drone 0.7

드론 감지 및 분류 AI모델

AIM

전남대학교

김주완(인공지능학부)
서가연(인공지능학부)
조하영(인공지능학부)

ENERGY+AI 2022 하계 마이크로 캡스톤 디자인

전남대학교 ENERGY+AI
핵심인재양성 교육연구단
CNU Energy+AI Education Research Center

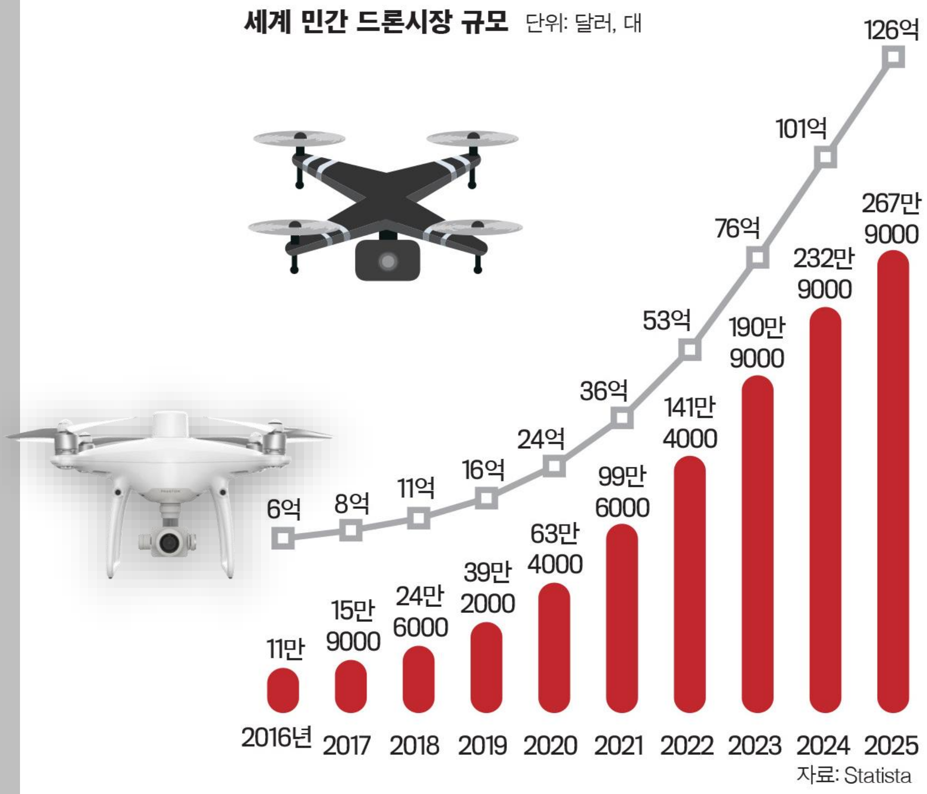
목 차

연구 배경과 목표

데이터 수집과 전처리 과정

모델링과 결과

연구 배경



드론에 대한 관심이 높아지면서

드론의 활용과 시장이 확대되고 있다.

드론이 상용화됨에 따라 불법촬영으로 인한

사생활 침해 문제가 발생하고 있다.

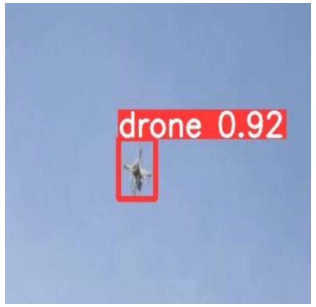
연구 목표

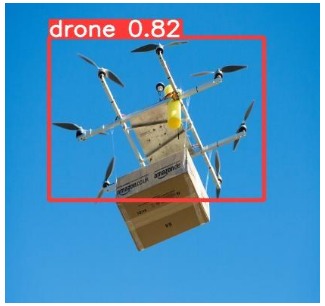
TASK1

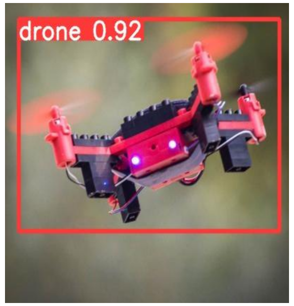
**Object Detection:
드론 객체 인식**

GOAL

이미지(영상) 데이터에서 드론 객체 탐지



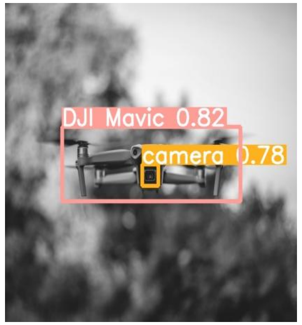




연구 목표

GOAL

드론의 종류를 구별하고, 카메라 객체 탐지



TASK2

Object Classification:

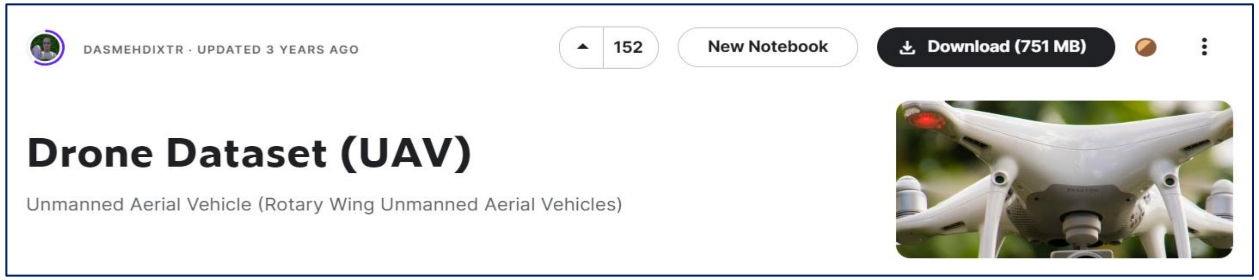
드론 모델 종류별 분류

데이터 수집

드론 데이터 수집 방법

플랫폼 활용

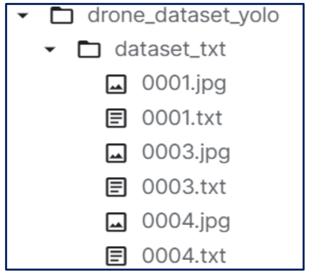
kaggle



Drone Dataset (UAV)

Unmanned Aerial Vehicle (Rotary Wing Unmanned Aerial Vehicles)

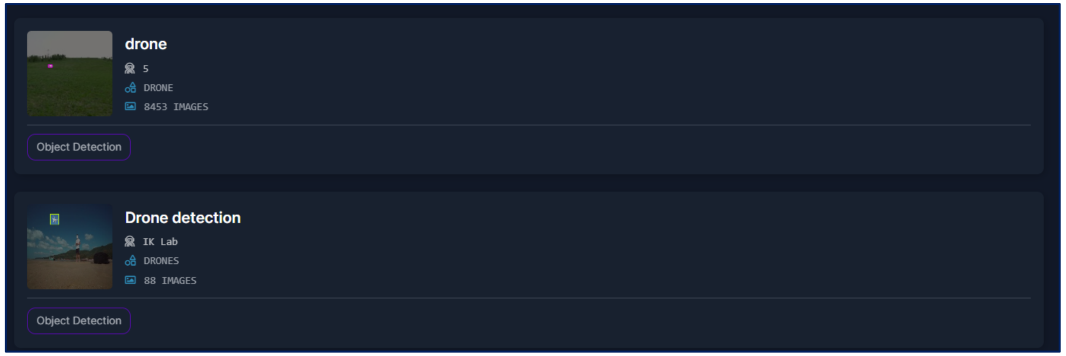
https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav



drone_dataset_yolo
dataset_txt
0001.jpg
0001.txt
0003.jpg
0003.txt
0004.jpg
0004.txt

라벨링 된 데이터 다운로드

roboflow



drone
5
DRONE
8453 IMAGES
Object Detection

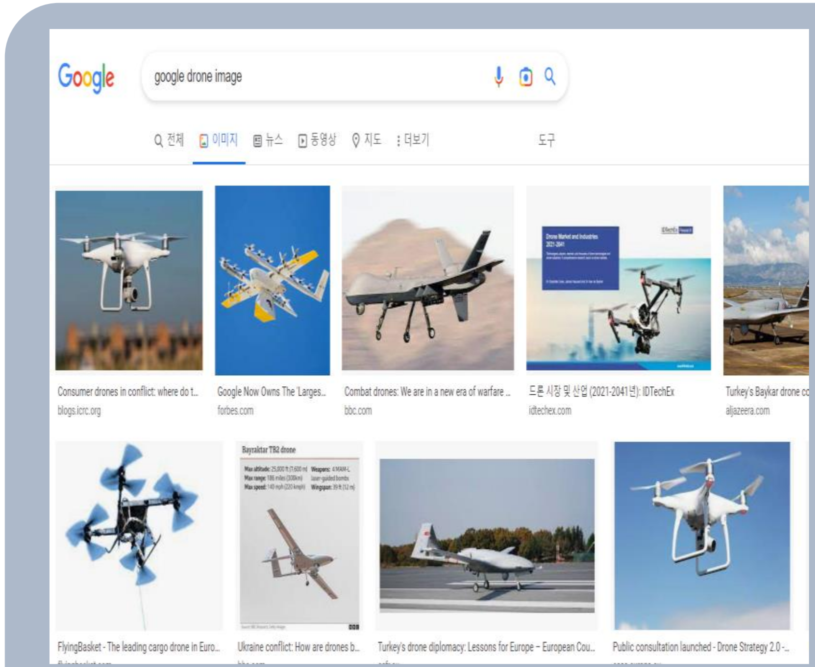
Drone detection
IK Lab
DRONES
88 IMAGES
Object Detection

https://universe.roboflow.com/search?q=drone

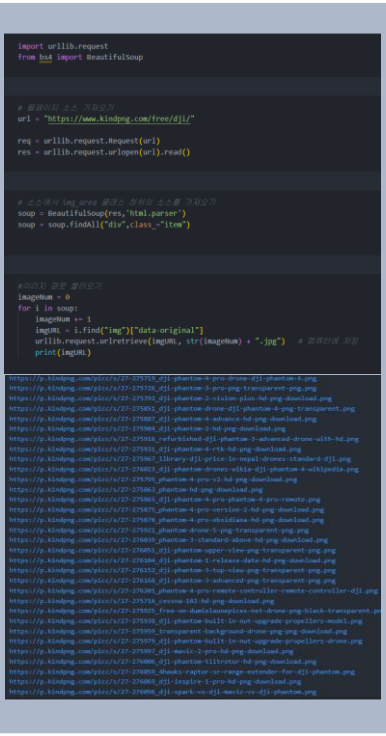
데이터 수집

드론 데이터 수집 방법

웹크롤링

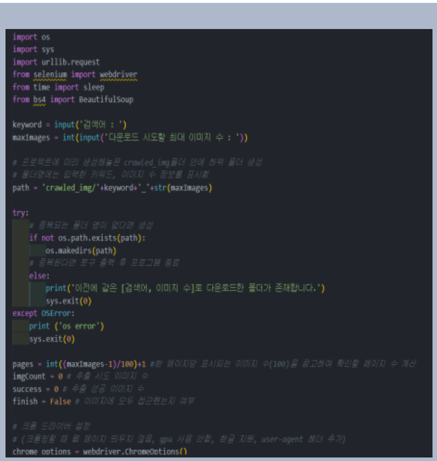

Google
google drone image
전체
이미지
뉴스
동영상
지도
더보기
도구
Consumer drones in conflict: where do t...
blogs.icrc.org
Google Now Owns The 'Larges...
forbes.com
Combat drones: We are in a new era of warfare ...
bbc.com
드론 시장 및 산업 (2021-2041년): IDTechEx
idtechex.com
Turkey's Baykar drone cc
aljazeera.com
Bayraktar TB2 drone
FlyingBasket - The leading cargo drone in Euro...
Ukraine conflict: How are drones b...
Turkey's drone diplomacy: Lessons for Europe - European Cou...
Public consultation launched - Drone Strategy 2.0 -...

이미지 크롤링으로 드론 데이터 수집

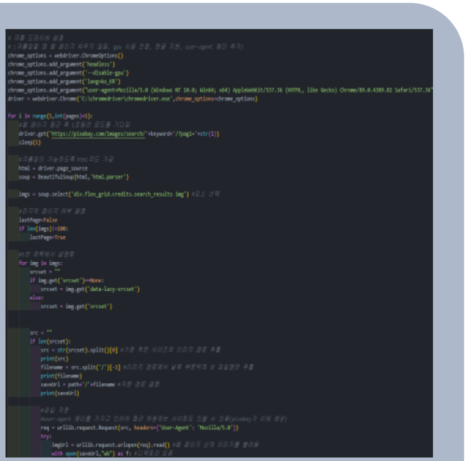


```python
import urllib.request
from bs4 import BeautifulSoup

# 웹페이지 소스 가져오기
url = "https://www.kindpng.com/free/dji/"

req = urllib.request.Request(url)
res = urllib.request.urlopen(url).read()

# 소스에서 img_area 클래스 하위의 소스를 가져오기
soup = BeautifulSoup(res,'html.parser')
soup = soup.findAll("div",class_="item")

#이미지 경로 불러오기
imageNum = 0
for i in soup:
    imageNum += 1
    imgURL = i.find("img")["data-original"]
    urllib.request.urlretrieve(imgURL, str(imageNum) + ".jpg")   # 컴퓨터에 저장
    print(imgURL)
```

```
https://p.kindpng.com/picc/s/27-275719_dji-phantom-4-pro-drone-dji-phantom-4.png
https://p.kindpng.com/picc/s/27-275726_dji-phantom-3-pro-png-transparent-png.png
https://p.kindpng.com/picc/s/27-275792_dji-phantom-2-vision-plus-hd-png-download.png
https://p.kindpng.com/picc/s/27-275851_dji-phantom-drone-dji-phantom-4-png-transparent.png
https://p.kindpng.com/picc/s/27-275887_dji-phantom-4-advance-hd-png-download.png
https://p.kindpng.com/picc/s/27-275904_dji-phantom-2-hd-png-download.png
https://p.kindpng.com/picc/s/27-275916_refurbished-dji-phantom-3-advanced-drone-with-hd.png
https://p.kindpng.com/picc/s/27-275931_dji-phantom-4-rtk-hd-png-download.png
https://p.kindpng.com/picc/s/27-275967_library-dji-price-in-nepal-drones-standard-dji.png
https://p.kindpng.com/picc/s/27-276023_dji-phantom-drones-wikia-dji-phantom-4-wikipedia.png
https://p.kindpng.com/picc/s/27-275795_phantom-4-pro-v2-hd-png-download.png
https://p.kindpng.com/picc/s/27-275863_phantom-hd-png-download.png
https://p.kindpng.com/picc/s/27-275865_dji-phantom-4-pro-phantom-4-pro-remote.png
https://p.kindpng.com/picc/s/27-275875_phantom-4-pro-version-2-hd-png-download.png
https://p.kindpng.com/picc/s/27-275878_phantom-4-pro-obsidiana-hd-png-download.png
https://p.kindpng.com/picc/s/27-275921_phantom-drone-5-png-transparent-png.png
https://p.kindpng.com/picc/s/27-276039_phantom-3-standard-above-hd-png-download.png
https://p.kindpng.com/picc/s/27-276051_dji-phantom-upper-view-png-transparent-png.png
https://p.kindpng.com/picc/s/27-276104_dji-phantom-1-release-date-hd-png-download.png
https://p.kindpng.com/picc/s/27-276152_dji-phantom-3-top-view-png-transparent-png.png
https://p.kindpng.com/picc/s/27-276168_dji-phantom-3-advanced-png-transparent-png.png
https://p.kindpng.com/picc/s/27-279201_phantom-4-pro-remote-controller-remote-controller-dji.png
https://p.kindpng.com/picc/s/27-275710_cessna-182-hd-png-download.png
https://p.kindpng.com/picc/s/27-275925_free-on-dumielauxepices-net-drone-png-black-transparent.pn
https://p.kindpng.com/picc/s/27-275938_dji-phantom-built-in-nut-upgrade-propellers-model.png
https://p.kindpng.com/picc/s/27-275959_transparent-background-drone-png-png-download.png
https://p.kindpng.com/picc/s/27-275979_dji-phantom-built-in-nut-upgrade-propellers-drone.png
https://p.kindpng.com/picc/s/27-275997_dji-mavic-2-pro-hd-png-download.png
https://p.kindpng.com/picc/s/27-276006_dji-phantom-tiltrotor-hd-png-download.png
https://p.kindpng.com/picc/s/27-276059_4hawks-raptor-sr-range-extender-for-dji-phantom.png
https://p.kindpng.com/picc/s/27-276069_dji-inspire-1-pro-hd-png-download.png
https://p.kindpng.com/picc/s/27-276098_dji-spark-vs-dji-mavic-vs-dji-phantom.png
```



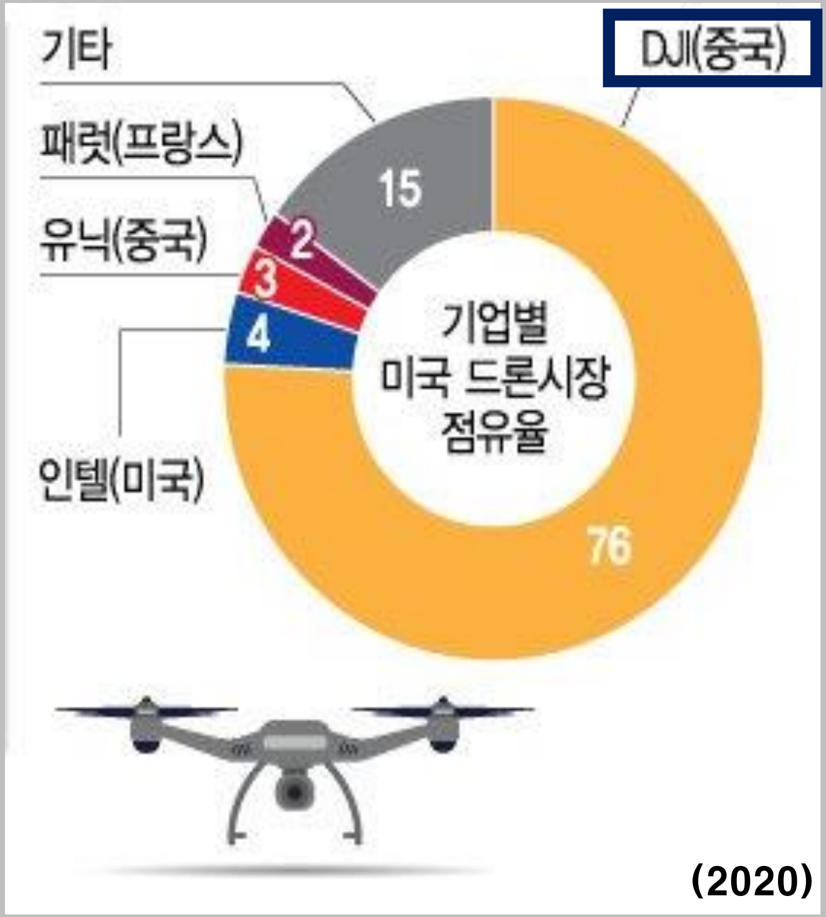
```python
import os
import sys
import urllib.request
from selenium import webdriver
from time import sleep
from bs4 import BeautifulSoup

keyword = input('검색어 : ')
maxImages = int(input('다운로드 시도할 최대 이미지 수 : '))

# 프로젝트에 미리 생성해놓은 crawled_img폴더 안에 하위 폴더 생성
# 폴더명에는 입력한 키워드, 이미지 수 정보를 표시함
path = 'crawled_img/'+keyword+'_'+str(maxImages)

try:
    # 중복되는 폴더 명이 없다면 생성
    if not os.path.exists(path):
        os.makedirs(path)
    # 중복된다면 문구 출력 후 프로그램 종료
    else:
        print('이전에 같은 [검색어, 이미지 수]로 다운로드한 폴더가 존재합니다.')
        sys.exit(0)
except OSError:
    print ('os error')
    sys.exit(0)

pages = int((maxImages-1)/100)+1 #한 페이지당 표시되는 이미지 수(100)을 참고하여 확인할 페이지 수 계산
imgCount = 0 # 추출 시도 이미지 수
success = 0 # 추출 성공 이미지 수
finish = False # 이미지에 모두 접근했는지 여부

# 크롬 드라이버 설정
# (크롤링할 때 웹 페이지 띄우지 않음, gpu 사용 안함, 한글 지원, user-agent 헤더 추가)
chrome_options = webdriver.ChromeOptions()
```



```python
# 크롬 드라이버 설정
# (크롤링할 때 웹 페이지 띄우지 않음, gpu 사용 안함, 한글 지원, user-agent 헤더 추가)
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('headless')
chrome_options.add_argument('--disable-gpu')
chrome_options.add_argument('lang=ko_KR')
chrome_options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
driver = webdriver.Chrome('C:\chromedriver\chromedriver.exe',chrome_options=chrome_options)

for i in range(1,int(pages)+1):
    #웹 페이지 접근 후 1초동안 로드를 기다림
    driver.get('https://pixabay.com/images/search/'+keyword+'/?pagi='+str(i))
    sleep(1)

    #크롤링이 가능하도록 html코드 가공
    html = driver.page_source
    soup = BeautifulSoup(html,'html.parser')

    imgs = soup.select('div.flex_grid.credits.search_results img') #요소 선택

    #마지막 페이지 여부 결정
    lastPage=False
    if len(imgs)!=100:
        lastPage=True

    #[illegible]
    for img in imgs:
        srcset = ""
        if img.get('srcset')==None:
            srcset = img.get('data-lazy-srcset')
        else:
            srcset = img.get('srcset')


        src = ""
        if len(srcset):
            src = str(srcset).split()[0] #가장 작은 사이즈의 이미지 경로 추출
            print(src)
            filename = src.split('/')[-1] #이미지 경로에서 날짜 부분뒤의 순 파일명만 추출
            print(filename)
            saveUrl = path+'/'+filename #저장 경로 결정
            print(saveUrl)

            #파일 저장
            #user-agent 헤더를 가지고 있어야 접근 허용하는 사이트도 있을 수 있음(pixabay가 이에 해당)
            req = urllib.request.Request(src, headers={'User-Agent': 'Mozilla/5.0'})
            try:
                imgUrl = urllib.request.urlopen(req).read() #웹 페이지 상의 이미지를 불러옴
                with open(saveUrl,"wb") as f: #디렉토리 오픈
```

드론 데이터 종류

미국 드론시장 내 점유율(%)



DJI mavic

DJI Phantom

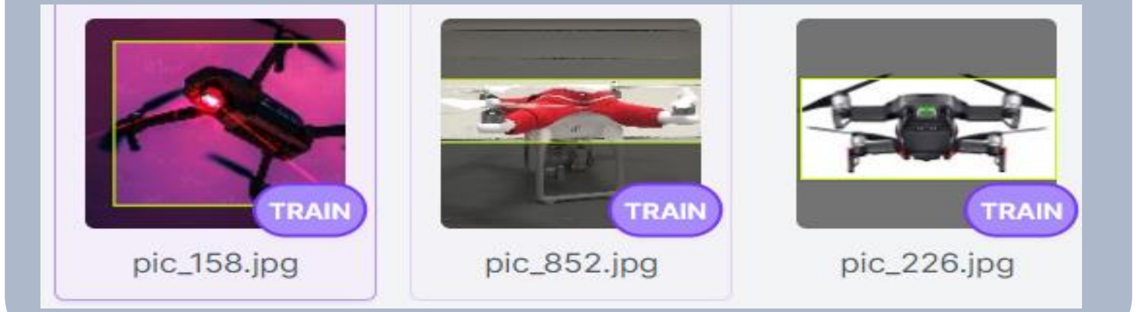
DJI Inspire

데이터 라벨링

- roboflow 플랫폼 이용
- 수집한 데이터 셋: yolov5-pytorch로 export하여 모델에 적용

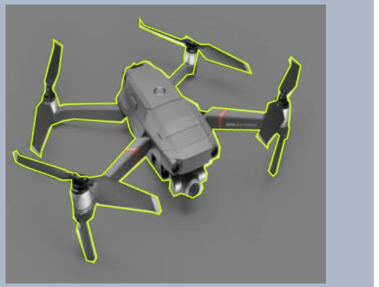
Detection

- class : 1 [“drone”]
- Kaggle + roboflow 6000장
- Annotation: bounding box



Classification

- class : 4
  [“DJI Phantom” , “DJI Mavic” , “DJI Inspire” , “camera”]
- 크롤링 데이터 144장
- Annotation: bounding box + polygon



데이터 전처리 및 증강

전처리 : resize(416*416)

original

»

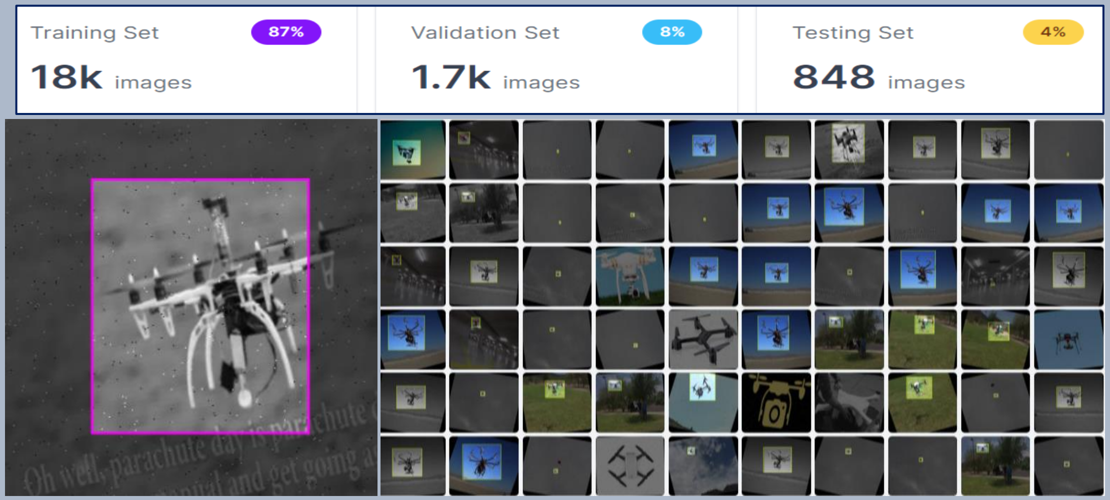
resized

416*416

데이터 전처리 및 증강

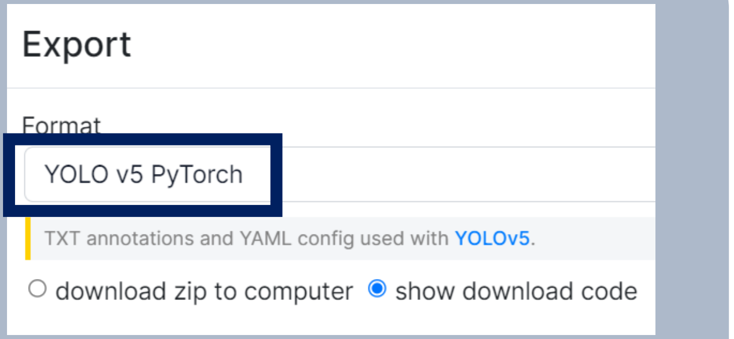
데이터 분리

train : valid : test = 70 : 20 : 10



Training Set 87%	Validation Set 8%	Testing Set 4%
18k images	1.7k images	848 images

dataset



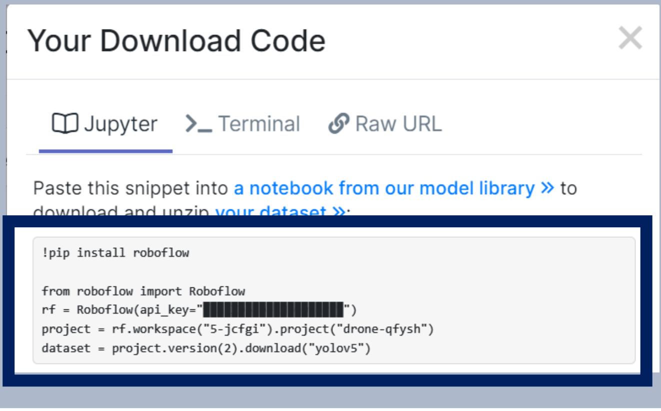
Export

Format

YOLO v5 PyTorch

TXT annotations and YAML config used with YOLOv5.

○ download zip to computer ● show download code



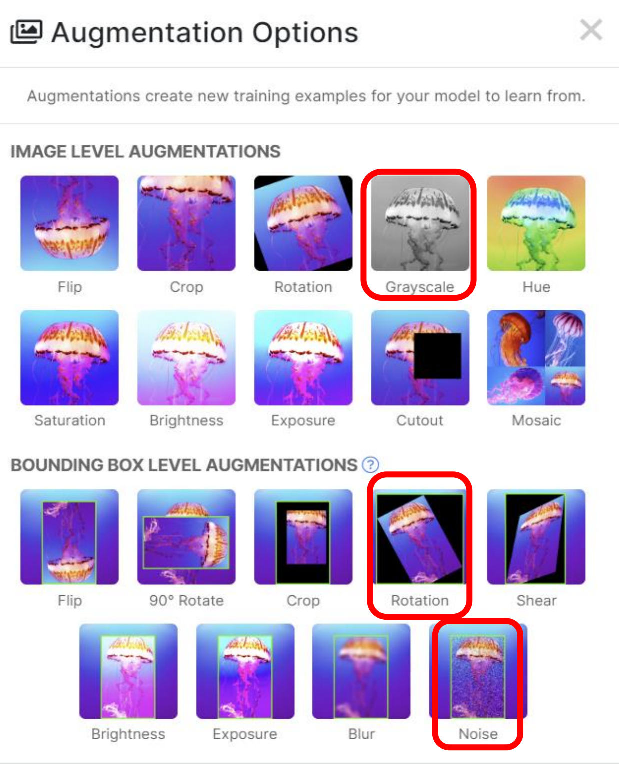
Your Download Code

Paste this snippet into a notebook from our model library » to download and unzip your dataset »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("5-jcfgi").project("drone-qfysh")
dataset = project.version(2).download("yolov5")
```

데이터 전처리 및 증강

데이터 증강(x3)



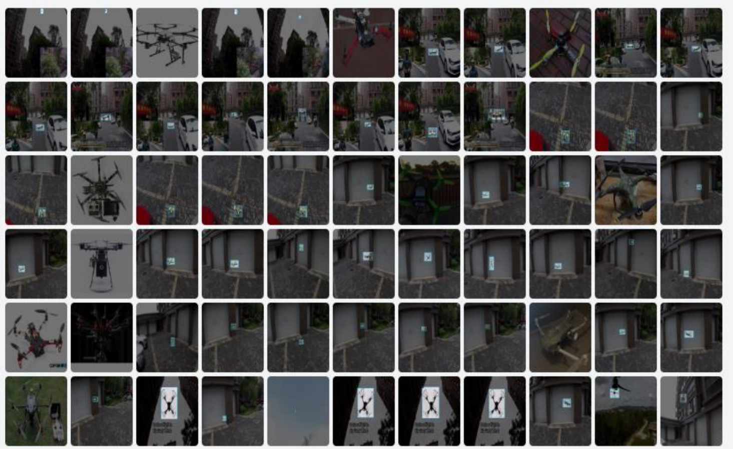
Augmentation Options

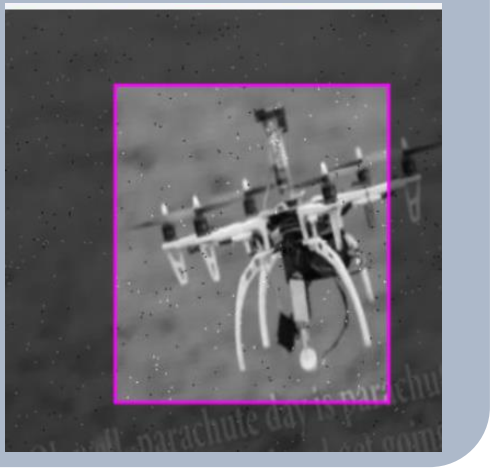
Augmentations create new training examples for your model to learn from.

IMAGE LEVEL AUGMENTATIONS

BOUNDING BOX LEVEL AUGMENTATIONS

- Rotation (−15° ~ +15°)
- Grayscale (Apply to 30% of images)
- Noise (5% of pixels)





모델 선택

Object Detection : ~~CNN, RCNN, Faster-RCNN, SSD-net,~~ ...

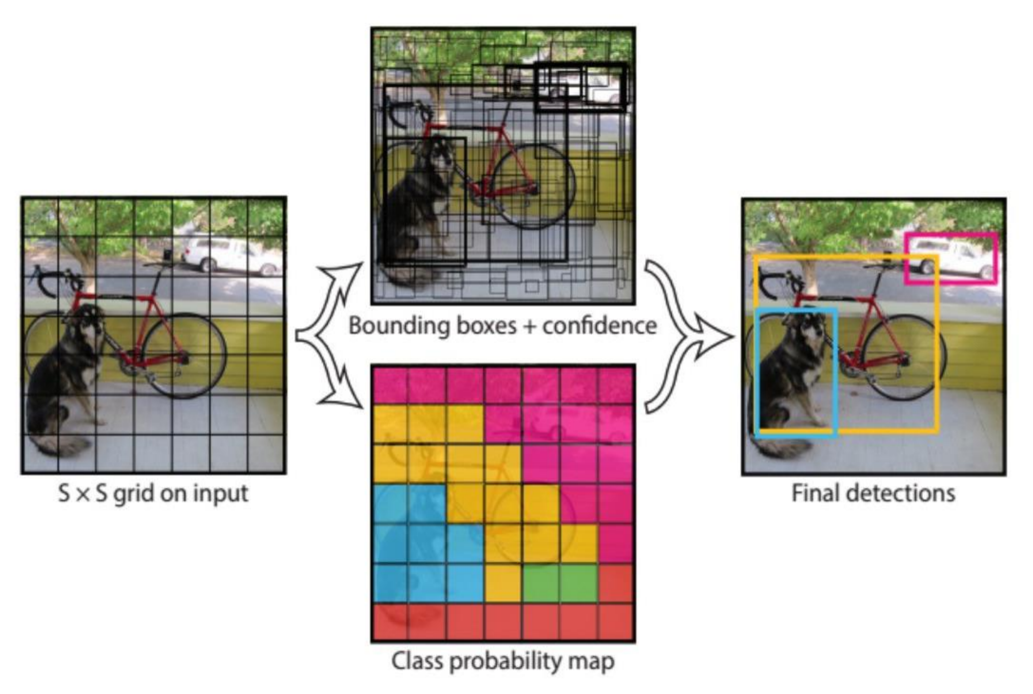
기존의 Object detection 알고리즘은 real-time으로 사용하기에는 속도가 느리다는 한계점

High performance & Fast detection speed

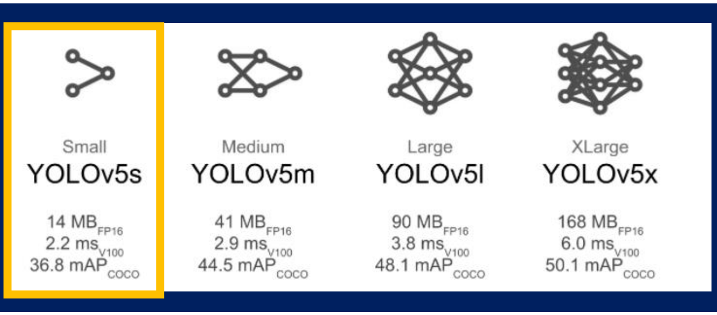
You Only Look Once

모델 선택

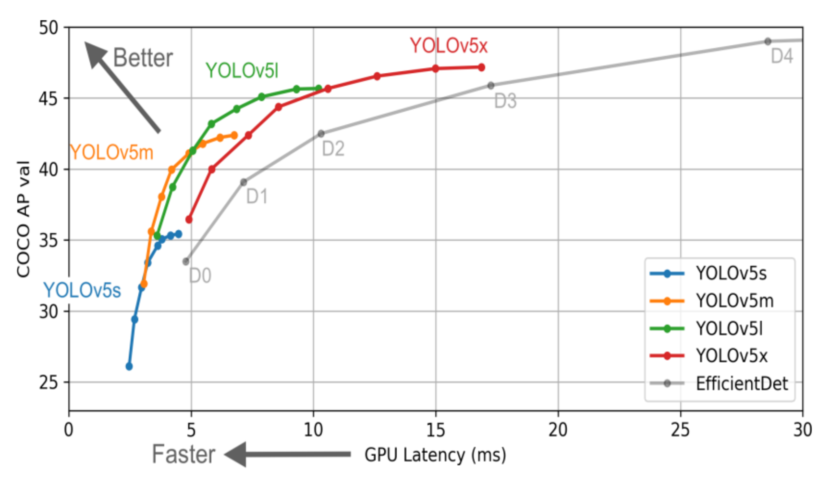
YOLO



YOLOv5 PyTorch



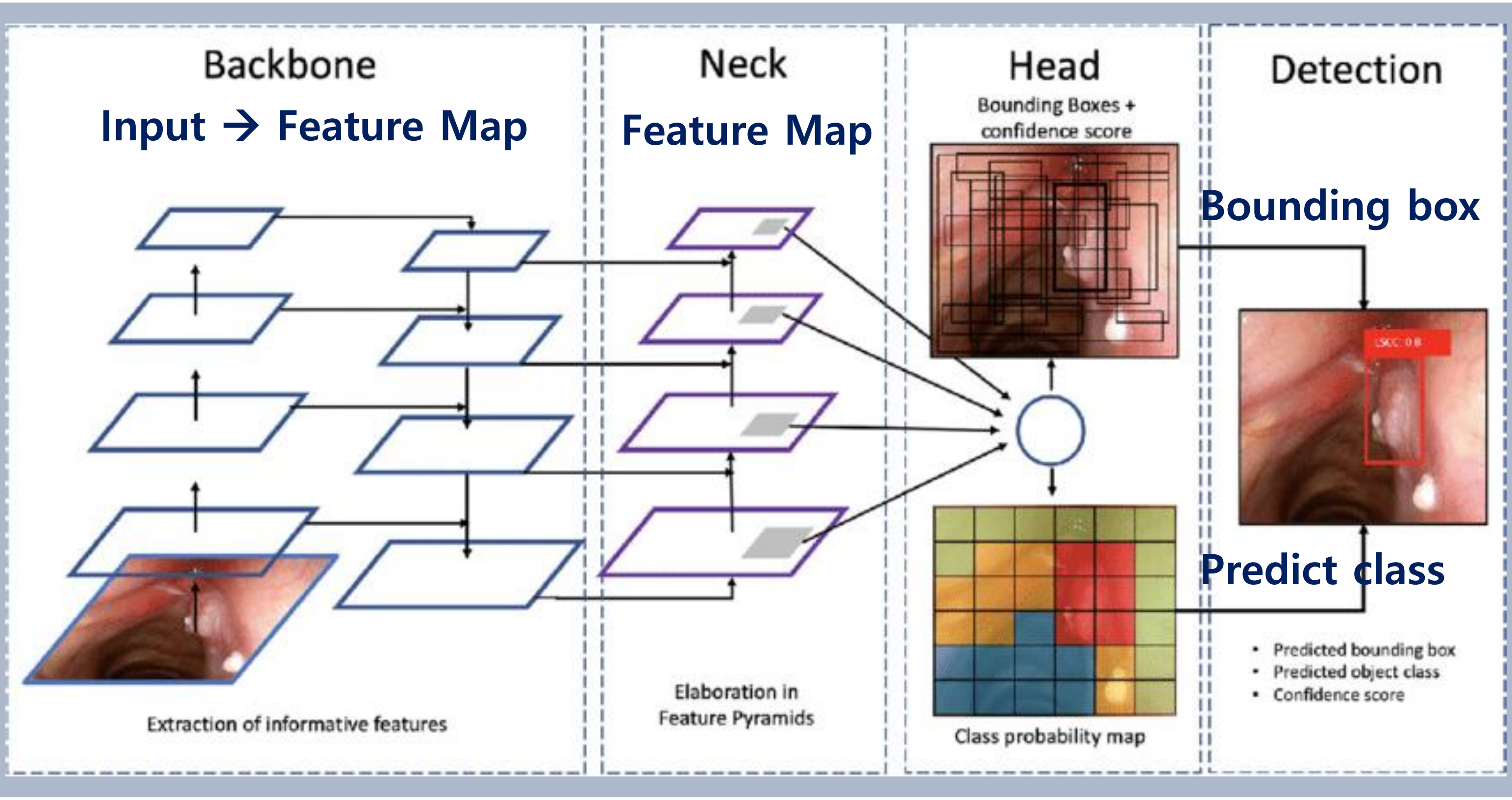
Small	Medium	Large	XLarge
YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
14 MB$_{\text{FP16}}$	41 MB$_{\text{FP16}}$	90 MB$_{\text{FP16}}$	168 MB$_{\text{FP16}}$
2.2 ms$_{\text{V100}}$	2.9 ms$_{\text{V100}}$	3.8 ms$_{\text{V100}}$	6.0 ms$_{\text{V100}}$
36.8 mAP$_{\text{COCO}}$	44.5 mAP$_{\text{COCO}}$	48.1 mAP$_{\text{COCO}}$	50.1 mAP$_{\text{COCO}}$



성능이 좋진 않지만, detection speed(fps)에 강점이 있는 YOLOv5s 선택!

모델 구조

Architecture



Input → Feature Map

Feature Map

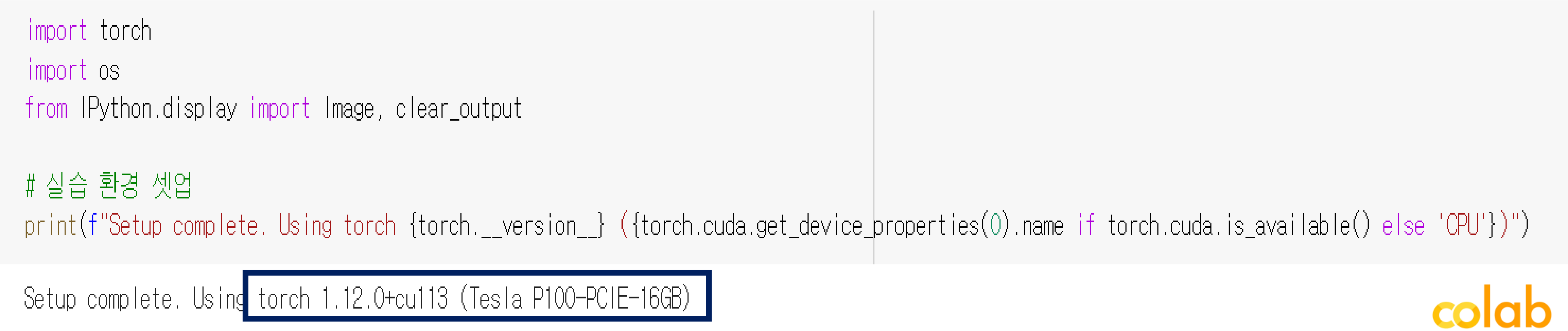
Bounding box

Predict class

모델링

Env Setup

Torch와 cuda 버전 확인



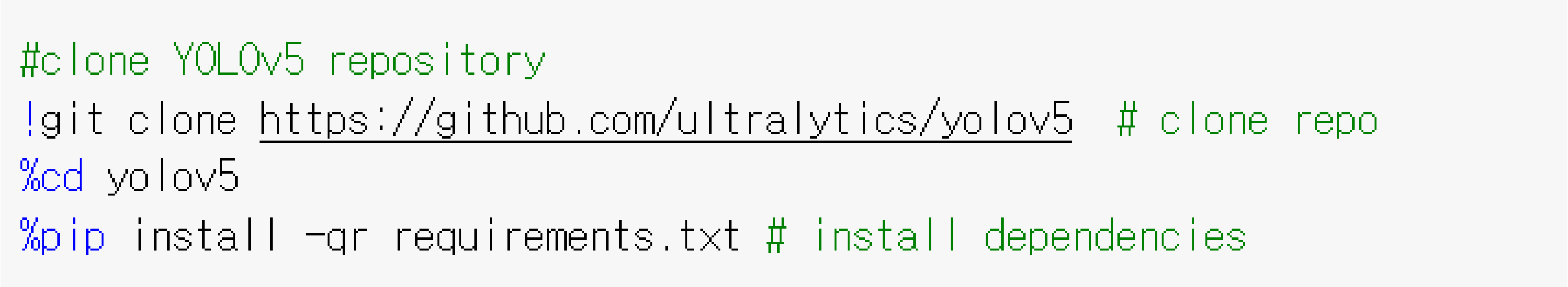
```python
import torch
import os
from IPython.display import Image, clear_output

# 실습 환경 셋업
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

```
Setup complete. Using torch 1.12.0+cu113 (Tesla P100-PCIE-16GB)
```

모델링

Install Dependencies

https://github.com/ultralytics/yolov5



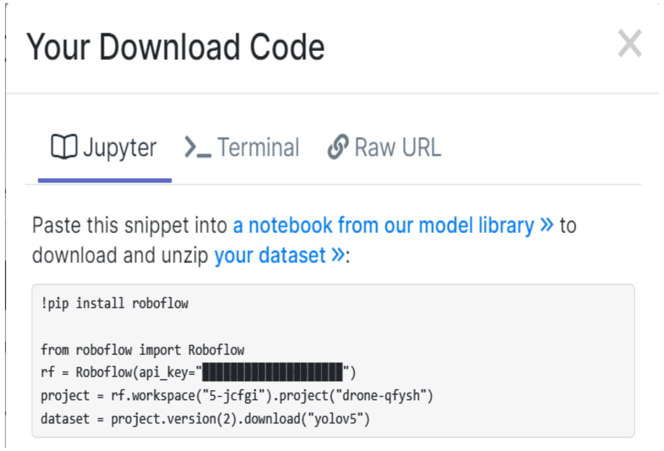
```
#clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
```

Yolov5 repository 다운로드 및 필요한 라이브러리 설치

모델 학습

Drone Detection

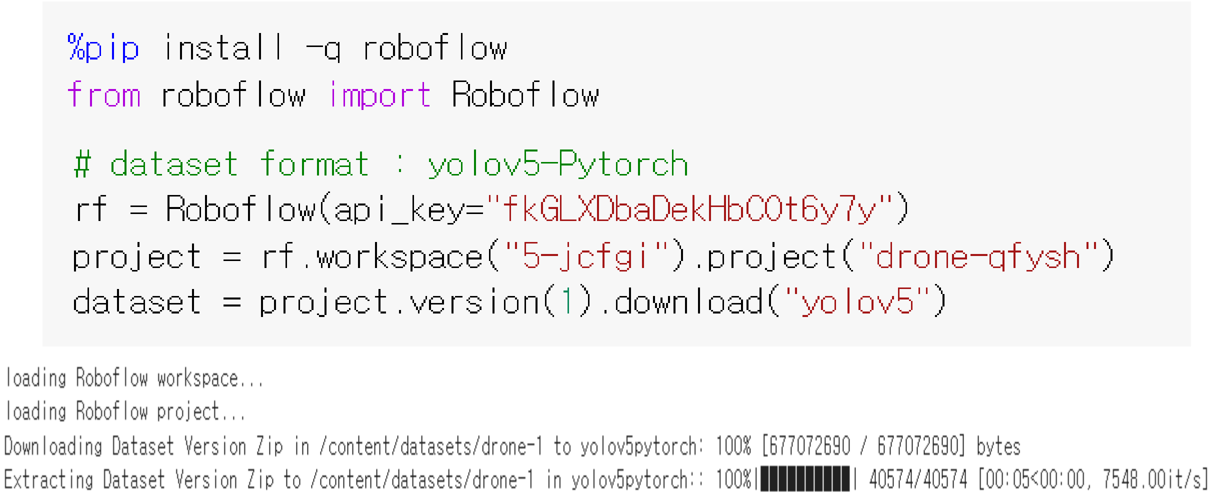
데이터셋 로드하기



Your Download Code

Paste this snippet into a notebook from our model library » to download and unzip your dataset »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("5-jcfgi").project("drone-qfysh")
dataset = project.version(2).download("yolov5")
```



```
%pip install -q roboflow
from roboflow import Roboflow

# dataset format : yolov5-Pytorch
rf = Roboflow(api_key="fkGLXDbaDekHbC0t6y7y")
project = rf.workspace("5-jcfgi").project("drone-qfysh")
dataset = project.version(1).download("yolov5")
```

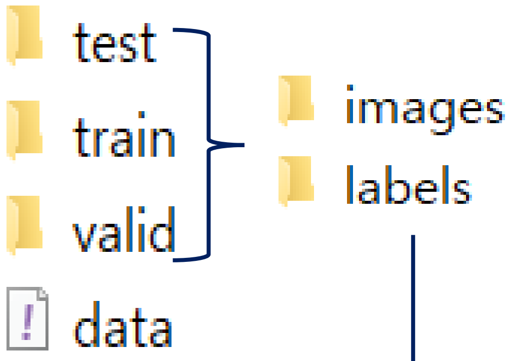
```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in /content/datasets/drone-1 to yolov5pytorch: 100% [677072690 / 677072690] bytes
Extracting Dataset Version Zip to /content/datasets/drone-1 in yolov5pytorch:: 100%|██████████| 40574/40574 [00:05<00:00, 7548.00it/s]
```

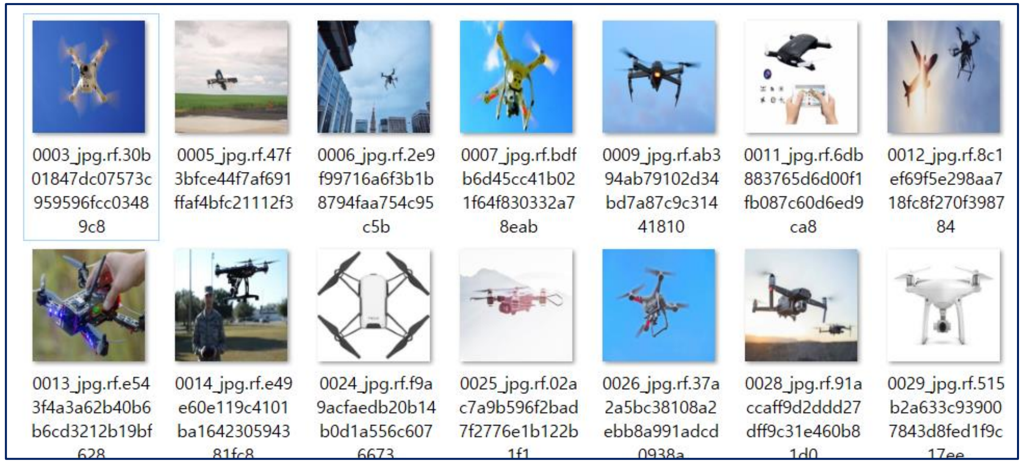
Roboflow에서 생성한 데이터셋 다운로드

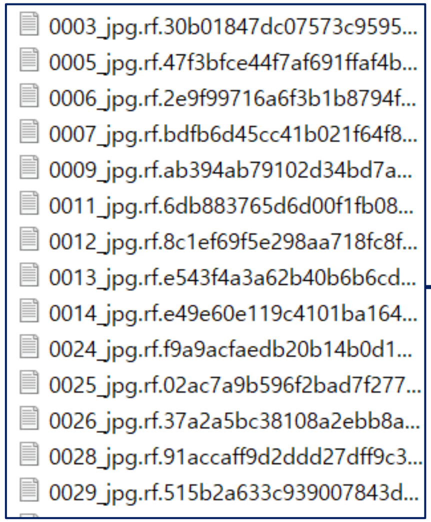
모델 학습

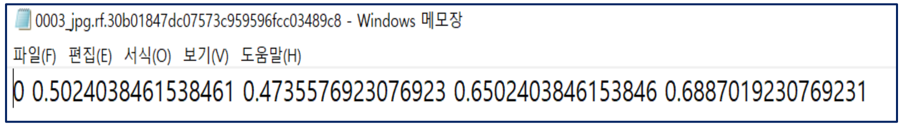
Drone Detection

데이터셋 로드하기




0003_jpg.rf.30b01847dc07573c959596fcc03489c8
0005_jpg.rf.47f3bfce44f7af691ffaf4bfc21112f3
0006_jpg.rf.2e9f99716a6f3b1b8794faa754c95c5b
0007_jpg.rf.bdfb6d45cc41b021f64f830332a78eab
0009_jpg.rf.ab394ab79102d34bd7a87c9c31441810
0011_jpg.rf.6db883765d6d00f1fb087c60d6ed9ca8
0012_jpg.rf.8c1ef69f5e298aa718fc8f270f398784
0013_jpg.rf.e543f4a3a62b40b6b6cd3212b19bf628
0014_jpg.rf.e49e60e119c4101ba164230594381fc8
0024_jpg.rf.f9a9acfaedb20b14b0d1a556c6076673
0025_jpg.rf.02ac7a9b596f2bad7f2776e1b122b1f1
0026_jpg.rf.37a2a5bc38108a2ebb8a991adcd0938a
0028_jpg.rf.91accaff9d2ddd27dff9c31e460b81d0
0029_jpg.rf.515b2a633c939007843d8fed1f9c17ee


0003_jpg.rf.30b01847dc07573c9595...
0005_jpg.rf.47f3bfce44f7af691ffaf4b...
0006_jpg.rf.2e9f99716a6f3b1b8794f...
0007_jpg.rf.bdfb6d45cc41b021f64f8...
0009_jpg.rf.ab394ab79102d34bd7a...
0011_jpg.rf.6db883765d6d00f1fb08...
0012_jpg.rf.8c1ef69f5e298aa718fc8f...
0013_jpg.rf.e543f4a3a62b40b6b6cd...
0014_jpg.rf.e49e60e119c4101ba164...
0024_jpg.rf.f9a9acfaedb20b14b0d1...
0025_jpg.rf.02ac7a9b596f2bad7f277...
0026_jpg.rf.37a2a5bc38108a2ebb8a...
0028_jpg.rf.91accaff9d2ddd27dff9c3...
0029_jpg.rf.515b2a633c939007843d...

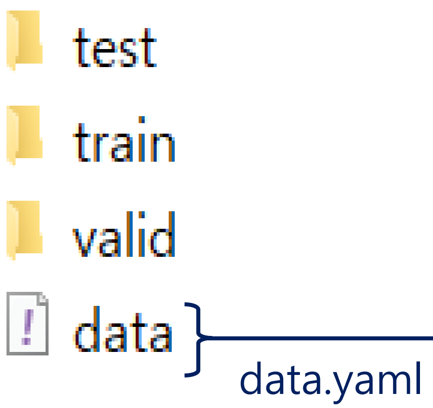

0003_jpg.rf.30b01847dc07573c959596fcc03489c8 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

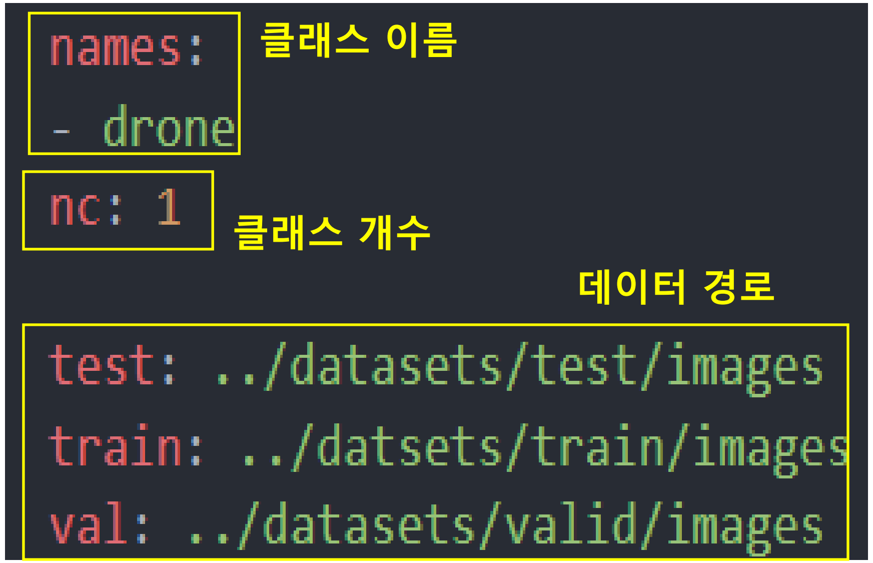
```
0 0.5024038461538461 0.4735576923076923 0.6502403846153846 0.6887019230769231
```

모델 학습

Drone Detection

데이터셋 로드하기





```
names:
- drone
nc: 1
test: ../datasets/test/images
train: ../datsets/train/images
val: ../datasets/valid/images
```

클래스 이름

클래스 개수

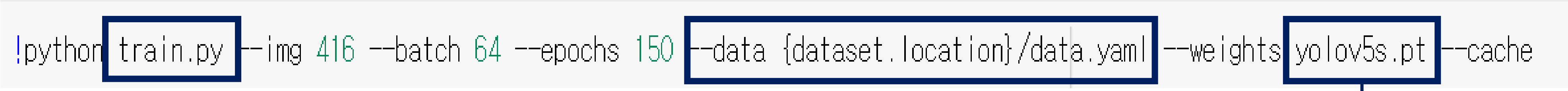
데이터 경로

모델 학습

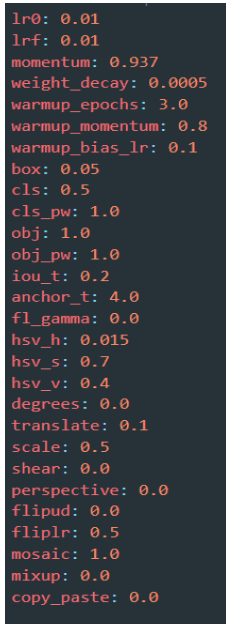
Drone Detection

모델로 학습하기

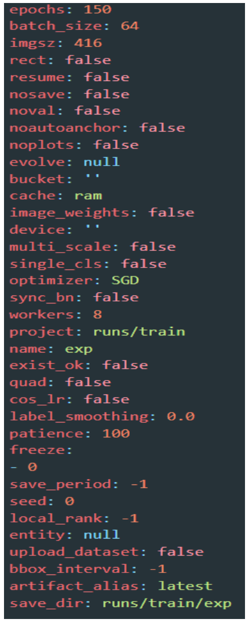
yolov5s 모델로 train data 학습시키기



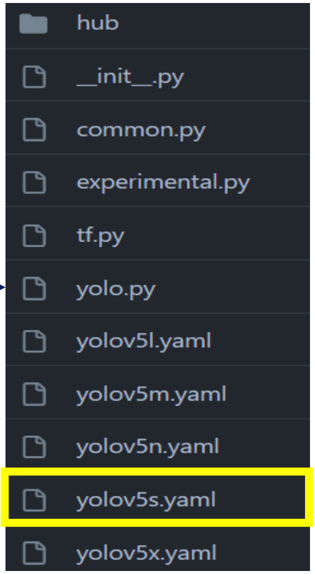
```
!python train.py --img 416 --batch 64 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
```



```
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.5
cls_pw: 1.0
obj: 1.0
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.1
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.0
copy_paste: 0.0
```



```
epochs: 150
batch_size: 64
imgsz: 416
rect: false
resume: false
nosave: false
noval: false
noautoanchor: false
noplots: false
evolve: null
bucket: ''
cache: ram
image_weights: false
device: ''
multi_scale: false
single_cls: false
optimizer: SGD
sync_bn: false
workers: 8
project: runs/train
name: exp
exist_ok: false
quad: false
cos_lr: false
label_smoothing: 0.0
patience: 100
freeze:
- 0
save_period: -1
seed: 0
local_rank: -1
entity: null
upload_dataset: false
bbox_interval: -1
artifact_alias: latest
save_dir: runs/train/exp
```

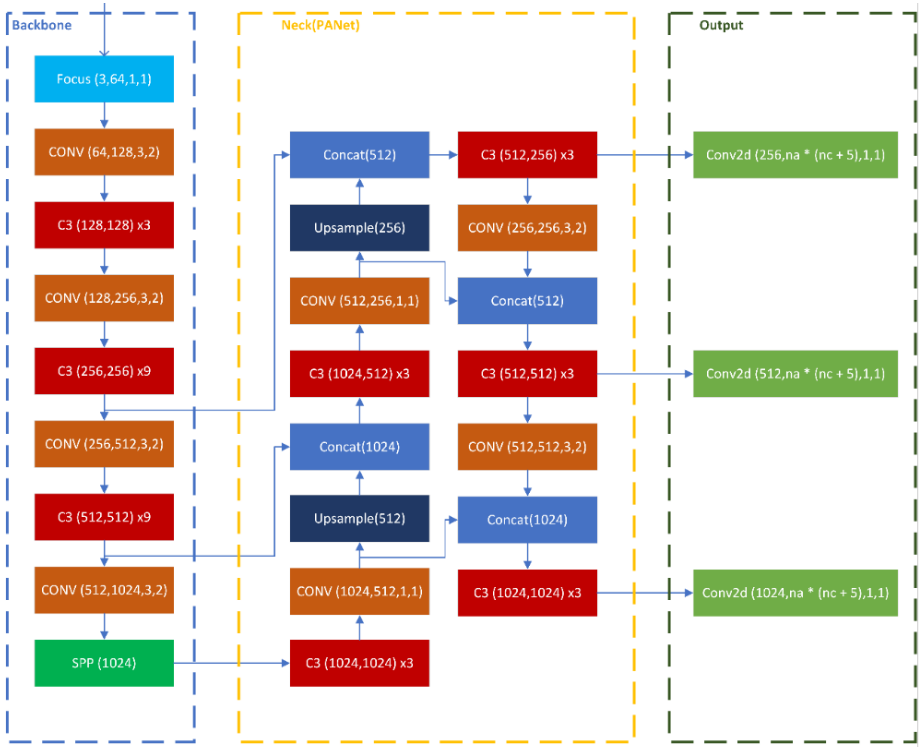


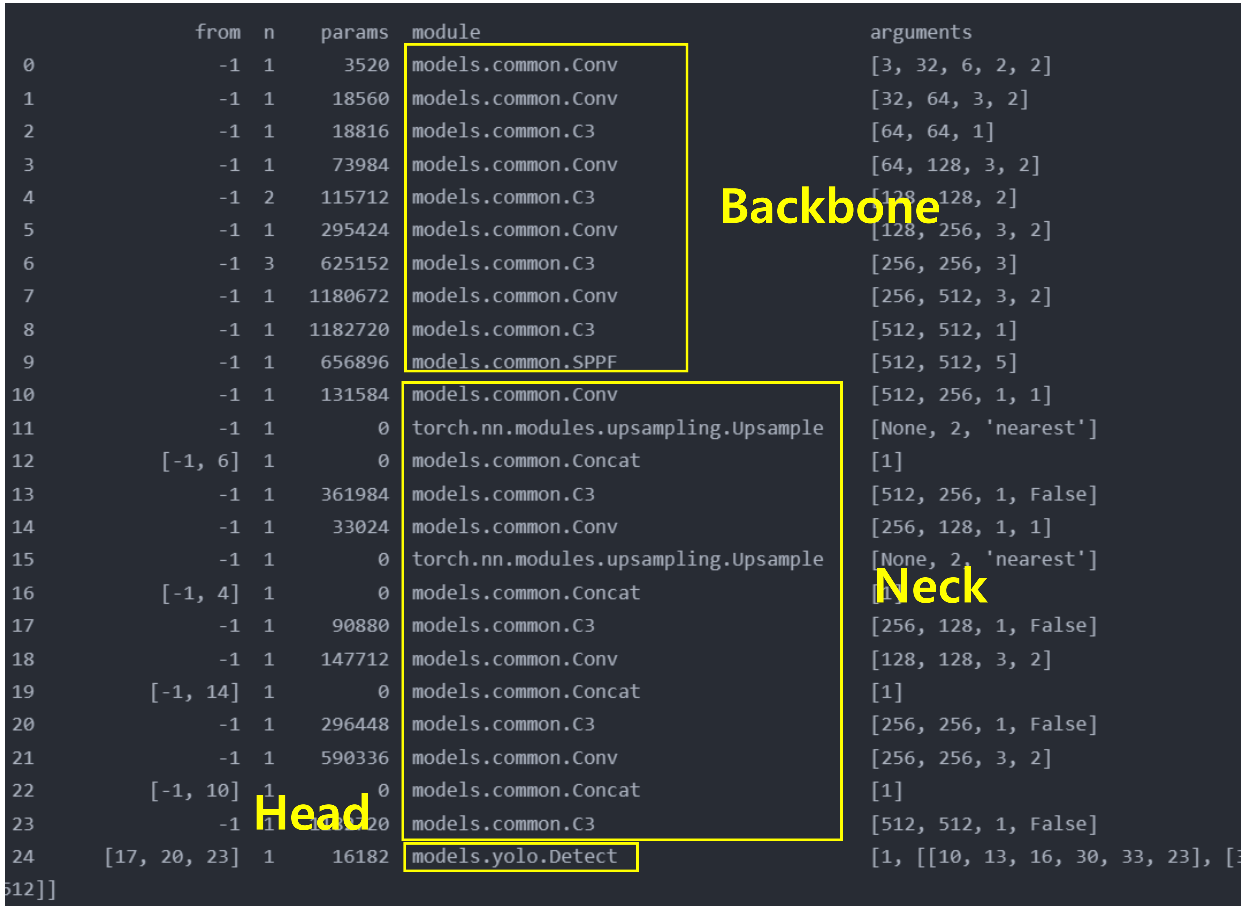
```
hub
__init__.py
common.py
experimental.py
tf.py
yolo.py
yolov5l.yaml
yolov5m.yaml
yolov5n.yaml
yolov5s.yaml
yolov5x.yaml
```

모델 학습

Drone Detection

모델로 학습하기





```
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1, [[10, 13, 16, 30, 33, 23], [
512]]
```

Backbone

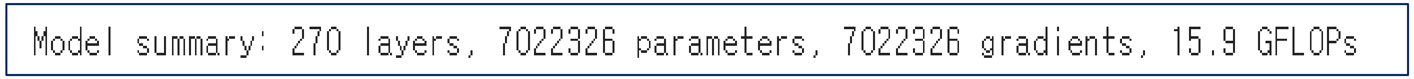
Neck

Head

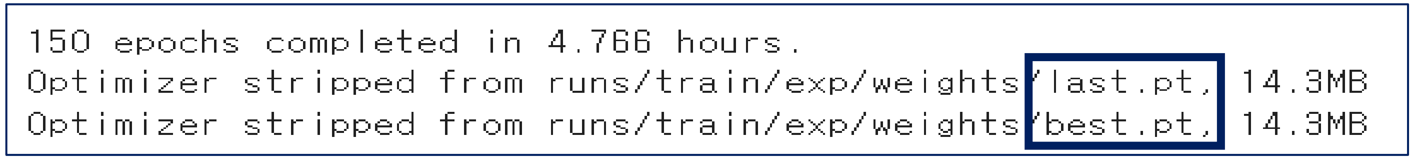
모델 학습

Drone Detection

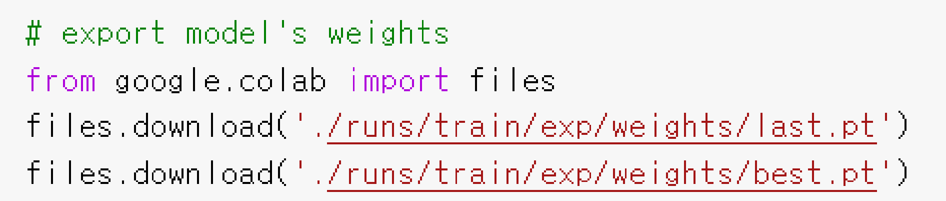
모델로 학습하기



```
Model summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.9 GFLOPs
```



```
150 epochs completed in 4.766 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB
```



```
# export model's weights
from google.colab import files
files.download('./runs/train/exp/weights/last.pt')
files.download('./runs/train/exp/weights/best.pt')
```

train의 마지막 epoch 가중치(last.pt)를 저장

train 중에 가장 좋게 기록된 epoch 가중치(best.pt)를 저장

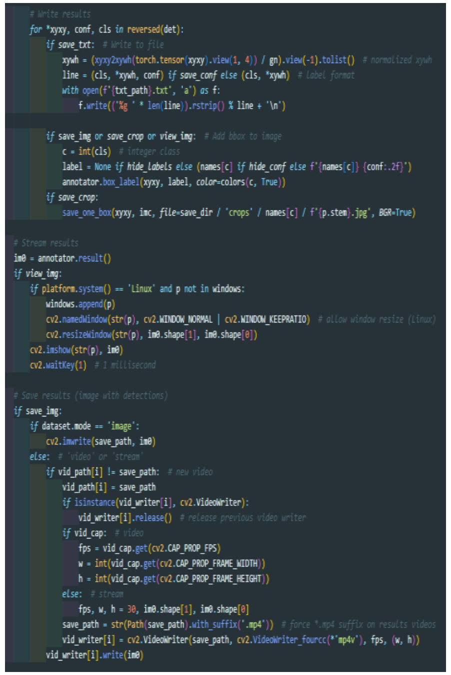
모델 학습

Drone Detection

학습결과로 추론하기

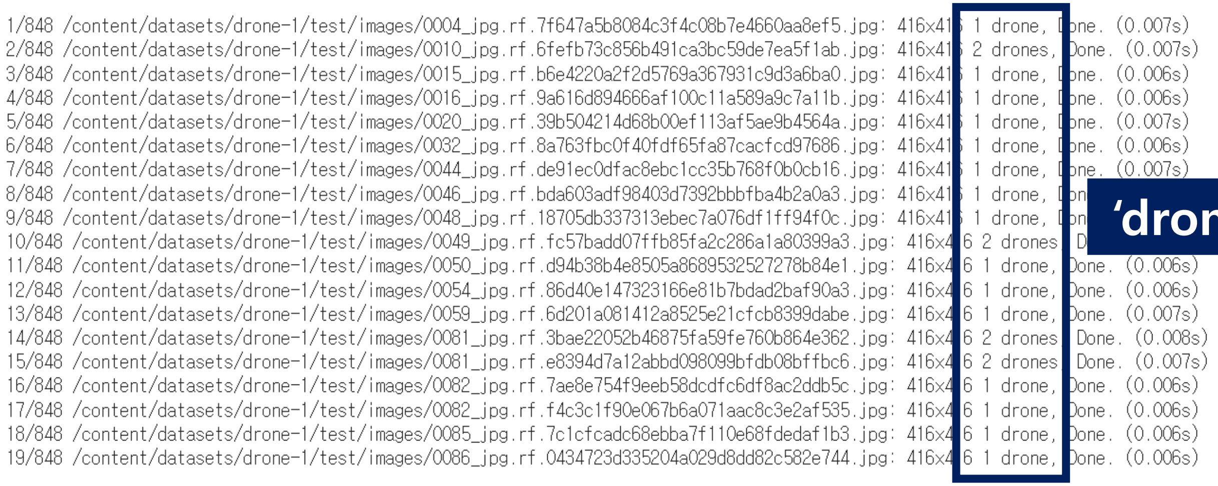
best.pt

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --data=/content/datasets/drone-1/data.yaml --source {dataset.location}/test/images
```



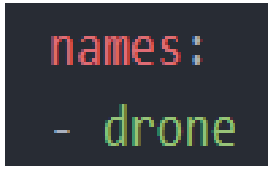
```
    # Write results
    for *xyxy, conf, cls in reversed(det):
        if save_txt:  # Write to file
            xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized xywh
            line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
            with open(f'{txt_path}.txt', 'a') as f:
                f.write(('%g ' * len(line)).rstrip() % line + '\n')

        if save_img or save_crop or view_img:  # Add bbox to image
            c = int(cls)  # integer class
            label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
            annotator.box_label(xyxy, label, color=colors(c, True))
        if save_crop:
            save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)

# Stream results
im0 = annotator.result()
if view_img:
    if platform.system() == 'Linux' and p not in windows:
        windows.append(p)
        cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
        cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
    cv2.imshow(str(p), im0)
    cv2.waitKey(1)  # 1 millisecond

# Save results (image with detections)
if save_img:
    if dataset.mode == 'image':
        cv2.imwrite(save_path, im0)
    else:  # 'video' or 'stream'
        if vid_path[i] != save_path:  # new video
            vid_path[i] = save_path
            if isinstance(vid_writer[i], cv2.VideoWriter):
                vid_writer[i].release()  # release previous video writer
            if vid_cap:  # video
                fps = vid_cap.get(cv2.CAP_PROP_FPS)
                w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            else:  # stream
                fps, w, h = 30, im0.shape[1], im0.shape[0]
            save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on results videos
            vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
        vid_writer[i].write(im0)
```

학습한 결과(best.pt)로 test image 데이터 사용하여 detection 추론하기



```
1/848 /content/datasets/drone-1/test/images/0004_jpg.rf.7f647a5b8084c3f4c08b7e4660aa8ef5.jpg: 416x416 1 drone, Done. (0.007s)
2/848 /content/datasets/drone-1/test/images/0010_jpg.rf.6fefb73c856b491ca3bc59de7ea5f1ab.jpg: 416x416 2 drones, Done. (0.007s)
3/848 /content/datasets/drone-1/test/images/0015_jpg.rf.b6e4220a2f2d5769a367931c9d3a6ba0.jpg: 416x416 1 drone, Done. (0.006s)
4/848 /content/datasets/drone-1/test/images/0016_jpg.rf.9a616d894666af100c11a589a9c7a11b.jpg: 416x416 1 drone, Done. (0.006s)
5/848 /content/datasets/drone-1/test/images/0020_jpg.rf.39b504214d68b00ef113af5ae9b4564a.jpg: 416x416 1 drone, Done. (0.007s)
6/848 /content/datasets/drone-1/test/images/0032_jpg.rf.8a763fbc0f40fdf65fa87cacfcd97686.jpg: 416x416 1 drone, Done. (0.006s)
7/848 /content/datasets/drone-1/test/images/0044_jpg.rf.de91ec0dfac8ebc1cc35b768f0b0cb16.jpg: 416x416 1 drone, Done. (0.007s)
8/848 /content/datasets/drone-1/test/images/0046_jpg.rf.bda603adf98403d7392bbbfba4b2a0a3.jpg: 416x416 1 drone, Don
9/848 /content/datasets/drone-1/test/images/0048_jpg.rf.18705db337313ebec7a076df1ff94f0c.jpg: 416x416 1 drone, Don
10/848 /content/datasets/drone-1/test/images/0049_jpg.rf.fc57badd07ffb85fa2c286a1a80399a3.jpg: 416x416 2 drones, D
11/848 /content/datasets/drone-1/test/images/0050_jpg.rf.d94b38b4e8505a8689532527278b84e1.jpg: 416x416 1 drone, Done. (0.006s)
12/848 /content/datasets/drone-1/test/images/0054_jpg.rf.86d40e147323166e81b7bdad2baf90a3.jpg: 416x416 1 drone, Done. (0.006s)
13/848 /content/datasets/drone-1/test/images/0059_jpg.rf.6d201a081412a8525e21cfcb8399dabe.jpg: 416x416 1 drone, Done. (0.007s)
14/848 /content/datasets/drone-1/test/images/0081_jpg.rf.3bae22052b46875fa59fe760b864e362.jpg: 416x416 2 drones, Done. (0.008s)
15/848 /content/datasets/drone-1/test/images/0081_jpg.rf.e8394d7a12abbd098099bfdb08bffbc6.jpg: 416x416 2 drones, Done. (0.007s)
16/848 /content/datasets/drone-1/test/images/0082_jpg.rf.7ae8e754f9eeb58dcdfc6df8ac2ddb5c.jpg: 416x416 1 drone, Done. (0.006s)
17/848 /content/datasets/drone-1/test/images/0082_jpg.rf.f4c3c1f90e067b6a071aac8c3e2af535.jpg: 416x416 1 drone, Done. (0.006s)
18/848 /content/datasets/drone-1/test/images/0085_jpg.rf.7c1cfcadc68ebba7f110e68fdedaf1b3.jpg: 416x416 1 drone, Done. (0.006s)
19/848 /content/datasets/drone-1/test/images/0086_jpg.rf.0434723d335204a029d8dd82c582e744.jpg: 416x416 1 drone, Done. (0.006s)
```

data.yaml



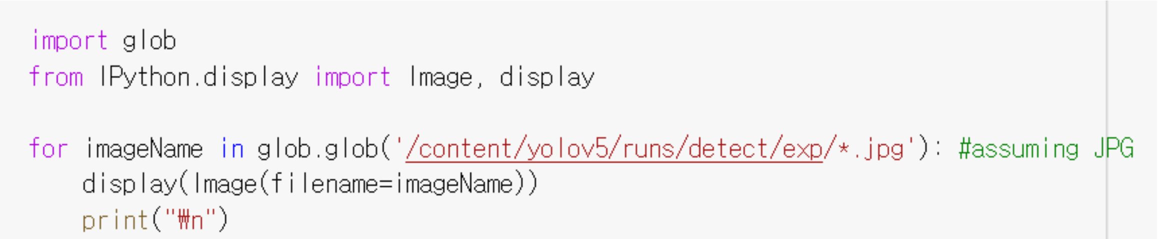
```
names:
  - drone
```

‘drone’으로 추론

모델 학습

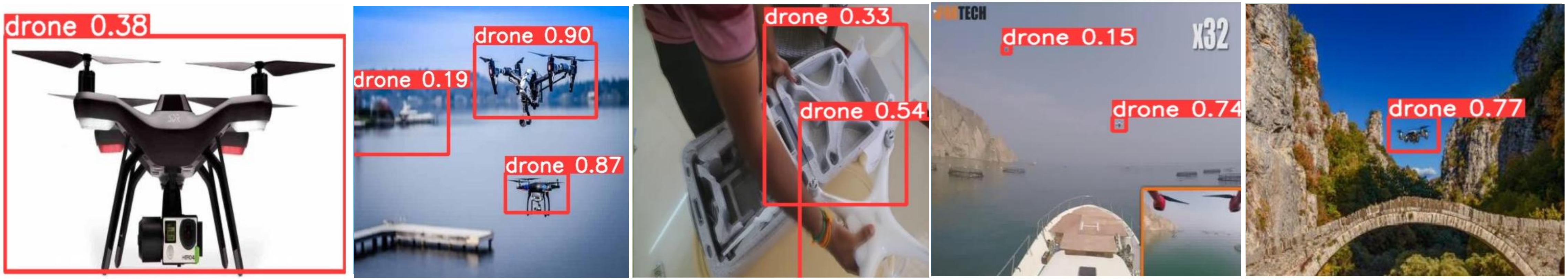
Drone Detection

추론 결과 확인



```
import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("₩n")
```

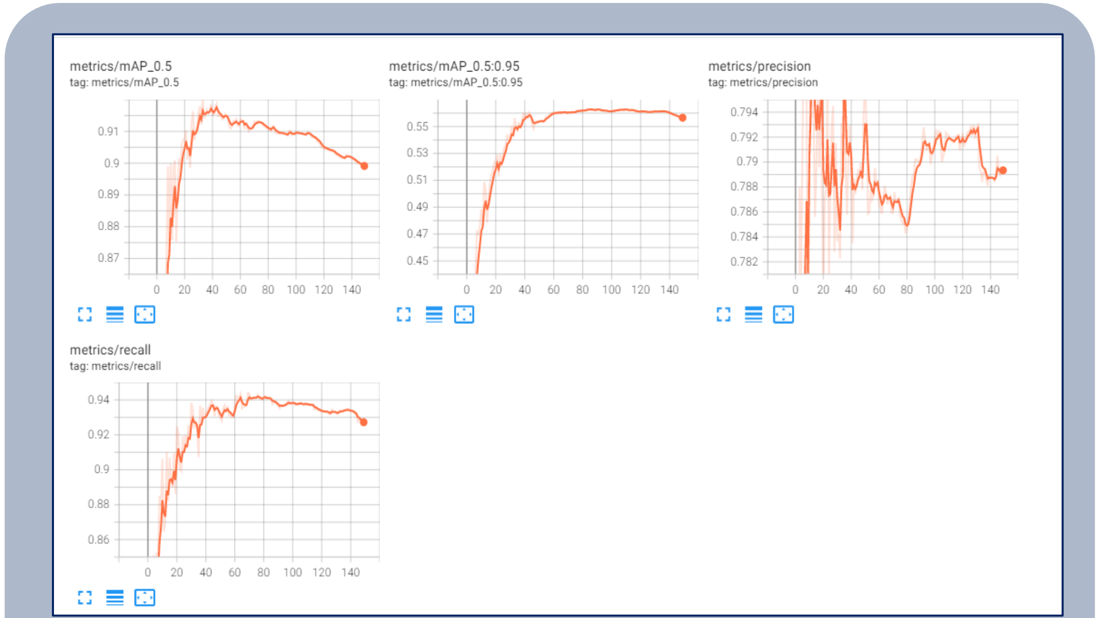
추론 결과 시각화



완벽하진 않지만, 그래도 **drone detection**이 가능해짐!!!

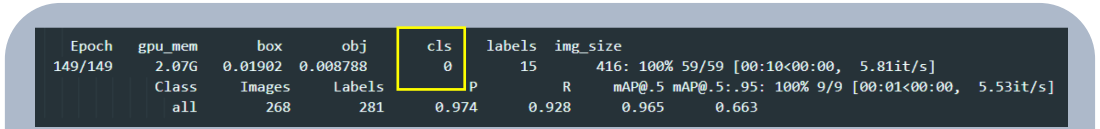
결과

Drone Detection

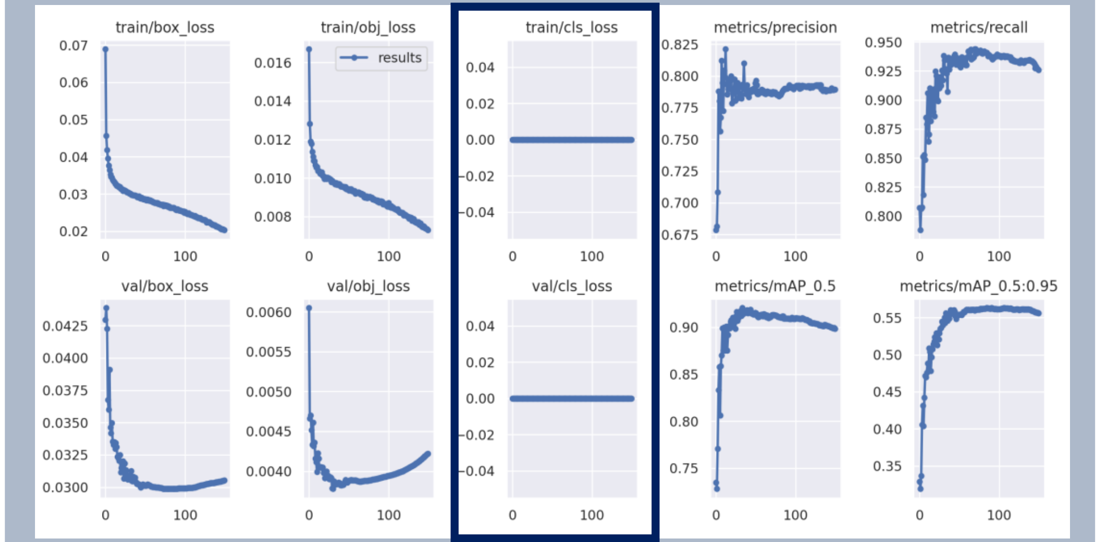


Tensorboard 시각화

mAP : 객체 탐지 **정확도** 평가 지표



```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
   149/149     2.07G   0.01902  0.008788         0        15       416: 100% 59/59 [00:10<00:00,  5.81it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 9/9 [00:01<00:00,  5.53it/s]
                 all        268        281      0.974      0.928      0.965      0.663
```

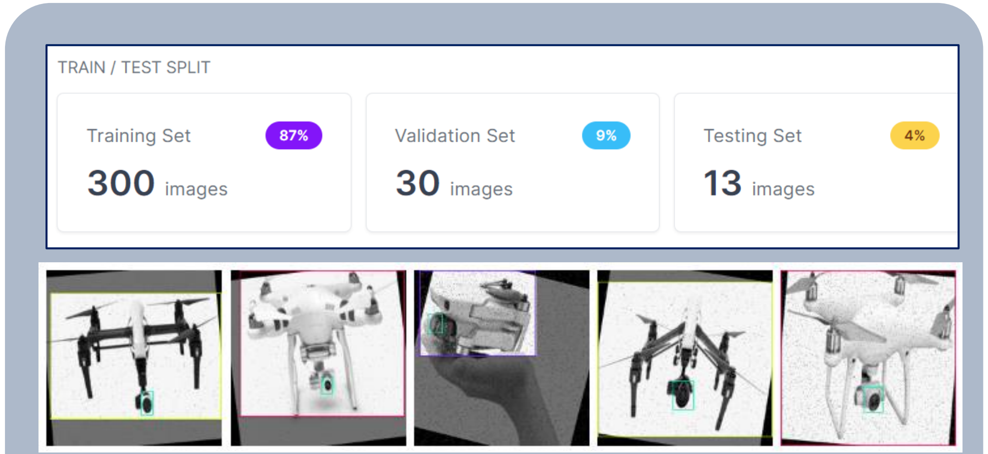


학습함에 따라 전체적으로 loss는 감소하고 정확도가 증가

모델 학습

Drone Classificatioin

데이터 로드하기

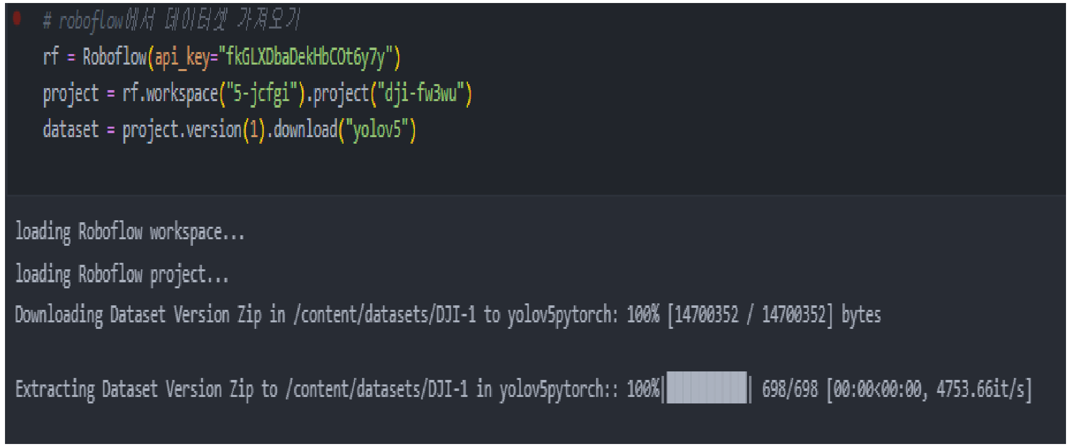


TRAIN / TEST SPLIT

Training Set 87%	Validation Set 9%	Testing Set 4%
300 images	30 images	13 images

class : 4
[DJI Phantom, DJI Mavic, DJI Inspire, camera]

EXPORT



```
# roboflow에서 데이터셋 가져오기
rf = Roboflow(api_key="fkGLXDbaDekHbCOt6y7y")
project = rf.workspace("5-jcfgi").project("dji-fw3wu")
dataset = project.version(1).download("yolov5")
```

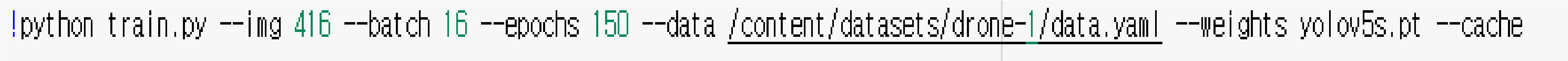
```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in /content/datasets/DJI-1 to yolov5pytorch: 100% [14700352 / 14700352] bytes

Extracting Dataset Version Zip to /content/datasets/DJI-1 in yolov5pytorch:: 100%|██████████| 698/698 [00:00<00:00, 4753.66it/s]
```

Roboflow에서 생성한 classification 데이터셋 다운로드

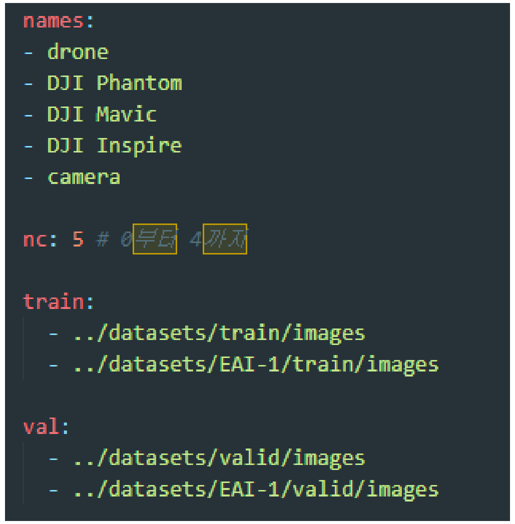
모델 학습

Drone Classificatioin

모델 학습하기



```
!python train.py --img 416 --batch 16 --epochs 150 --data /content/datasets/drone-1/data.yaml --weights yolov5s.pt --cache
```



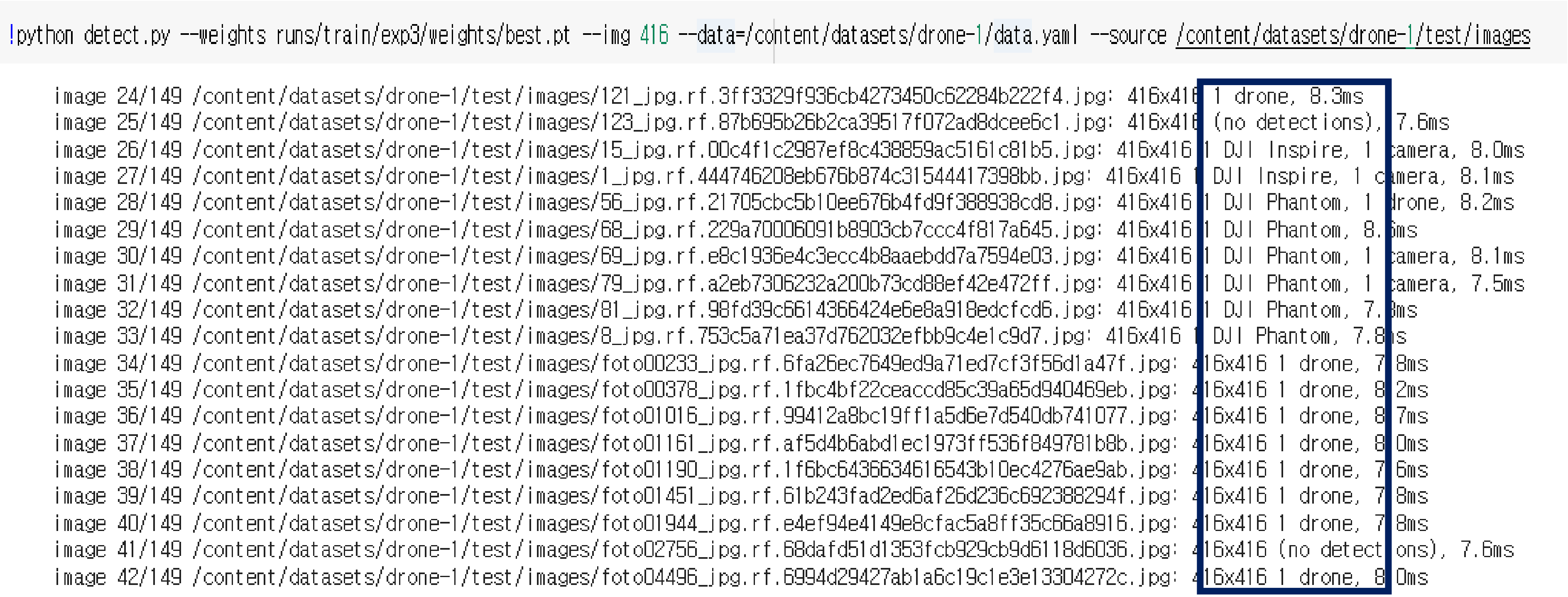
```
names:
- drone
- DJI Phantom
- DJI Mavic
- DJI Inspire
- camera

nc: 5 # 0부터 4까지

train:
  - ../datasets/train/images
  - ../datasets/EAI-1/train/images

val:
  - ../datasets/valid/images
  - ../datasets/EAI-1/valid/images
```

Detection에 사용한 데이터셋과 merge하여 train data 학습시키기

모델 학습

Drone Classificatioin

모델로 추론하기



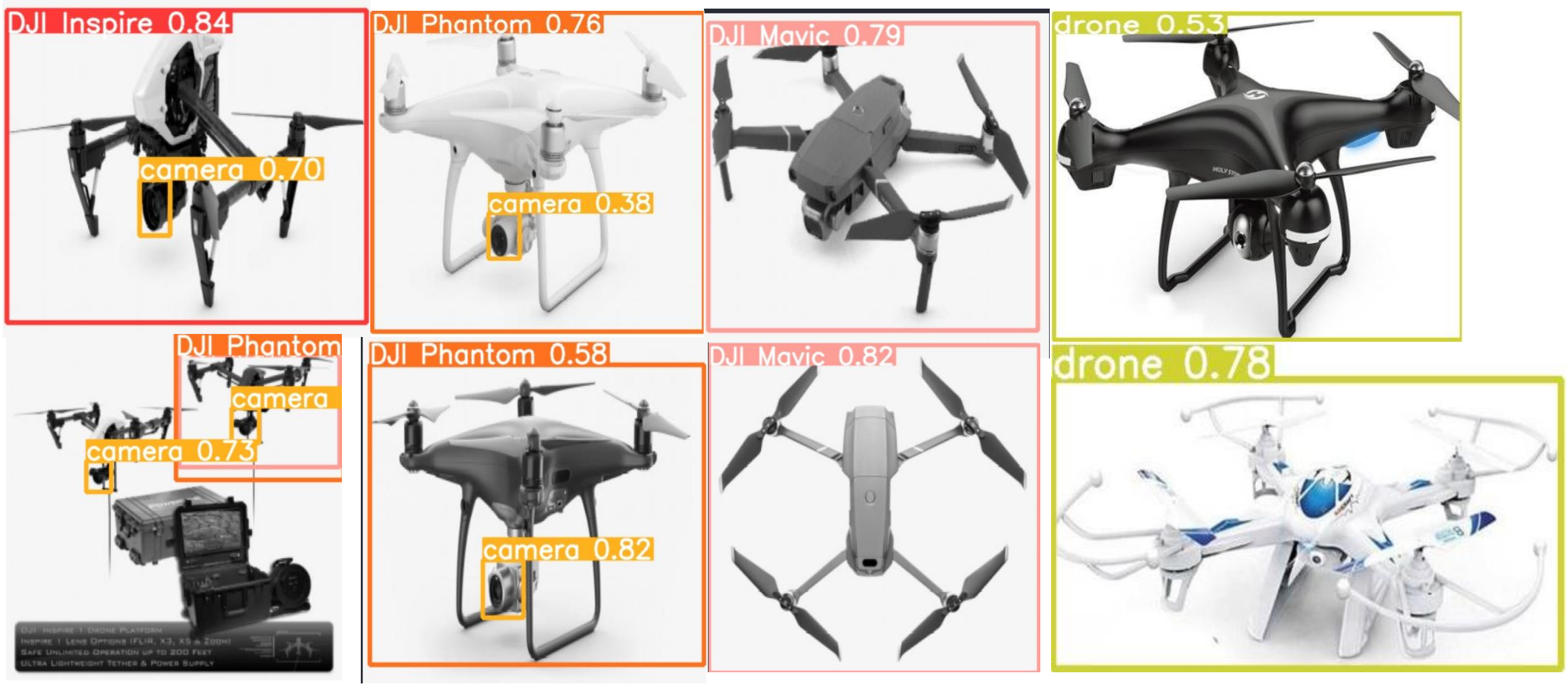
```
!python detect.py --weights runs/train/exp3/weights/best.pt --img 416 --data=/content/datasets/drone-1/data.yaml --source /content/datasets/drone-1/test/images
```

```
image 24/149 /content/datasets/drone-1/test/images/121_jpg.rf.3ff3329f936cb4273450c62284b222f4.jpg: 416x416 1 drone, 8.3ms
image 25/149 /content/datasets/drone-1/test/images/123_jpg.rf.87b695b26b2ca39517f072ad8dcee6c1.jpg: 416x416 (no detections), 7.6ms
image 26/149 /content/datasets/drone-1/test/images/15_jpg.rf.00c4f1c2987ef8c438859ac5161c81b5.jpg: 416x416 1 DJI Inspire, 1 camera, 8.0ms
image 27/149 /content/datasets/drone-1/test/images/1_jpg.rf.444746208eb676b874c3154441739bb.jpg: 416x416 1 DJI Inspire, 1 camera, 8.1ms
image 28/149 /content/datasets/drone-1/test/images/56_jpg.rf.21705cbc5b10ee676b4fd9f388938cd8.jpg: 416x416 1 DJI Phantom, 1 drone, 8.2ms
image 29/149 /content/datasets/drone-1/test/images/68_jpg.rf.229a70006091b8903cb7ccc4f817a645.jpg: 416x416 1 DJI Phantom, 8.6ms
image 30/149 /content/datasets/drone-1/test/images/69_jpg.rf.e8c1936e4c3ecc4b8aaebdd7a7594e03.jpg: 416x416 1 DJI Phantom, 1 camera, 8.1ms
image 31/149 /content/datasets/drone-1/test/images/79_jpg.rf.a2eb7306232a200b73cd88ef42e472ff.jpg: 416x416 1 DJI Phantom, 1 camera, 7.5ms
image 32/149 /content/datasets/drone-1/test/images/81_jpg.rf.98fd39c6614366424e6e8a918edcfcd6.jpg: 416x416 1 DJI Phantom, 7.8ms
image 33/149 /content/datasets/drone-1/test/images/8_jpg.rf.753c5a71ea37d762032efbb9c4e1c9d7.jpg: 416x416 1 DJI Phantom, 7.8ms
image 34/149 /content/datasets/drone-1/test/images/foto00233_jpg.rf.6fa26ec7649ed9a71ed7cf3f56d1a47f.jpg: 416x416 1 drone, 7.8ms
image 35/149 /content/datasets/drone-1/test/images/foto00378_jpg.rf.1fbc4bf22ceaccd85c39a65d940469eb.jpg: 416x416 1 drone, 8.2ms
image 36/149 /content/datasets/drone-1/test/images/foto01016_jpg.rf.99412a8bc19ff1a5d6e7d540db741077.jpg: 416x416 1 drone, 8.7ms
image 37/149 /content/datasets/drone-1/test/images/foto01161_jpg.rf.af5d4b6abd1ec1973ff536f849781b8b.jpg: 416x416 1 drone, 8.0ms
image 38/149 /content/datasets/drone-1/test/images/foto01190_jpg.rf.1f6bc6436634616543b10ec4276ae9ab.jpg: 416x416 1 drone, 7.6ms
image 39/149 /content/datasets/drone-1/test/images/foto01451_jpg.rf.61b243fad2ed6af26d236c692388294f.jpg: 416x416 1 drone, 7.8ms
image 40/149 /content/datasets/drone-1/test/images/foto01944_jpg.rf.e4ef94e4149e8cfac5a8ff35c66a8916.jpg: 416x416 1 drone, 7.8ms
image 41/149 /content/datasets/drone-1/test/images/foto02756_jpg.rf.68dafd51d1353fcb929cb9d6118d6036.jpg: 416x416 (no detections), 7.6ms
image 42/149 /content/datasets/drone-1/test/images/foto04496_jpg.rf.6994d29427ab1a6c19c1e3e13304272c.jpg: 416x416 1 drone, 8.0ms
```

모델 학습

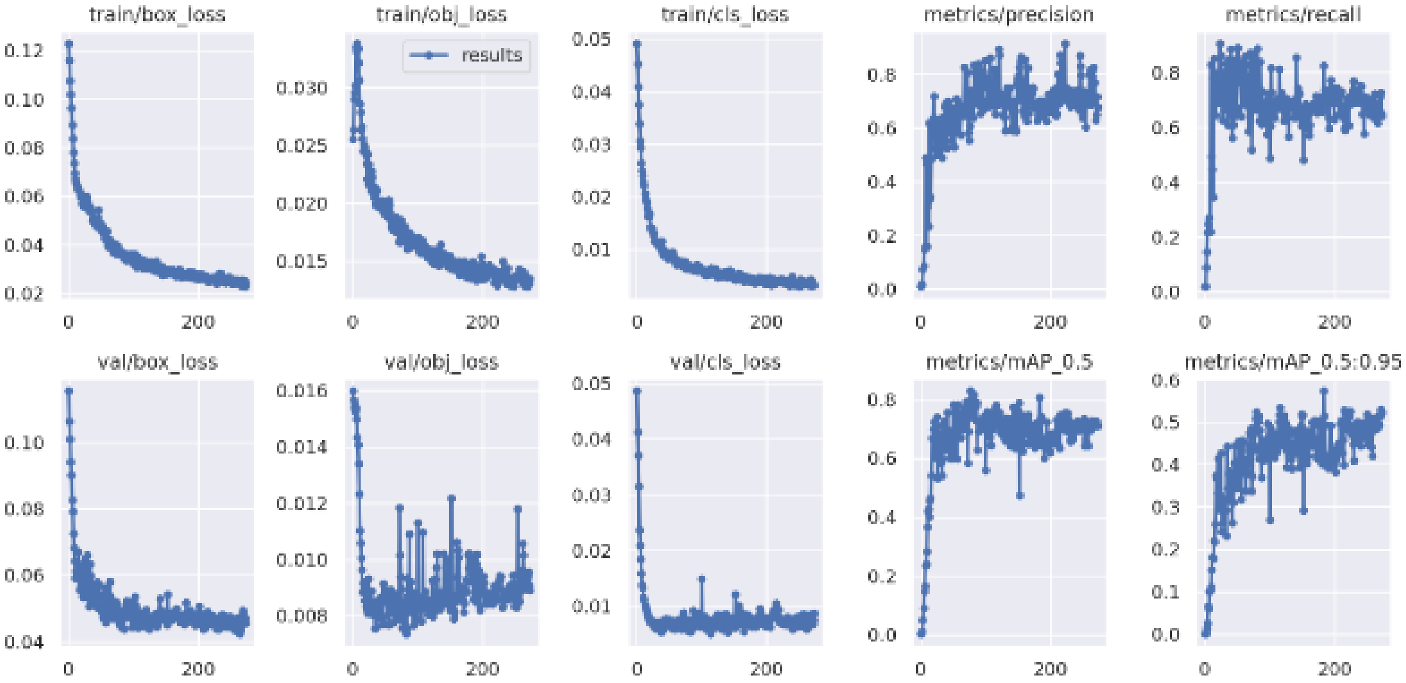
Drone Classificatioin

모델로 추론하기



결과

Drone Classificatioin

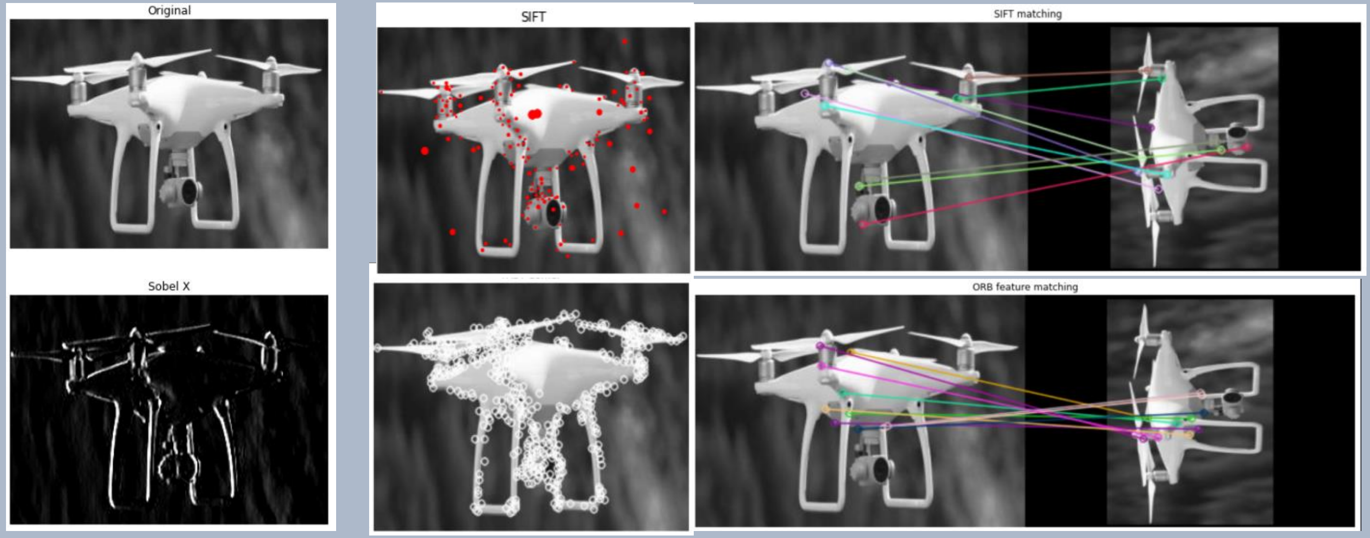


Drone detection에 비해 성능이 떨어짐

Object Tracking

Drone Feature Mapping

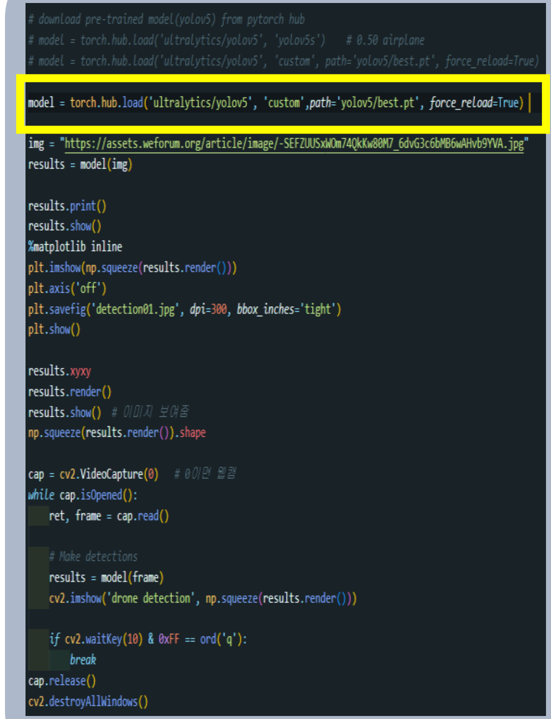
- Harris, SIFT, FAST, ORB : 드론 이미지에서 객체 특징 추출 및 매칭하는 Image Processing



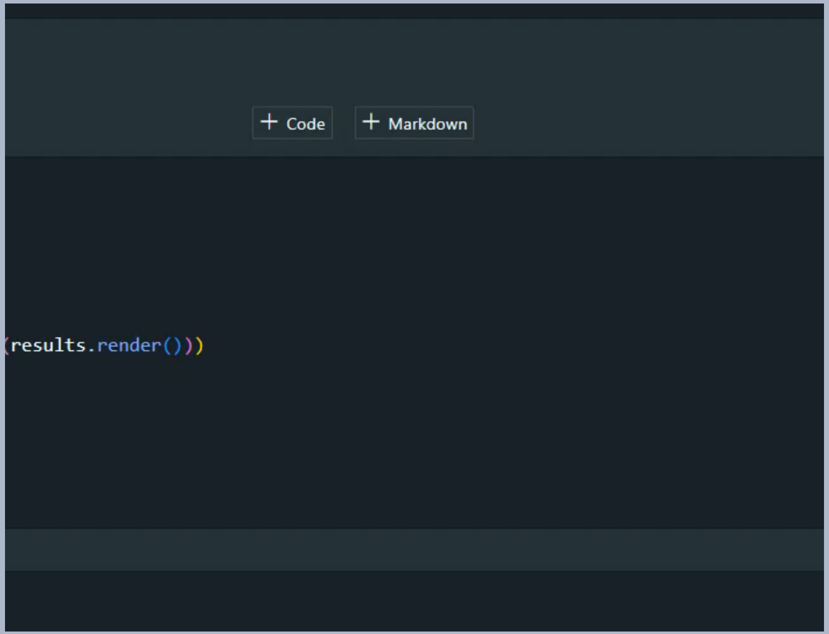
결과

Real-time detection

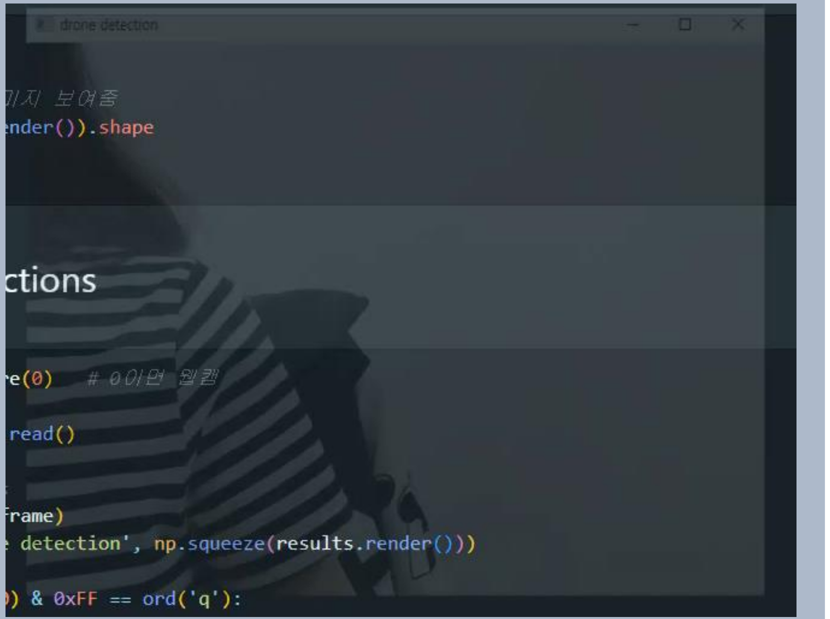
Using webcam



```
# download pre-trained model(yolov5) from pytorch hub
# model = torch.hub.load('ultralytics/yolov5', 'yolov5s')    # 0.50 airplane
# model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/best.pt', force_reload=True)

model = torch.hub.load('ultralytics/yolov5', 'custom',path='yolov5/best.pt', force_reload=True)

img = "https://assets.weforum.org/article/image/-SEFZUUSxWOm74QkKw80M7_6dvG3c6bMB6wAHvb9YVA.jpg"
results = model(img)

results.print()
results.show()
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.axis('off')
plt.savefig('detection01.jpg', dpi=300, bbox_inches='tight')
plt.show()

results.xyxy
results.render()
results.show()  # 이미지 보여줌
np.squeeze(results.render()).shape

cap = cv2.VideoCapture(0)   # 0이면 웹캠
while cap.isOpened():
    ret, frame = cap.read()

    # Make detections
    results = model(frame)
    cv2.imshow('drone detection', np.squeeze(results.render()))

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



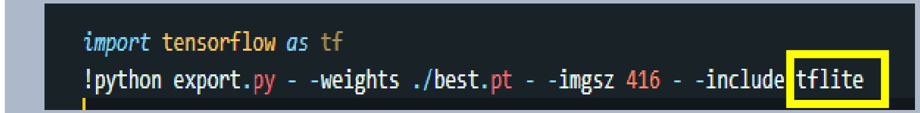
```
(results.render()))
```



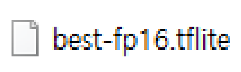
```
미지 보여줌
ender()).shape

ctions

e(0)   # 0이면 웹캠

read()

frame)
 detection', np.squeeze(results.render()))

) & 0xFF == ord('q'):
```

Application

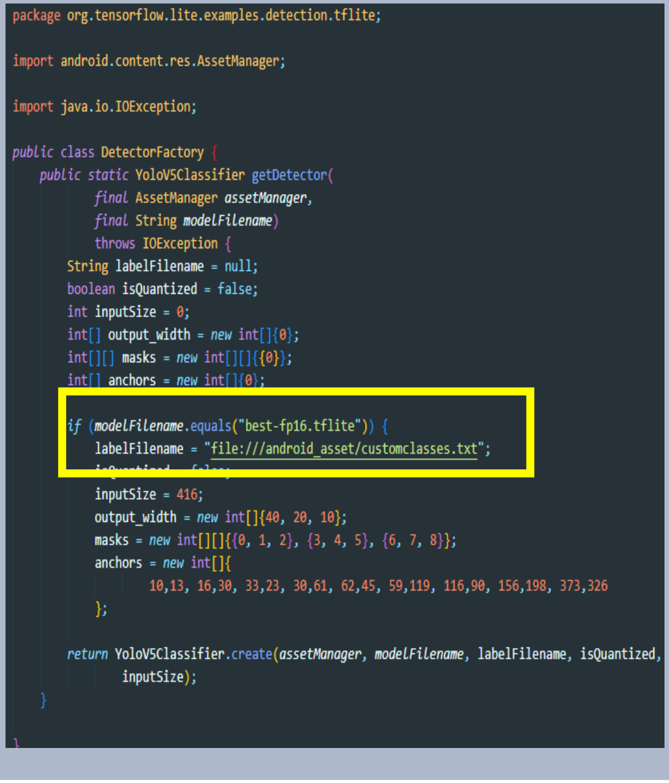
tflite- web



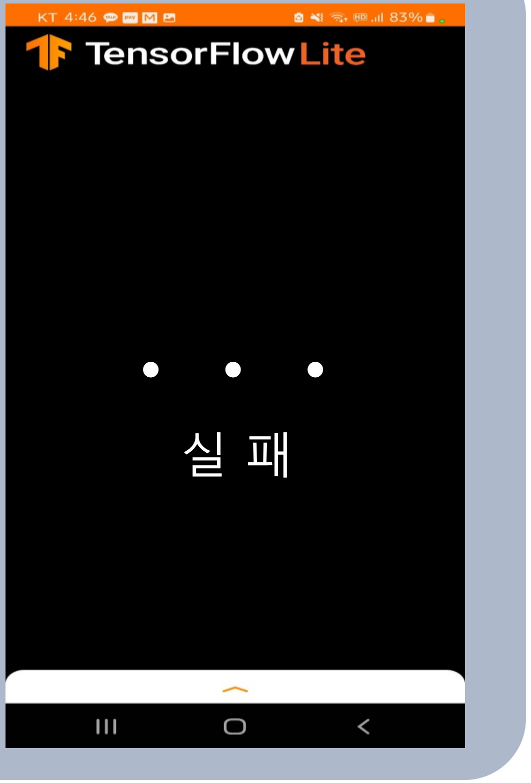
```
import tensorflow as tf
!python export.py - -weights ./best.pt - -imgsz 416 - -include tflite
```



best-fp16.tflite



```
package org.tensorflow.lite.examples.detection.tflite;

import android.content.res.AssetManager;

import java.io.IOException;

public class DetectorFactory {
    public static YoloV5Classifier getDetector(
            final AssetManager assetManager,
            final String modelFilename)
            throws IOException {
        String labelFilename = null;
        boolean isQuantized = false;
        int inputSize = 0;
        int[] output_width = new int[]{0};
        int[][] masks = new int[][]{{0}};
        int[] anchors = new int[]{0};

        if (modelFilename.equals("best-fp16.tflite")) {
            labelFilename = "file:///android_asset/customclasses.txt";
            isQuantized = false;
            inputSize = 416;
            output_width = new int[]{40, 20, 10};
            masks = new int[][]{{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
            anchors = new int[]{
                    10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
            };

        return YoloV5Classifier.create(assetManager, modelFilename, labelFilename, isQuantized,
                inputSize);
    }
}
```

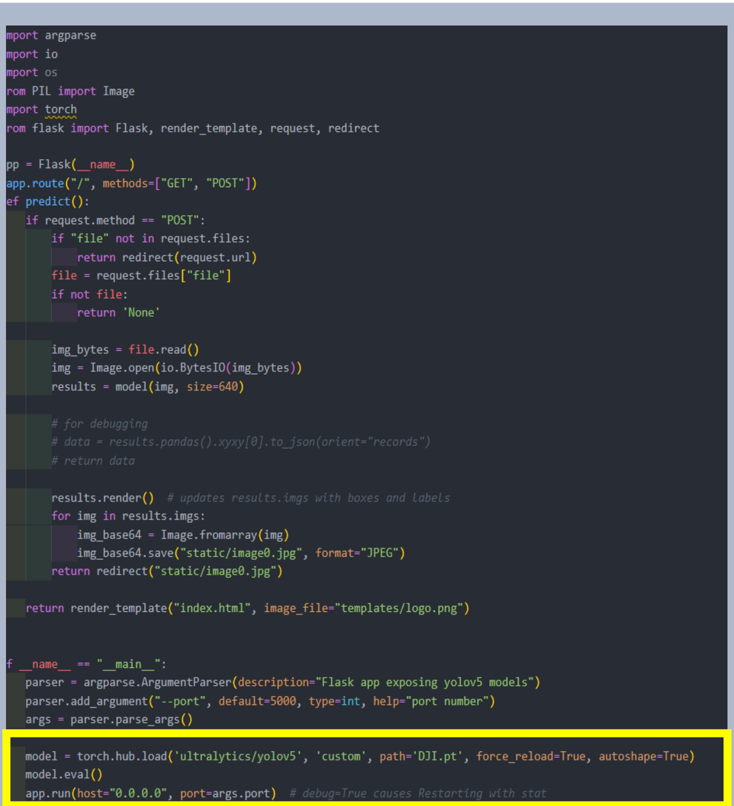


• • •

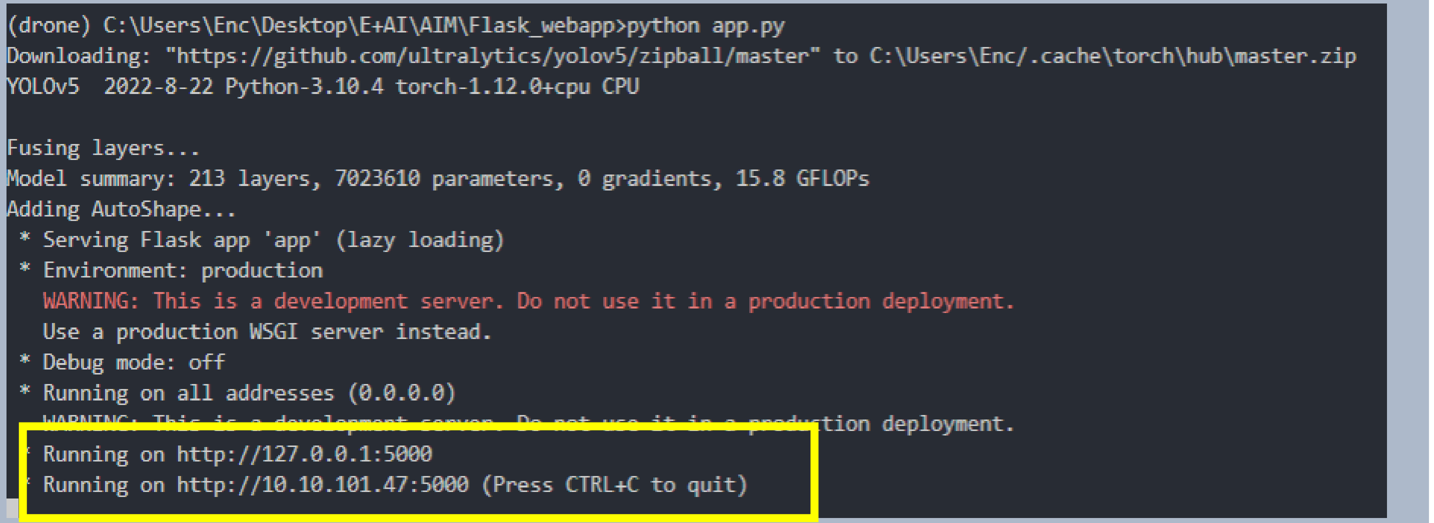
실 패

Application

Flask – web



```
mport argparse
mport io
mport os
rom PIL import Image
mport torch
rom flask import Flask, render_template, request, redirect

pp = Flask(__name__)
app.route("/", methods=["GET", "POST"])
ef predict():
    if request.method == "POST":
        if "file" not in request.files:
            return redirect(request.url)
        file = request.files["file"]
        if not file:
            return 'None'

        img_bytes = file.read()
        img = Image.open(io.BytesIO(img_bytes))
        results = model(img, size=640)

        # for debugging
        # data = results.pandas().xyxy[0].to_json(orient="records")
        # return data

        results.render()  # updates results.imgs with boxes and labels
        for img in results.imgs:
            img_base64 = Image.fromarray(img)
            img_base64.save("static/image0.jpg", format="JPEG")
        return redirect("static/image0.jpg")

    return render_template("index.html", image_file="templates/logo.png")


f __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Flask app exposing yolov5 models")
    parser.add_argument("--port", default=5000, type=int, help="port number")
    args = parser.parse_args()

    model = torch.hub.load('ultralytics/yolov5', 'custom', path='DJI.pt', force_reload=True, autoshape=True)
    model.eval()
    app.run(host="0.0.0.0", port=args.port)  # debug=True causes Restarting with stat
```

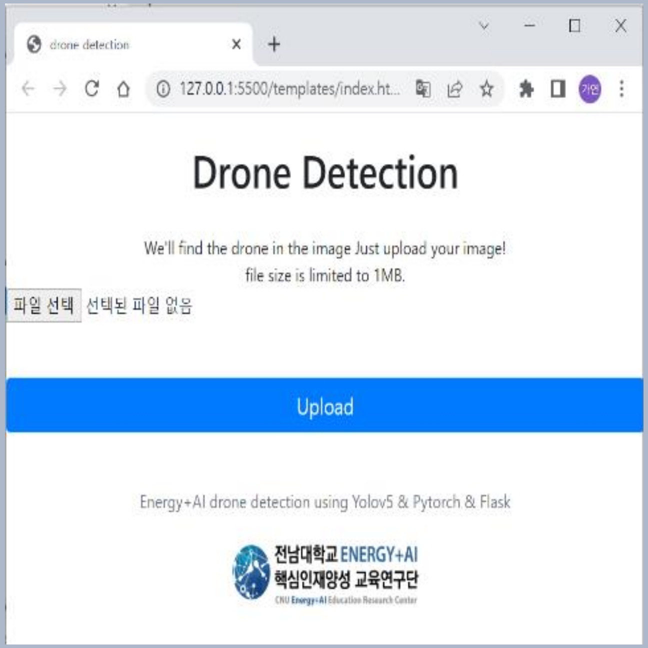


```
(drone) C:\Users\Enc\Desktop\E+AI\AIM\Flask_webapp>python app.py
Downloading: "https://github.com/ultralytics/yolov5/zipball/master" to C:\Users\Enc/.cache\torch\hub\master.zip
YOLOv5  2022-8-22 Python-3.10.4 torch-1.12.0+cpu CPU

Fusing layers...
Model summary: 213 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://10.10.101.47:5000 (Press CTRL+C to quit)
```

Application

전남대학교 ENERGY+AI
핵심인재양성 교육연구단
CNU Energy+AI Education Research Center

Flask - web



drone detection
127.0.0.1:5500/templates/index.ht...

Drone Detection

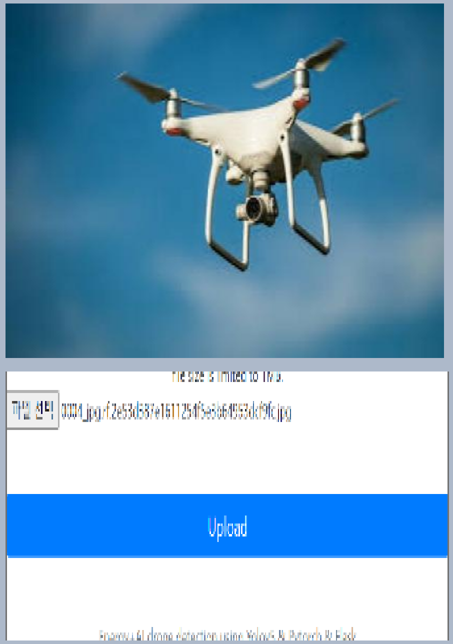
We'll find the drone in the image Just upload your image!
file size is limited to 1MB.

파일 선택 선택된 파일 없음

Upload

Energy+AI drone detection using Yolov5 & Pytorch & Flask

Main 화면



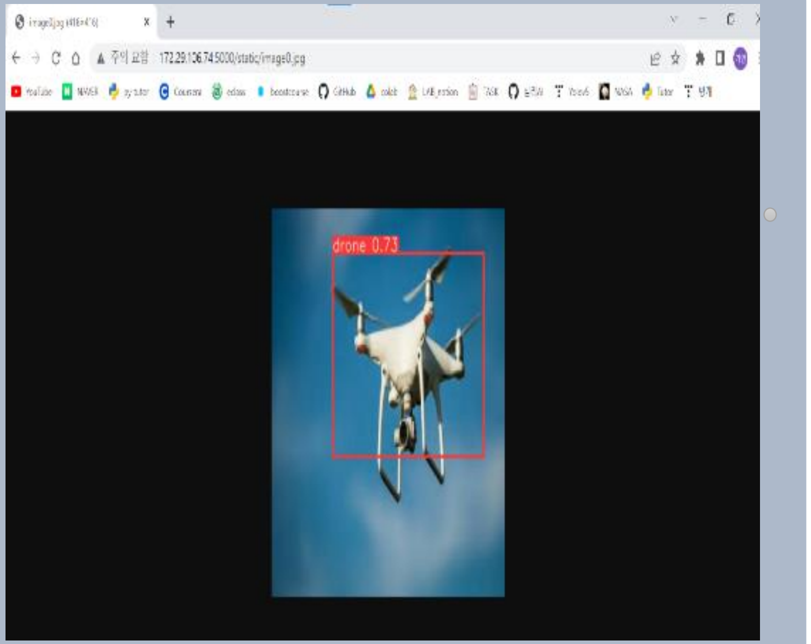
file size is limited to 1MB.

파일 선택 0004_jpg.rf.2e53d587e1611254f5e3b64953dcf9fc.jpg

Upload

Energy+AI drone detection using Yolov5 & Pytorch & Flask

Image 파일 업로드



Detection 결과 출력

결론

YOLO 모델로 드론 이미지 및 영상데이터를 학습시켜
객체를 감지하고 종류 구분이 가능한 프로토타입 AI모델을 제작함

학습한 모델(best.pt)을 바탕으로
flask를 활용한 detection web-애플리케이션에 적용함

충분하지 않는 데이터셋, GPU 용량 문제, 부족한 지식 등으로
인한 다양한 모델 적용 및 학습 시 튜닝시도 부족

Reference

- https://github.com/ultralytics/yolov5
- Unauthorized Unmanned Aerial Vehicle Detection using YOLOv5 and Transfer Learning, doi:10.20944/preprints202202.0185.v1
- https://github.com/AarohiSingla/TFLite-Object-Detection-Android-App-Tutorial-Using-YOLOv5
- https://github.com/ViAsmit/YOLOv5-Flask
- YOLOv4: Optimal Speed and Accuracy of Object Detection
- https://www.youtube.com/watch?v=yqkISICHH-U
- https://www.youtube.com/watch?v=WQeoO7MI0Bs
- https://www.youtube.com/results?search_query=yolov5+tflite
- https://www.tensorflow.org/lite/guide?hl=ko

감사합니다

전남대학교 ENERGY+AI
핵심인재양성 교육연구단
CNU Energy+AI Education Research Center

전남대학교
CHONNAM NATIONAL UNIVERSITY

AIM

김주완(인공지능학부)
서가연(인공지능학부)
조하영(인공지능학부)