

편의점 CCTV영상 내 절도 범죄 탐지 시스템



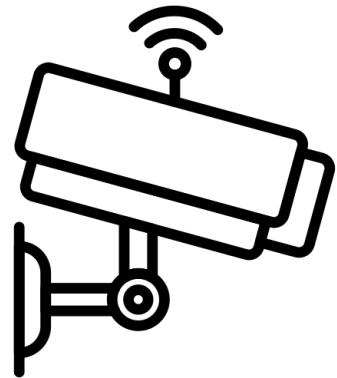
독수리 오형제

팀장: 주원석

팀원: 박진서, 박진석, 심주성, 홍수만



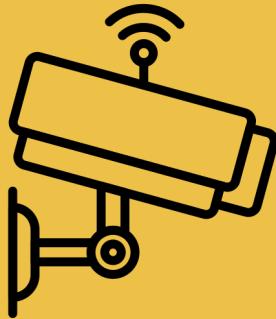
01 프로젝트 개요
주제 소개



EAGLE EYE

편의점 CCTV 영상 내의
절도 범죄 탐지 시스템

시스템 개요



EAGLE EYE

편의점 CCTV 영상 내의
절도 범죄 탐지 시스템



편의점에서 주로 발생하는
도난으로 예상되는
두 가지 절도 행위 분류

- ✓ 1. 호주머니에 물품을 넣는 행위
- ✓ 2. 가방에 물품을 넣는 행위



02 프로젝트 분석 설계 개발 환경



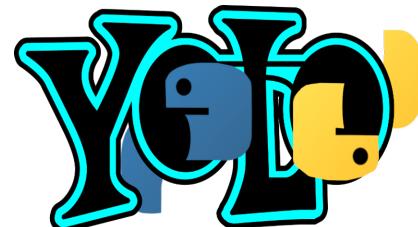
라이브러리



PyTorch

파이토치(Pytorch)는 파이썬(Python) 기반의 오픈 소스 머신러닝 라이브러리로 딥러닝을 구현하기 위해 적합하다.

인공지능



YOLO는 **딥러닝 기반 물체 인식 알고리즘**으로 매우 빠른 객체 검출 속도라는 장점을 가지고 있어 이상 행동 감지시스템에 적합하다.

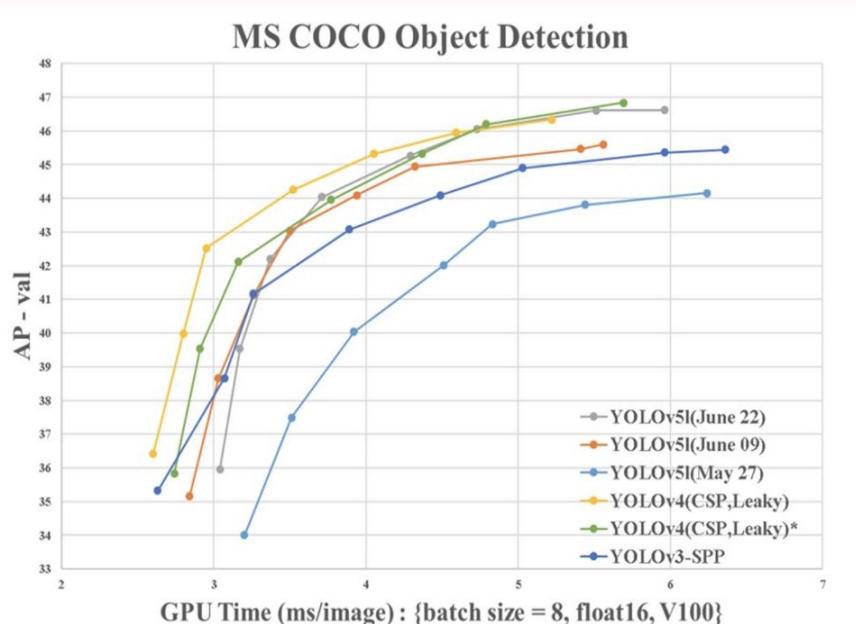
서버



정보통신산업진흥원
National IT Industry Promotion Agency

NIPA(정보통신산업진흥원)의 GPU기반 인공지능 고성능 컴퓨팅 자원 및 개발 인프라를 제공받아 우분투 서버에서 시스템 개발을 진행함

YOLO v5를 선택한 이유



- ✓ 기존 object detection과 달리 YOLO는 One-shot 감지로 정확도가 조금은 떨어지지만 **매우 빠른 객체 검출 속도**를 가지고 있기 때문에 즉각적으로 대처해야 하는 이상 행동 감지 시스템 특성상 적절하다.
- ✓ Darknet을 사용하는 이전 버전들과는 달리 Version 5는 **PyTorch 구현**이기 때문에 더 쉽게 환경을 구성하고 구현할 수 있다.
- ✓ 제일 최근에 나온 버전5는 낮은 용량과 빠른 속도를 가지고 있다. 즉, **제일 좋은 성능을** 가지고 있다.

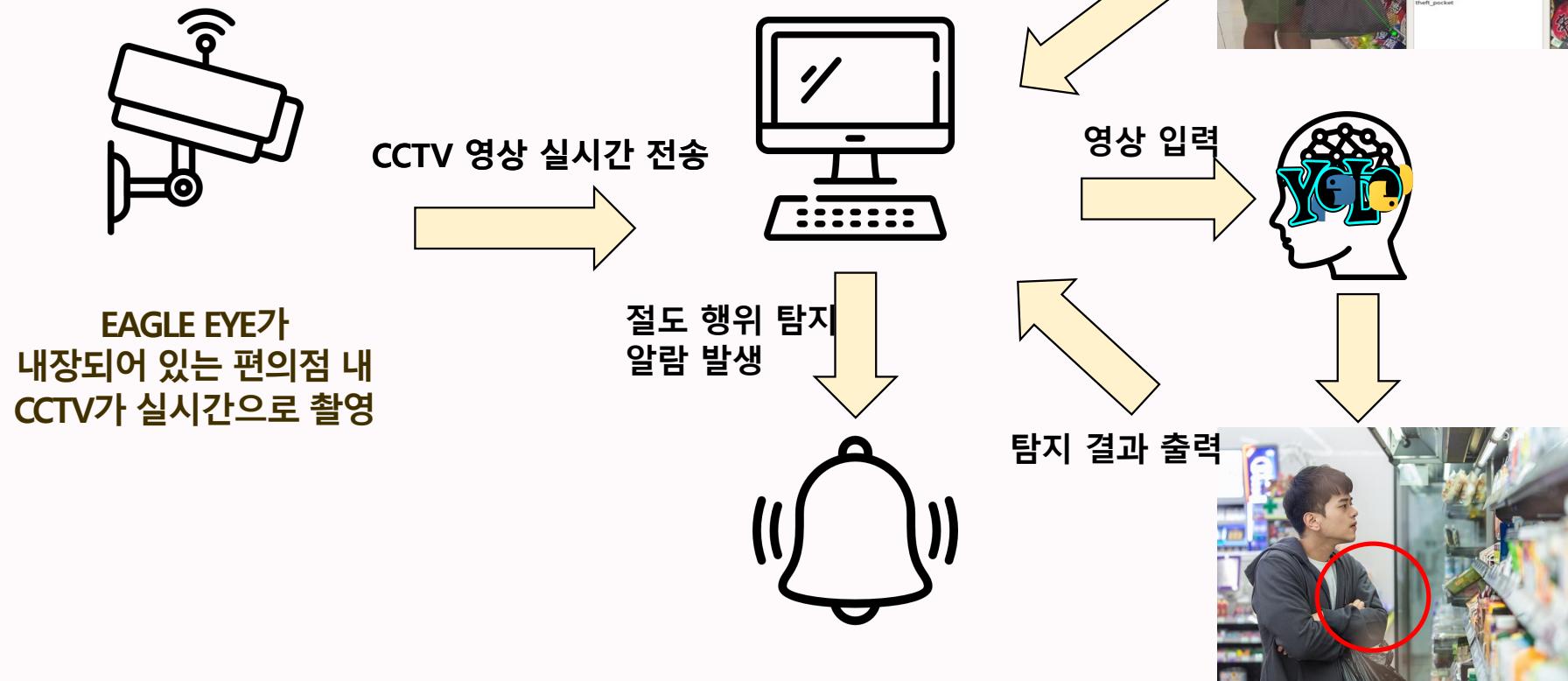
02 프로젝트 분석 설계

설계 방향

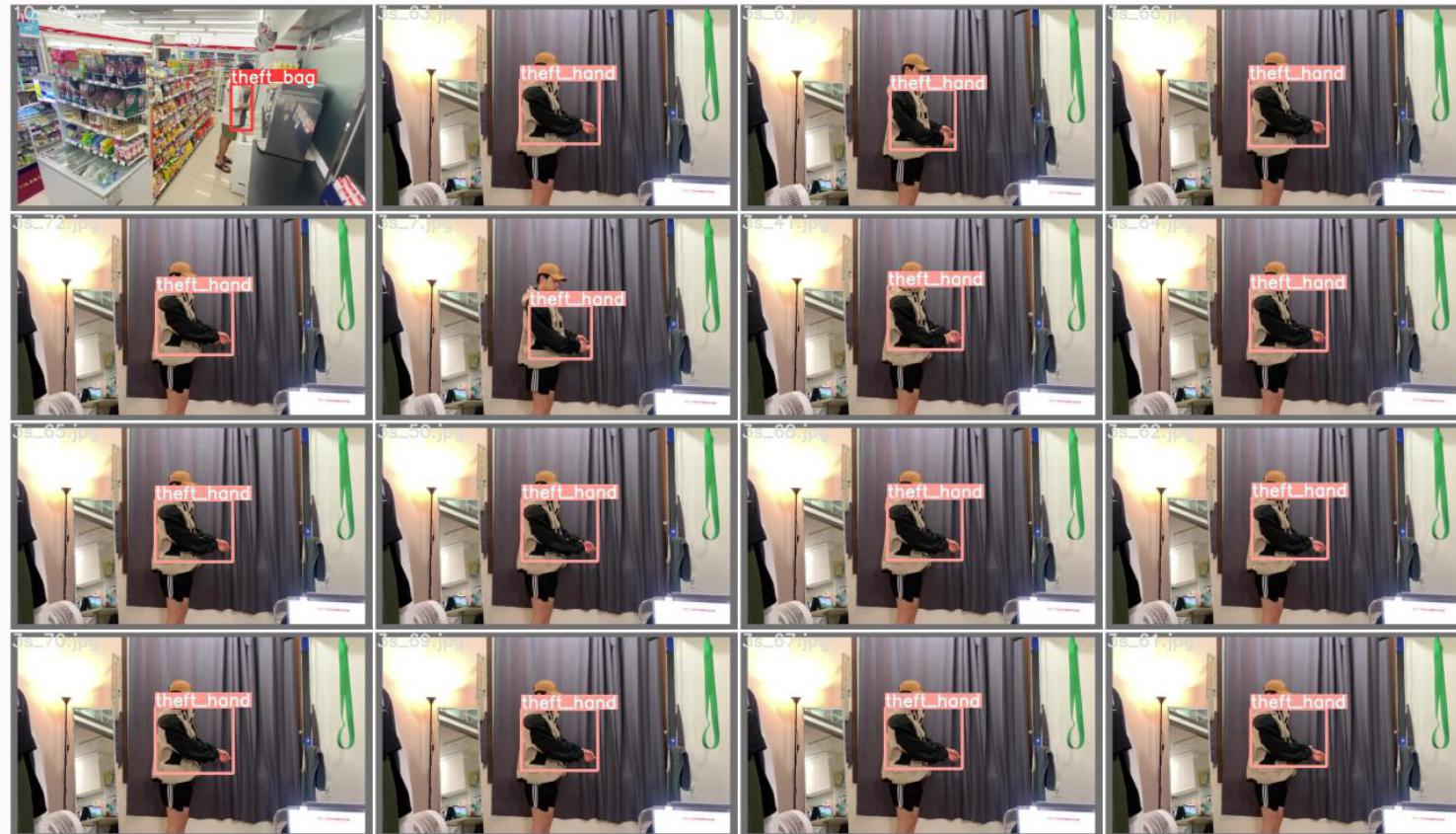


- ✓ 영상 속 절도 의심 상황 중 반복적으로 발견할 수 있는 요소들 **의 데이터를 찾고 라벨링 과정을 통해 행위를 지정해준** 후 학습을 시켜 궁극적으로 높은 인식률을 가진 절도 탐지 인공지능을 완성시키는 것이다.
- ✓ 절도 행위로 의심되는 행동이 발견된다면 **알람이 울려** 사용자가 그 장면을 쉽게 확인할 수 있도록 한다.
- ✓ 기존 YOLO 알고리즘은 객체나 사물 탐지에 그쳤는데 우리는 이 알고리즘을 사용하여 **절도 “행위”를 탐지하고자** 한다. 즉, 절도가 의심되는 행위 자체를 하나의 객체로 인식함과 동시에 가방, 주머니, 소매 등 절도에 자주 등장하는 범죄 도구들도 학습을 시켜 여러가지 범죄 상황을 탐지하게 한다.

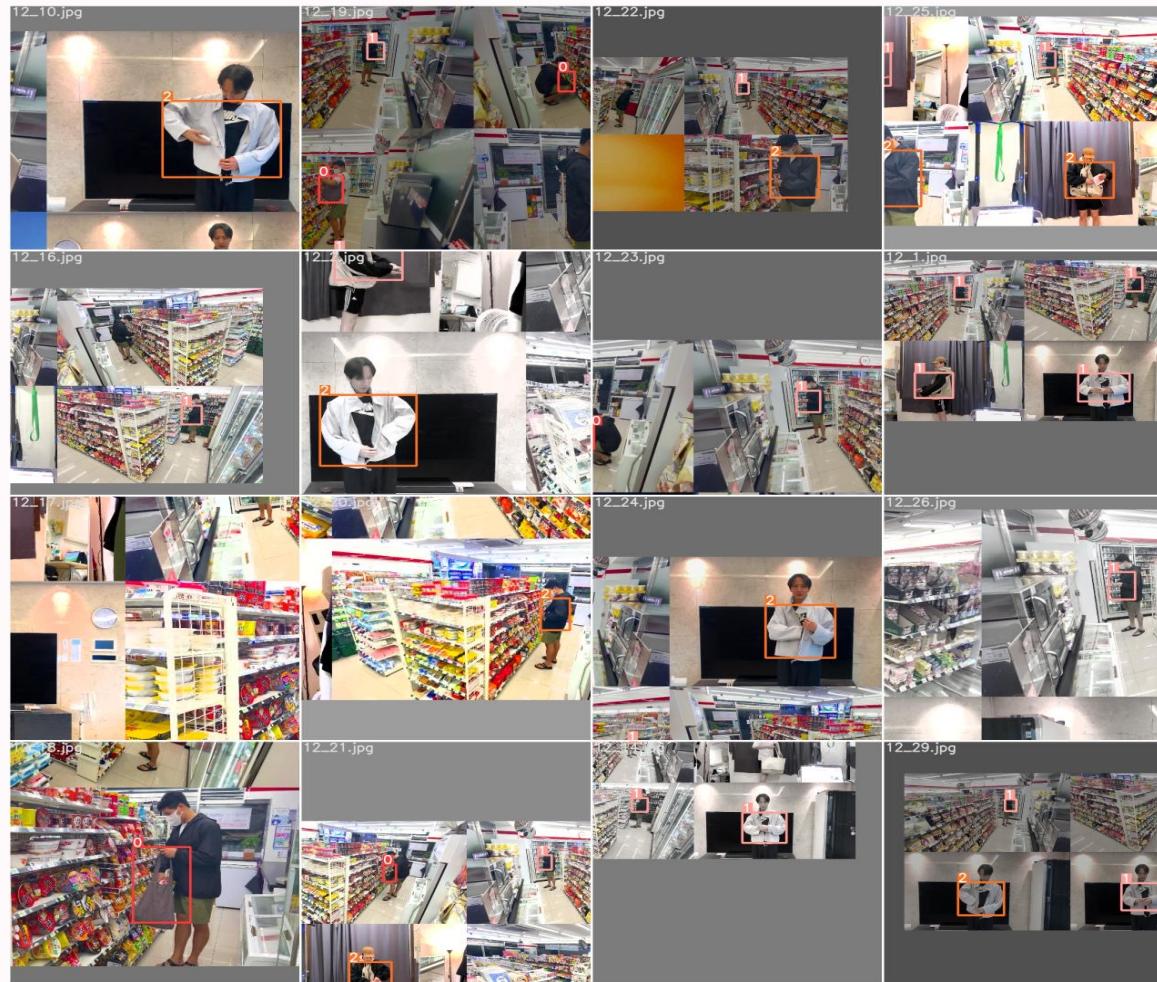
02 프로젝트 분석 설계 시스템 시나리오



03 프로젝트 구현 데이터셋 수집



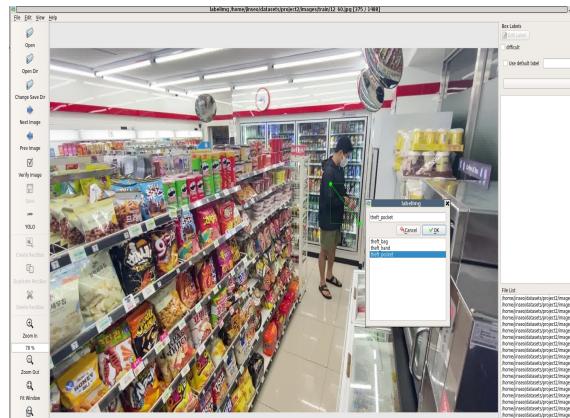
03 프로젝트 구현 데이터셋 수집



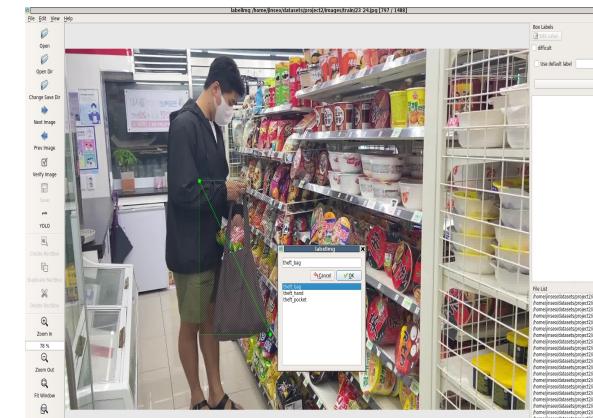
03 프로젝트 구현 데이터셋 분류



- ✓ 두 가지 절도 행위를 분류한 후 각 행위에 해당하는 데이터를 검색과 제작을 통해 수집한다.



✓ Theft_pocket
호주머니에 물품을 넣는 행위

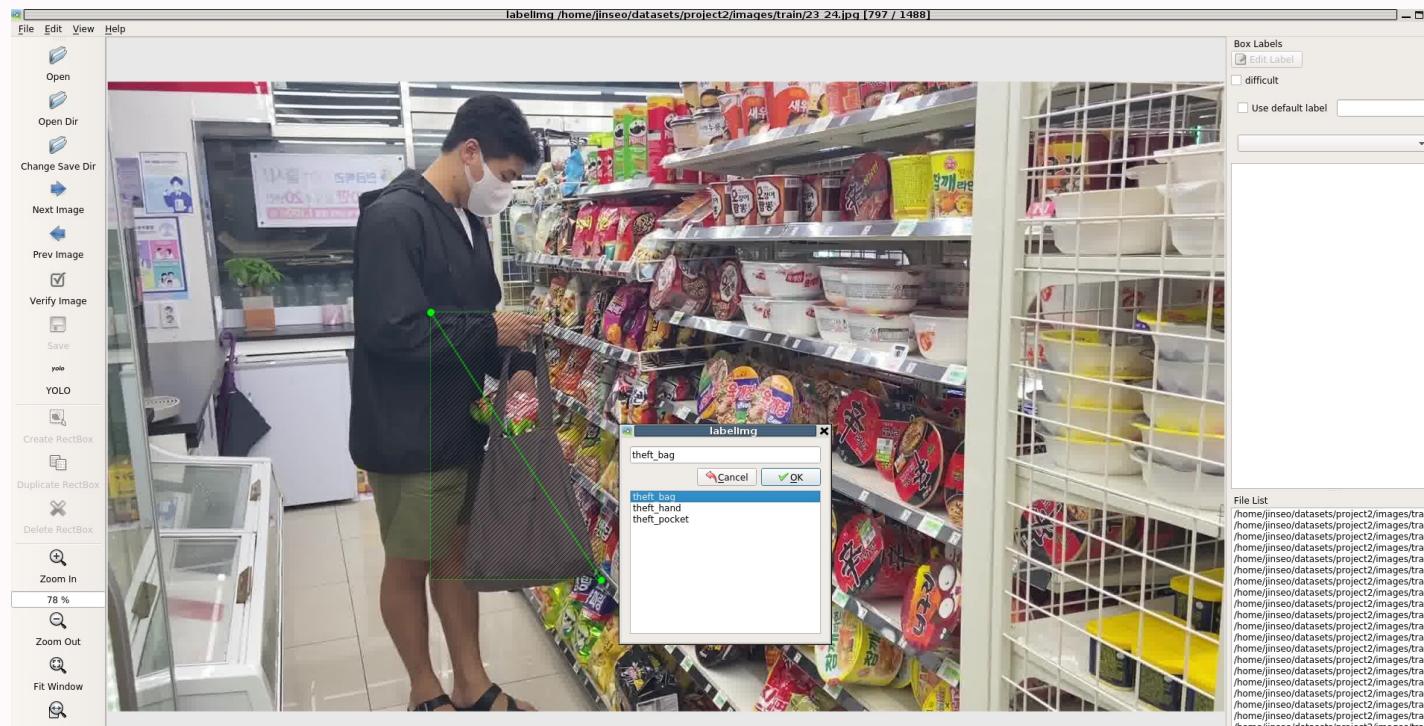


✓ Theft_bag
가방에 물품을 넣는 행위

03 프로젝트 구현 데이터셋 분류



- ✓ 각 행위에 사용되는 범죄도구와 손과 물체를 하나의 객체로 인식하여 라벨링을 진행한다.



03 프로젝트 구현 데이터셋 학습



✓ 라벨링이 완료된 동영상 및 이미지들을 인공지능을 통해 학습시킨다.

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default='yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default='data/coco128.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default='data/hyps/hyp.scratch.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')
    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 check')
    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('--singlecls', action='store_true', help='train multi-class data as single-class')
    parser.add_argument('--adam', action='store_true', help='use torch.optim.Adam() 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='maximum number of dataloader workers')
    parser.add_argument('--project', default='runs/train', help='save to project/name')
    parser.add_argument('--entity', default=None, help='W&B entity')
    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('--linear-lr', action='store_true', help='linear LR')
    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
    parser.add_argument('--upload-dataset', action='store_true', help='Upload dataset as W&B artifact table')
    parser.add_argument('--bbox-interval', type=int, default=-1, help='Set bounding-box image logging interval for W&B')
    parser.add_argument('--save-period', type=int, default=-1, help='Log model after every "save_period" epoch')
    parser.add_argument('--artifact-alias', type=str, default="latest", help='version of dataset artifact to be used')
```

(1) 기존 YOLO에서 설정된 Parameter들 중에서 Weights, epochs,name 들을 직접 설정해준다.

03 프로젝트 구현 데이터셋 학습



✓ 라벨링이 완료된 동영상 및 이미지들을 인공지능을 통해 학습시킨다.

```
jinseo@nipa2021-17247: ~/yolov5
File Edit View Search Terminal Help
jinseo@nipa2021-17247: ~/yolov5$ python3 train.py --data ./data/final.yanf --cfg ./model
s/yolov5.yanf --weights ./runs/train/final2/weights/best.pt --batch 32 --epochs 1200
--name final
train: weights=/runs/train/final2/weights/best.pt, cfg=./model/s/yolov5.yanf, dat
a=/d
ata/final.yanf, hyp=dat/hyps/hyp.scratch.yanf, epochs=1200, batch_size=32, imgsz=640,
rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, evol=False, b
ucket_size=False, cache=False, image_weights=False, devicex=multiprocessing=False, singl
e_cls=False, adam=False, sync_bn=False, workers=8, project=runs/train, entity=False, name=f
inal, exist_ok=False, quad=False, linear_lr=False, label_smoothing=0.0, uploda
d_dataset=False, bbox_interval=1, save_period=1, artifact_aliastest, local_rank=1, freeze=
False
github: skip pip check (not a git repository), for updates see https://github.com/ultra
Yolov5/yolov5
YOLOv5 [2021-8-5] torch 1.8.1+cu102 CUDA 0 (NVIDIA Tesla V100-SXM2-32GB, 32510.5MB)
hyperparameters: lr=0.01, lr_f=0.2, momentum=0.937, weight_decay=0.0005, var_nup_epochs=
3.0, var_nup_momentum=0.8, var_nup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, o
bj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, de
grees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, filp_lr=0.5
, mosaic=1.0, nhup=0.0, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5
runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:600
6/
2021-09-28 23:38:32.441659: I tensorflow/stream_executor/platform/default/dso_l
oader.cc:53] Successfully opened dynamic library libcuda.so.11.0
from n    params module                                arguments
0       -1 1      7040 model.s.common.Focus           [ 3, 64, 3]
1       -1 1      73984 model.s.common.Conv            [ 64, 128, 3]
2       -1 1     156928 model.s.common.C3             [ 128, 128,
3]      -1 1     295424 model.s.common.Conv            [ 128, 256,
```

(2) Train 코드 입력

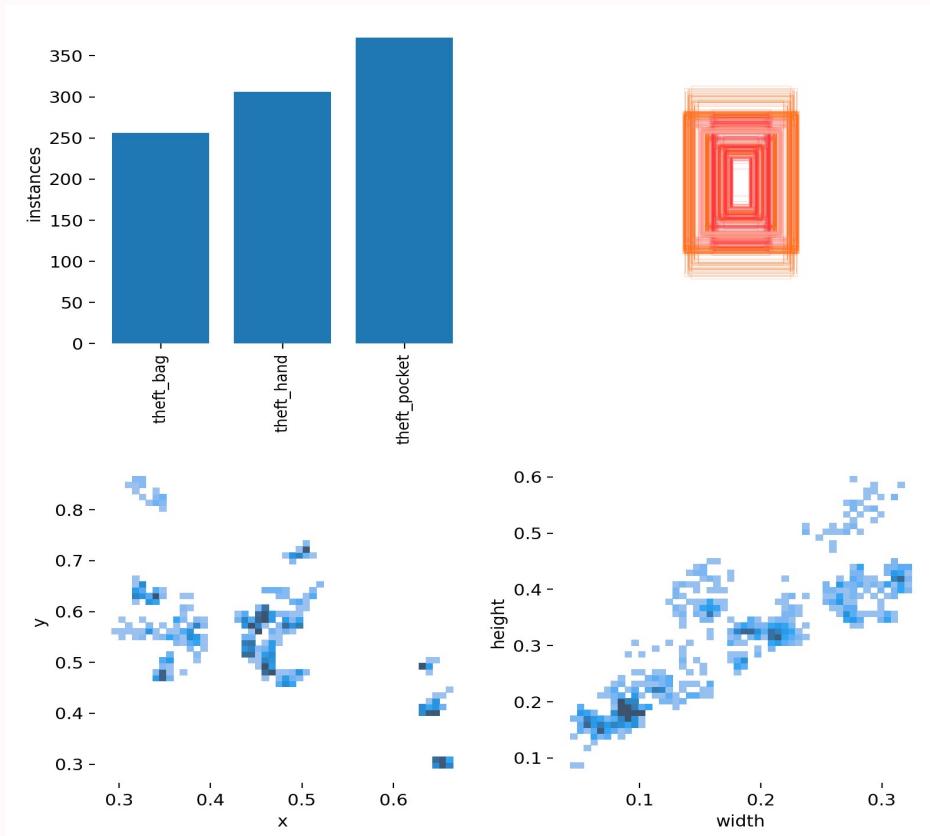
	Epoch	gpu mem	box	obj	cls	label s	img size	640: 100%	mAP@5 mAP@
1142/1199	19.7G	0.005765	0.00166	8.967e-05	8	R	640: 100%	mAP@5 mAP@	0.539
	Class	Images	Labels	P					
	all	382	233	0.666	0.537				
0.343									
1143/1199	19.7G	0.00594	0.001713	0.0001012	9	R	640: 100%	mAP@5 mAP@	0.537
	Class	Images	Labels	P	9	R	640: 100%	mAP@5 mAP@	0.537
	all	382	233	0.659	0.538				
0.342									
1144/1199	19.7G	0.005813	0.001661	9.002e-05	10	R	640: 100%	mAP@5 mAP@	0.537
	Class	Images	Labels	P	10	R	640: 100%	mAP@5 mAP@	0.537
	all	382	233	0.661	0.538				
0.342									
1145/1199	19.7G	0.005866	0.001693	9.885e-05	7	R	640: 100%	mAP@5 mAP@	0.538
	Class	Images	Labels	P	7	R	640: 100%	mAP@5 mAP@	0.538
	all	382	233	0.662	0.538				
0.343									
1146/1199	19.7G	0.005856	0.001715	9.25e-05	13	R	640: 100%	mAP@5 mAP@	0.537
	Class	Images	Labels	P	13	R	640: 100%	mAP@5 mAP@	0.537
	all	382	233	0.662	0.537				
0.341									
1147/1199	19.7G	0.005863	0.001629	9.157e-05	7	R	640: 100%	mAP@5 mAP@	0.537
	Class	Images	Labels	P	7	R	640: 100%	mAP@5 mAP@	0.537
	all	382	233	0.663	0.538				
0.342									

(3) 학습이 돌아가는 과정

03 프로젝트 구현 데이터셋 학습



✓ 라벨링이 완료된 동영상 및 이미지들을 인공지능을 통해 학습시킨다.



(4) 학습이 끝난 후 분석 그래프가 파일로 저장됨. 그래프는 이미지 사이즈와 바운딩 박스 x, y좌표를 나타낸다.

03 프로젝트 구현 프로그램 구현



✓ 알람이 울리는 기능을 YOLO 알고리즘에 추가

```
if save_img or save_crop or view_img: # Add bbox to image
    chunk = 1024

    path='siren.wav'

    with wave.open(path, 'rb') as f:
        p = pyaudio.PyAudio()
        stream = p.open(format = p.get_format_from_width(f.getsampwidth()),
                         channels = f.getnchannels(),
                         rate = f.getframerate(),
                         output = True)

        data = f.readframes(chunk)
        while data:
            stream.write(data)
            data = f.readframes(chunk)

        stream.stop_stream()
        stream.close()

        p.terminate()
    c = int(cls) # integer class
    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
    annotator.box_label(xyxy, label, color=colors(c, True))
    if save_crop:
        save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)
```

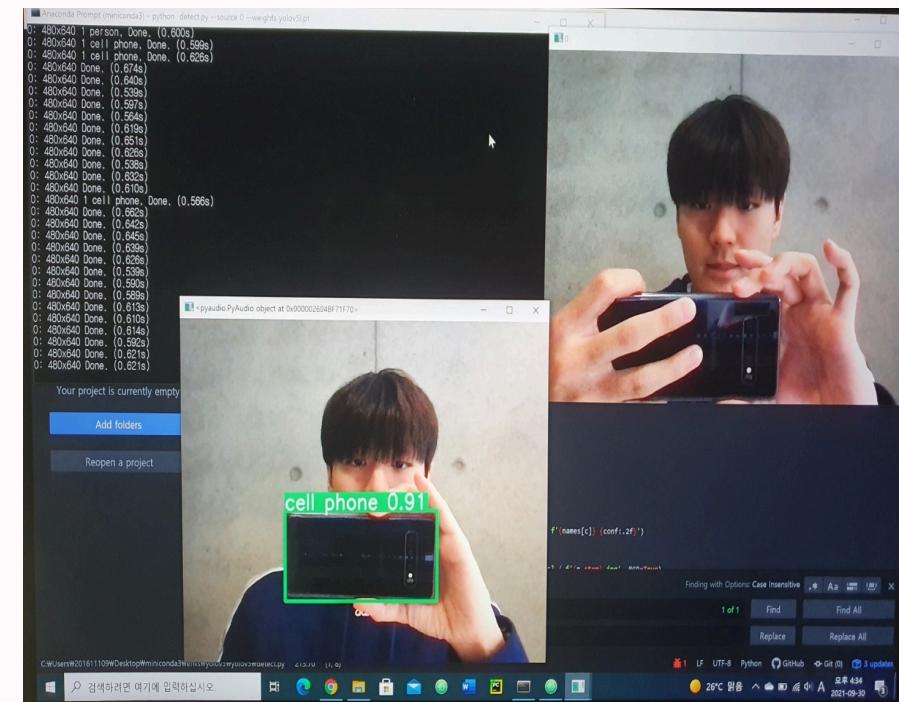
03 프로젝트 구현 프로그램 구현



✓ 카메라 연동

```
1 # source 설명
2 python detect.py --source 0 # webcam
3             file.jpg # image
4             file.mp4 # video
5             path/ # directory
6             path/*.jpg # glob
7             'https://youtu.be/NUs0V1DFqZg' # YouTube video
8             'rtsp://example.com/media.mp4' # RTSP, RTMP, HTTP stream
9
10
11 # 코드입력
```

```
1 python detect.py --weights ./runs/train/coco_yolov5/weights/best.pt --img 640 --conf 0.5 --source
2 ~/valid/images/00000010707.jpg.rf.6d5d82303a417707aab4f1cd2f465e8a.jpg
```



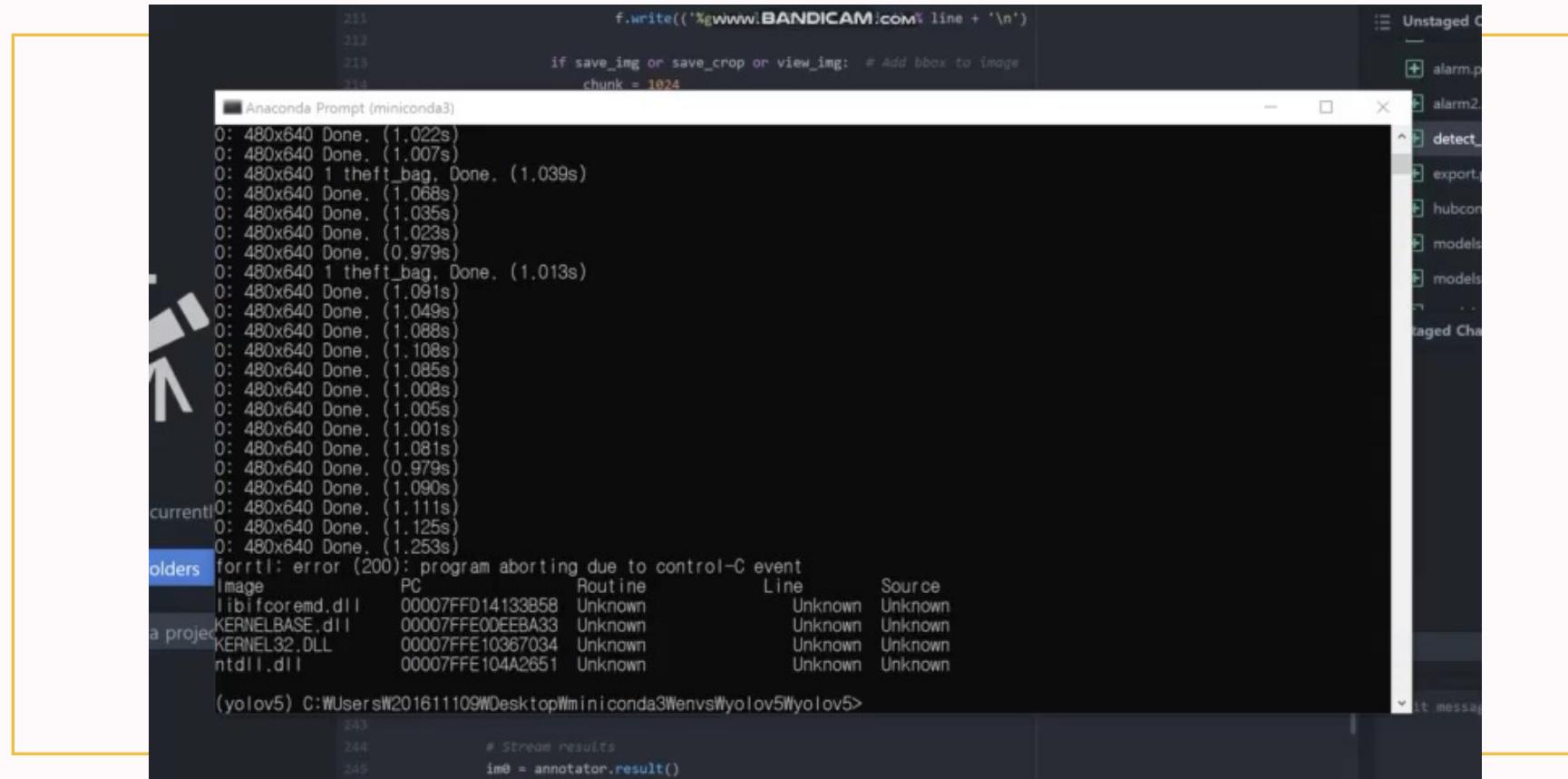
04 프로젝트 결론
시연 영상



04 프로젝트 결론
시연 영상



04 프로젝트 결론 시연 영상



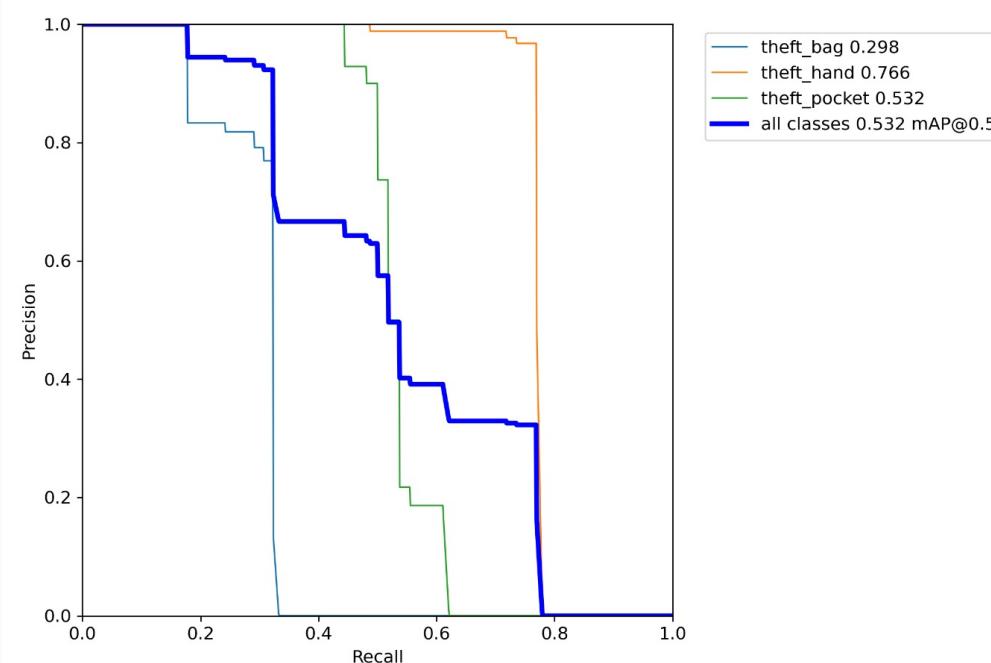
04 프로젝트 결론 시연 영상



```
210         with open(txt_path + '.txt', 'a') as f:
211             f.write((('%g' * len(line)).rstrip() % line) + '\n')
212
213     if save_img or save_crop or view_img: # Add bbox to image
214
215         detect(weights=[best.pt'], source=0, imgsz=[640, 640], conf_thres=0.5, iou_thres=0.5, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False)
216         YOLOv5 5fa7601 torch 1.9.1 CPU
217
218         Fusing layers...
219         Model Summary: 476 layers, 87212152 parameters, 0 gradients, 217.1 GFLOPs
220         1/1: 0... success (inf frames 640x480 at 30.00 FPS)
221
222         0: 480x640 Done. (1.058s)
223         0: 480x640 Done. (1.050s)
224         0: 480x640 Done. (1.029s)
225         0: 480x640 1 theft_pocket, Done. (1.037s)
226         0: 480x640 Done. (1.136s)
227         0: 480x640 Done. (1.079s)
228         0: 480x640 Done. (1.074s)
229         0: 480x640 Done. (1.096s)
230         0: 480x640 Done. (1.016s)
231         forrtl: error (200): program aborting due to control-C event
232         Image          PC          Routine          Line      Source
233         libifcoremd.dll 00007FFD14F13B58 Unknown           Unknown Unknown
234         KERNELBASE.dll 00007FFE0DEEBAA3 Unknown           Unknown Unknown
235         KERNEL32.DLL   00007FFE10367034 Unknown           Unknown Unknown
236         ntdll.dll    00007FFE104A2651 Unknown           Unknown Unknown
237
238         (yolov5) C:\Users\201611109\Desktop\miniconda3\envs\yolov5\yolov5>
239         print(f'{s}Done. ({t3 - t2:.3f}s)')
240
241         # Stream results
242         im0 = annotator.result()
```

The screenshot shows a terminal window running a Python script for object detection using YOLOv5. The script performs file operations, runs the detection algorithm, and prints performance metrics. A stack trace is shown for a control-C interrupt. To the right of the terminal is a GitHub commit interface showing staged changes.

04 프로젝트 결론 성능 지표



기대효과

- ✓ 범죄행위들을 단절하기 위해선 많은 노력과 시간이 든다. 그중에서도 가장 흔히 일어나는 범죄인 절도행위를 근절하는데에 있어 절도 행위 탐지 알고리즘이 학습된 인공지능을 이용하여 **인력과 인건비를 절감함**과 동시에 더 나아가 여러 범죄들의 데이터셋과 관련된 이미지와 객체들의 학습을 통해 **폭행, 강도, 방화, 주취행동** 등의 여러 범죄를 쉽게 예방할 수 있다.
- ✓ 또한 cctv와 도난방지 시스템이 널려있는 대형마트에 비해 상대적으로 절도에 취약한 편의점 또는 중소형 상점, 노점상에서 cctv 한대만으로 **점주들과 근로자들의 범죄에 대한 부담과 걱정을 덜어줄** 수 있다.
- ✓ 나날이 발전하고 성장하는 이론과 기술들로 인간 그 이상의 능력과 잠재력을 보여주는 인공지능과 알고리즘을 효율적으로 이용하기 위해 만들어진 "EAGLE EYE"는 이러한 기대효과를 불러 일으킬 수 있다.

감사합니다!!!



독수리 오형제

팀장: 주원석

팀원: 박진서, 박진석, 심주성, 홍수만