Road Scanner팀 제출물 매뉴얼

(2021 인공지능 학습용 데이터 해커톤 대회)

Road Scanner
팀장 : 문광표
팀원 : 노성철 | 신덕용 | 신준혁 | 이은지

1. 제출 폴더 및 파일 목록

Road Scanner는 구글 드라이브 공유 폴더 형식으로 파일을 제출하며, 제출한 폴더 및 파일의 목록은 다음과 같습니다.

- 학습된 가중치 폴더(weights) / 최종 가중치 파일 이름 : Roadscanner_best.pt
- YOLO 구동 시 필요한 기초 파일 폴더(models, utils 형식으로 구성)
- 학습 시 이용한 train.py 및 객체 탐지 코드인 detect.py로 구성
- Test Data Set (이미지 120장)

2. 프로그램 설명

Road Scanner에서 제시한 프로그램은 파이썬 파일(.py) 형태로, 코드 실행을 통해 도로시설물 이미지 파일을 로딩하고, 이미지 내 도로시설물 유형(30종)을 판단하며, 해당 시설물의 양호/불량 여부를 판별합니다.

도로시설물 유형(30종)

자동차진입 억제용 말뚝 (bollard)_스테인리스, 자동차진입 억제용 말뚝 (bollard)_탄성고무, 시선유도봉_ 2줄, 시선유도봉_3줄, 보행자용 방호울타리, 교량용 방호울타리, 턱낮추기, 경사로, 점자블럭, (도로안내표지) 지주, 시멘트 콘크리트, 보도블록, 자전거도로, 연석, 무단횡단 방지 울타리, 맨홀, 현장 신호제어기, 시각장애인용 음향신호기, 과속방지턱, 횡단보도, 고원식횡단보도, 통합표지판, 정주식, 부착식 표지, 가로등, (전봇대) 빗금표시, 자동차진입 억제용 말뚝 (bollard)_대리석, 자동차진입 억제용 말뚝 (bollard)_U자형, 소화전, 주차멈춤턱_블럭형, 미끄럼방지 계단

저희 팀에서는 학습에 활용할 이미지에 구조물 형태별 정상/불량(불량/불량전체/불량부분)을 구분하여, 아래 표와 같이 Label Number를 부여했습니다.

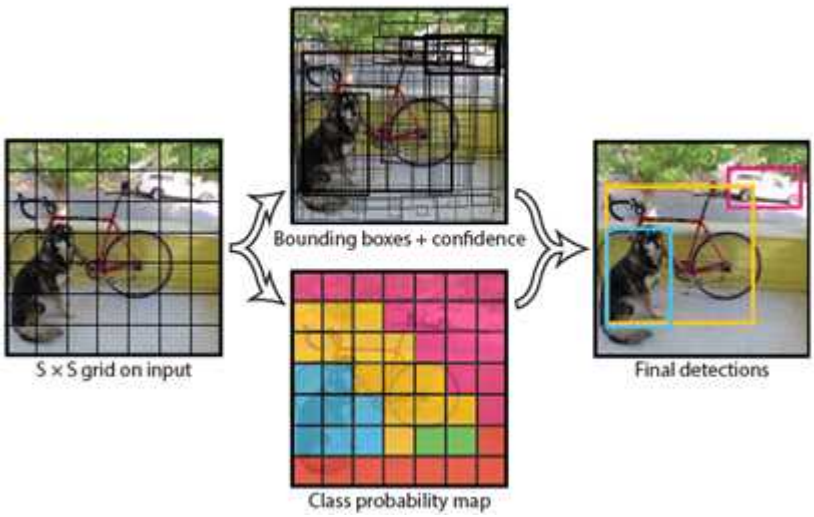
<표 1> 구조물 정상/불량 Label Number

순번	구조물 이름	Label Number		
01	자동차진입 억제용 말뚝 (bollard)_스테인리스	0 (정상)	1 (불량)	
02	자동차진입 억제용 말뚝 (bollard)_탄성고무	2 (정상)	3 (불량)	
03	시선유도봉_ 2줄	4 (정상)	5 (불량)	
04	시선유도봉_3줄	6 (정상)	7 (불량)	
05	보행자용 방호울타리	8 (정상)	9 (불량전체)	10 (불량부분)
06	교량용 방호울타리	11 (정상)	12 (불량전체)	13 (불량부분)
07	턱낮추기	14 (정상)	15 (불량전체)	16 (불량부분)
08	경사로	17 (정상)	18 (불량)	
09	점자블럭	19 (정상)	20 (불량전체)	21 (불량부분)
10	(도로안내표지) 지주	22 (정상)	23 (불량)	
11	시멘트 콘크리트	24 (정상)	25 (불량전체)	26 (불량부분)
12	보도블록	27 (정상)	28 (불량전체)	29 (불량부분)
13	자전거 도로	30 (정상)	31 (불량전체)	32 (불량부분)
14	연석	33 (정상)	34 (불량전체)	35 (불량부분)
15	무단횡단 방지 울타리	36 (정상)	37 (불량전체)	38 (불량부분)
16	맨홀	39 (정상)	40 (불량)	
17	현장 신호제어기	41 (정상)	42 (불량)	
18	시각장애인용 음향신호기	43 (정상)	44 (불량)	
19	과속방지턱	45 (정상)	46 (불량전체)	47 (불량부분)
20	횡단보도	48 (정상)	49 (불량전체)	50 (불량부분)
21	고원식횡단보도	51 (정상)	52 (불량전체)	53 (불량부분)
22	통합표지판	54 (정상)	55 (불량)	
23	정주식, 부착식 표지	56 (정상)	57 (불량)	
24	가로등	58 (정상)	59 (불량)	
25	(전봇대) 빗금표시	60 (정상)	61 (불량)	
26	자동차진입 억제용 말뚝 (bollard)_대리석	62 (정상)	63 (불량)	
27	자동차진입 억제용 말뚝 (bollard)_U자형	64 (정상)	65 (불량)	
28	소화전	66 (정상)	67 (불량)	
29	주차멈춤턱_블럭형	68 (정상)	69 (불량)	
30	미끄럼방지 계단	70 (정상)	71 (불량)	

3. 주요 알고리즘

본 프로그램의 주요 알고리즘으로는 YOLO를 활용했습니다. 저희 팀이 사용한 YOLO는 대표적인 객체 탐지 모델로, 이미지에서 관심 객체를 배경과 구분해 식별하는 자동화 기법입니다. 이미지를 그리드 형식으로 분할 후, 그리드 중앙을 중심으로 경계박스(Anchor Box) 개수를 예측하고, 이를 기반으로 신뢰도를 계산한다는 특징을 가지고 있습니다.

저희 팀은 비교적 최신버전인 YOLO V5를 채택하였습니다. 그 중 학습 속도가 빠르고, 가중치 용량도 가벼운 YOLOV5S 모델을 활용하였습니다.



4. 객체탐지 실행 (구조 및 탐지 실행 순서)

1) 실행 디렉토리 파일 구성

: 공유 드라이브 파일을 다운받고, 실행 디렉토리에 다음과 같이 파일을 구성합니다.

2) 필요파일 설치

: 터미널 환경에서 실행 디렉토리로 이동 후, Road_scanner 폴더까지 이동 후 다음의 명

령어를 통해 필요파일을 설치합니다.

```
pip install -r ./Road_scanner/requirements.txt
```

```
실행 디렉토리        >pip install -r./Road_scanner/requirements.txt
Collecting matplotlib>=3.2.2
  Downloading matplotlib-3.5.1-cp38-cp38-win_amd64.whl (7.2 MB)
     |████████████████████████████████| 7.2 MB 6.8 MB/s
Collecting numpy>=1.18.5
  Downloading numpy-1.21.4-cp38-cp38-win_amd64.whl (14.0 MB)
     |████████████████████████████████| 14.0 MB 6.8 MB/s
Collecting opencv-python>=4.1.2
  Downloading opencv_python-4.5.4.60-cp38-cp38-win_amd64.whl (35.1 MB)
     |████████████████████████████████| 35.1 MB 18 kB/s
Collecting Pillow>=7.1.2
  Downloading Pillow-8.4.0-cp38-cp38-win_amd64.whl (3.2 MB)
     |████████████████████████████████| 3.2 MB 6.4 MB/s
Collecting PyYAML>=5.3.1
  Downloading PyYAML-6.0-cp38-cp38-win_amd64.whl (155 kB)
     |████████████████████████████████| 155 kB 6.8 MB/s
Collecting requests>=2.23.0
  Using cached requests-2.26.0-py2.py3-none-any.whl (62 kB)
Collecting scipy>=1.4.1
  Downloading scipy-1.7.3-cp38-cp38-win_amd64.whl (34.2 MB)
     |████████████████████████████████| 34.2 MB 615 kB/s
Collecting torch>=1.7.0
```

3) 가중치 설정 및 detect 실행

: 터미널에서 Road_scanner 내 detect.py 파일을 다음과 같은 명령어로 실행합니다.

```
python ./Road_scanner/detect.py --weights ./Road_scanner/Roadscanner_best.pt --img 416 --conf 0.25 --save-txt --source ./test_input/
```

```
실행 디렉토리    >python ./Road_scanner/detect.py -weights ./Road_scanner/Roads
canner.pt --img 416 --conf 0.25 --save-txt --source ./test_input/
```

4) 실행 확인

: 코드가 정상적으로 작동되면, 아래와 같이 특정 위치에 탐지 결과가 저장되었다는 문구 및 경로가 출력됩니다.

```
detect_수정진행: weights=['./Road_scanner/weigh
ts/1212_2342_best.pt'], source=C:/python/yolov5
-master/data/images/test, imgsz=[416, 416], con
f_thres=0.25, iou_thres=0.45, max_det=1000, dev
ice=, view_img=False, save_txt=True, save_conf=
False, save_crop=False, nosave=False, classes=N
one, agnostic_nms=False, augment=False, visuali
ze=False, update=False, project=Road_scanner, n
ame=test_output, exist_ok=False, line_thickness
=3, hide_labels=False, hide_conf=False, half=Fa
lse, dnn=False
YOLOv5  2021-12-5 torch 1.10.0+cpu CPU

Fusing layers...
Model Summary: 213 layers, 7204309 parameters,
0 gradients, 16.4 GFLOPs
해당 경로에 파일별 객체 탐지 결과를 저장하였습
니다 : c:₩python₩제출용₩test_output
```

5. 탐지 결과 (이상탐지 결과 확인)

1) test_output 폴더

: 정상적으로 detect가 완료되면, 다음과 같이 test_output 폴더에 Input 이미지의 탐지 결과가 각각의 텍스트 파일(.txt) 형태로 저장됩니다.

이름	수정한 날짜	유형	크기
1_01_1_1_01_1_20210719_0000028607	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210720_0000091853	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210724_0000088598	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210724_0000128991	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210728_0000001660	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210728_0000165936	2021-12-09 오후 5:04	텍스트 문서	1KB
1_01_1_1_01_1_20210806_0000347470	2021-12-09 오후 5:04	텍스트 문서	1KB

: 텍스트 파일을 열면 다음의 구조로 탐지 결과를 포함하고 있습니다. YOLO format BBOX를 기존 Json 형태에 맞게 xmin, ymin, xmax, ymax로 반환합니다.

[정상 : 0 / 불량 : 1] [시설물 코드(0~71)][1] : [xmin, ymin, xmax, ymax]

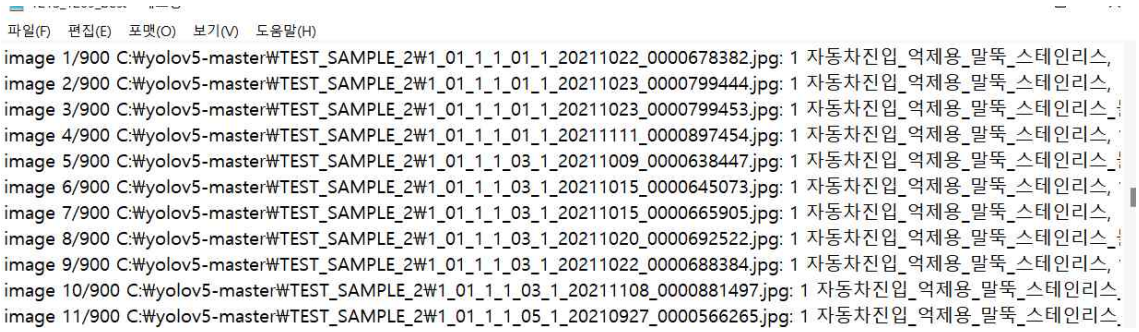
2) test_summary 폴더

: 마찬가지로 정상적으로 detect가 완료되면, [test_summary] 폴더에 Input된 각 이미지

1) <표 1> 구조물 정상/불량 Label Number

별 도로 시설물 유형 및 불량 탐지 여부를 기록한 로그 파일이 하나의 텍스트 파일(.txt)로 저장됩니다. 그 내용은 다음과 같이 시설물의 이름 및 개수를 나타냅니다.

Image 현재/총개수 [경로] : [탐지 개수] [탐지 시설물 이름]



6. 부록

제출한 파일 중 [Data_preprocessing.py]는 저희 팀에서 진행한 데이터 전처리 과정에 사용한 코드입니다. 이 코드를 통해 진행한 내용은 다음과 같습니다.

- 폴리곤형태 데이터 시각화
- 이미지 Resizing
- JSON 내 annotation 을 YOLO 포맷으로 변환