

Yolov5 실습 정리 - I

MS AI 스쿨



YOLOv5 실습 내용 정리

YOLOv5 zip 파일 다운로드 주소

<https://github.com/ultralytics/yolov5/archive/refs/heads/master.zip>

다운로드 후 압축 풀고 파이썬에서 열기

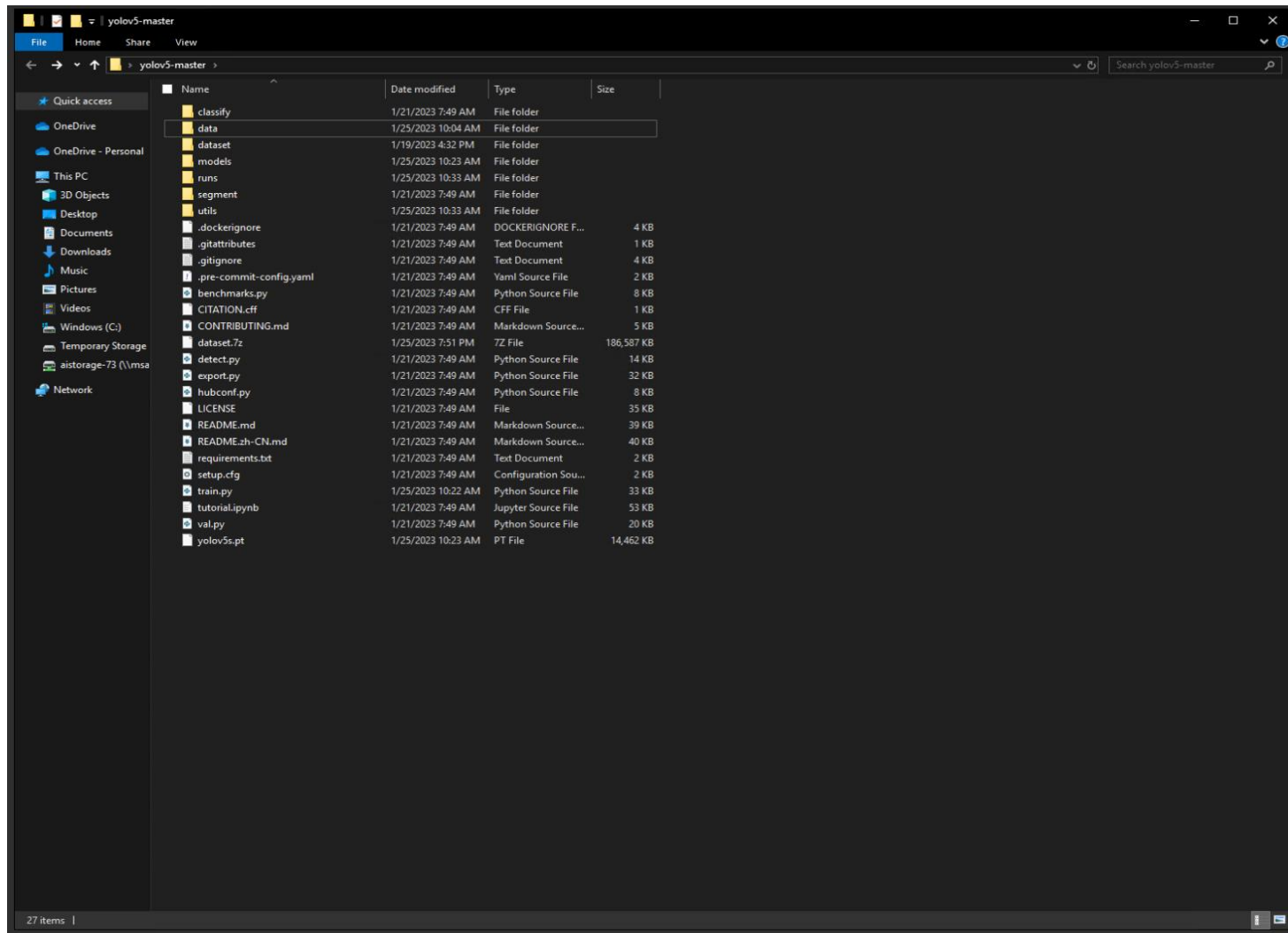
실습 데이터 다운로드 주소

https://drive.google.com/drive/folders/1seTuMKg4Bj3ebVIXGAuuGQYdEGOAeY4a?usp=share_link

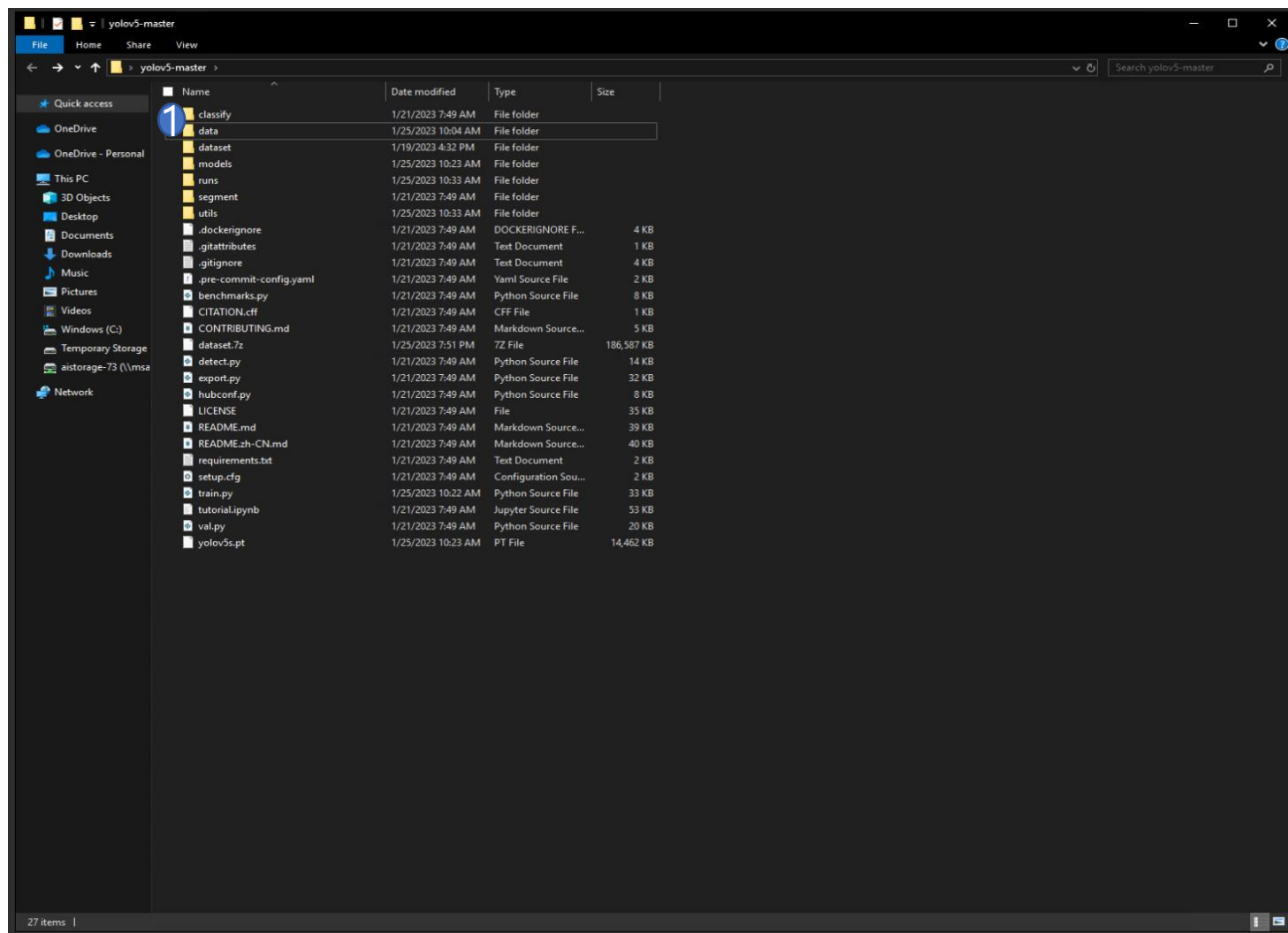
실습 데이터 다운로드 후 압축 풀고 yolov5 폴더로 이동시키기

YOLOv5 실습 내용 정리

압축 풀고 dataset 추가한 모습

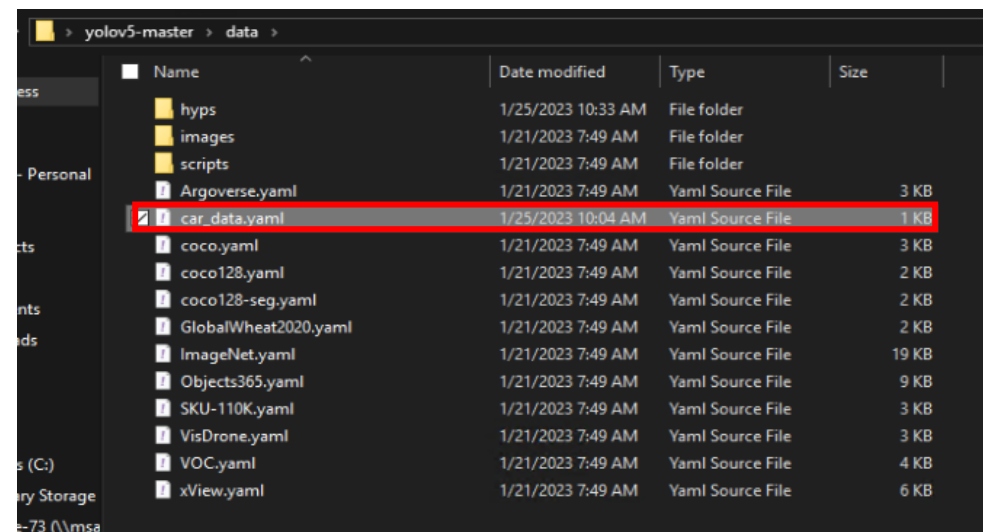


YOLOv5 실습 내용 정리

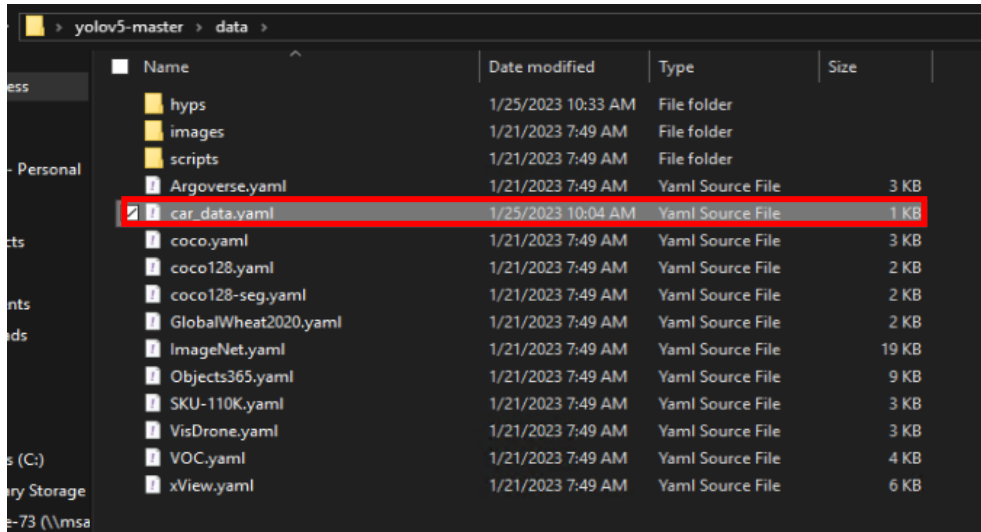


1. data 폴더에서 학습할 파일 경로 위치
라벨 정보 config 파일 생성하기

ex) car_data.yaml 파일 생성



YOLOv5 실습 내용 정리



Name	Date modified	Type	Size
hyps	1/25/2023 10:33 AM	File folder	
images	1/21/2023 7:49 AM	File folder	
scripts	1/21/2023 7:49 AM	File folder	
Argoverse.yaml	1/21/2023 7:49 AM	Yaml Source File	3 KB
car_data.yaml	1/25/2023 10:04 AM	Yaml Source File	1 KB
coco.yaml	1/21/2023 7:49 AM	Yaml Source File	3 KB
coco128.yaml	1/21/2023 7:49 AM	Yaml Source File	2 KB
coco128-seg.yaml	1/21/2023 7:49 AM	Yaml Source File	2 KB
GlobalWheat2020.yaml	1/21/2023 7:49 AM	Yaml Source File	2 KB
ImageNet.yaml	1/21/2023 7:49 AM	Yaml Source File	19 KB
Objects365.yaml	1/21/2023 7:49 AM	Yaml Source File	9 KB
SKU-110K.yaml	1/21/2023 7:49 AM	Yaml Source File	3 KB
VisDrone.yaml	1/21/2023 7:49 AM	Yaml Source File	3 KB
VOC.yaml	1/21/2023 7:49 AM	Yaml Source File	4 KB
xView.yaml	1/21/2023 7:49 AM	Yaml Source File	6 KB

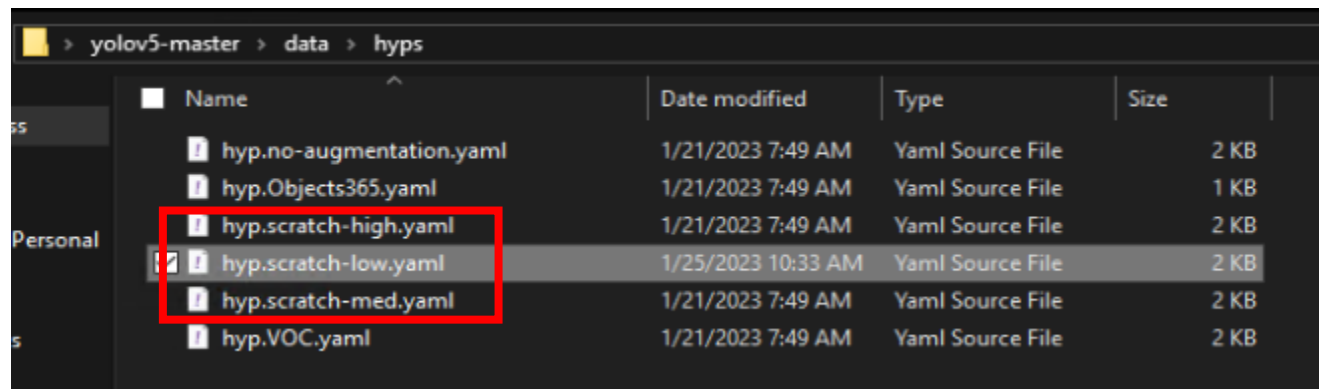
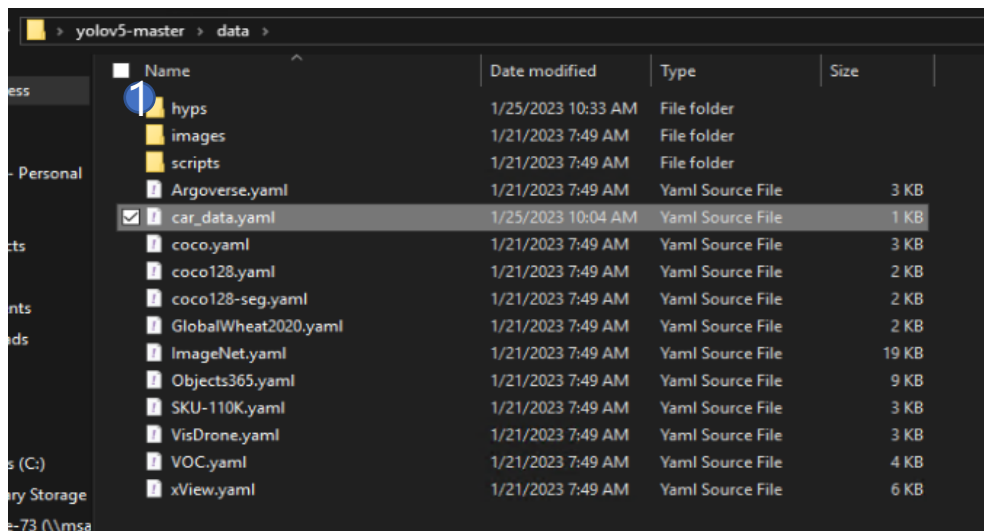


```
! car_data.yaml X
C:\> Users > labadmin > Desktop > yolov5-master > data > ! car_data.yaml
1 train: ./dataset/train/images
2 val: ./dataset/valid/images
3
4 # classes
5 names :
6 0: big bus
7 1: big truck
8 2: bus-l-
9 3: bus-s-
10 4: car
11 5: mid truck
12 6: small bus
13 7: small truck
14 8: truck-l-
15 9: truck-m-
16 10: truck-s-
17 11: truck-xl-
18
```

car_data.yaml 파일 내용

1. Train, val 데이터 경로 위치 지정
2. Classes 정의 (학습하고자 하는 라벨 정보)

YOLOv5 실습 내용 정리



1. 모델 크기에 따라서 설정하는 hyp 값이 다름
2. 해당하는 yaml 파일을 선택하여 값을 수정

hyps 값 수정하는 방법

1. hyps 폴더 클릭

모델별 선택해야하는 yaml 파일

yolov5n, yolov5s -> hyp.scratch-low.yaml


yolov5m -> hyp.scratch-med.yaml

yolov5l, yolov5x -> hyp.scratch-high.yaml


YOLOv5 실습 내용 정리

hyps 값 수정

```
! hyp.scratch-low.yaml X
C: > Users > labadmin > Desktop > yolov5-master > data > hyps > ! hyp.scratch-low.yaml
1  # YOLOv5 by Ultralytics, GPL-3.0 license
2  # Hyperparameters for low-augmentation COCO training from scratch
3  # python train.py --batch 64 --cfg yolov5n6.yaml --weights '' --data coco.yaml --img 640 --epochs 300 --linear
4  # See tutorials for hyperparameter evolution https://github.com/ultralytics/yolov5#tutorials
5
6  lr0: 0.001 # initial learning rate (SGD=1E-2, Adam=1E-3)
7  lrf: 0.01 # final OneCycleLR learning rate (lr0 * lrf)
8  momentum: 0.937 # SGD momentum/Adam beta1
9  weight_decay: 0.0005 # optimizer weight decay 5e-4
10 warmup_epochs: 3.0 # warmup epochs (fractions ok)
11 warmup_momentum: 0.8 # warmup initial momentum
12 warmup_bias_lr: 0.1 # warmup initial bias lr
13 box: 0.05 # box loss gain
14 cls: 0.5 # cls loss gain
15 cls_pw: 1.0 # cls BCELoss positive_weight
16 obj: 1.0 # obj loss gain (scale with pixels)
17 obj_pw: 1.0 # obj BCELoss positive_weight
18 iou_t: 0.20 # IoU training threshold
19 anchor_t: 4.0 # anchor-multiple threshold
20 # anchors: 3 # anchors per output layer (0 to ignore)
21 fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)
22 hsv_h: 0.015 # image HSV-Hue augmentation (fraction)
23 hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)
24 hsv_v: 0.4 # image HSV-Value augmentation (fraction)
25 degrees: 0.5 # image rotation (+/- deg)
26 translate: 0.1 # image translation (+/- fraction)
27 scale: 0.5 # image scale (+/- gain)
28 shear: 0.0 # image shear (+/- deg)
29 perspective: 0.0 # image perspective (+/- fraction), range 0-0.001
30 flipud: 0.0 # image flip up-down (probability)
31 fliplr: 0.5 # image flip left-right (probability)
32 mosaic: 1.0 # image mosaic (probability)
33 mixup: 0.0 # image mixup (probability)
34 copy_paste: 0.0 # segment copy-paste (probability)
35
```

수정 가능한 hyps 값

학습 lr, momentum, weight_decay

Augmentation 지정 값들 수정가능

YOLOv5 실습 내용 정리

train.py parser 값 수정

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='./models/yolov5s.yaml', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/data.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=100, help='total training epochs')
    parser.add_argument('--batch-size', type=int, default=64, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument('--noval', action='store_true', help='only validate final epoch')
    parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
    parser.add_argument('--noplots', action='store_true', help='save no plot files')
    parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache', type=str, nargs='?', const='ram', help='image --cache ram/disk')
    parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
    parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
    parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW', 'SGDP', 'AdamP'], default='AdamP', help='optimizer')
    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
    parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')
    parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')
    parser.add_argument('--name', default='exp_0125', help='save to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--quad', action='store_true', help='quad dataloader')
    parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
    parser.add_argument('--label-smoothing', type=float, default=0.3, help='Label smoothing epsilon')
    parser.add_argument('--patience', type=int, default=100, help='EarlyStopping patience (epochs without improvement)')
    parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers: backbone=10, first3=0 1 2')
    parser.add_argument('--save-period', type=int, default=-1, help='Save checkpoint every x epochs (disabled if < 1)')
    parser.add_argument('--seed', type=int, default=0, help='Global training seed')
    parser.add_argument('--local_rank', type=int, default=-1, help='Automatic DDP Multi-GPU argument, do not modify')

    # Logger arguments
    parser.add_argument('--entity', default=None, help='Entity')
    parser.add_argument('--upload_dataset', nargs='?', const=True, default=False, help='Upload data, "val" option')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval')
    parser.add_argument('--artifact_alias', type=str, default='latest', help='Version of dataset artifact to use')

    return parser.parse_known_args()[0] if known else parser.parse_args()
```

weights : 학습하고자 하는 모델 pt 파일 지정
yolov5s.pt, yolov5m.pt, yolov5l.pt, yolov5x.pt

cfg : 해당하는 모델 아키텍처 경로 지정
models 폴더안에 아키텍처들 있음

data : dataset.yaml path
ex : data/car_data.yaml

epoch : 학습 횟수
batch-size : 배치 사이즈 값

optimizer : choices 값 을 보고 default 값 수정
label-smothing : 오버피팅을 방지하기 위한 기법
위의 내용을 수정 후 학습 진행
터미널 : Python train.py 혹은 파이참 실행

YOLOv5 실습 내용 정리

Inf.py 인퍼런스 파이스크립트

```

1 import torch
2 import os
3 import glob
4 import cv2
5
6 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
7
8 # model call
9 model = torch.hub.load('ultralytics/yolov5', 'custom', path='./runs/train/exp_0125/weights/best.pt')
10
11 # inference Settings
12 model.conf = 0.5 # NMS confidence threshold
13 model.iou = 0.45 # NMS IoU threshold
14 model.to(device)
15
16 # image loader
17 image_dir = './dataset/test/images/'
18 image_path = glob.glob(os.path.join(image_dir, "*.jpg"))
19 label_dict = {0:"big bus", 1:"big truck", 2:"bus-l-", 3:"bus-s-",4:"car",
20              5:"mid truck", 6:"small bus", 7:"small truck", 8:"truck-l-", 9:"truck-m-",
21              10:"truck-s-",11:"truck-xl-"}
22
23 for img_path in image_path :
24     # Image
25     img = cv2.imread(img_path)
26
27     # Inference
28     results = model(img, size=640)
29
30     # Results
31     bbox = results.xyxy[0]
32
33     # image name
34     image_name = os.path.basename(img_path)
35
36     # image w h
37     h, w, c = img.shape
38
39     for box in bbox :
40         x1 = box[0].item()
41         y1 = box[1].item()
42         x2 = box[2].item()
43         y2 = box[3].item()
44         print(x1,y1,x2,y2)
45         cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), (0,255,0),2)
46
47     cv2.imshow("Test", img)
48     cv2.waitKey(0)

```

device 설정

학습 한 모델 호출 path 인자에 학습한 모델 경로 지정

모델 인퍼런스를 위한 conf, iou 설정

테스트할 이미지 경로 지정 및 이미지 경로 리스트 생성
라벨 딕셔너리 생성For 문을 이용하여 이미지 경로 리스트에서 하나씩 경로 값을
추출 하여 cv2.imread() 함수를 통해 이미지 읽기

학습한 모델에 입력 값으로 전달 model(이미지, 학습한 이미지 사이즈)

bbox = results.xyxy[0] # 객체인식한 바운딩박스 좌표, 스코어, 라벨
>> 이차원 리스트 존재For 을 통해서 bbox 값을 [x,y,x,y,스코어,라벨] 추출 후 인덱스를 이용
하여 x1,y1,x2,y2 값을 추출 후 cv2.rectangle() 함수를 통해 시각화

YOLOv5 실습 내용 정리

Inf.py 인퍼런스 파이스크립트 추가 : CVAT XML 양식으로 추출하는 방법

```
import torch
import os
import glob
import cv2
import xml.etree.ElementTree as ET
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# model call
model = torch.hub.load('ultralytics/yolov5', 'custom', path='./runs/train/exp_0125/weights/best.pt')

# Inference Settings
model.conf = 0.5 # NMS confidence threshold
model.iou = 0.45 # NMS IoU threshold
model.to(device)

# image loader
image_dir = "./dataset/test/images/"
image_path = glob.glob(os.path.join(image_dir, "*.jpg"))
label_dict = {0:"big bus", 1:"big truck", 2:"bus-l-", 3:"bus-s-", 4:"car",
              5:"mid truck", 6:"small bus", 7:"small truck", 8:"truck-l-", 9:"truck-m-",
              10:"truck-s-", 11:"truck-xl-"}

tree = ET.ElementTree()
root = ET.Element("annotations")
"""
<annotations>
</annotations>
"""

seen_count = 0

for img_path in image_path :
    # Image
    img = cv2.imread(img_path)

    # Inference
    results = model(img, size=640)

    # Results
    bbox = results.xyxy[0]

    # image name
    image_name = os.path.basename(img_path)

    # image w h
    h, w, c = img.shape

    # xml fix code
    # <image id="0" name="adit_mp4-1002_jpg.rf.5e4018e963af1251b3f7e6fd487c479e.jpg" width="640" height="480">
    xml_frame = ET.SubElement(root, "image", id="%d"%seen_count, name=image_name,
                                width="%d"%w, height="%d"%h)
    """
```

Xml 파일에서 가장 상단

<annotations>

</annotations>

단 한번만 생성하면 되므로 for 밖에서 생성

YOLOv5 실습 내용 정리

Inf.py 인퍼런스 파이스크립트 추가 : CVAT XML 양식으로 추출하는 방법

```

import torch
import os
import glob
import cv2
import xml.etree.ElementTree as ET
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# model call
model = torch.hub.load('ultralytics/yolov5', 'custom', path='./runs/train/exp_0125/weights/best.pt')

# Inference Settings
model.conf = 0.5 # NMS confidence threshold
model.iou = 0.45 # NMS IoU threshold
model.to(device)

# image loader
image_dir = "./dataset/test/images/"
image_path = glob.glob(os.path.join(image_dir, "*.jpg"))
label_dict = {0:"big bus", 1:"big truck", 2:"bus-l-", 3:"bus-s-", 4:"car",
              5:"mid truck", 6:"small bus", 7:"small truck", 8:"truck-l-", 9:"truck-m-",
              10:"truck-s-", 11:"truck-xl-"}

tree = ET.ElementTree()
root = ET.Element("annotations")
"""
<annotations>
</annotations>
"""
seen_count = 0

for img_path in image_path :
    # Image
    img = cv2.imread(img_path)

    # Inference
    results = model(img, size=640)

    # Results
    bbox = results.xyxy[0]

    # image name
    image_name = os.path.basename(img_path)

    # image w h
    h, w, c = img.shape

    # xml fix code
    # <image id="0" name="adit_mp4-1002_jpg.rf.5e4018e963af1251b3f7e6fd487c479e.jpg" width="640" height="480">
    xml_frame = ET.SubElement(root, "image", id="%d"%seen_count, name=image_name,
                               width="%d"%w, height="%d"%h)
    """

```

Xml 파일에서 annotations 안에 <image> 태그 생성
<annotations>
</annotations>

이미지 하나당 <image> 태그 생성되어야 하므로
For img_path in image_path : 안에서 생성하기

생성방법은 빨간색 박스 참고

YOLOv5 실습 내용 정리

Inf.py 인퍼런스 파이스크립트 추가 : CVAT XML 양식으로 추출하는 방법

```

for box in bbox :
    # <box label="car" occluded="0" source="manual" xtl="346.68"
    # ytl="325.63" xbr="427.46" ybr="404.08" z_order="0"> </box>
    # box
    """
    <annotations>
    <image id="0" name="adit_mp4-1002_jpg.rf.5e4018e963af1251b3f7e6fd487c479e.jpg" width="640" height="480">
        <box></box>
        <box></box>
        <box></box>
        <box></box>
    </image>
    <image id="0" name="adit_mp4-1002_jpg.rf.5e4018e963af1251b3f7e6fd487c43249e.jpg" width="640" height="480">
        <box></box>
        <box></box>
        <box></box>
        <box></box>
    </image>
    </annotations>
    """
    x1 = box[0].item()
    y1 = box[1].item()
    x2 = box[2].item()
    y2 = box[3].item()
    xtl = str(round(x1,3))
    ytl = str(round(y1,3))
    xbr = str(round(x2,3))
    ybr = str(round(y2,3))

    # class
    class_number = box[5].item()
    class_number_int = int(class_number)
    labels = label_dict[class_number_int]

    # score
    sc = box[4].item()

    # bbox xml
    # xtl="346.68" ytl="325.63" xbr="427.46" ybr="404.08" z_order="0"
    ET.SubElement(xml_frame, "box", label=labels, occluded="0", source="manual",
                  xtl=xtl, ytl=ytl, xbr=xbr, ybr=ybr, z_order="0")

    seen_count +=1

tree._setroot(root)
tree.write("test.xml", encoding="utf-8")

```

Xml 파일에서 annotations 안에 <image> 태그 생성

```

<annotations>
  <image>
    <box> </box>
  <image>
    <box> </box>
  </annotations>

```

해당 이미지에 객체 탐지한 박스 만큼 <box> 태그 생성되어야 하므로 바운딩 박스를 추출 하는 for 안에서 정의

생성방법은 빨간색 박스 참고

그리고 마지막에 최종적으로 tree에서 최종 완성 되어야합니다. 그리고 완성된 tree 값을 write 해서 xml파일로 저장



감사합니다.