YOLO를 활용한

CCTV Object Tracking

Deep Learning Project

3조 김미정 김예지 이기중 최재철

Contents

프로젝트 개요

프로젝트 목표

주어진 CCTV에서 지정된 인물을 다른 CCTV에서 검출 해낼 수 있는가?

한 개의 CCTV 영상에 등장한 특정 인물을 지정해 학습한 후

다른 CCTV에서 지정한 인물을 검출

데이터셋 소개

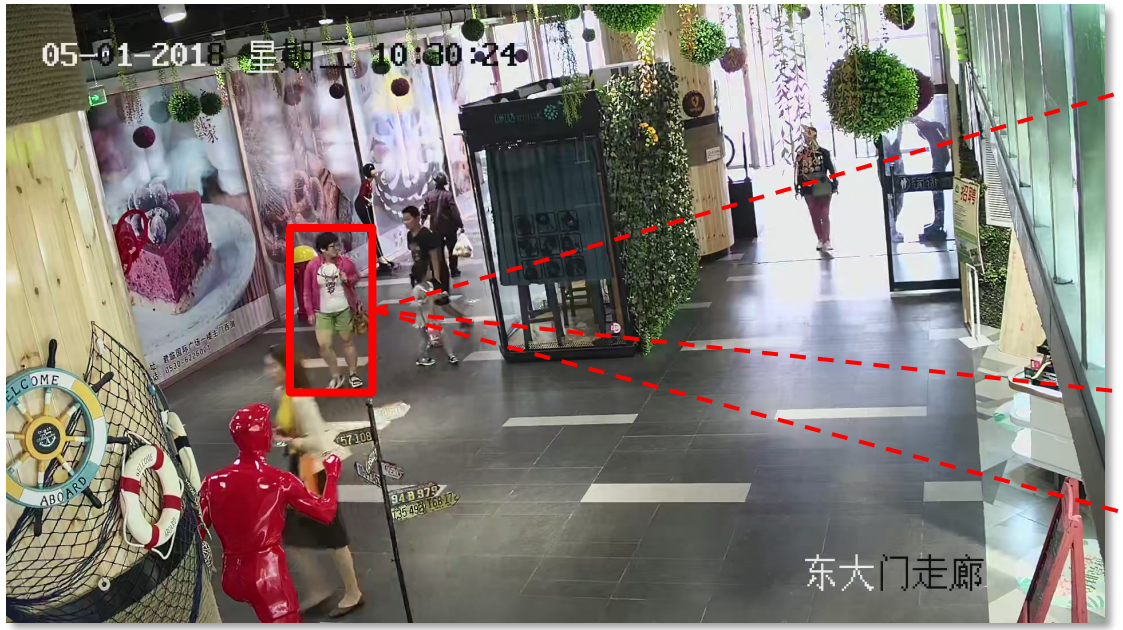

data
Multi-view_tracking_indoor
Multi-view_tracking_outdoor
00001_market
00002_market
... 10541_market
30006_street
30008_street
...35021_street
frame
json
samples_frames
video
frame
json
samples_frames
video
01
02
03
...
1-5
6-21
01
02
03
...
01
02
03
...
1-5
6-21
01
02
03
...

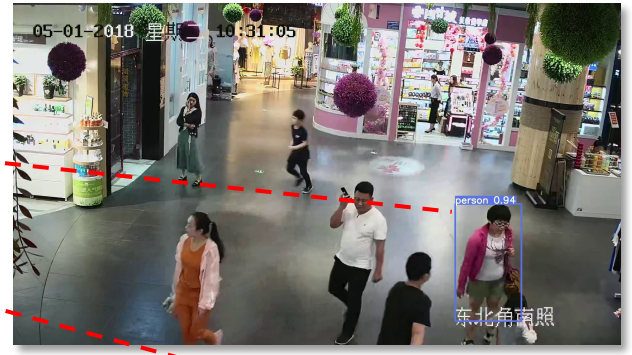
데이터셋 소개

videos	백화점 CCTV 176개 영상 & 야외 도로 CCTV 160개 영상
sample_frames	각 영상 마다 검지할 인물이 포함된 이미지 자료 포함 백화점 CCTV 50명 & 야외 도로 CCTV 50명
frames	백화점 영상 캡쳐 프레임 총 2314개 & 야외 도로 영상들에서 캡쳐된 프레임 총 1116개
json	대상의 class와 Bounding Box 좌표가 담겨있는 json 파일 클래스 (2개) : Pedestrian: 2571개 & Rider: 858개

프로젝트 컨셉

ex) Indoor_Market_00001 데이터





- 1개의 CCTV 영상에서 학습을 진행하여,
  나머지 CCTV 영상에서 해당 물체 검출 및 추적
- 추적하고자 하는 타겟 검출에 포커스

YOLO 모델 소개

Why YOLO?

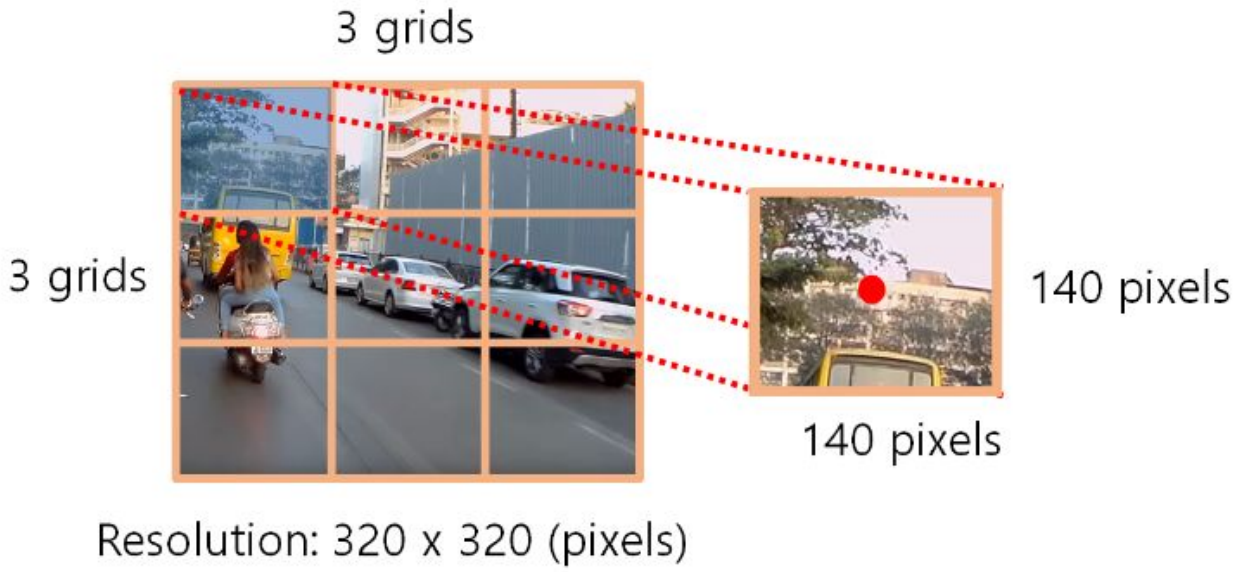
YOLO 이전의 Objec

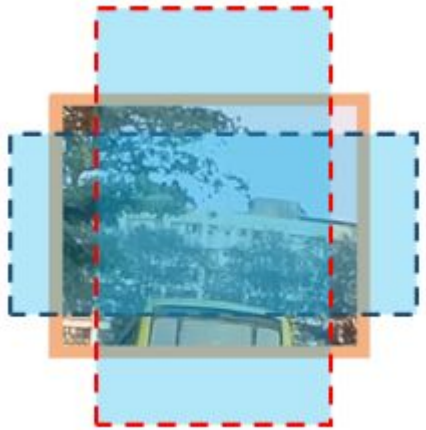
- Sliding window: c 곳곳을 지나다니면서
  S x S size의 픽셀에
- 문제점:
  - 연산량이 많고
  - 한 window에

YOLO 기본 개념

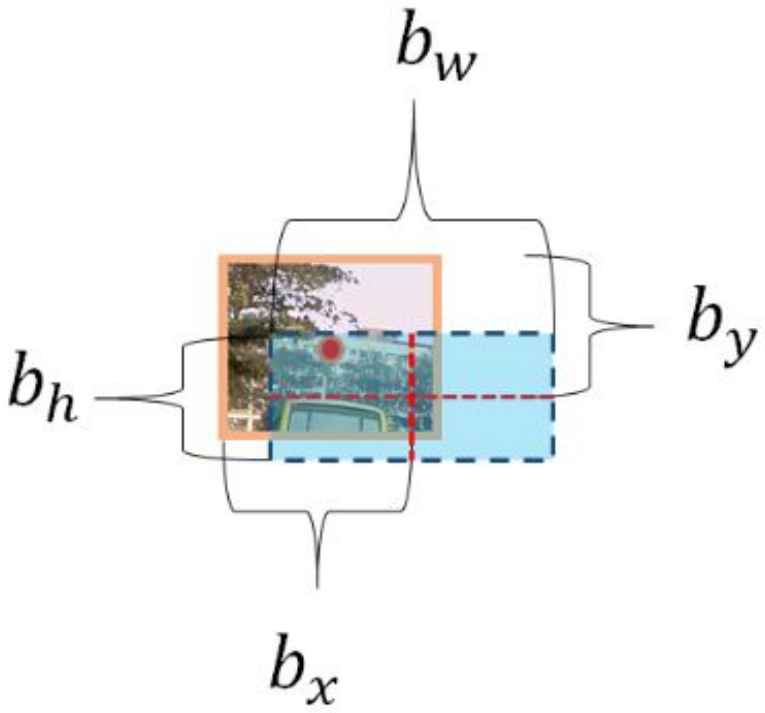
YOLO에는 grid와 anchor box를 도입하여 단점 보완

- Grid 분할: S x S의 grid cell로 분할하면 각 셀의 정보를 갖게 되며 각각 pixel size와 중앙점을 감게 됨
- Anchor box: 각각의 cell에 보통 1:1, 1:2, 2:1등의 비율이 정해져 있고 크기가 다른 박스들이 있음





YOLO 기본 개념



140 x 140 (pixels)

$$P_o, b_x, b_y, b_w, b_h$$

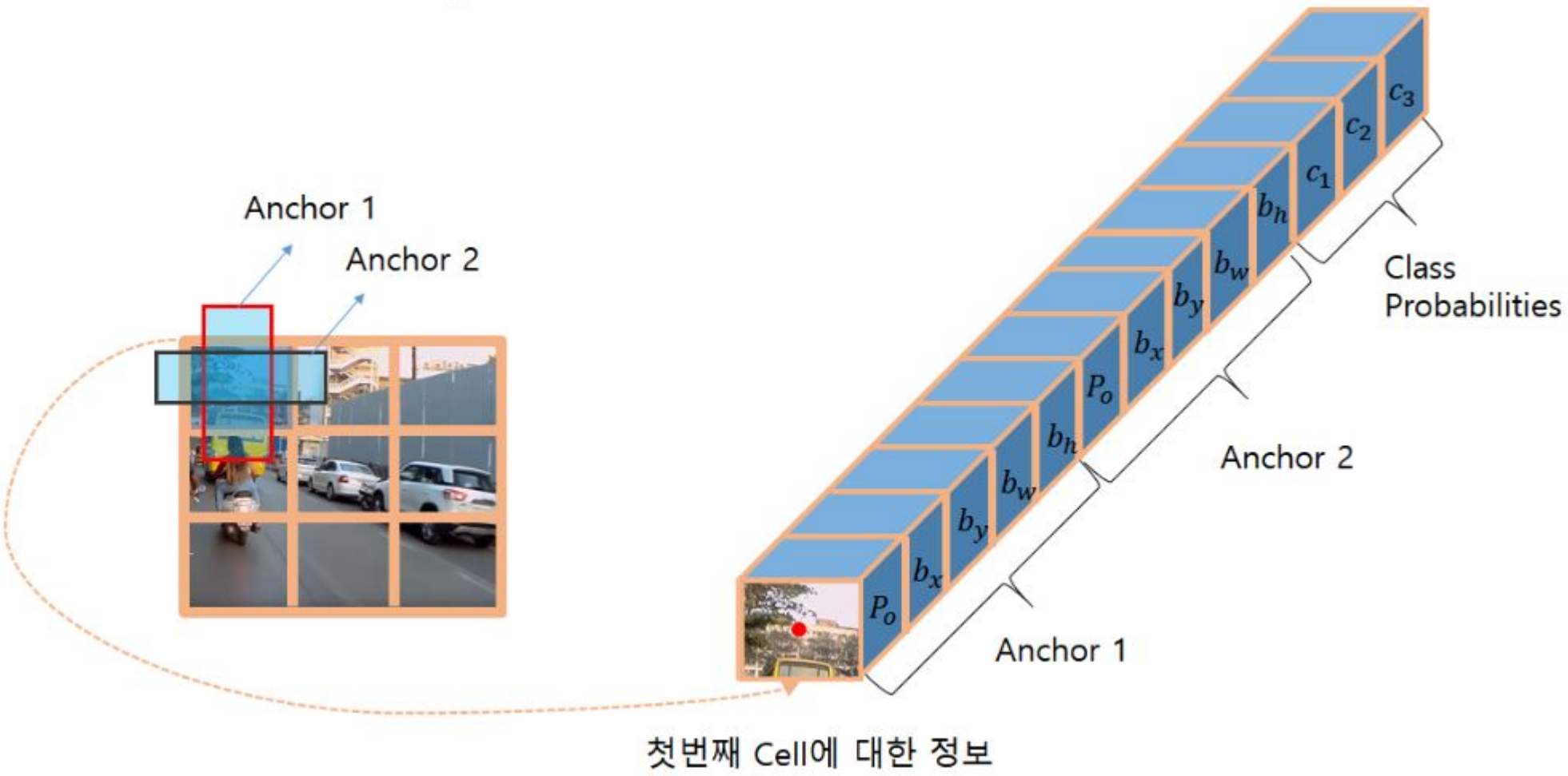
P_o: 해당 cell의 Objectness (물체가 있을 확률)

b_x, b_y: 해당 cell의 x, y 값

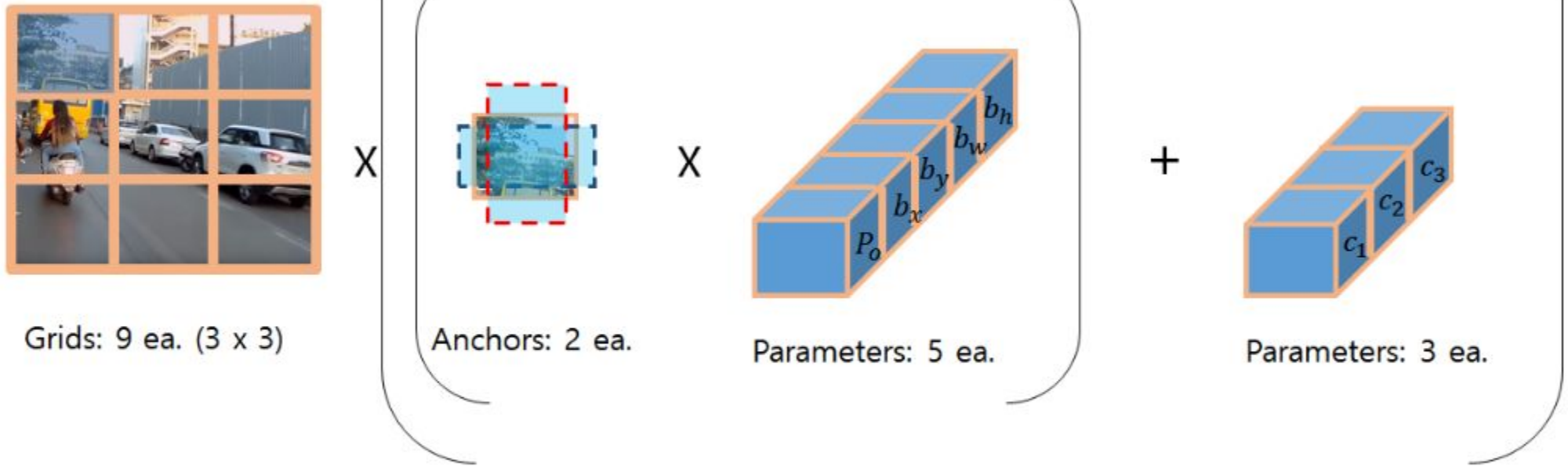
b_w, b_h: 해당 cell에 있는 Anchor Box의 w, h 값

(위 모든 값은 는 0~1의 값으로 normalize 됨)

YOLO 기본 개념


Anchor 1
Anchor 2
P_o
b_x
b_y
b_w
b_h
c_1
c_2
c_3
Class Probabilities
Anchor 2
Anchor 1
첫번째 Cell에 대한 정보

YOLO 기본 개념



최종 파라미터 개수 = 117 ea. (9 x ((2 x 5) + 3).

YOLO version

YOLO	최초의 one-stage object detection network
YOLO v2	7x7 그리드 맵에서 13x13으로 증가하여 small object 예측에 효과적, anchor box에 K-평균 클러스터링 적용
YOLO v3	multi-label을 위해 class prediction으로 logistic classifier을 이용
YOLO v4	Bag of Freebies, Bag of Specials 적용
YOLO v5	PyTorch 기반으로 더 많은 개발자들이 쉽게 이용 가능, 속도도 빨라지며 정확도 또한 좋아짐

프로젝트 진행

JSON -> TXT 전처리

data > Multi-view_tracking_indoor > 00001_market > json > 1-5 > 02 >


```
{
  "mark": [
    {
      "coordinates": [
        [
          1300.6516,
          139.90450000000007
        ],
        [
          1351.3968,
          139.90450000000007
        ],
        [
          1351.3968,
          297.48170000000005
        ],
        [
          1300.6516,
          297.48170000000005
        ]
      ],
      "label": {
        "Category": "pedestrian",
        "Clear face": "no",
        "Truncated": "non truncated",
        "Orientation": "left",
        "Occluded": "non occluded"
      }
    }
  ],
  "title": "00001_market/frames/02/00009.jpg"
}
```


(left, top) (right, top)
(left, bottom) (right, bottom)

labels_txt > Multi-view_tracking_indoor > 00001_market > json > 1-5 > 01 > 00002.txt

```
0 0.5802382031249995 0.0915860185185185 0.05251182291666664 0.183172037037037
```

center_x, center_y, w, h

Code

```
for filepath in json_filepath:
    with open(filepath) as json_file:
        # json 파일 로드, 파싱, yolo 좌표값으로변환
        json_data = json.load(json_file)
        center_x = str((json_data['mark'][0]['coordinates'][1][0] + json_data['mark'][0]['coordinates'][0][0]) / 2)
        center_y = str((json_data['mark'][0]['coordinates'][2][1] + json_data['mark'][0]['coordinates'][0][1]) / 2)
        w = str(json_data['mark'][0]['coordinates'][1][0] - json_data['mark'][0]['coordinates'][0][0])
        h = str(json_data['mark'][0]['coordinates'][2][1] - json_data['mark'][0]['coordinates'][0][1])

        # 좌표값 저장
        data = str("0 {} {} {} {}".format(center_x, center_y, w, h))

        # 레이블 텍스트 저장할 경로 설정
        label_path = './labels' + filepath[6:-4:] + 'txt'

        # 텍스트 쓰기
        f = open(label_path, 'w')
        f.write(data)
        f.close()
```

제공된 json 라벨링 파일을 yolo training에 적합한 txt 파일로 변환 작업 진행

Image Augmentation

실제 CCTV 데이터 특성에 맞추어 Image Augmentation 진행

원본

좌우 반전

밝기 조절
(밝게)

밝기 조절
(어둡게)

CoarseDropout

ShearX

MotionBlur

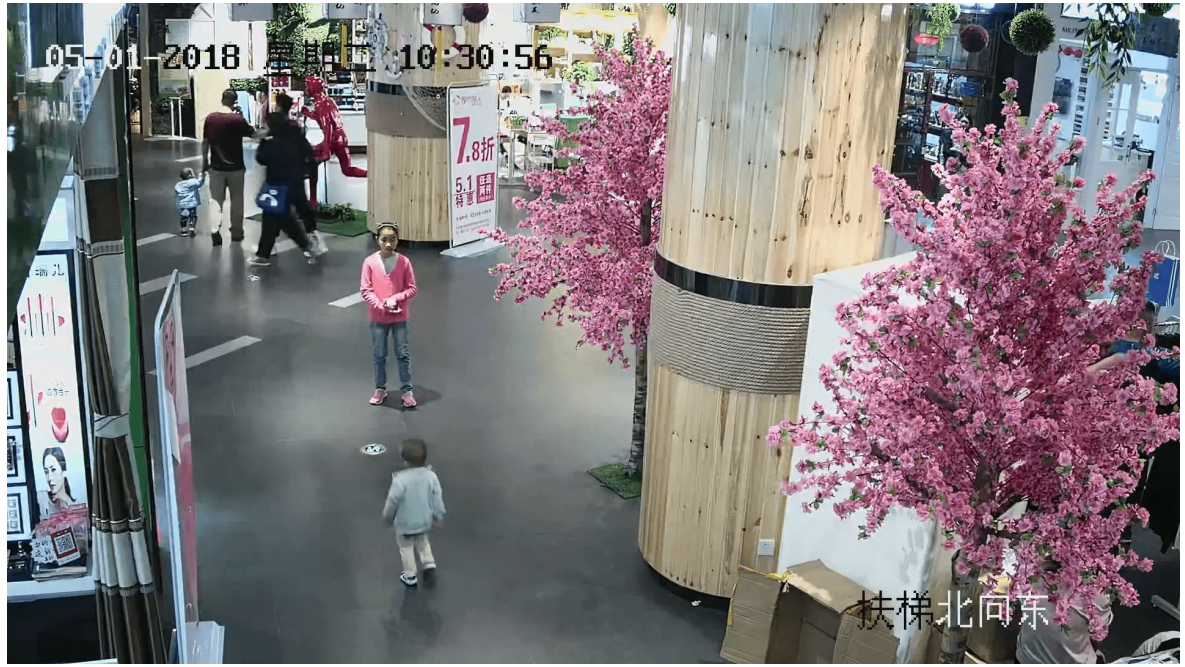
YOLO v3

YOLOv3-tiny

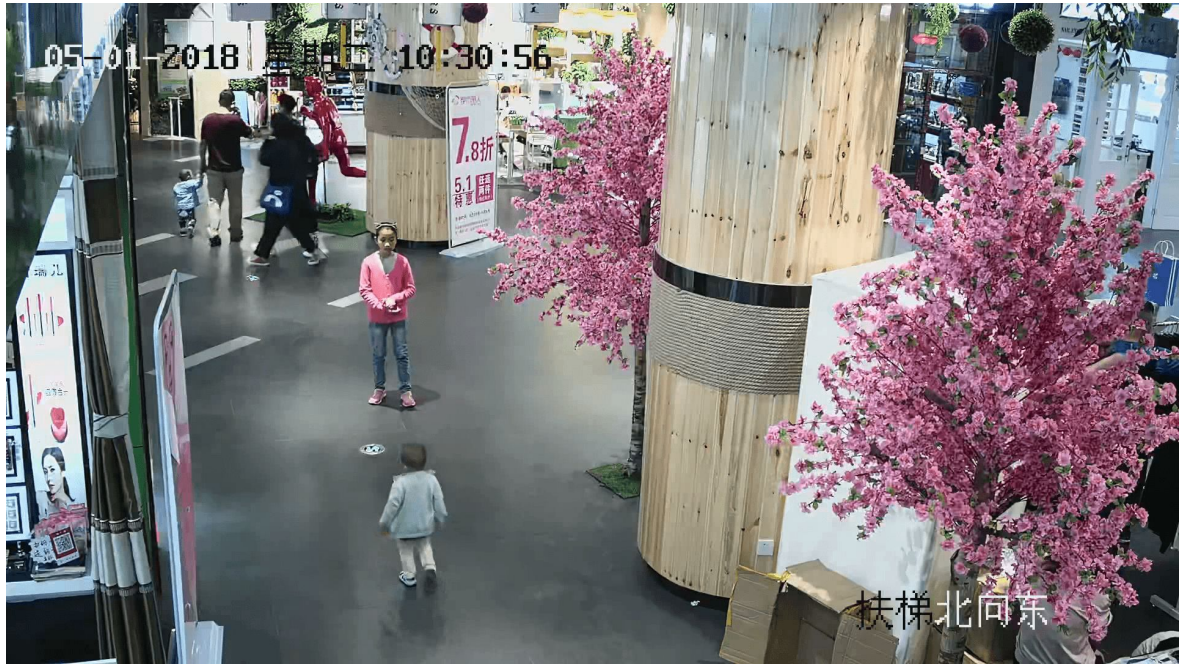
YOLOv3

YOLO v4 + DeepSORT

YOLOv3-tiny



YOLOv4 + DeepSORT



5개의 파일 중 3개의 폴더 즉 원본 frame 39장을 증강 후 학습시켜 테스트한 결과

버전별 비교

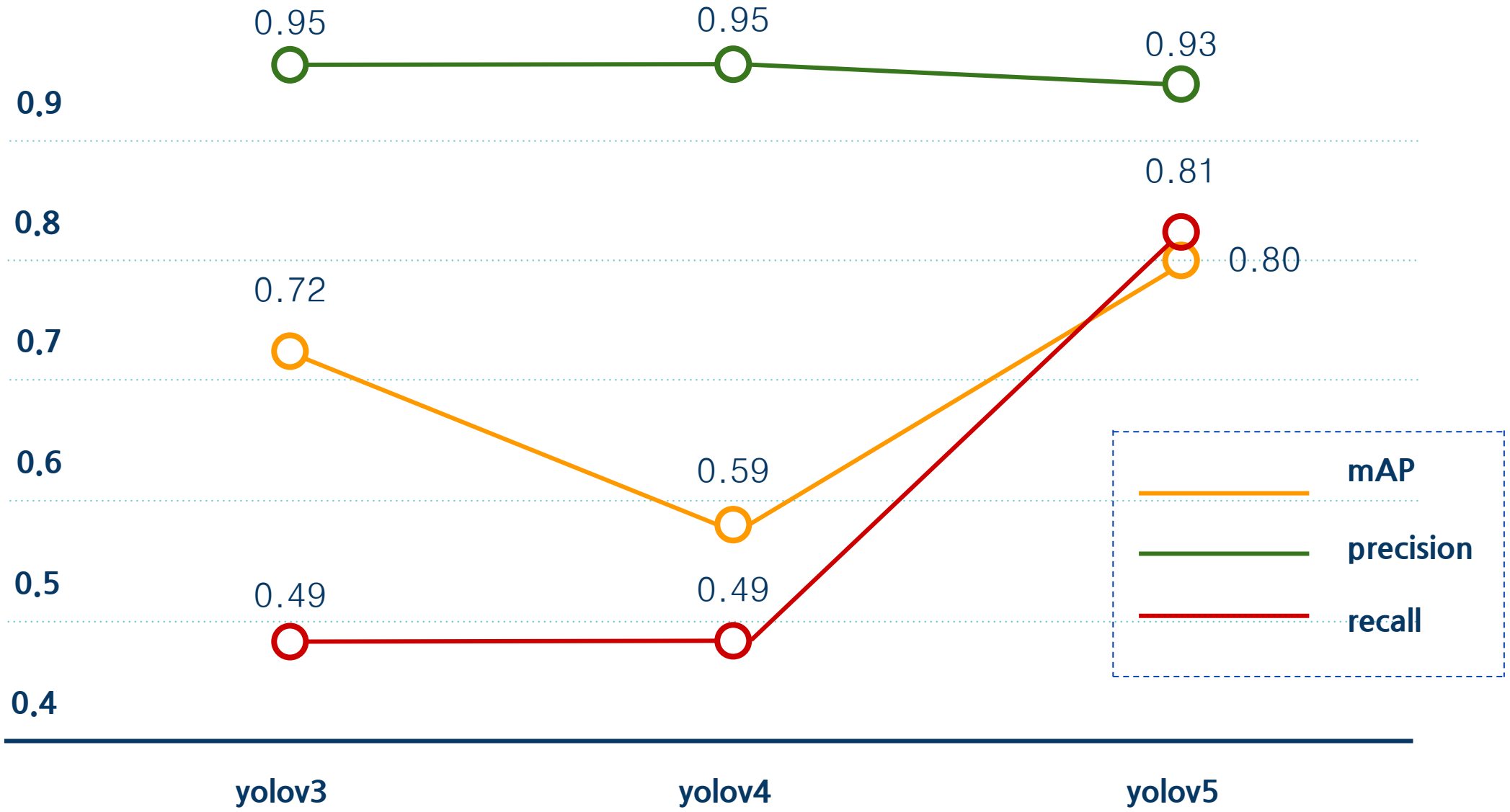
yolov3

yolov4

yolov5

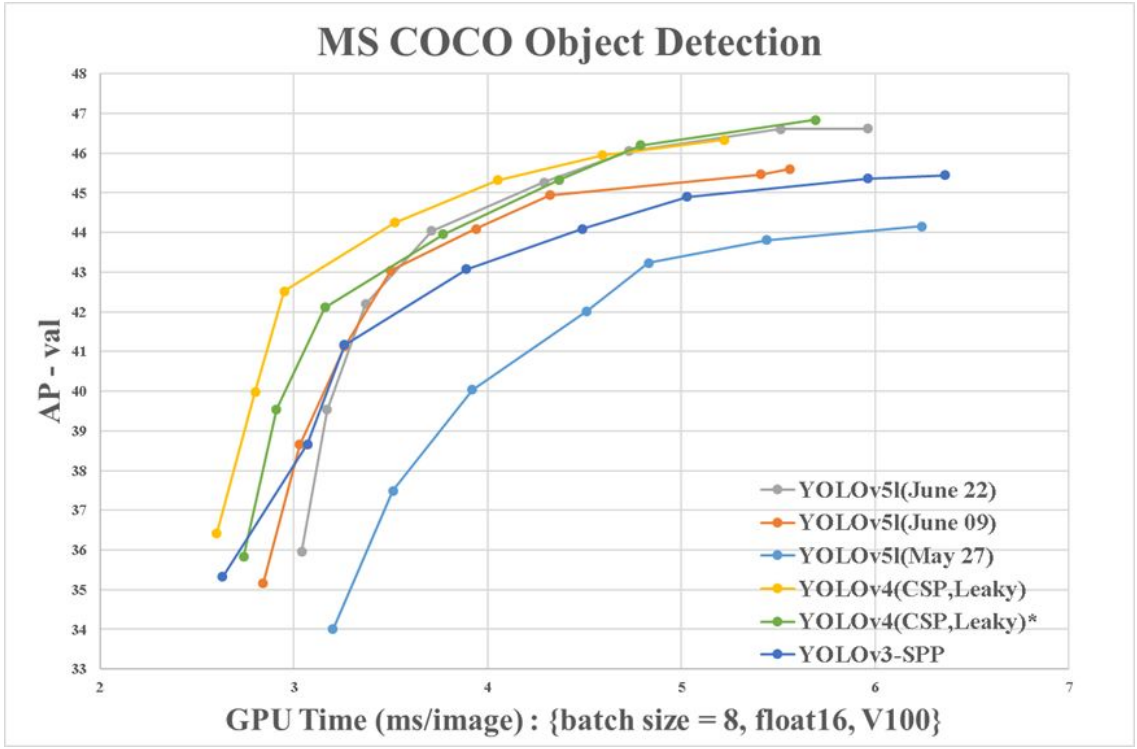
5개의 폴더 중 1개의 폴더에 있던 17장의 frame을 68장으로 증강 하여 학습 후 테스트한 결과

버전별 비교


0.95
0.95
0.93
0.9
0.81
0.8
0.80
0.72
0.7
0.6
0.59
0.5
0.49
0.49
0.4
mAP
precision
recall
yolov3
yolov4
yolov5

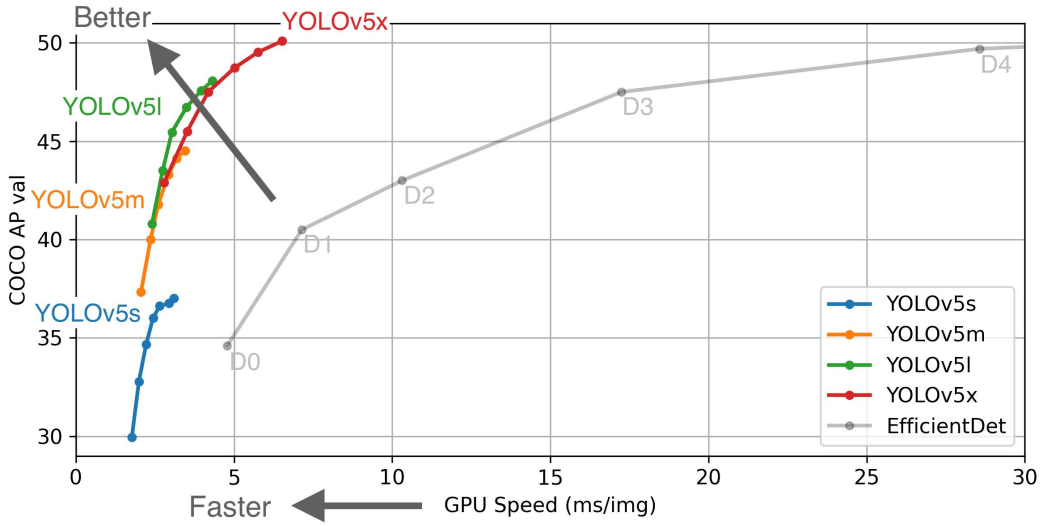
YOLO v5

YOLO 버전별 성능



출처: https://github.com/pjreddie/darknet/issues/2198

YOLOv5 모델 크기별 성능



Model	size	AP^{val}	AP^{test}	AP_{50}	$Speed_{V100}$	FPS_{V100}	params	GFLOPS
YOLOv5s	640	36.8	36.8	55.6	**2.2ms**	**455**	7.3M	17.0
YOLOv5m	640	44.5	44.5	63.1	2.9ms	345	21.4M	51.3
YOLOv5l	640	48.1	48.1	66.4	3.8ms	264	47.0M	115.4
YOLOv5x	640	**50.1**	**50.1**	**68.7**	6.0ms	167	87.7M	218.8
YOLOv5x + TTA	832	**51.9**	**51.9**	**69.6**	24.9ms	40	87.7M	1005.3

출처: https://github.com/ultralytics/yolov5

YOLO v5 settings

이미지 증강

- YOLO v5에서 밝기와 mosaic 등의 증강을 기본 제공
- 기존 이미지에서 flip, rotation, shear, coarse dropout만 사용

ex) yolov5 train_batch

앵커박스 최적화

- COCO dataset에 pre-trained된 anchor box 값을 우리 데이터에 맞게 최적화

```
from utils.autoanchor import *
_ = kmean_anchors(path='./data.yaml', n=9, img_size=640, thr=4.0, gen=1000, verbose=True)

Scanning 'project/train.cache' for images and labels... 56 found, 0 missing, 0 empty, 0 corrupted: 100%|
autoanchor: Evolving anchors with Genetic Algorithm:: 0%| | 0/1000 [00:00<?, ?it/s]autoanchor: Running kme
autoanchor: thr=0.25: 1.0000 best possible recall, 9.00 anchors past thr
autoanchor: n=9, img_size=640, metric_all=0.728/0.968-mean/best, past_thr=0.728-mean: 24,60,  22,64,  27,64,  27,72,
autoanchor: thr=0.25: 1.0000 best possible recall, 9.00 anchors past thr
autoanchor: n=9, img_size=640, metric_all=0.728/0.968-mean/best, past_thr=0.728-mean: 24,60,  22,65,  28,64,  27,72,
autoanchor: Evolving anchors with Genetic Algorithm: fitness = 0.9685: 100%| 1000/1000 [00:00<00:00,
autoanchor: n=9, img_size=640, metric_all=0.728/0.969-mean/best, past_thr=0.728-mean: 24,60,  22,65,  27,64,  27,72,
autoanchor: thr=0.25: 1.0000 best possible recall, 9.00 anchors past thr
autoanchor: n=9, img_size=640, metric_all=0.728/0.969-mean/best, past_thr=0.728-mean: 24,60,  22,65,  27,64,  27,72,
```

Single Detect

- 컨피던스가 높은 1개의 타겟만 검출하도록 general.py 소스 변경

```
# Settings
min_wh, max_wh = 2, 4096  # (pixels) minimum and maxi
max_det = 1  # maximum number of detections per image
max_nms = 30000  # maximum number of boxes into torch
time_limit = 10.0  # seconds to quit after
```

Confidence threshold

- 컨피던스가 낮은 다른 물체 detect를 막기 위해 0.25(default) -> 0.6으로 confidence threshold 상향

```
!LD_LIBRARY_PATH=/usr/local/cuda/lib64/ python3 detect.py
--source ../../project/indoor_01.mp4 \
--weights ./runs/train/xlarge_with_aug_anchor2/weights/best.pt
--conf-thres 0.6
```

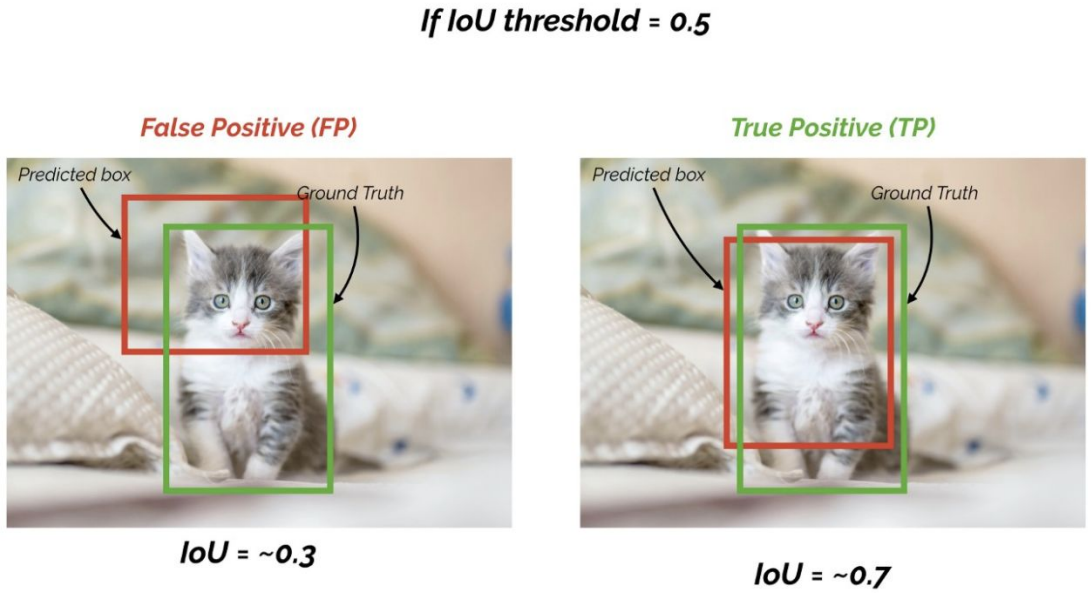
프로젝트 결과

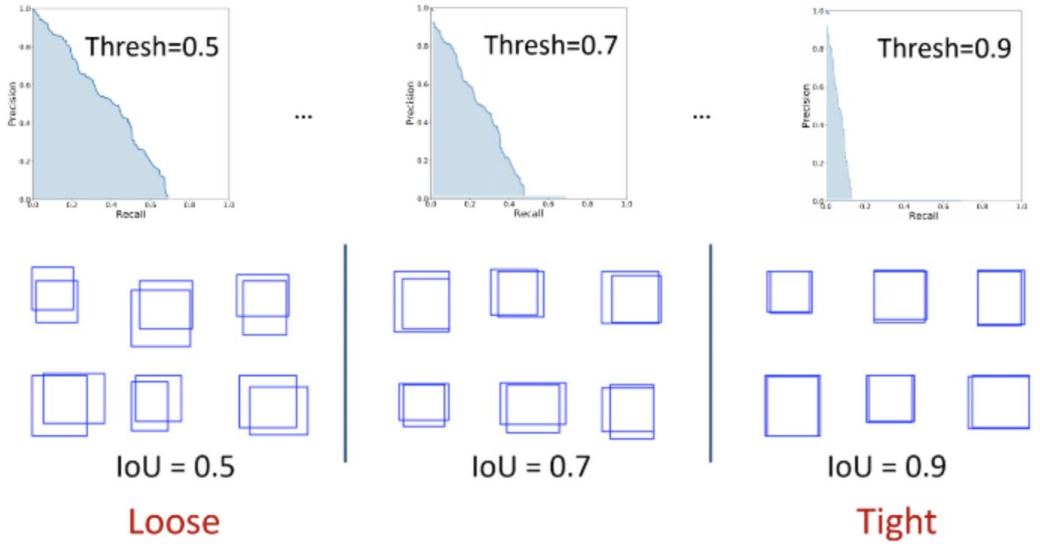
성능 평가 지표 - 1

1. mAP

(1) $$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

(2) $$[.5:.95]mAP = \frac{mAP_{0.50} + mAP_{0.55} + \ldots + mAP_{0.95}}{10}$$





성능 평가 지표 - 2

2. 자체 수기 산출

1. 기존 모델에서 제공하는 mAP 지표의 한계 존재
   - mAP 값이 높아도 물체를 잘 Tracking 하는 것은 아니었음(타물체 오감지 / 중복감지 문제 발생)
   - 추적보다는 검출에 포커스

2. 초당 3프레임 간격으로 프레임 샘플링

3. 샘플링 프레임을 바탕으로 수기로 산출

Indoor_Market_00001

영상 길이 00초, 총 00개 프레임

		ACTUAL CLASS	
		Target O	Target X
PREDICTED CLASS	Box O		
	Box X		

$$\text{Accuracy} = \frac{\quad}{\quad}$$

$$\text{Precision} = \frac{\quad}{\quad}$$

$$\text{Recall} = \frac{\quad}{\quad}$$

$$\text{F1 Score} = 2 * \frac{P * R}{P+R}$$

Target O : 타겟이 나오는 프레임
Target X : 타겟이 나오지 않는 프레임
Box O : 박스를 그린 프레임
Box X : 박스를 그리지 않은 프레임

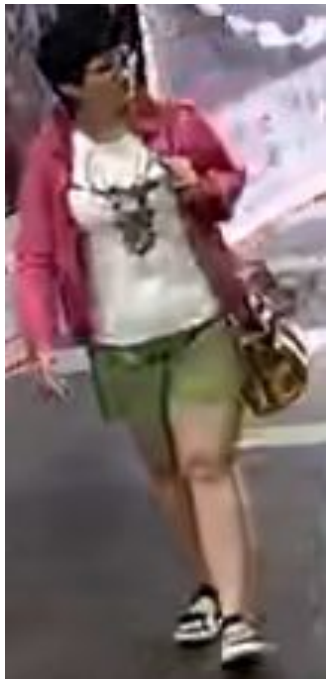
*프레임 내에 타겟이 있으나, 다른 물체에 박스를 그린 프레임은 Target O+Box X 프레임으로 카운트 (False Negative)

테스트 데이터셋

인물의 다양성 고려하여 성인 남성/여성, 어린이 남성/여성 총 4명의 테스트 타겟 선정

Indoor_market

Target 1



market_00001
핑크 자켓 여성

Target 2

market_06277
빨간 줄무늬 남자아이

Outdoor_street

Target 3

street_30009
중절모 남성

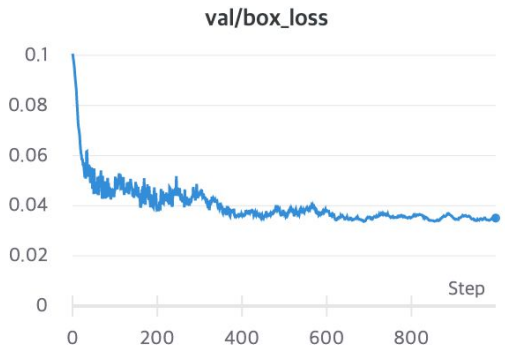
Target 4

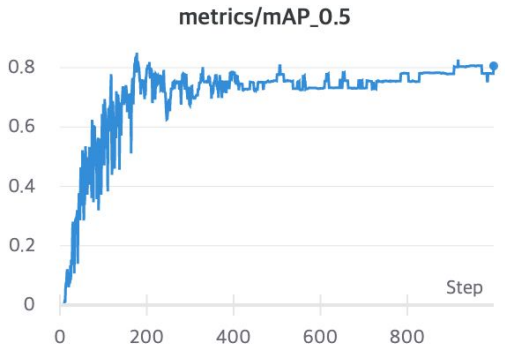
street_30012
분홍티 여자아이

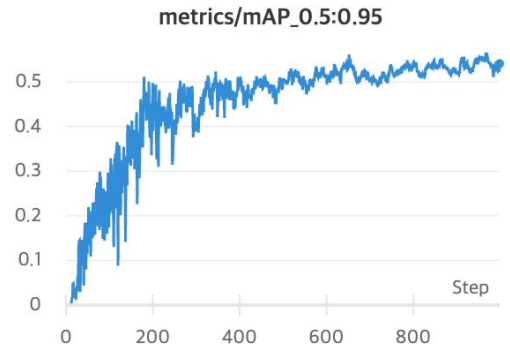
테스트 결과

Target 1 : Metrics

market_00001
핑크 자켓 여성







영상 길이 44초, 총 111개 프레임

		ACTUAL CLASS	
		Target O	Target X
PREDICTED CLASS	Box O	47	0
	Box X	44	20

$$\text{Accuracy} = \frac{47 + 20}{47 + 0 + 44 + 20} = 0.6036$$

$$\text{Precision} = \frac{47}{47 + 0} = 1.0000$$

$$\text{Recall} = \frac{47}{47 + 44} = 0.5165$$

$$\text{F1 Score} = 2 * \frac{P * R}{P + R} = 0.6812$$

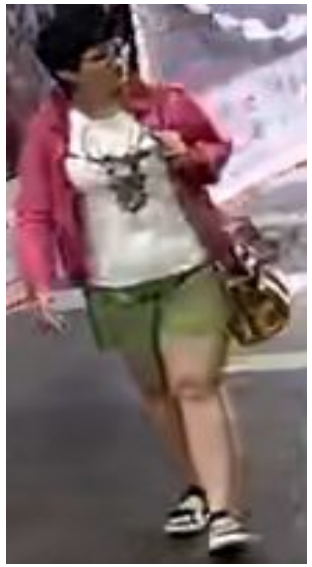
best weight 산출 방식 (yolov5 default):

```
w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
```

테스트 결과

Target 2 : Tracking Result

학습 영상

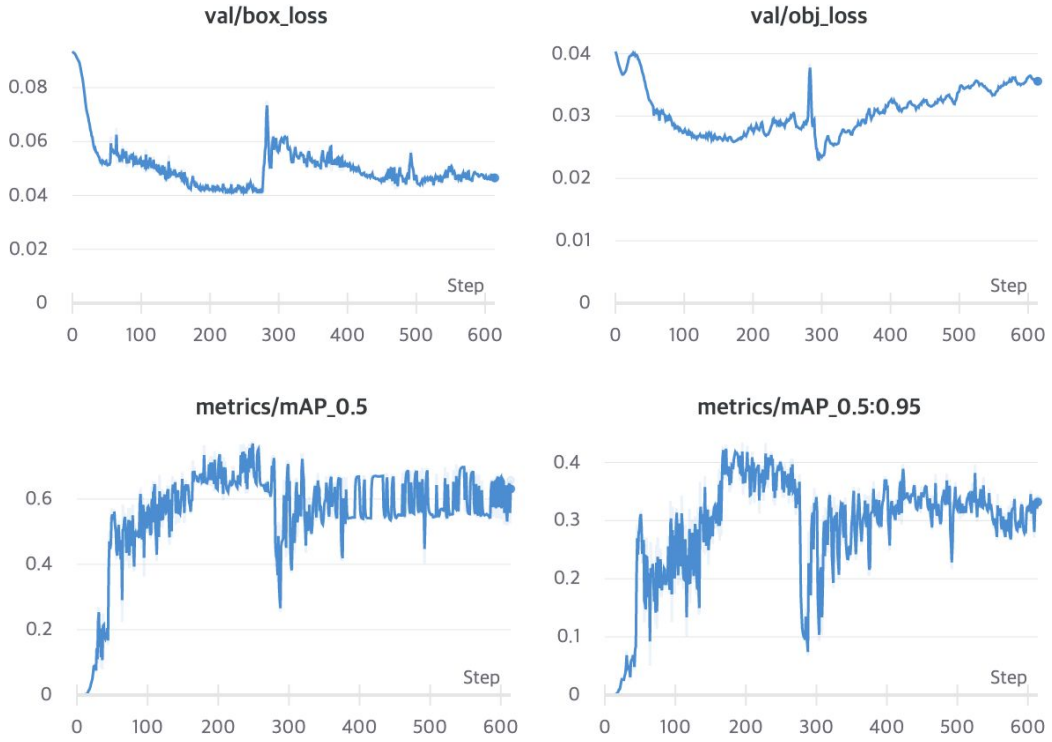


market_00001
핑크 자켓 여성

테스트 결과

Target 2 : Metrics

market_06277
빨간 줄무늬 남자아이



영상 길이 17초, 총 41개 프레임

		ACTUAL CLASS	
		Target O	Target X
PREDICTED CLASS	Box O	15	0
	Box X	10	16

$$\text{Accuracy} = \frac{15 + 16}{15 + 0 + 10 + 16} = 0.7561$$

$$\text{Precision} = \frac{15}{15 + 0} = 1.0000$$

$$\text{Recall} = \frac{15}{15 + 10} = 0.6000$$

$$\text{F1 Score} = 2 * \frac{P * R}{P + R} = 0.7500$$

best weight 산출 방식 (yolov5 default):

```
w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
```

테스트 결과

Target 2 : Tracking Result

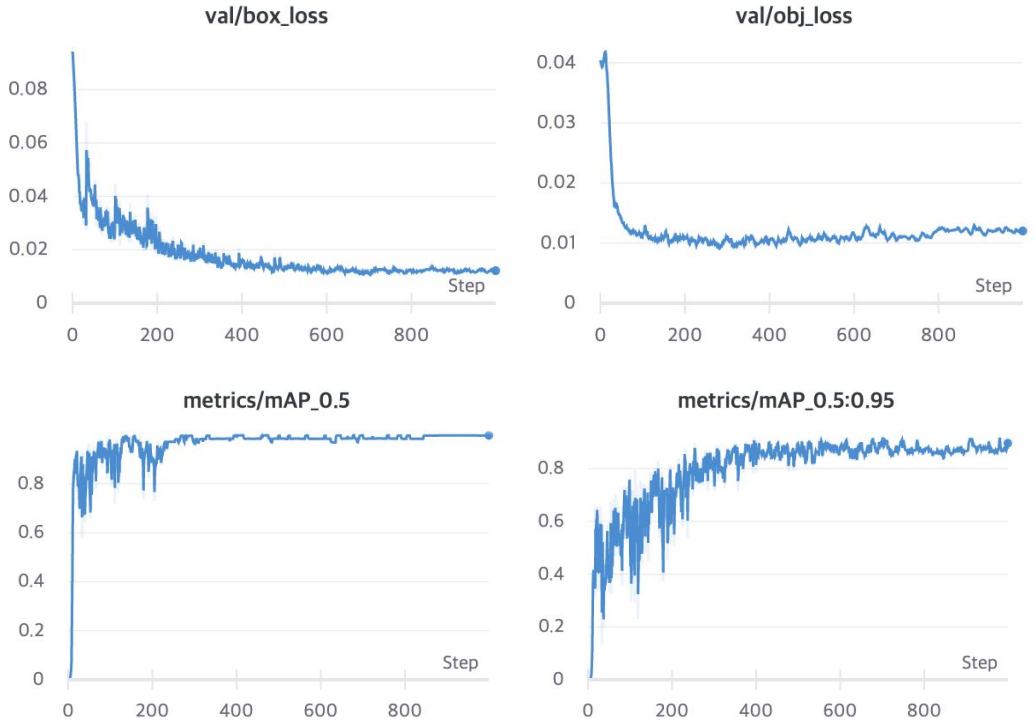
학습 영상

market_06277
빨간 줄무늬 남자아이

테스트 결과

Target 3 : Metrics

street_30009
중절모 남성



영상 길이 23초, 총 66개 프레임

		ACTUAL CLASS	
		Target O	Target X
PREDICTED CLASS	Box O	51	5
	Box X	0	10

$$\text{Accuracy} = \frac{51 + 10}{51 + 5 + 0 + 10} = 0.9242$$

$$\text{Precision} = \frac{51}{51 + 5} = 0.9107$$

$$\text{Recall} = \frac{51}{51 + 0} = 1.0000$$

$$\text{F1 Score} = 2 * \frac{P * R}{P + R} = 0.9533$$

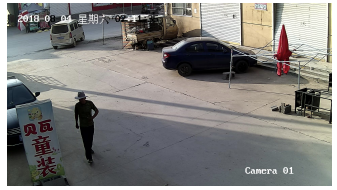
best weight 산출 방식 (yolov5 default):

```
w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
```

테스트 결과

Target 3 : Tracking Result

학습 영상

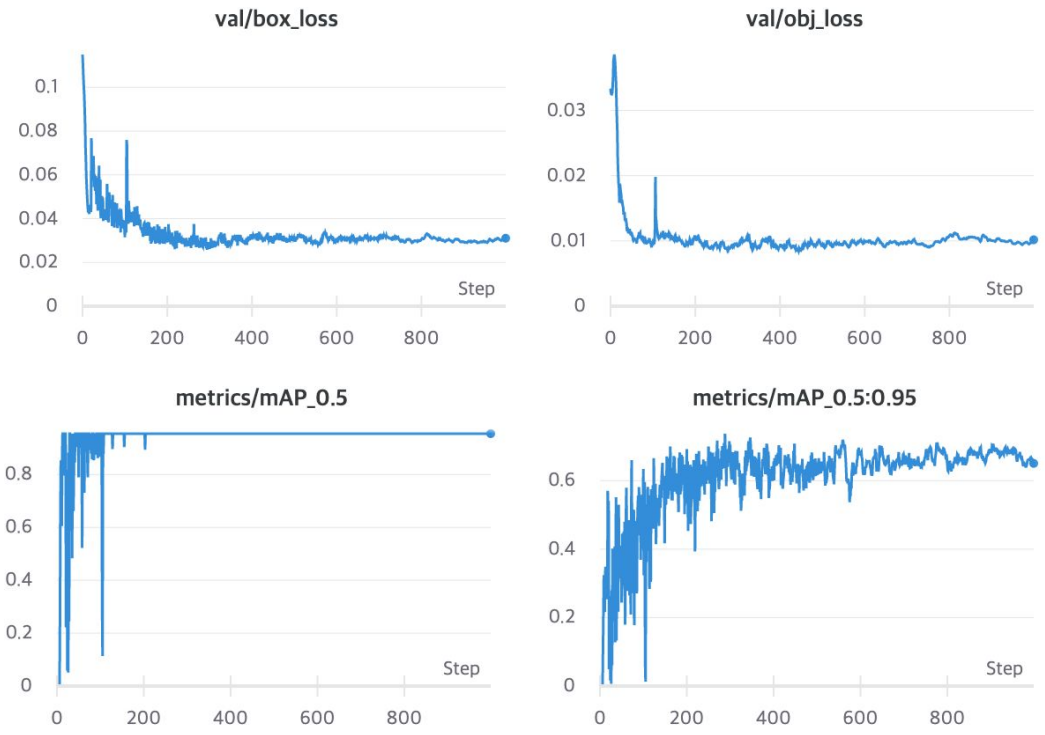


street_30009
중절모 남성

테스트 결과

Target 4 : Metrics

street_30012
분홍티 여자아이



영상 길이 25초, 총 62개 프레임

		ACTUAL CLASS	
		Target O	Target X
PREDICTED CLASS	Box O	43	0
	Box X	6	13

$$\text{Accuracy} = \frac{43 + 13}{43 + 0 + 6 + 13} = 0.9032$$

$$\text{Precision} = \frac{43}{43 + 0} = 1.0000$$

$$\text{Recall} = \frac{43}{43 + 6} = 0.8776$$

$$\text{F1 Score} = 2 * \frac{P * R}{P + R} = 0.9348$$

best weight 산출 방식 (yolov5 default):

```
w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
```

테스트 결과

Target 4 : Tracking Result

학습 영상

street_30012
분홍티 여자아이

마치며...

배운 점

- 다양한 이미지 증강 기법
- Object detection 알고리즘의 발전
- YOLO 모델의 원리 학습
- IoU, mAP 등 Object detection 평가 지표에 대한 이해

개선할 점

- yolo 소스 코드 분석을 통한 데이터에 최적화 된 튜닝
- 보다 Target에 특화된 Custom Training을 통한 모델 개선
- 모든 시도는 소중하니, 결과를 기록하는 습관 개선 (별 5개)

Thanks

취뽀 화이팅