AI 영상 데이터 수집 및 모델 준비
-YOLOv8-

0. 수집 이미지 일부

Fig 1. 수집 이미지(cat B)

1. 사전 개념 및 준비

a. 전체 디렉터리 구조 예시

```
your_project/
├── images/
│   ├── all/
│   │   ├── catA/          ← 원본 수집 이미지 50장
│   │   └── catB/          ← 원본 수집 이미지 50장
│   ├── train/
│   │   ├── catA/          ← 학습용(80%) 이미지
│   │   └── catB/
│   └── val/
│       ├── catA/          ← 검증용(20%) 이미지
│       └── catB/
├── labels/
│   ├── train/
│   │   ├── catA/          ← train 바운딩박스 라벨(.txt)
│   │   └── catB/
│   └── val/
│       ├── catA/          ← val 바운딩박스 라벨(.txt)
│       └── catB/
├── scripts/
│   ├── split_data.py      ← train/val 랜덤 분할 스크립트
│   ├── clean_images.py    ← 해상도 + 중복 제거 스크립트
│   └── infer.py           ← 학습된 모델로 추론 스크립트
├── data.yaml              ← YOLOv8 설정 파일
└── requirements.txt       ← 필요한 패키지 목록
```

b. requirements.txt

```
ultralytics
opencv-python
bing-image-downloader
Pillow
duplicate-image-finder
labelImg
```

터미널 :pip install -r requirements.txt

c. 데이터 분할 스크립트 : scripts/split_data.py

```
import os, random, shutil

def split(src_dir, train_dir, val_dir, ratio=0.8):
    os.makedirs(train_dir, exist_ok=True)
    os.makedirs(val_dir, exist_ok=True)
    files = [f for f in os.listdir(src_dir) if f.lower().endswith(('.jpg','.png'))]
    random.shuffle(files)
    split_idx = int(len(files) * ratio)
    for f in files[:split_idx]:
        shutil.copy(os.path.join(src_dir, f), os.path.join(train_dir, f))
    for f in files[split_idx:]:
        shutil.copy(os.path.join(src_dir, f), os.path.join(val_dir, f))

if __name__ == '__main__':
    split('images/all/catA', 'images/train/catA', 'images/val/catA')
    split('images/all/catB', 'images/train/catB', 'images/val/catB')
    print("데이터 분할 완료: train 80%, val 20%")
```

터미널 : python scripts/split_data.py

d. 전처리 스크립트 : scripts/clean_images.py

```
import os
from PIL import Image
from imagededup.methods import PHash

def remove_small(root, min_w=200, min_h=200):
    for subset in ('train','val'):
        for cls in ('catA','catB'):
            folder = os.path.join(root, subset, cls)
            for fn in os.listdir(folder):
                fp = os.path.join(folder, fn)
                try:
                    w,h = Image.open(fp).size
                    if w<min_w or h<min_h:
                        os.remove(fp)
                except:
                    os.remove(fp)
```

```
def remove_dups(root, threshold=5):
    phasher = PHash()
    for subset in ('train','val'):
        for cls in ('catA','catB'):
            folder = os.path.join(root, subset, cls)
            dups = phasher.find_duplicates(
                image_dir=folder,
                max_distance_threshold=threshold
            )
            for orig, lst in dups.items():
                for dup in lst:
                    os.remove(os.path.join(folder, dup))

if __name__ == '__main__':
    base = 'images'
    print("1) 작은 이미지 제거…")
    remove_small(base)
    print("2) 중복 이미지 제거…")
    remove_dups(base)
    print("전처리 완료!")
```

터미널 :python scripts/clean_images.py

e. labelImg

visual studio에서는 라벨링 설치가 되지 않거나 오류가 걸렸으므로 https://github.com/tzutalin/labelImg/releases 에서 labelImg 프로그램을 다운받아서 실행

image 폴더의

```
|    ├── train/
|    |    ├── catA/          ← 학습용(80%) 이미지
|    |    └── catB/
|    └── val/
|         ├── catA/          ← 검증용(20%) 이미지
|         └── catB/
```

에 있는 파일들을 모두 네모박스 만들어서 labels 폴더의

```
├── labels/
|    ├── train/
|    |    ├── catA/          ← train 바운딩박스 라벨(.txt)
|    |    └── catB/
```

```
|    └── val/
|        ├── catA/          ← val 바운딩박스 라벨(.txt)
|        └── catB/
```
로 .txt를 만드는 과정

labelImg 화면 왼쪽:
① File > Open Dir > your_project/images/train
② File > Change Save Dir > your_project/labels/train
③ Yolo 선택
④ 각 이미지에 라벨링 한 뒤 저장

생성된 txt 파일 형식:
0 0.532 0.412 0.364 0.278
1 0.214 0.623 0.301 0.345

.txt 맨 앞 부분에 0 or 1이 아닌 15 or 16으로 생성됨 -> 15는 0으로 16은 1로 바꿔주기

f. 설정 파일 : data.yaml

```
train: images/train
val: images/val

# class 수와 이름
nc: 2
names:
  0: cat
  1: dog
```

g. YOLOv8 학습실행

환경변수설정 :set KMP_DUPLICATE_LIB_OK=TRUE

학습명령실행 : yolo detect train model=yolov8n.pt data=data.yaml epochs=50 imgsz=640 batch=16 project=runs/train name=catAB_exp

2. 결과 및 해석

a. Confidence Curve

: 모델이 검출한 모든 바운딩 박스의 “confidence score(신뢰도)” 분포를 시각화한 그래프로 어떤 confidence 값을 기준(threshold)으로 삼으면 성능-검출 수의 균형이 좋은지 파악할 때 사용

F1 스코어 : 정밀도와 재현율의 조화 평균 / Confidence Threshold : 신뢰도 임계값

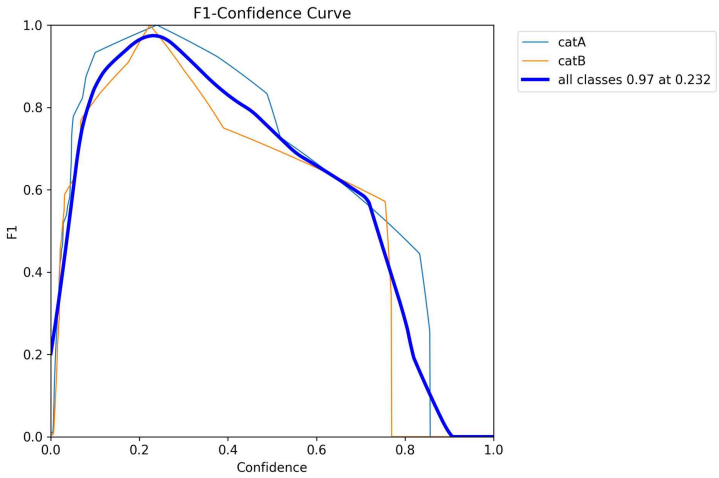

Fig 2. 신뢰도 분포 시각화

그래프 해석 :

Threshold가 0일 때 : 거의 모든 예측을 “유효”로 받아들여 재현율은 높아지지만, 정밀도가 급격히 떨어져 F1도 낮아진다.

Threshold가 0.25일 때 : 적절히 불확실한 예측을 걸러내고, 충분히 많은 객체를 잡아내며 정밀도 역시 유지하기 때문에 F1스코어가 최고점을 찍는다.

Threshold가 1일 때 : 아주 자신 있는 예측만 남기다 보니 정밀도는 높아지지만, 대부분의 진짜 객체를 놓쳐 재현율이 바닥을 치며 F1이 낮아진다.

confidence는 정밀도(Precision)와 비례하고 재현율(Recall)과 반비례함

Trade-off 조절

Precision이 더 중요할 시 Peak보다 조금 오른쪽(높은 Threshold)를 선택 : 오탐지 줄임

Recall이 더 중요할 시 Peak보다 왼쪽(낮은 Threshold)를 선택 : 놓치는 사례를 줄임

b. Precision-Confidence Curve

: confidence threshold 를 높여가며 그때그때의 정밀도(precision) 변화를 그린 곡선으로 “얼마만큼 확신하는 예측만 골라낼 때(예: conf ≥ 0.5), 얼마나 정확해지는가” 확인

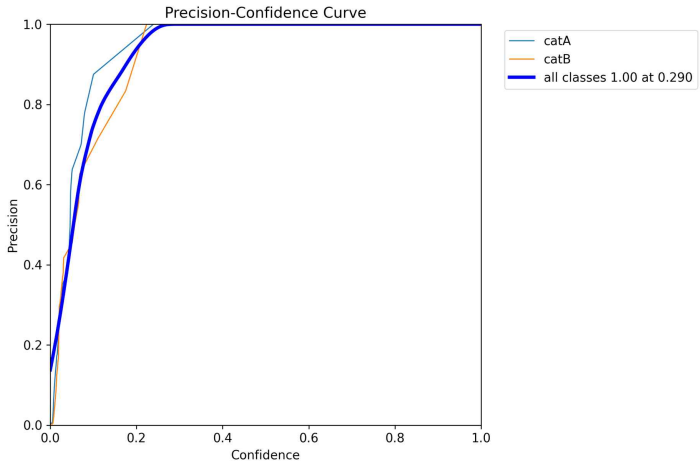

Fig 3. 정밀도 변화 곡선

c. Precision-Recall Curve

: 재현율(recall) 을 가로축, 정밀도(precision) 를 세로축으로 놓고 변화 추이를 그린 곡선으로 임계값(threshold) 선택에 따른 전체 트레이드오프(precision vs recall) 성능을 한눈에 파악

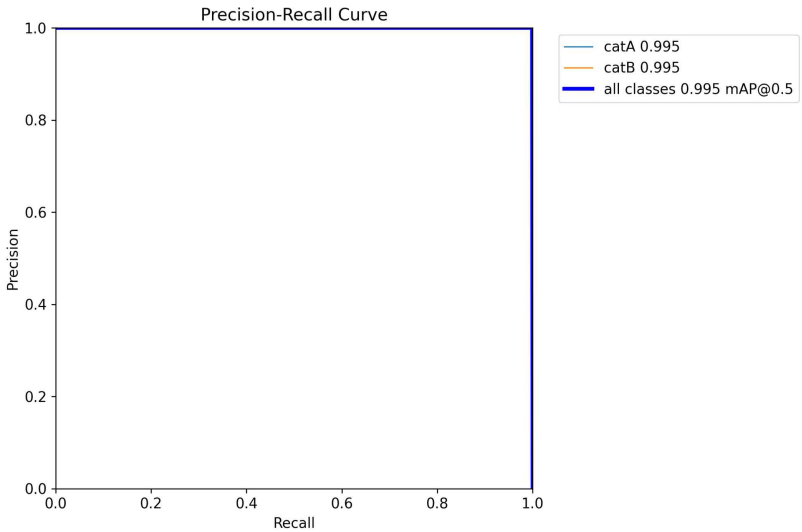

Fig 4. 변화 추이 곡선(정밀도-재현율)

d. Recall-Confidence Curve

: confidence threshold 를 높여가며 그때그때의 재현율(recall) 변화를 그린 곡선으로 “높은 확신만 골라내면 검출되는 객체 수(recall)는 얼마나 떨어지는가” 확인

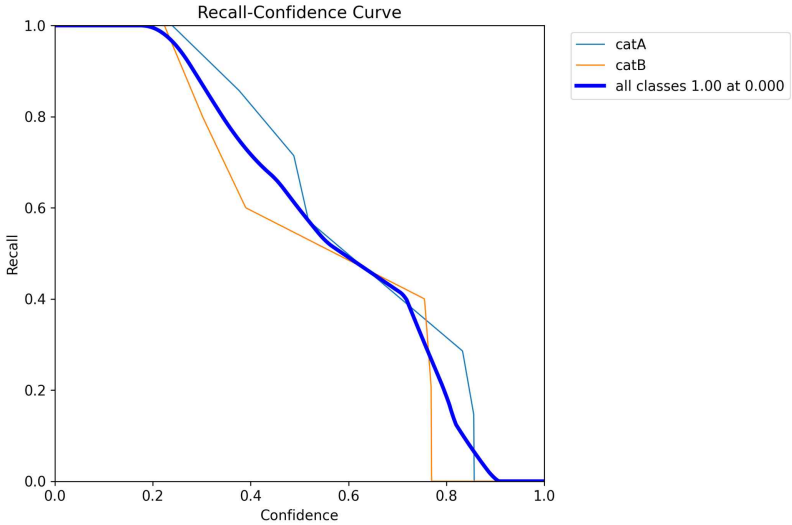

Fig 5. 재현율 변화 곡선

e. Confusion Matrix

: 실제 클래스(행) vs 예측 클래스(열) 의 개수 분포표(매트릭스)로 어떤 클래스가 다른 클래스로 잘못 예측되는지(오분류 패턴) 확인

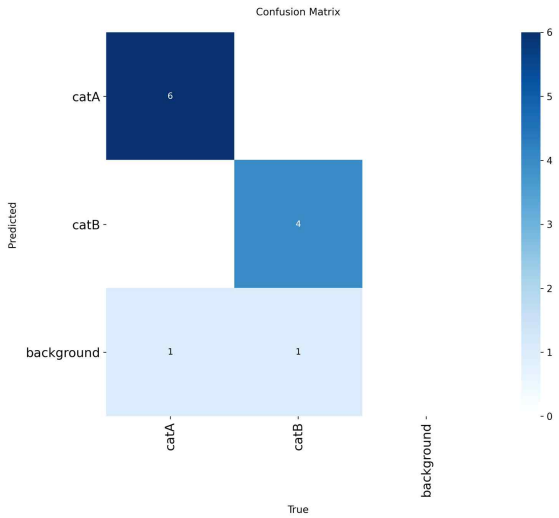

Fig 6. 매트릭스

f. Confusion Matrix Normalized

: 위 매트릭스를 각 행(실제 샘플 수) 기준으로 정규화(백분율)한 형태로 클래스별 샘플 수 차이를 보정하고, 오분류 비율을 직관적으로 비교

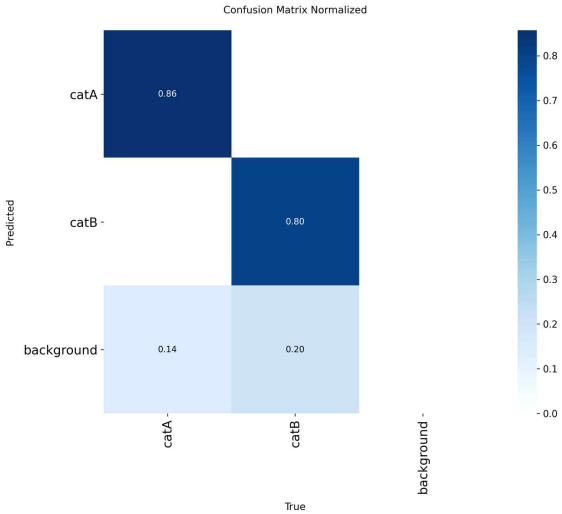

Fig 7. 매트릭스의 정규화

g. Labels Correlogram

: 클래스 간 상호 예측(“catA⇄catB”) 상관관계를 히트맵으로 시각화한 것으로 어떤 클래스 쌍 사이에 혼동(confusion)이 특히 심한지 파악

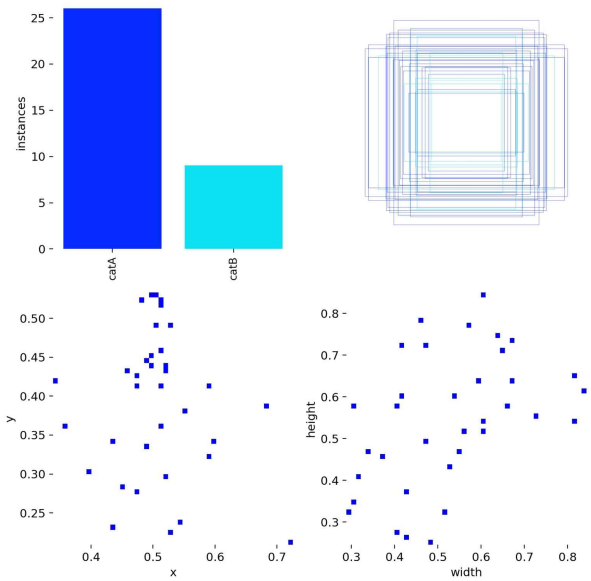

Fig 8. 상호 예측 상관관계 히트맵

h. Labels

: 학습·검증 데이터에서 각 클래스별 라벨(ground-truth) 분포를 보여 주는 바(Bar) 그래프로 클래스 불균형 여부, 각 클래스별 이미지·인스턴스 개수를 확인

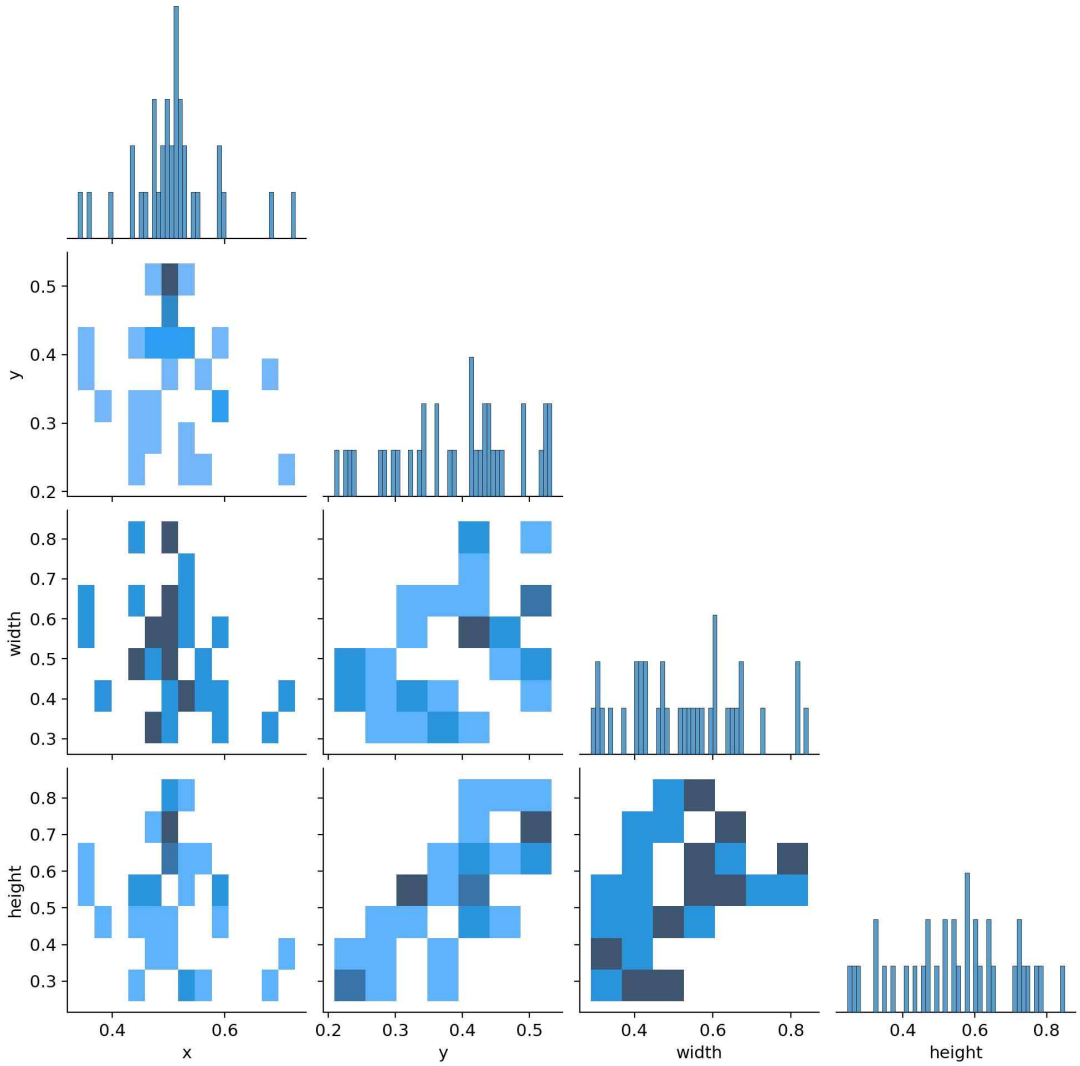

Fig 9. 라벨 분포

I. **results.csv**

: 학습 과정(epoch)마다 기록된 주요 지표(precision, recall, mAP, loss 등) 테이블을 CSV로 저장한 파일로 엑셀·판다스 등으로 열어 “어느 epoch에서 성능이 가장 좋았는지” 정밀하게 분석

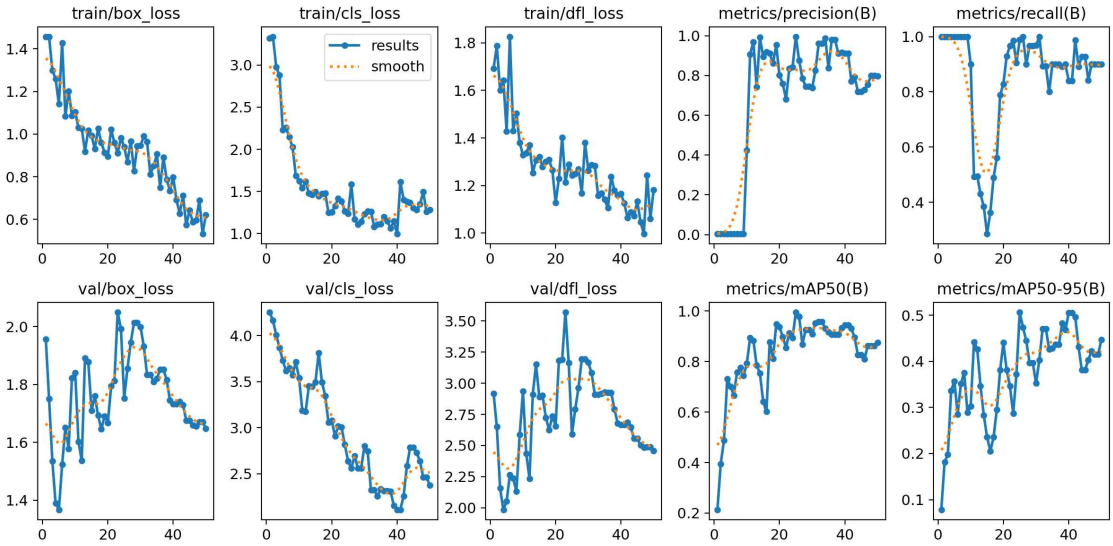

Fig 10. 학습 과정 기록 주요 지표

J. **results.png**

: results.csv 의 지표들을 시계열(epochs)로 그린 종합 플롯(학습 곡선) 이미지

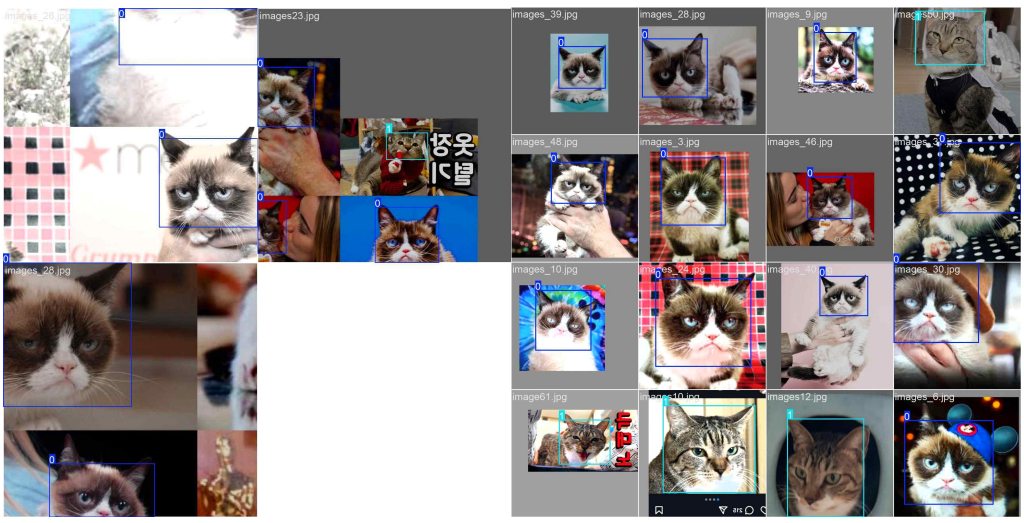

Fig 11. 라벨링 결과

+) 추론 스크립트 : scripts/infer.py

```
import cv2
from ultralytics import YOLO

if __name__ == '__main__':
    model = YOLO('runs/train/catAB_exp/weights/best.pt')

    # (1) 이미지 추론
    img = cv2.imread('images/val/catA/00001.jpg')
    res = model(img)[0]
    cv2.imshow('Result', res.plot())
    cv2.waitKey(0)

    # (2) 웹캠 실시간 추론
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret: break
        out = model(frame)[0]
        cv2.imshow('Webcam', out.plot())
        if cv2.waitKey(1) & 0xFF == 27: break
    cap.release()
    cv2.destroyAllWindows()
```

터미널 : python scripts/infer.py