
심화 교육과정
Object Detection 유명 프레임 워크 사용 방법
THINK LIFE SYNC AI
Infinyx

Yolov5

코드 주소 : https://github.com/ultralytics/yolov5

* 코드 다운로드 방법 ZIP 파일 내려 받기 혹은 Git Clone 명령어 이용한 방법

Git Clone 명령어
- git clone https://github.com/ultralytics/yolov5.git

Zip 파일 주소
- https://github.com/ultralytics/yolov5/archive/refs/heads/master.zip

Yolov5

환경 세팅

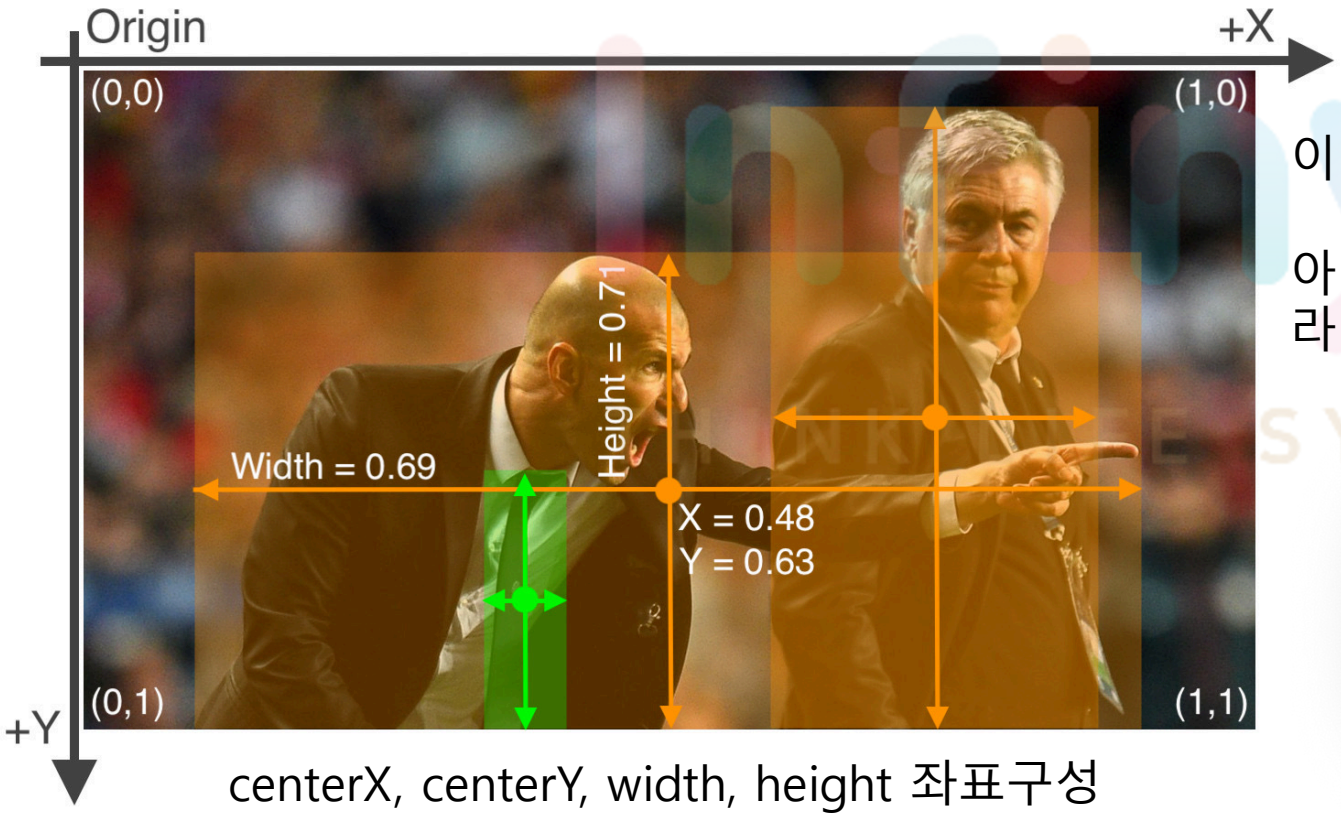
1. 아나콘다 활용하여 가상환경 만든 후 진행

2. Pytorch 버전에 맞는 것을 설치 할 것

3. pip install -r requirements.txt
   단 리스트에서 torch, torchvison 삭제 후 실행 할 것 위에서 이미 설치 하여서 불필요

Yolov5

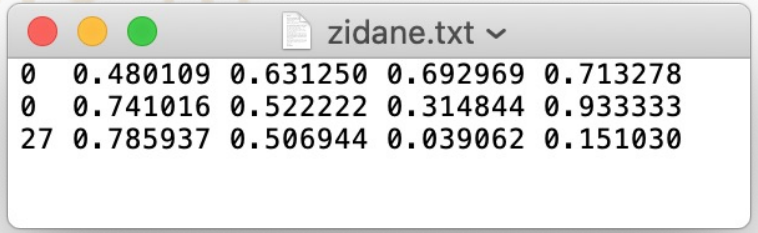
라벨링 데이터 : 바운딩 박스 YOLO 좌표 표기



centerX, centerY, width, height 좌표구성

이미지 하나당 하나의 라벨 txt 파일 필요

아래와 같이 라벨.txt 파일은 구성
라벨번호 좌표(centerX, centerY, width, height) 구성

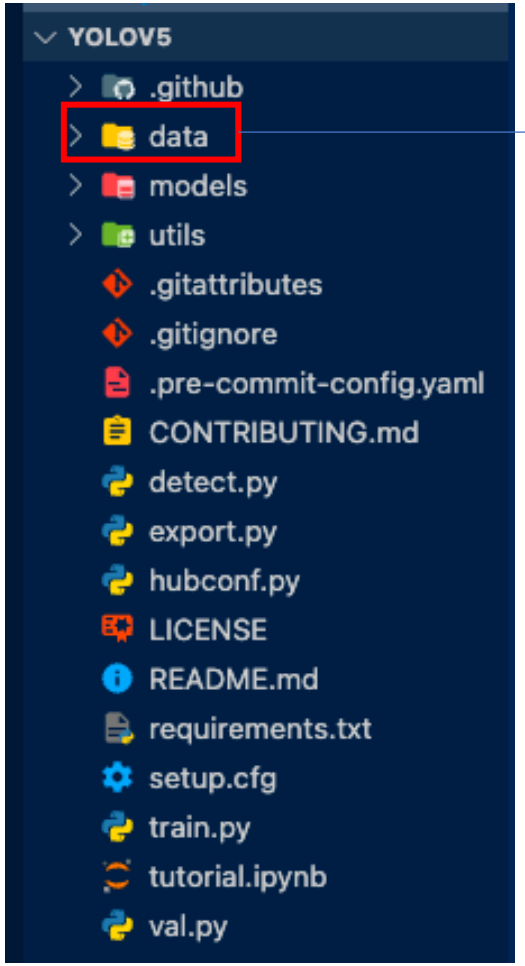


이미지명 = 라벨.txt 명칭 동일 zidane.jpg - zidane.txt

Infinyx

Yolov5

폴더구성



학습할 데이터 셋팅(학습 데이터 위치 경로, hyps 폴더안에는 하이퍼 파라메타 값 수정 가능)

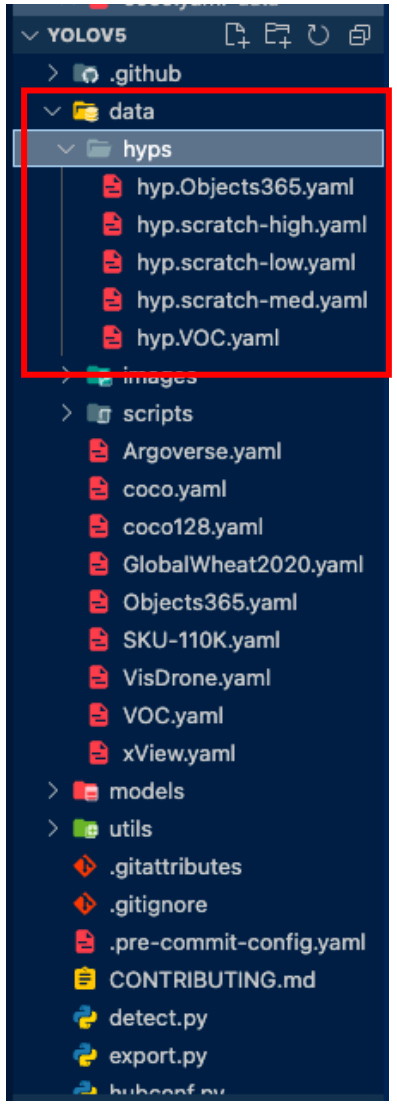
1. 학습하고자 하는 데이터 yaml 파일생성하기

2. Yaml 파일 구성 요소

Train : 학습하고자 데이터 경로
Val : 중간 평가 데이터 경로

Nc : 클래스 갯수(타입 : int) ex) 80
Name : 클래스 이름(타입: list) ex) names : ['person', 'bicycle', 'car']

Yolov5



Hyperparameter 값이 yaml 파일로 지정되어있음

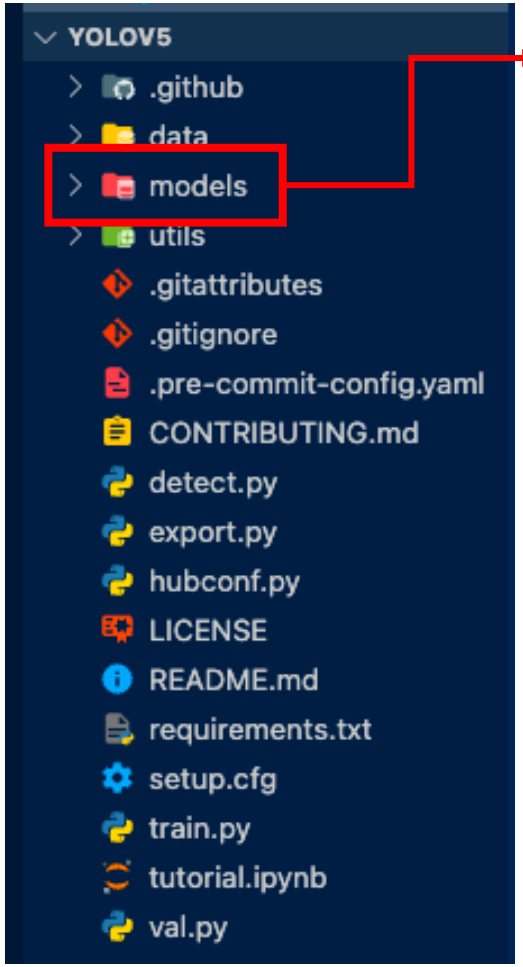
필요시 수정가능

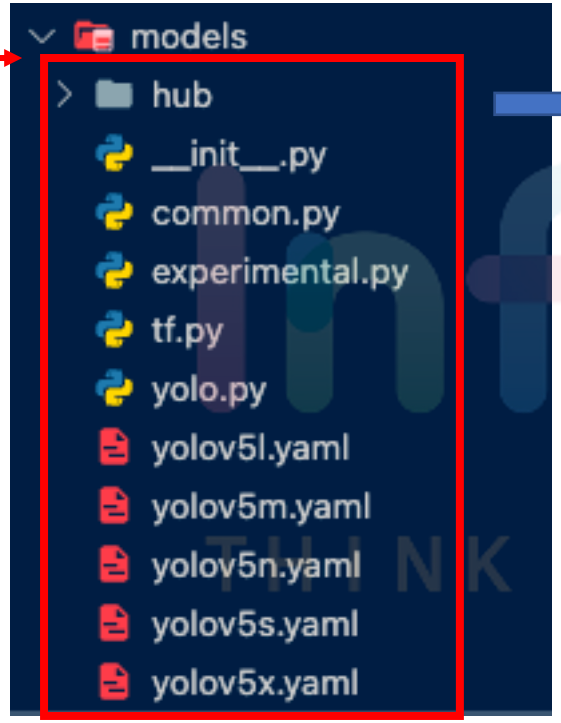
아래와 같이 Hyperparameter 값을 조정가능

```yaml
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
```

Infinyx

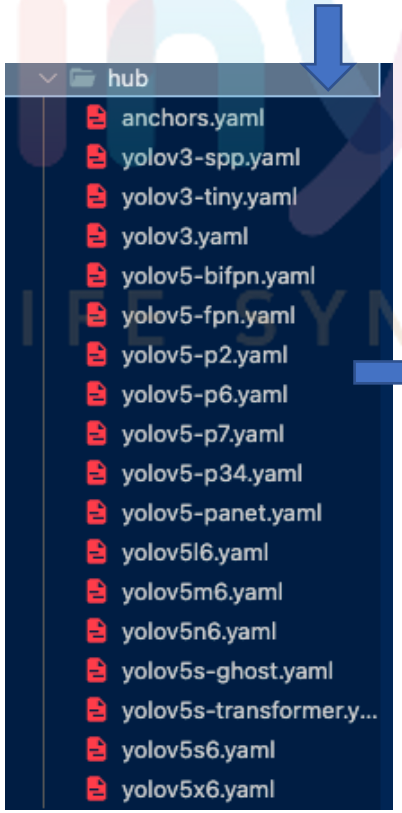
Yolov5





Yolov5 모델 아키텍처 위치하고 폴더

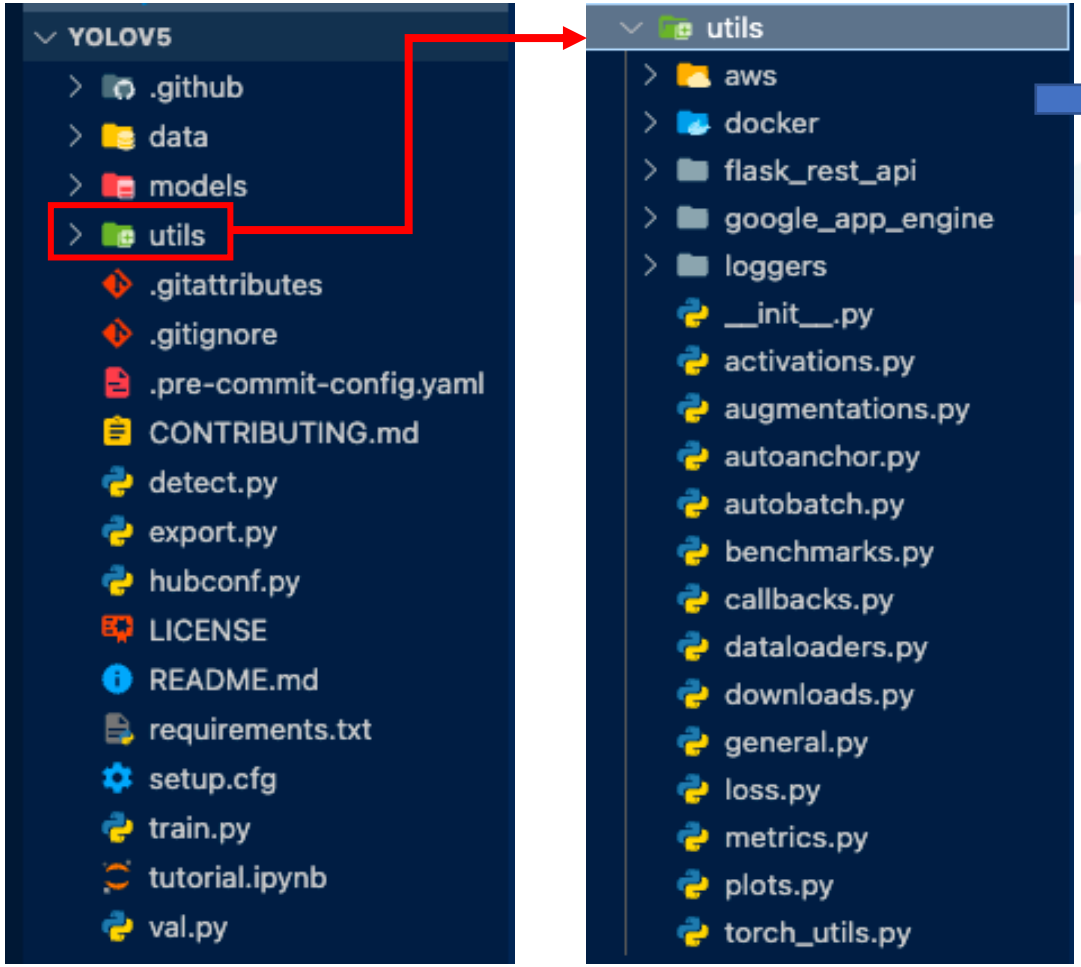
Input size 1280 모델 아키텍처는 Hub 폴더 안에 존재



다양한 아키텍처 모델 존재
- 패치 내용을 확인 필요
- 실험 요소 아키텍처 저장 장소

Infinyx

Yolov5



Yolov5 구성하기 위한 각종 Utils 파일

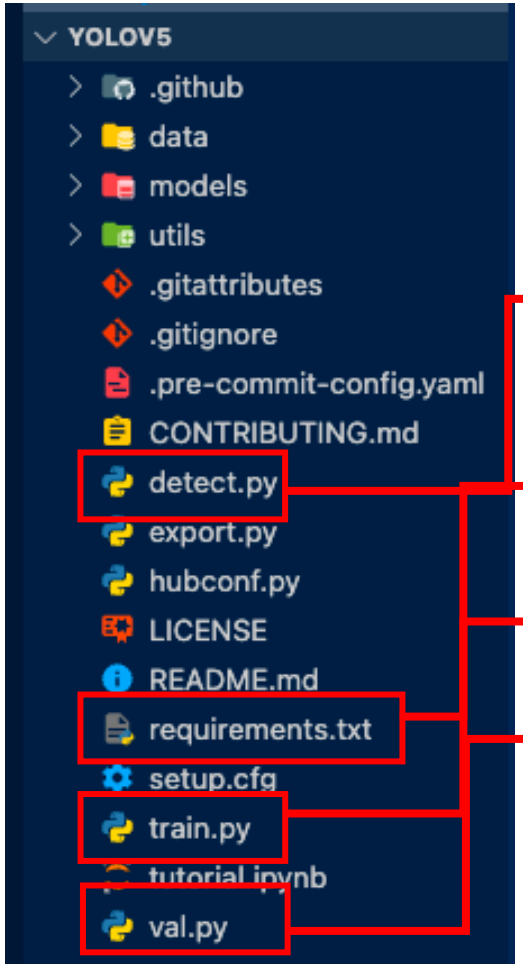
- Augmentations.py : Augmentations 코드 추가 하고싶은 aug 추가가능

- dataloaders.py : 데이터 전처리 및 불러오는 코드

- Loss.py : 각종 적용가능한 Loss 함수 코드

Infinyx

Yolov5



학습한 모델을 이용하여 테스트 데이터 결과를 확인 할 수 있는 코드

Yolov5 학습을 위한 설치 파일 리스트 주의 pytorch 리스트에서 제거 따로 설치 되어있는경우 !

Yolov5 학습 코드

평가 코드

Infinyx

Yolov5 Train.py parser 세팅

```
parser = argparse.ArgumentParser()
parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.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=300)
parser.add_argument('--batch-size', type=int, default=16, 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='--cache images in "ram" (default) or "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'], default='SGD', 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', 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=0.0, 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('--local_rank', type=int, default=-1, help='DDP parameter, do not modify')
```

Weights : initial weights path
- 초기 weights 경로 지정

cfg : 모델 아키텍처 경로 지정

data : dataset.yaml 경로 지정

hyp : hyperparameters 경로 지정

Epochs : 학습을 얼마나 할 것 인가 지정

Batch_size : batch 사이즈 지정

Resume : 컴퓨터 강제로 종료되어 학습을 재시작 원하는 경우

Device : cuda device i.e 0 or 0 1 2 3 or CPU

Opeimizer : opeimizer 설정

Lr 스케줄 : 스케줄 변경 설정

Label-smothing : 라벨 스무딩 설정

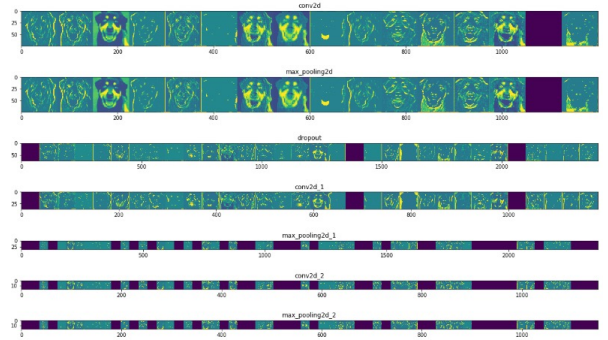
mmdetection

- 칭화 대학(중국, 베이징시)의 주도로 만들어진 Computer vision Open Source project인 OpenMMLab에서 출발
- 2018년 MS-COCO Challenge에서 우승 후 모듈을 확장하여 다수의 알고리즘 수용
- 최신의 다양한 Object Detection, Segmentation 알고리즘을 Package로 구현 제공
- 뛰어난 구현 성능, 효율적인 모듈 설계, Config 기반으로 데이터부터 모델 학습/평가 까지 이어지는 간편한 파이프라인 적용
- Pytorch 기반으로 구현

Infinyx

mmdetection

MM Detection 모델 아키텍처

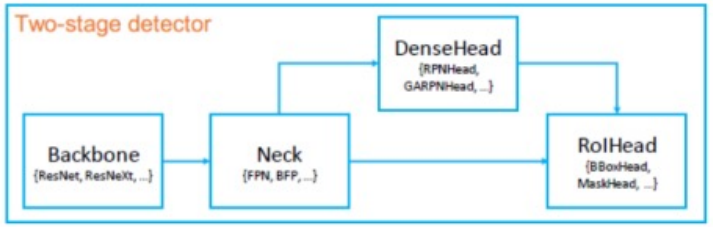


- Backbone : Feature Extractor (이미지 -> Feature Map)

- Neck : Backbone 과 Heads 연결하면서 heads 가
  feature map의 특성을 보다 잘 해석하고 처리할 수 있도록 정제 작업 수행.

- DenseHead (AnchorHead / AnchorFreeHead)
  - Feature Map 에서 Object 위치와 Classification을 처리하는 부분

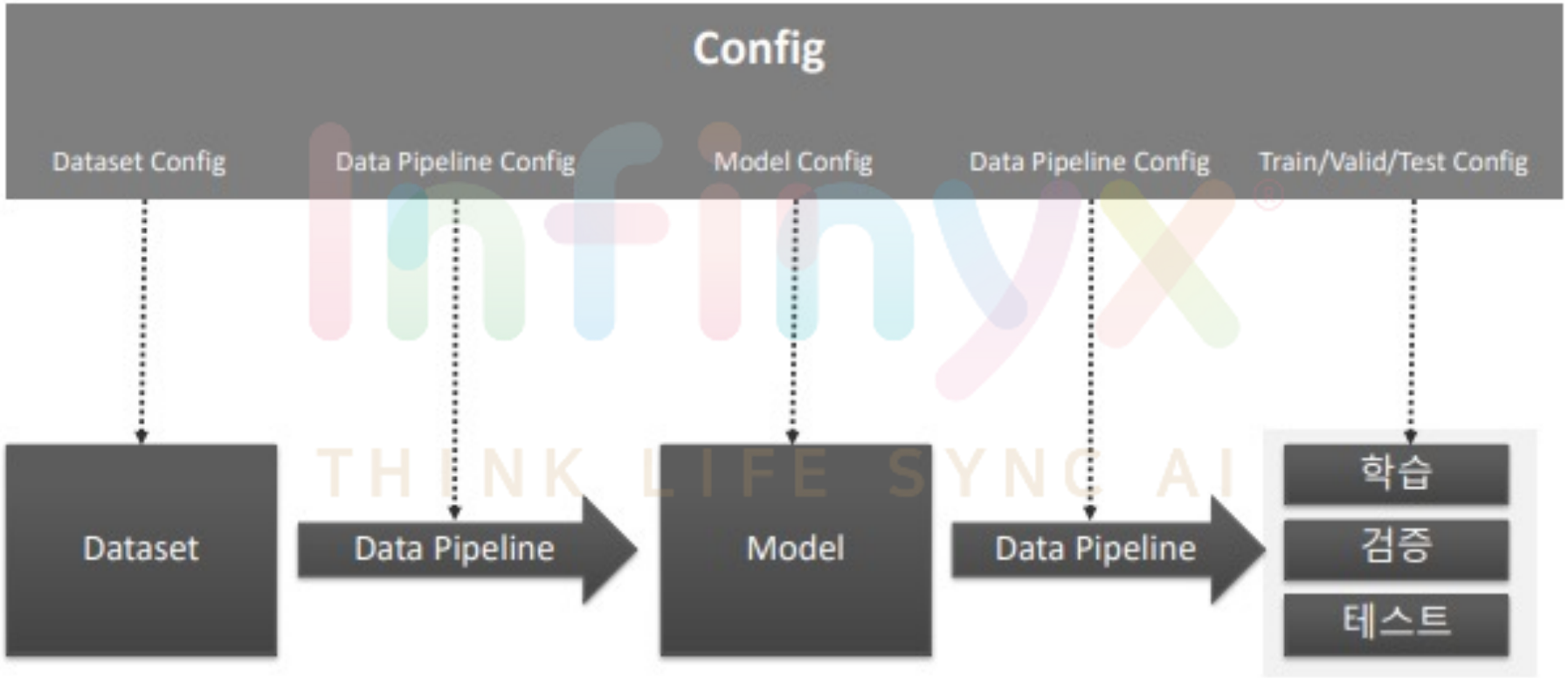
- RoIExtractor
  - Feature Map 에서 ROI 정보를 뽑아내는 부분



- RoIHead (BBoxHead / MaskHead)
  - ROI 정보를 기반으로 Object 위치와 Classification 을 수행하는 부분

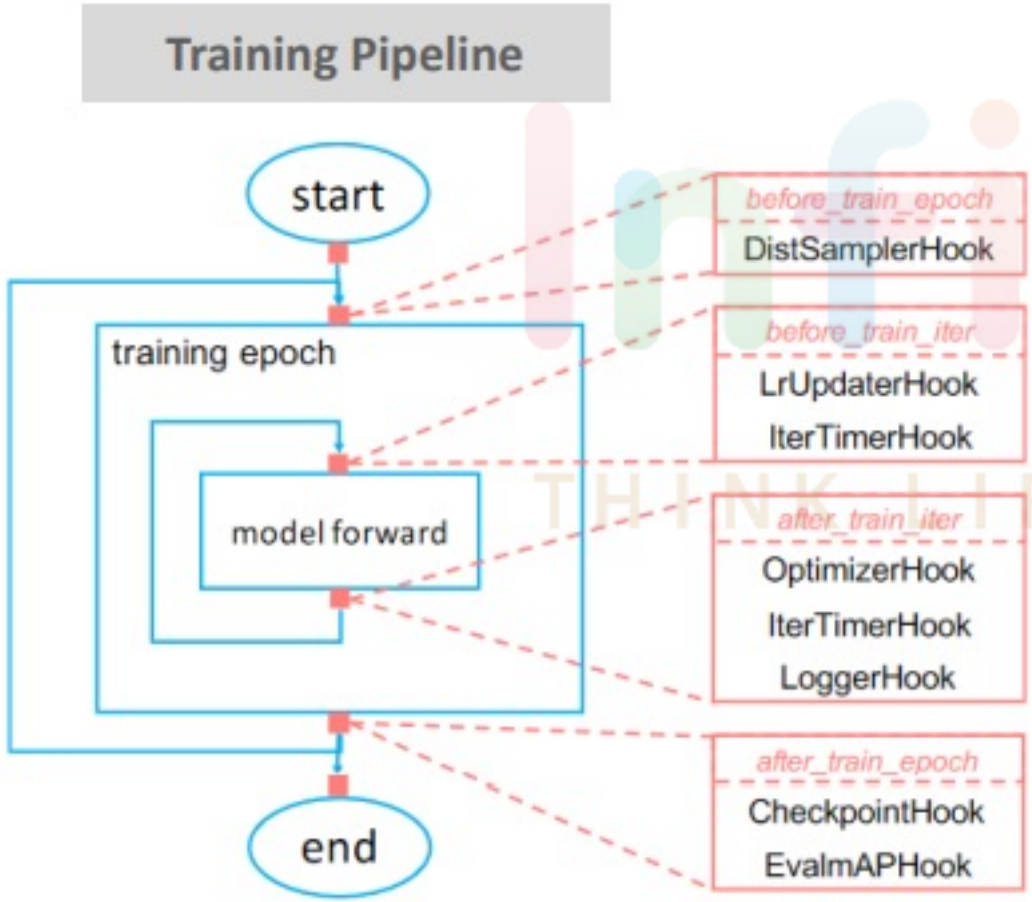
Infinyx

mmdetection



mmdetection

MM Detection Training Pipeline



- Hook(Callback)을 통해 학습에 필요한 여러 설정들을 Customization 가능
- 대부분 Configuration에서 이를 설정함.

Yolov5

코드 주소 : https://github.com/ultralytics/yolov5

* 코드 다운로드 방법 ZIP 파일 내려 받기 혹은 Git Clone 명령어 이용한 방법

Git Clone 명령어
- git clone https://github.com/ultralytics/yolov5.git

Zip 파일 주소
- https://github.com/ultralytics/yolov5/archive/refs/heads/master.zip

Mmdetection 환경 세팅

코드 주소 : https://github.com/open-mmlab/mmdetection

Step 1. Create a conda environment and activate it.

```
conda create --name openmmlab python=3.8 -y conda
activate openmmlab
```

Step 2. Install PyTorch following official instructions, e.g. https://pytorch.org/get-started/previous-versions/

```
On GPU platforms:
  - conda install pytorch torchvision -c pytorch
```

```
On CPU platforms:
  - conda install pytorch torchvision cpuonly -c pytorch
```

- Pytorch 홈페이지 에서 버전 체크 하여 Pytroch 설치 해야함
- MMdetection 경우 설치된 torch 버전과 cuda 버전이 동일하게 설치 되어야함.

Mmdetection 환경 세팅

코드 주소 : https://github.com/open-mmlab/mmdetection

Step 0. Install MMCV using MIM.

```
pip install -U openmim
pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu113/torch1.10/index.html
```

Step 2. Install PyTorch following official instructions, e.g. https://pytorch.org/get-started/previous-versions/

```
git clone https://github.com/open-mmlab/mmdetection.git
cd mmdetection
pip install -v -e .
```

Mmdetection 환경 세팅

Install on Google Colab

Step 1. Install MMCV using MIM.

```
!pip3 install openmim
!mim install mmcv-full
```

Step 2. Install MMDetection from the source.

```
!git clone https://github.com/open-mmlab/mmdetection.git
%cd mmdetection
!pip install -v -e .
```

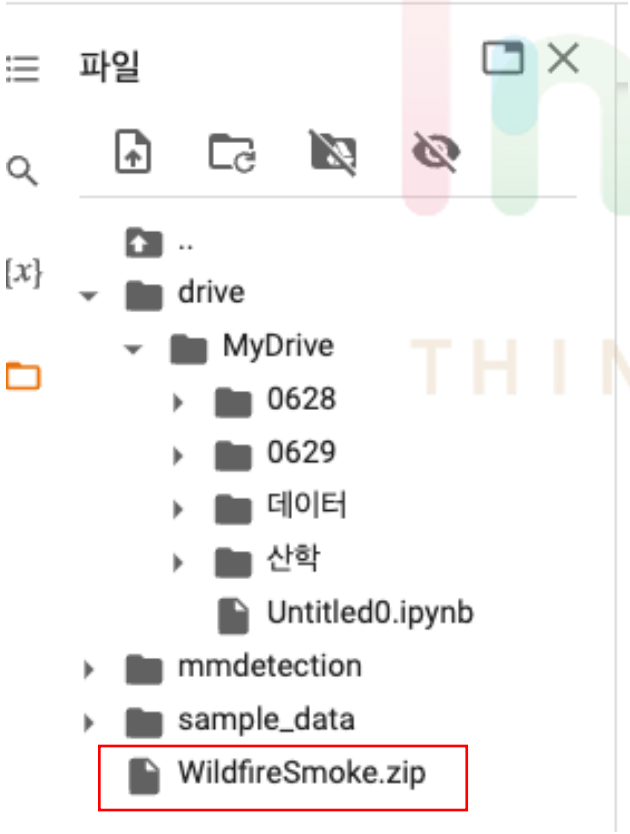
Step 3. Verification.

```
import mmdet
print(mmdet.__version__)
# Example output: 2.23.0
```

Mmdetection 환경 세팅

Install on Google Colab

구글 드라이브 연결 후 산불 데이터 업로드 하고



* 산불 데이터를 구글 드라이버에서 드래그 해서 옆에 사진처럼 위치 하시면 됩니다.

Zip 파일 압축 해제 방법

```
!unzip /content/WildfireSmoke.zip -d ./dataset
```

해제 후 -> dataset 폴더안에 산불 데이터가 있습니다 .

Infinyx

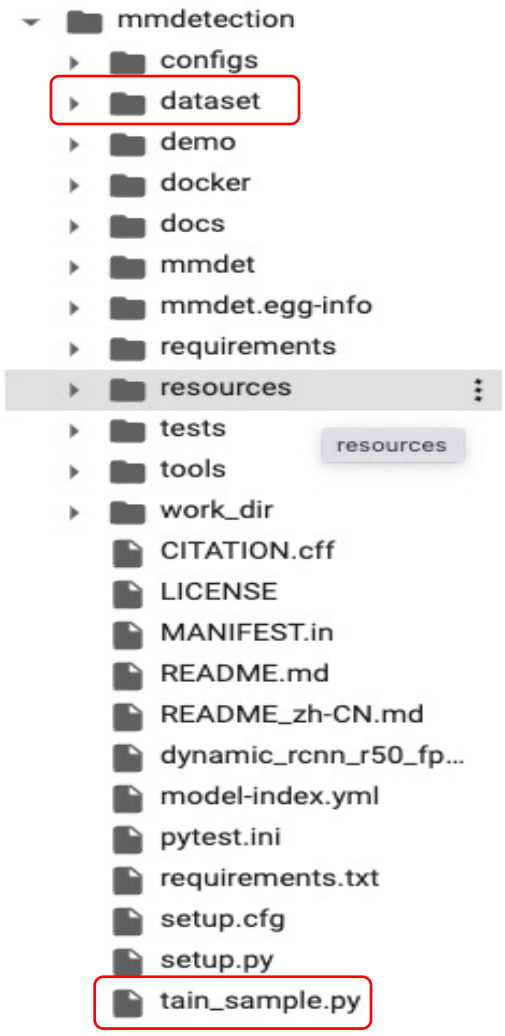
Mmdetection 환경 세팅

Install on Google Colab

강의자료 18페이지 부분에 있는 Install on Google Colab 설치 진행 하기

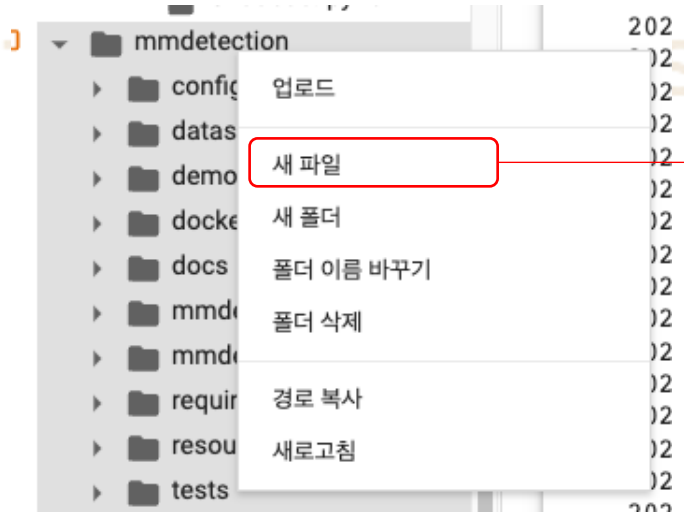
환경 세팅 완료 후에 mmdetection 폴더 에 dataset 폴더를 넣어줍니다.

다음과 같이 mmdetection 폴더가 구성 된 것을 확인 가능합니다.



train_sample.py 생성합니다

생성방법은 mmdetection 폴더에서 RIGHT CLICK 하시면 파일 생성 버튼 있습니다.



train_sample.py 생성하시면 됩니다.

Mmdetection sample train code

학습 코드는 train_sample.py 확인 하세요 주석으로 설명 해뒀습니다.


감사합니다.
THINK LIFE SYNC AI
Infinyx