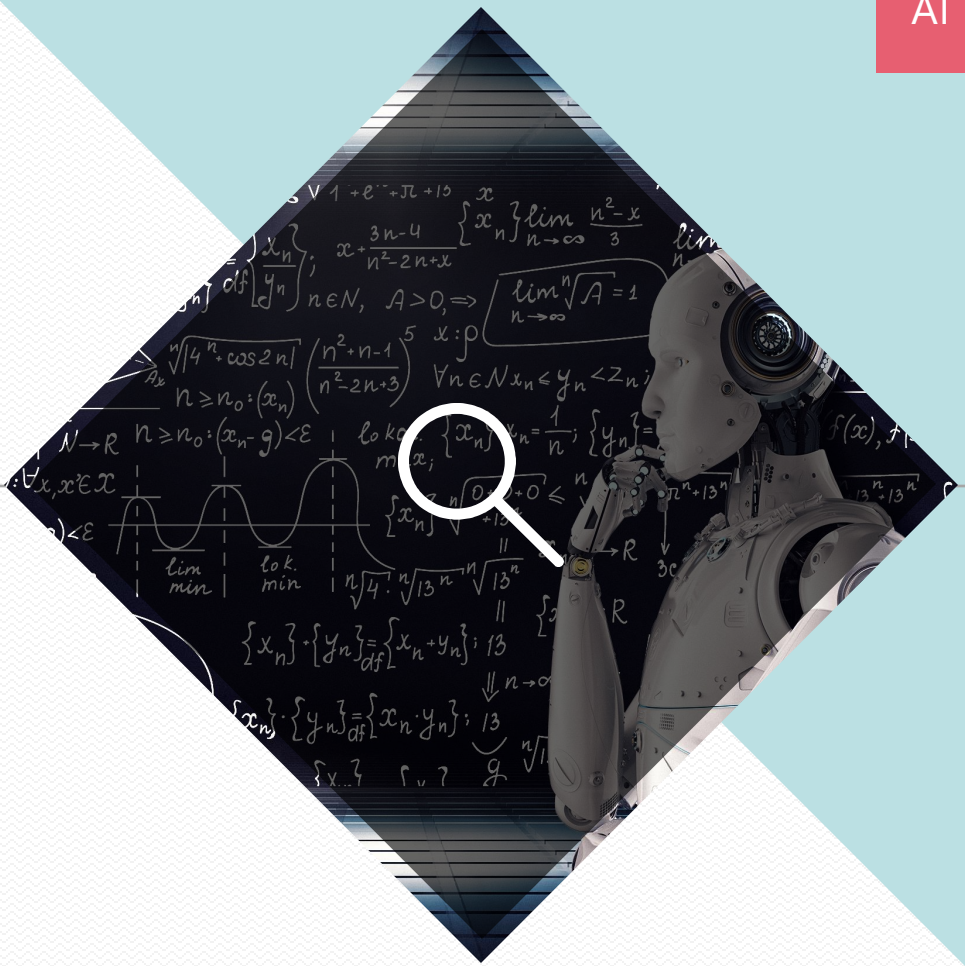
AI TRAINING SERIES



YOLO실습

CONTENTS

YOLOv5

Yolo PRACTICE

YOLOv5설치

▼ Install

Clone repo and install requirements.txt in a Python>=3.8.0 environment, including PyTorch>=1.8.

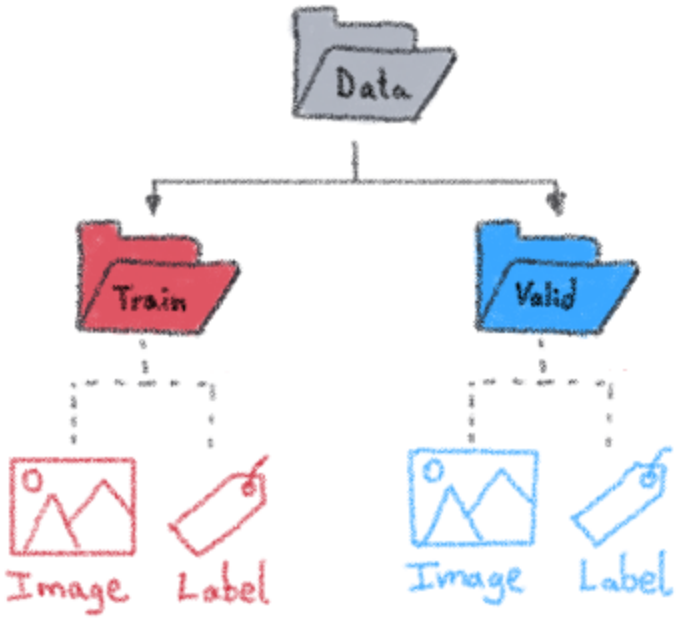
```
git clone https://github.com/ultralytics/yolov5  # clone
cd yolov5
pip install -r requirements.txt  # install
```

YOLOv5 training

python3 train.py --data dataset.yaml --epochs 300 --weights '' --cfg yolov5n.yaml --batch-size 128

- dataset.yaml 파일 작성 방법
- Dataset 폴더 구성 방법
- Cat and Dog dataset download (https://universe.roboflow.com/kct-yh2hv/cat-dog-4paux/dataset/2)
- Train/Image의 파일 이름과 Train/Label의 파일 이름이 일치 함.

```
data/train/images/img0.jpg # image
data/train/labels/img0.txt # label
```



- **Set up files and directory structure**: to train the YOLOv5 model, we need to add a **.yaml** file to describe the parameters of our dataset.

YOLOv5 training

python3 train.py --data dataset.yaml --epochs 300 --weights '' --cfg yolov5n.yaml --batch-size 128

- dataset.yaml 파일 작성 방법

```
names: ['aeroplane', 'apple', 'backpack', 'banana', 'baseball bat', 'baseball glove', 'bear', 'bed', 'bench', 'bicycle',
'bird', 'boat', 'book', 'bottle', 'bowl', 'broccoli', 'bus', 'cake', 'car', 'carrot', 'cat', 'cell phone', 'chair', 'clock',
'cow', 'cup', 'diningtable', 'dog', 'donut', 'elephant', 'fire hydrant', 'fork', 'frisbee', 'giraffe', 'hair drier', 'handbag',
'horse', 'hot dog', 'keyboard', 'kite', 'knife', 'laptop', 'microwave', 'motorbike', 'mouse', 'orange', 'oven', 'parking
meter', 'person', 'pizza', 'pottedplant', 'refrigerator', 'remote', 'sandwich', 'scissors', 'sheep', 'sink', 'skateboard',
'skis', 'snowboard', 'sofa', 'spoon', 'sports ball', 'stop sign', 'suitcase', 'surfboard', 'teddy bear', 'tennis racket',
'tie', 'toaster', 'toilet', 'toothbrush', 'traffic light', 'train', 'truck', 'tvmonitor', 'umbrella', 'vase', 'wine glass',
'zebra']
```

YOLOv5 training

python3 train.py --data dataset.yaml --epochs 300 --weights '' --cfg yolov5n.yaml --batch-size 128

- dataset.yaml 예지

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3
imgs1, path/to/imgs2, ..]
path: /home/team996/workspace/D04_CNN/issue2_catanddogs/  # dataset root dir
train: train/images  # train images (relative to 'path') 80% images
val: valid/images  # val images (relative to 'path') 20% images
test:  # test images (optional)

# Classes (2 classes)
names: ['Cat', 'Dog']
```

YOLOv5 training

```
python3 train.py --data dataset.yaml --epochs 300 --weights '' --cfg yolov5n.yaml --batch-size 128
```

yolov5s	64
yolov5m	40
yolov5l	24
yolov5x	16

2. Select a Model

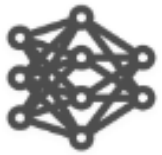
Select a pretrained model to start training from. Here we select YOLOv5s, the second-smallest and fastest model available. See our README table for a full comparison of all models.

Nano
YOLOv5n
4 MB_{FP16}
6.3 ms_{V100}
28.4 mAP_{COCO}

Small
YOLOv5s
14 MB_{FP16}
6.4 ms_{V100}
37.2 mAP_{COCO}

Medium
YOLOv5m
41 MB_{FP16}
8.2 ms_{V100}
45.2 mAP_{COCO}

Large
YOLOv5l
89 MB_{FP16}
10.1 ms_{V100}
48.8 mAP_{COCO}



XLarge
YOLOv5x
166 MB_{FP16}
12.1 ms_{V100}
50.7 mAP_{COCO}

YOLOv5 Inference

```
python3 detect.py --weights yolov5s.pt --source 0          # webcam
                                        img.jpg            # image
                                        vid.mp4            # video
```

```
import torch
import cv2
import argparse
import os
from utils.general import (
cv2,
print_args,
xyxy2xywh,
)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", nargs="+", type=str, help="model path or triton URL")
    parser.add_argument("--source", type=str, help="file/dir/URL/glob/screen/0(webcam)")
    # parser.add_argument("--imgsz", "--img", "--img-size", nargs="+", type=int, default=[640], help="l size h,w")
    opt = parser.parse_args()
    #opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand
    print_args(vars(opt))
    return opt
```

```
args = parse_opt()
weights = (os.getcwd()+'/').join(args.weights)
print(weights)
source = str(args.source)
print("model load", weights, type(weights))
print("source", source)
model = torch.hub.load('ultralytics/yolov5', 'custom', path=weights, _verbose=False)
#model = torch.hub.load('ultralytics/yolov5', 'yolov5s', _verbose=False)


print("image load")
im2 = cv2.imread(source)
results = model(im2)
results.save()
pred = results.pandas().xyxy[0]
print(pred)
predNP = pred.to_numpy()
nj, ni = predNP.shape


for n, i in enumerate(pred.columns):
#print(n, i)
    if i == "name":
        #print(((predNP.shape)))
        for n2, j in enumerate(predNP[:, n]):
            print(predNP[n2,0], predNP[n2,1], predNP[n2,2], predNP[n2,3], j)
```

THANK YOU