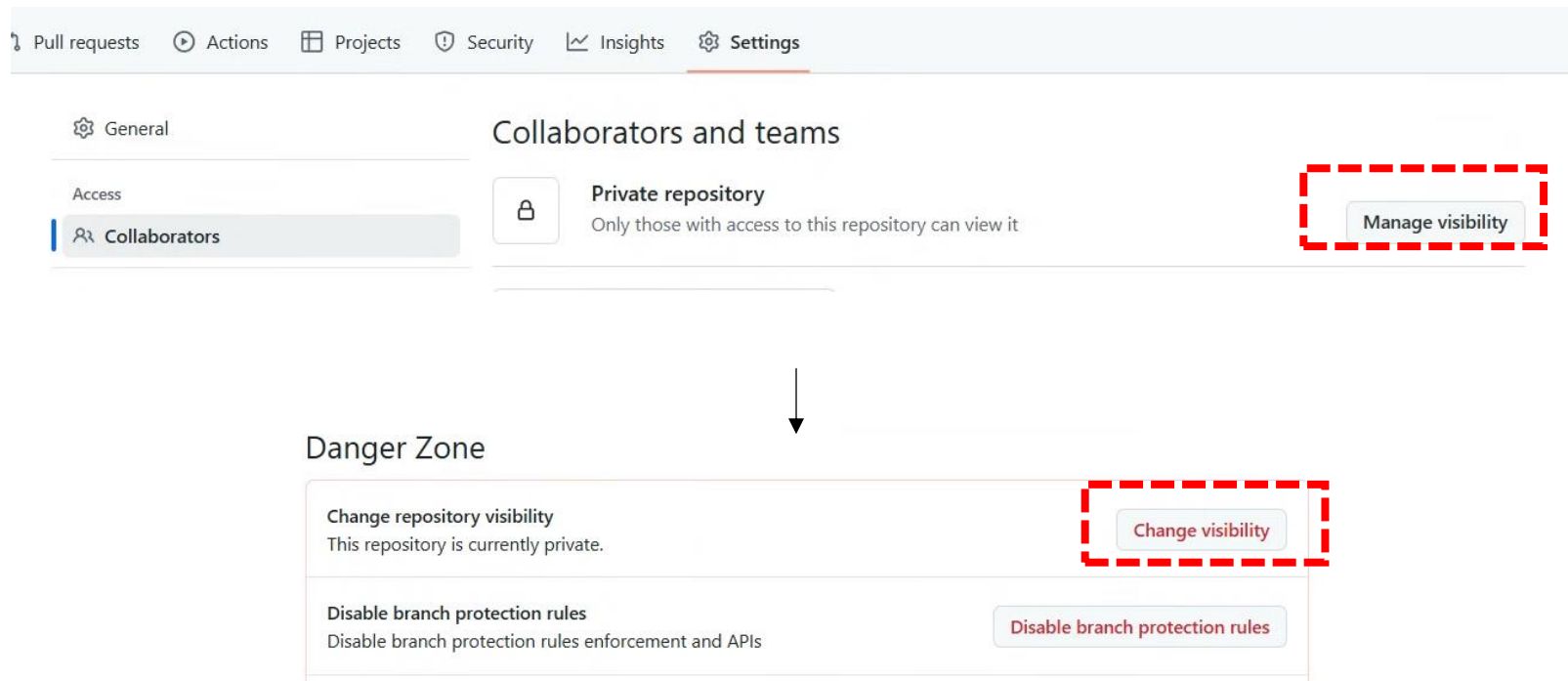


- 완료되면 다시 깃허브에 튜닝 내용을 반영한 모델 파일만 업로드



- README.md 파일에서 제출할 모델 파일에 “(Proposed)” 입력 후 커밋
- 리파지토리 공개 설정

데모 당일



- 리파지토리 공개 설정(끝)