

KISTI 4차인재 양성과정 팀프로젝트 최종 발표

시각장애인을 위한 횡단보도 보행 도움 앱

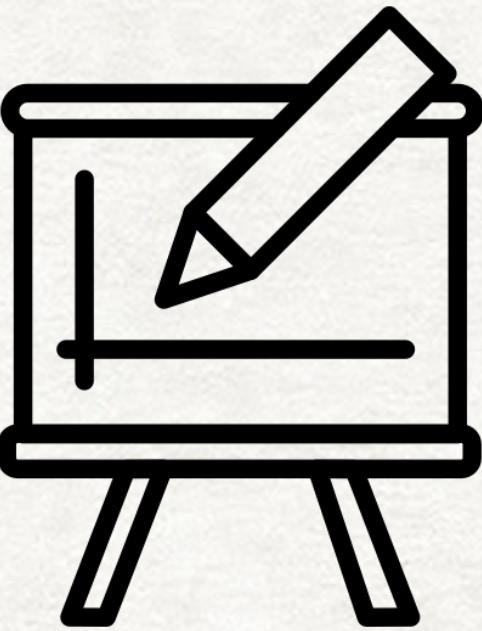
(불사조) 신승배 김종현 이다혜 유정용

목차

주제 선정 배경

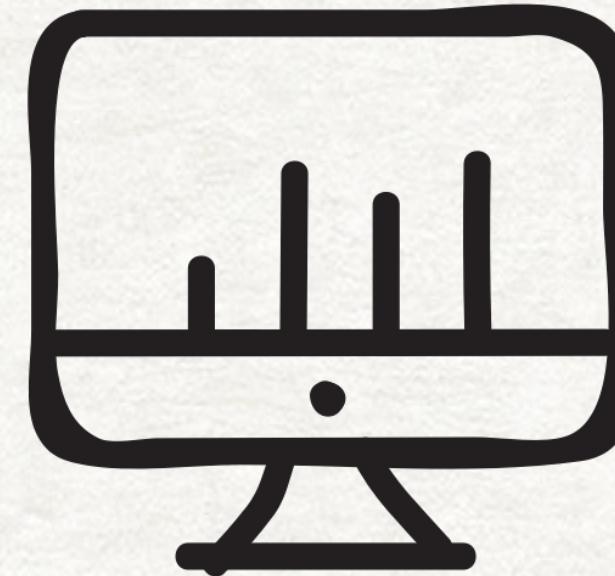


프로젝트 계획



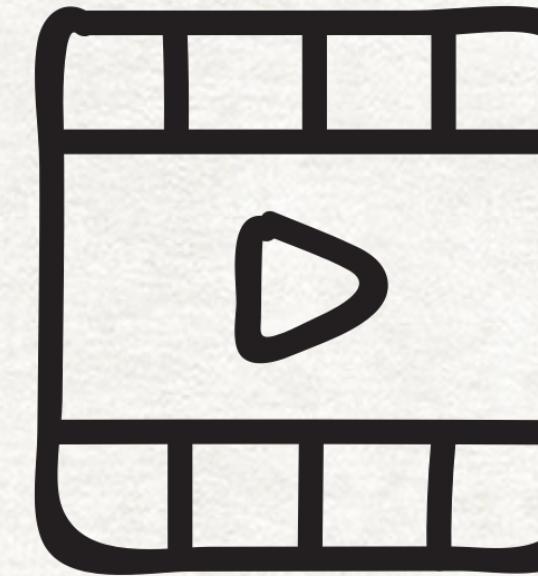
1. 업무 일정
2. 조원 역할 분담
3. 전체 프로세스

진행 과정



1. 데이터셋 구성
2. 모델 선정 및 학습
3. 안드로이드 app 개발

영상 시연



소감



1. 개선 사항
2. 힘들었던 점

주제 선정 배경

주제에 대한 의문점

“시각장애인용 음향신호기가 이미 존재한다”

- ▶ 모든 신호등에 설치되어 있지 않다
- ▶ 초록불인 걸 알더라도,
정지선을 넘어선 차량은
인지하기 어렵다



신호등의 초록불과 빨간불을 구별하면서,
동시에 차량 경고도 할 수 있는 APP을 만들어보자!



프로젝트 계획

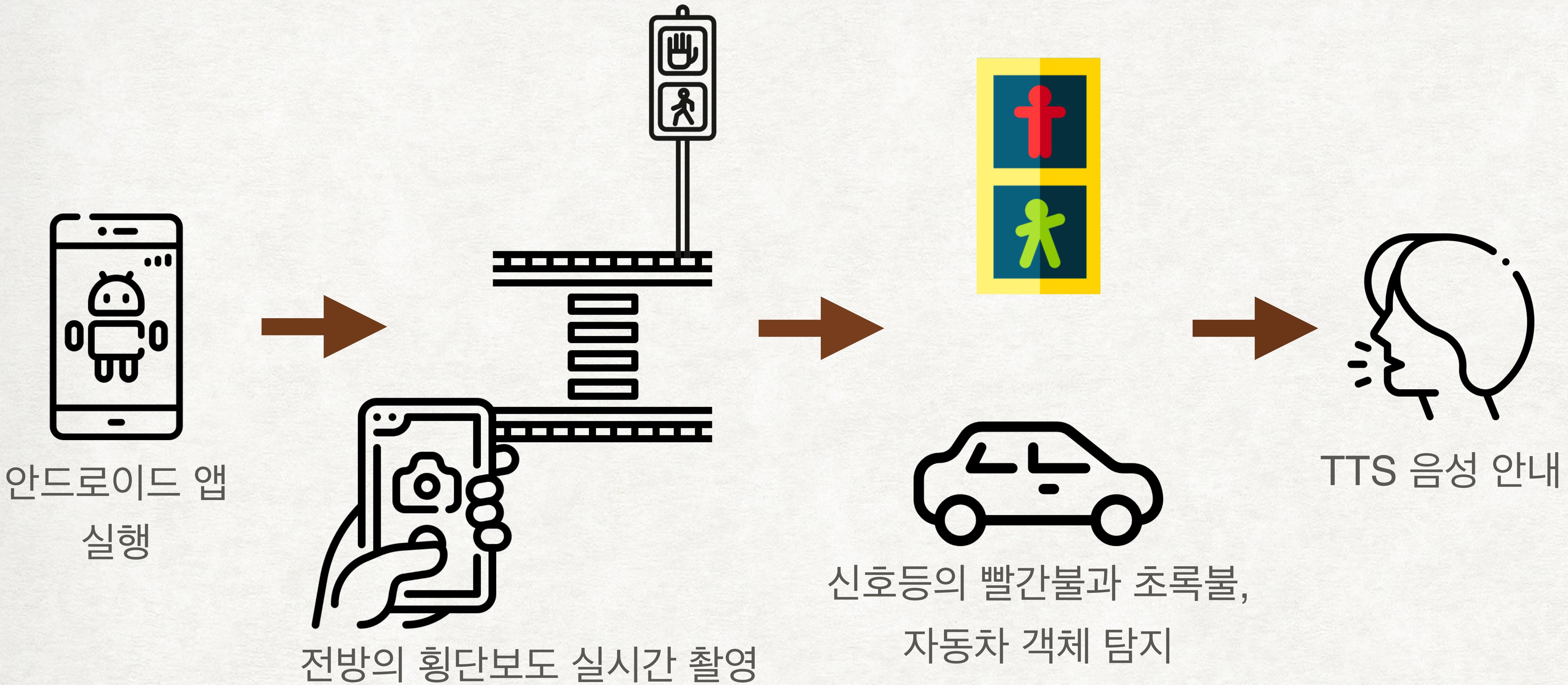
업무일정

업무	1주차	2주차	3주차	4주차	5주차	6주차
주제 선정 및 프로젝트 구체화						
데이터 수집 및 전처리						
모델 선정 및 학습						
프로젝트 배포준비						
발표 준비						

역할

신승배	시나리오 관련 로직 구상, 안드로이드 앱 개발
김종현	모델 선정 및 학습, 안드로이드 앱 개발
이다혜	발표 자료 작성, 데이터 수집 및 라벨링
유정용	데이터 수집 및 라벨링, 안드로이드 앱 개발

PROCESS



데이터셋 구성

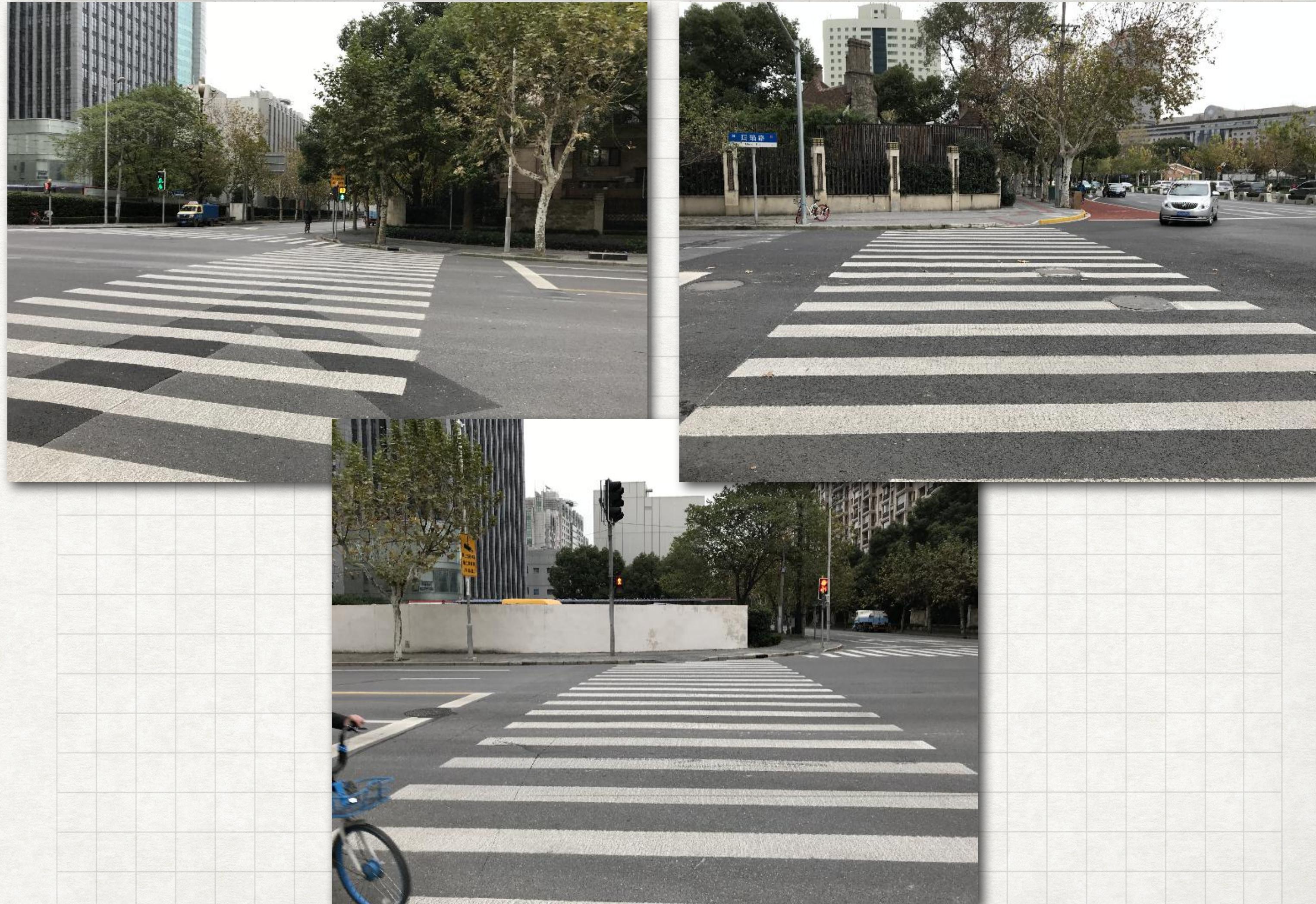
데이터 수집

데이터 라벨링

데이터 보완

최종 데이터셋

▶ 중국의 횡단보도 사진 5,219장 확보
(<https://github.com/samuelyu2002/ImVisible>)



데이터셋 구성

데이터 수집

데이터 라벨링

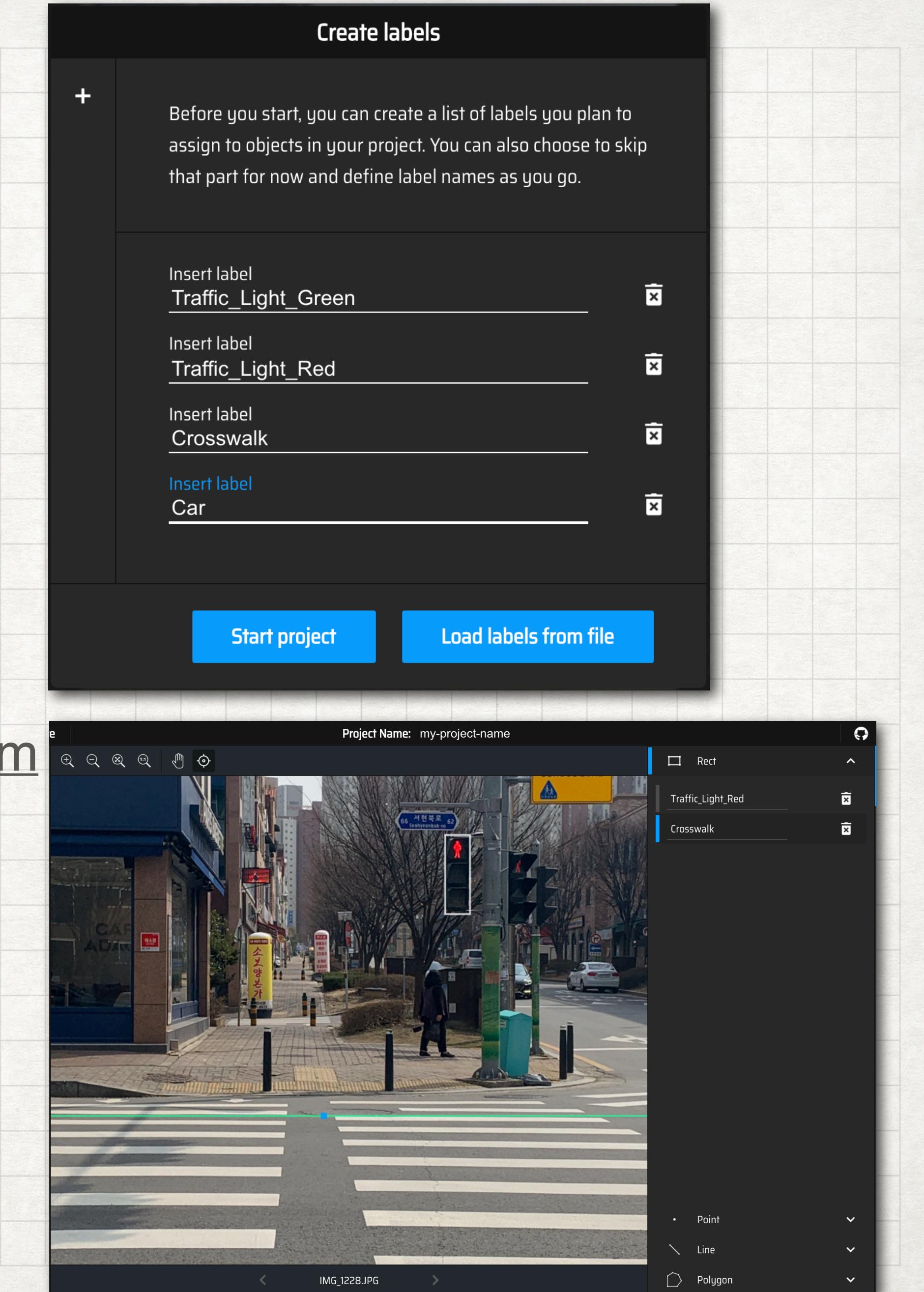
데이터 보완

최종 데이터셋

▶ 라벨링 대상 4가지

1. 횡단보도
2. 신호등의 빨간불
3. 신호등의 초록불
4. 자동차

▶ <http://makesenseai.com> 웹서비스를 이용해서 라벨링을 진행하였다



데이터셋 구성

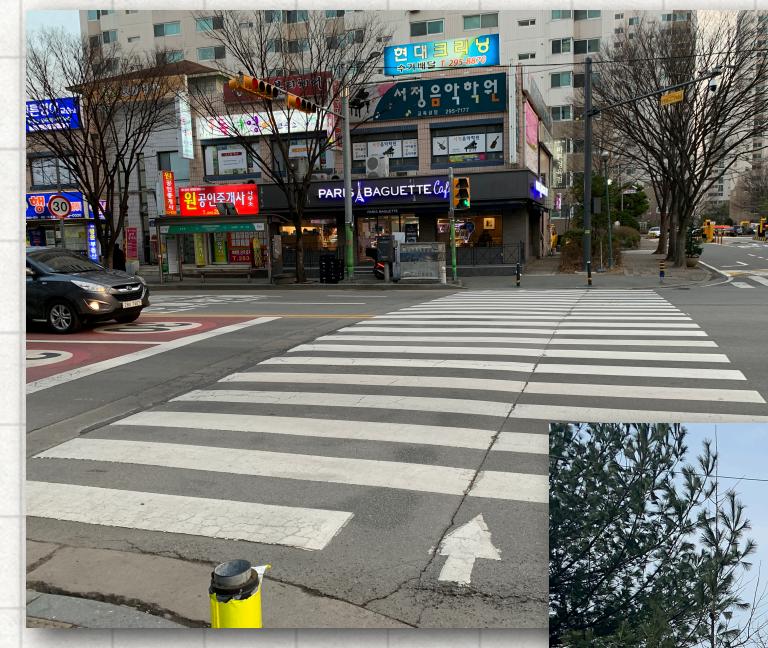
데이터 수집



데이터 라벨링

- ▶ 중국의 횡단보도는 1줄, 우리나라의 횡단보도는 2줄로 구성되어 있음
- ▶ 중국의 신호등은 빨간불이 들어오는 칸에 노란색으로 숫자가 표시됨

데이터 보완



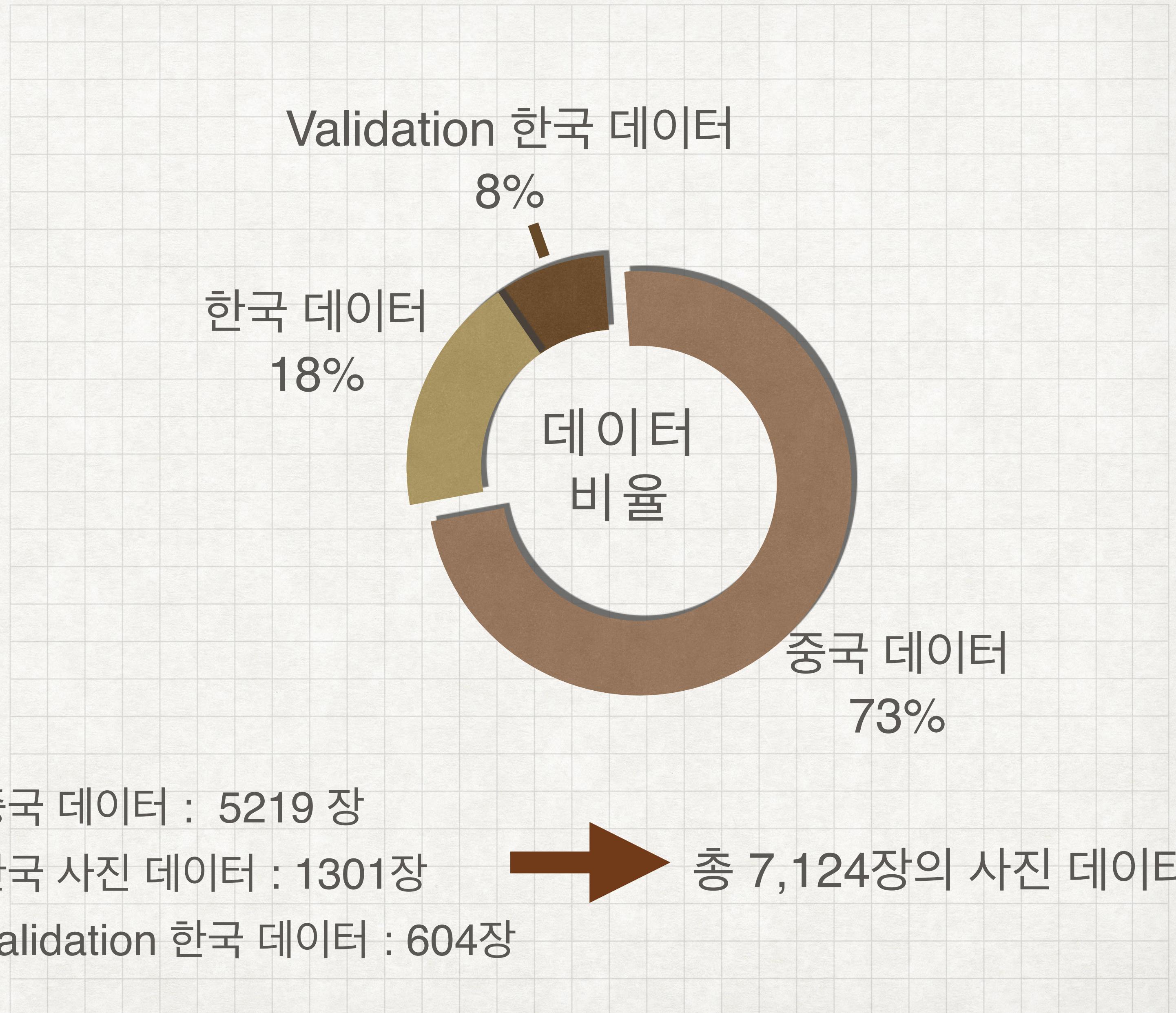
최종 데이터셋



한국의 횡단보도 사진을 직접 수집



데이터셋 구성



모델 선정 및 학습

모델 소개

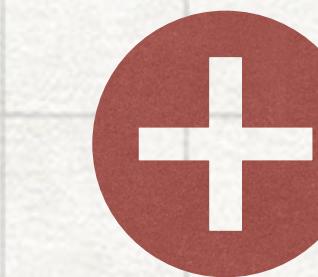
모델 선정

모델 학습

모델 평가

<Object detection>

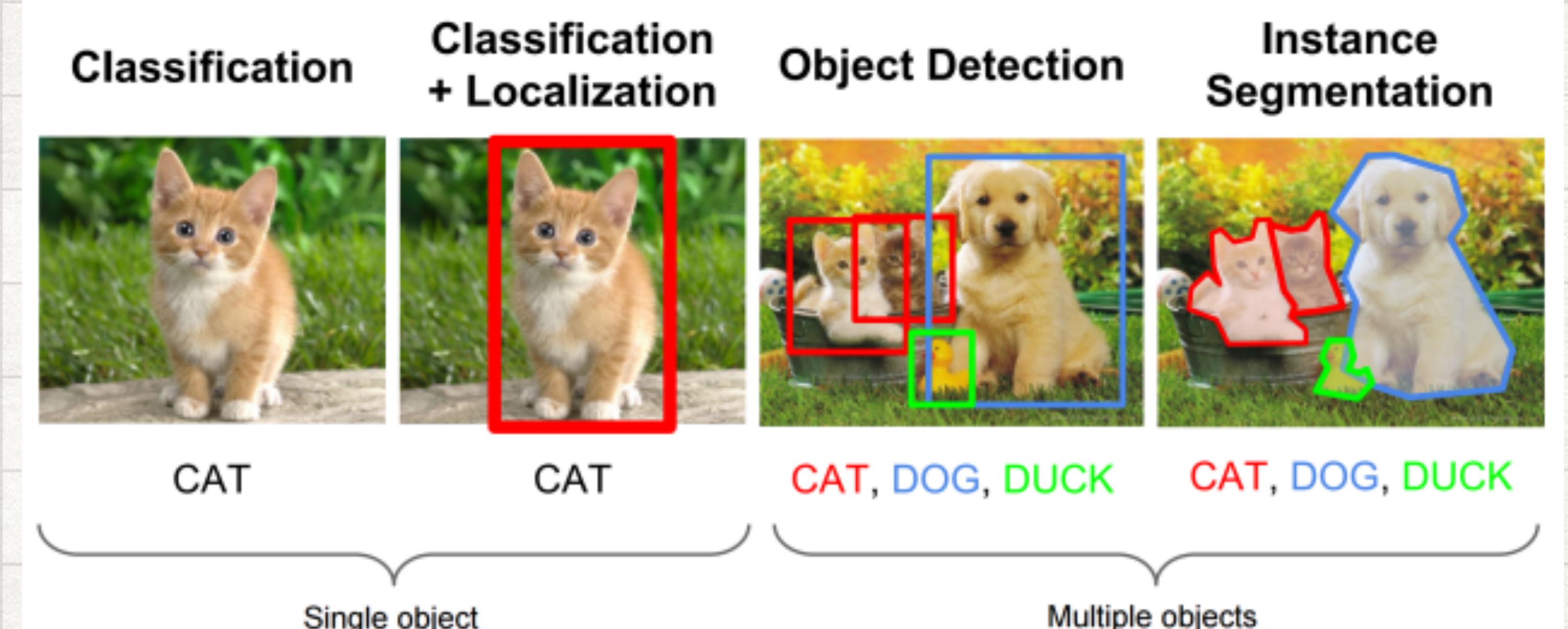
Multi-Label Classification



Bounding Box Regression
(Localization)

한 사진에 대해 여러 물체가 있을 때,
이것이 ‘어떤 물체’인지 분류하는 것

‘그 물체’가 사진 속 어디에 위치하는지
Bounding Box의 위치 정보를 나타내는 것



모델 선정 및 학습

<Object detection Model>

모델 소개

모델 선정

모델 학습

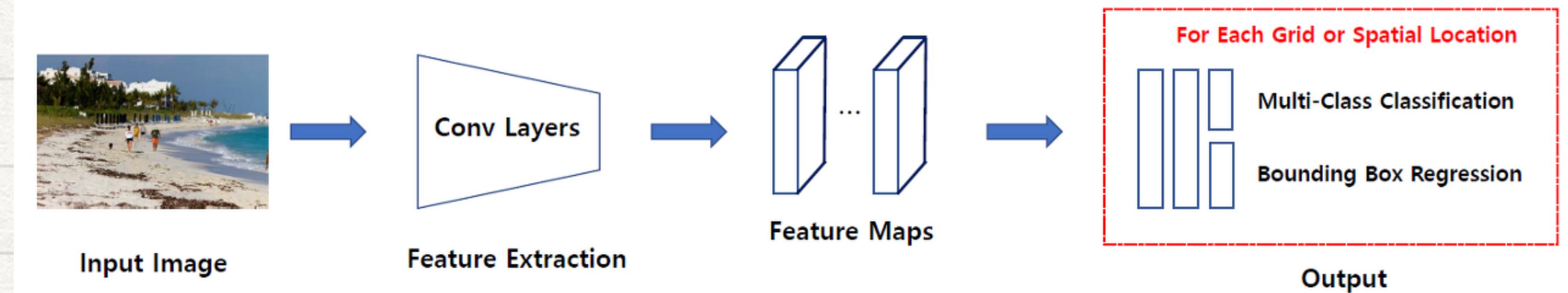
모델 평가

Backbone (Feature Extraction)

: 이미지의 특징을 추출하여 Feature map으로 변형
(VGG16 / ResNet-50 / DarkNet53 / MobileNet 등)

Head (Detector)

: Backbone에서 추출된 Feature map을 통해
Localization과 Classification을 수행
(R-CNN 계열 / YOLO / SSD 등)



모델 선정 및 학습

모델 소개

모델 선정

모델 학습

모델 평가

<Tensorflow Object detection API>

장점

- ◆ 고성능의 모델을 비교적 쉽게 학습 시킬 수 있다
- ◆ 다양한 객체 탐지 모델을 선택할 수 있다
- ◆ Pre-trained Model을 통해 전이학습을 할 수 있다

모바일 환경에서 작동하기 가벼운

<SSD MobileNet V2> 모델

Model name	Speed (ms)	COCO mAP	Outputs
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes

모델 선정 및 학습

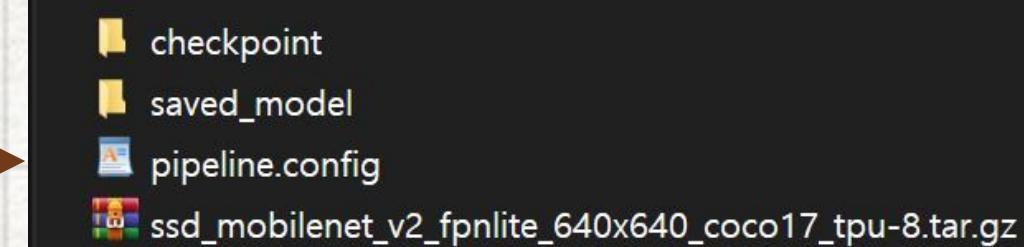
모델 소개

모델 선정

모델 학습

모델 평가

▶ 다운 받은 모델 압축 파일에서 Pipeline.config 파일을 찾는다



2021-03-16 오전 11:10	파일 폴더
2021-03-16 오전 11:10	파일 폴더
2021-03-16 오전 11:31	CONFIG 파일
2021-03-16 오전 11:09	WinRAR archive

5KB
20,038KB

◆ Pipeline.config 파일이란?

모델의 구조, 하이퍼 파라미터 값,
학습에 필요한 데이터 경로 등을
지정하는 Pipeline 설정 파일

▶ 우리의 시나리오에 맞게 파일 수정한다

```
model {  
    ssd {  
        num_classes: 4  
        image_resizer {  
            fixed_shape_resizer {  
                height: 640  
                width: 640  
            }  
        }  
        feature_extractor {  
            type: "ssd_mobilenet_v2_fpn_keras"  
            depth_multiplier: 1.0  
            min_depth: 16  
            conv_hyperparams {  
                regularizer {  
                    l2_regularizer {  
                        weight: 3.9999998989515007e-05  
                    }  
                }  
                initializer {  
                    random_normal_initializer {  
                        mean: 0.0  
                        stddev: 0.00999999776482582  
                    }  
                }  
                activation: RELU_6  
                batch_norm {  
                    decay: 0.996999979019165  
                    scale: true  
                    epsilon: 0.001000000474974513  
                }  
            }  
            use_depthwise: true  
            override_base_feature_extractor_hyperparams: true  
            fpn {  
                min_level: 3  
                max_level: 7  
                additional_layer_depth: 128  
            }  
        }  
    }  
}
```

모델 선정 및 학습

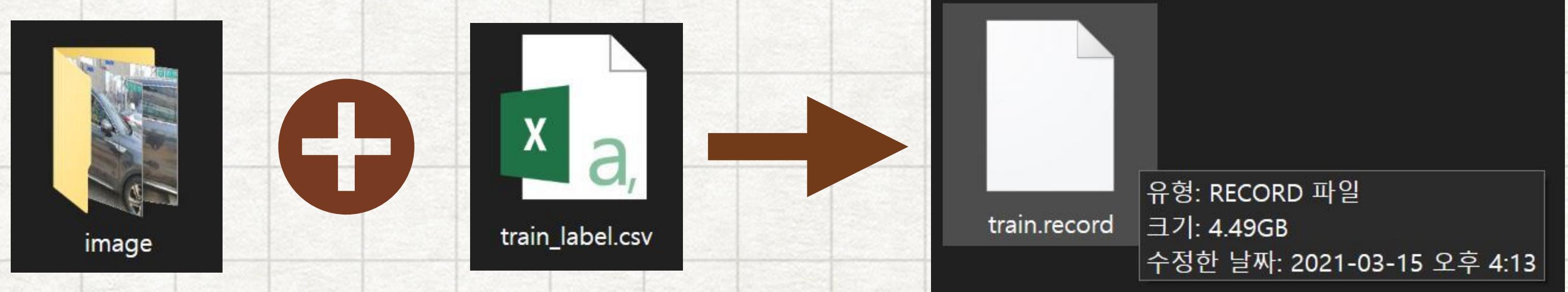
모델 소개

모델 선정

모델 학습

모델 평가

▶ 데이터 Label 파일과 데이터 이미지 파일을 Tensorflow Record파일로 변환한다



TFRecord 파일의 특징

- 1 Label과 이미지를 통합하여 한 파일에 저장
- 2 바이너리 데이터 형식으로 저장되어 이미지 형식 별 인코딩/디코딩의 시간을 절약해서 모델 학습 시간 단축시킴
- 3 이미지를 모델이 학습할 size로 resize 할 때, 데이터 용량이 줄어든다

모델 선정 및 학습

모델 소개

모델 선정

모델 학습

모델 평가

▶ 100step마다 log가 출력되며 모델 학습이 시작됐다

Tensorboard도 지원이 되어 훈련 상황을 시각화하여 확인할 수 있다

```
in Training Steps
INFO:tensorflow:Step 100 per-step time 0.487s loss=1.407
I0322 14:30:37.200727 4260 model_lib_v2.py:672] Step 100 per-step time 0.487s loss=1.407
INFO:tensorflow:Step 200 per-step time 0.487s loss=1.698
I0322 14:31:24.936470 4260 model_lib_v2.py:672] Step 200 per-step time 0.487s loss=1.698
INFO:tensorflow:Step 300 per-step time 0.477s loss=1.473
I0322 14:32:12.369146 4260 model_lib_v2.py:672] Step 300 per-step time 0.477s loss=1.473
INFO:tensorflow:Step 400 per-step time 0.447s loss=0.975
I0322 14:32:59.727805 4260 model_lib_v2.py:672] Step 400 per-step time 0.447s loss=0.975
INFO:tensorflow:Step 500 per-step time 0.469s loss=1.243
I0322 14:33:47.666594 4260 model_lib_v2.py:672] Step 500 per-step time 0.469s loss=1.243
INFO:tensorflow:Step 600 per-step time 0.457s loss=0.860
I0322 14:34:35.357328 4260 model_lib_v2.py:672] Step 600 per-step time 0.457s loss=0.860
INFO:tensorflow:Step 700 per-step time 0.480s loss=1.316
I0322 14:35:23.008053 4260 model_lib_v2.py:672] Step 700 per-step time 0.480s loss=1.316
INFO:tensorflow:Step 800 per-step time 0.457s loss=0.864
I0322 14:36:10.520746 4260 model_lib_v2.py:672] Step 800 per-step time 0.457s loss=0.864
INFO:tensorflow:Step 900 per-step time 0.470s loss=1.177
I0322 14:36:57.836396 4260 model_lib_v2.py:672] Step 900 per-step time 0.470s loss=1.177
```

모델 선정 및 학습

모델 소개

모델 선정

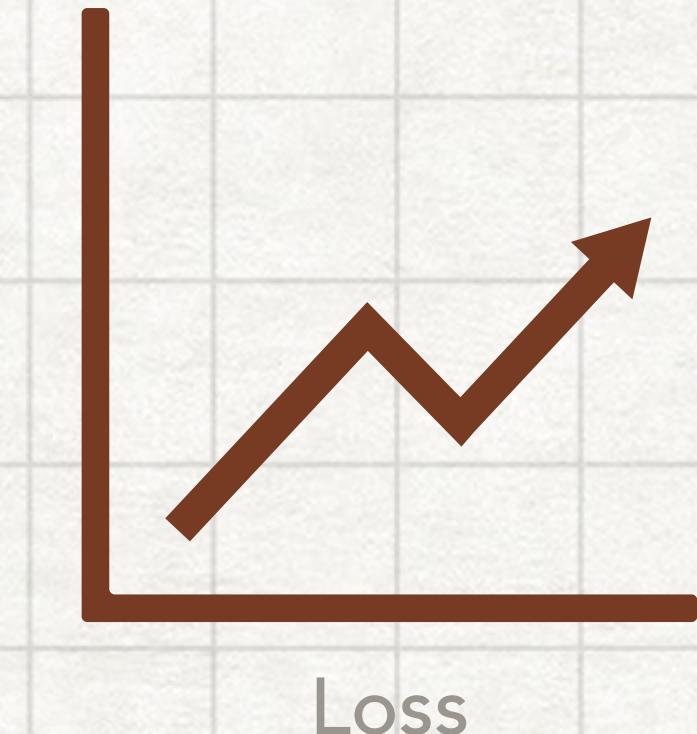
모델 학습

모델 평가

문제 발생

- ▶ SSD-MobileNet 320X320 모델이 학습 도중

Loss가 급격하게 튀는 현상이 발생했다
(최소 100,000~억 단위 까지)



- ▶ 이미지가 resize되면서 라벨링 Box의 좌표값이 손실되는 등
문제가 되는 것으로 추측했다
(신호등이 워낙 작다보니 Box 라벨링의 크기가 작은 경우)



더 큰 이미지 size를 소화하는 다른 모델로 학습을 시켜보자

모델 선정 및 학습

모델 소개

두 번째 모델
<SSD-MobileNet 640X640>

모델 선정

모델 학습

모델 평가

▶ 이전 모델보다 input size가 4배 커졌다

다

학습 시 Loss가 안정적으로 떨어졌다

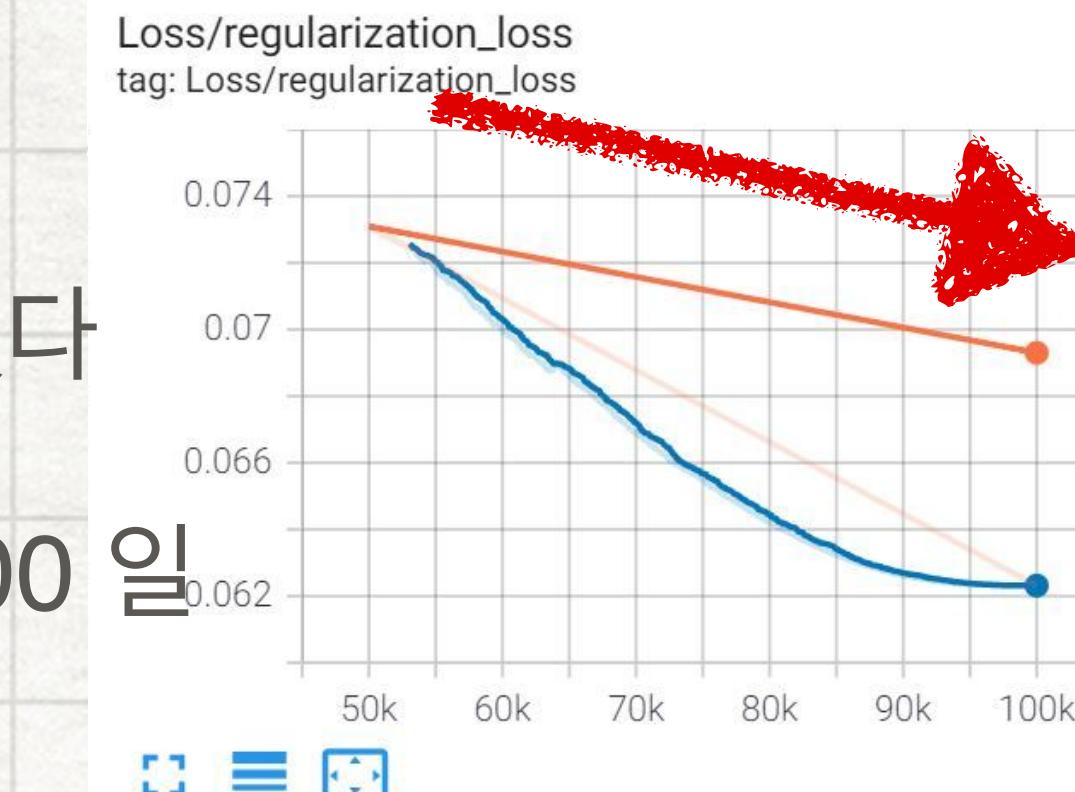
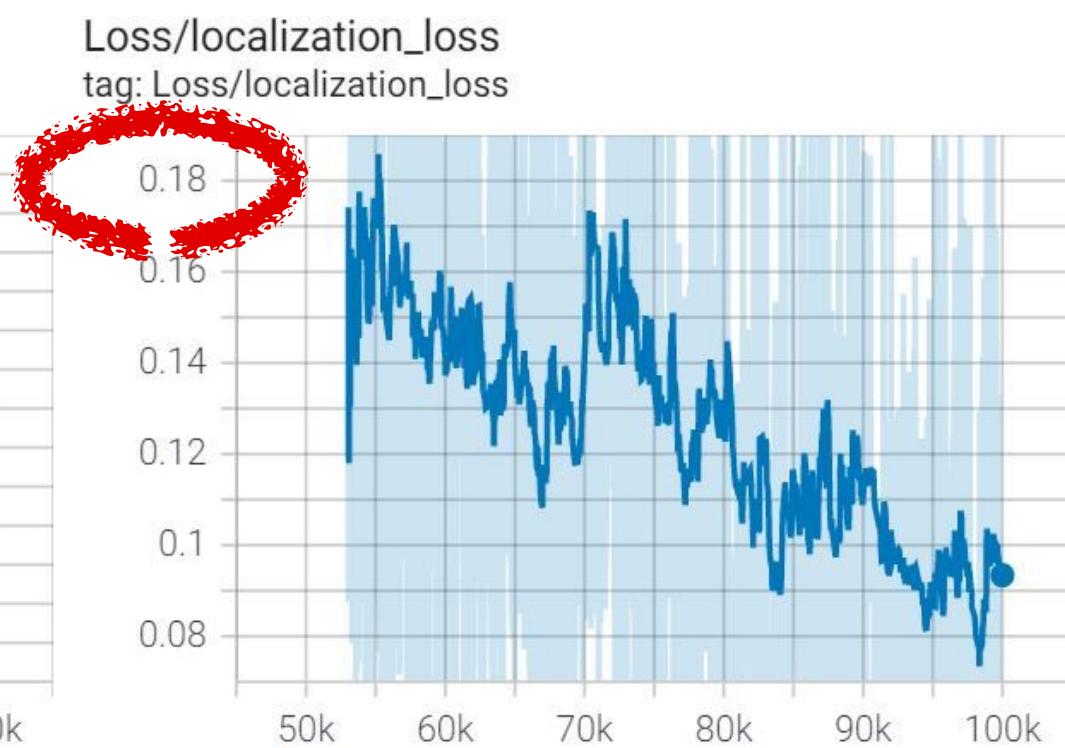
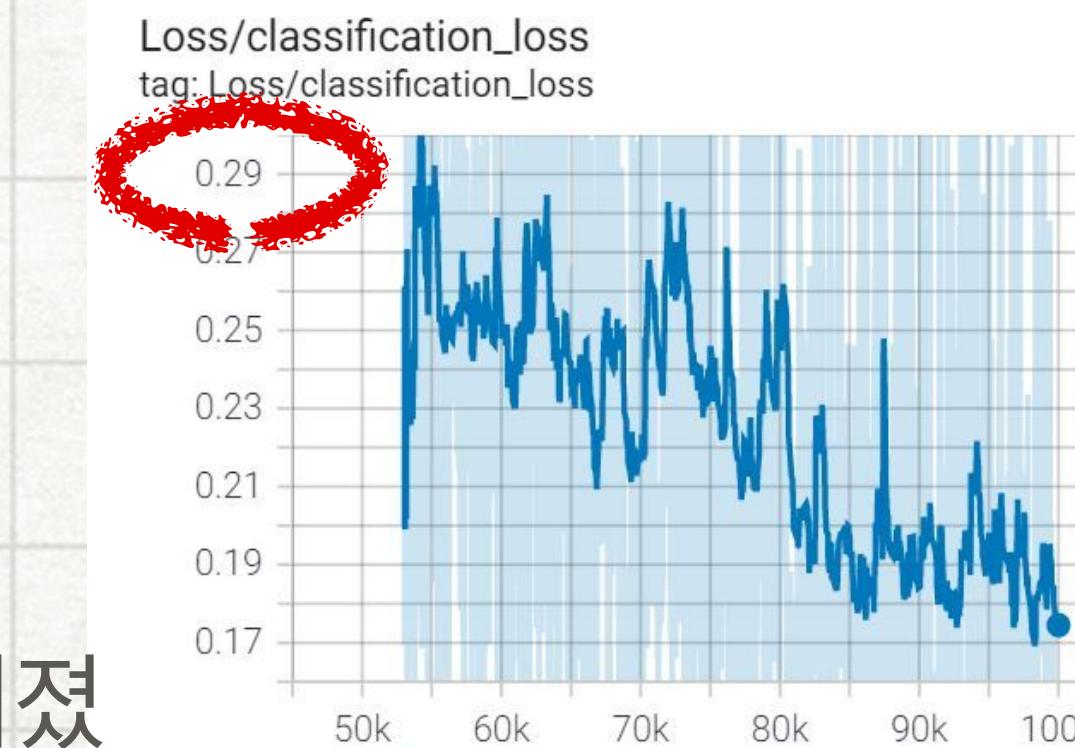
▶ Batch size 4, Total step 100,000 일

때,

전체 Loss가 1 이하인 것을 확인했다

▶ 50k, 100k step에서 모델의 validation을 실시했다

다



모델 선정 및 학습

모델 소개

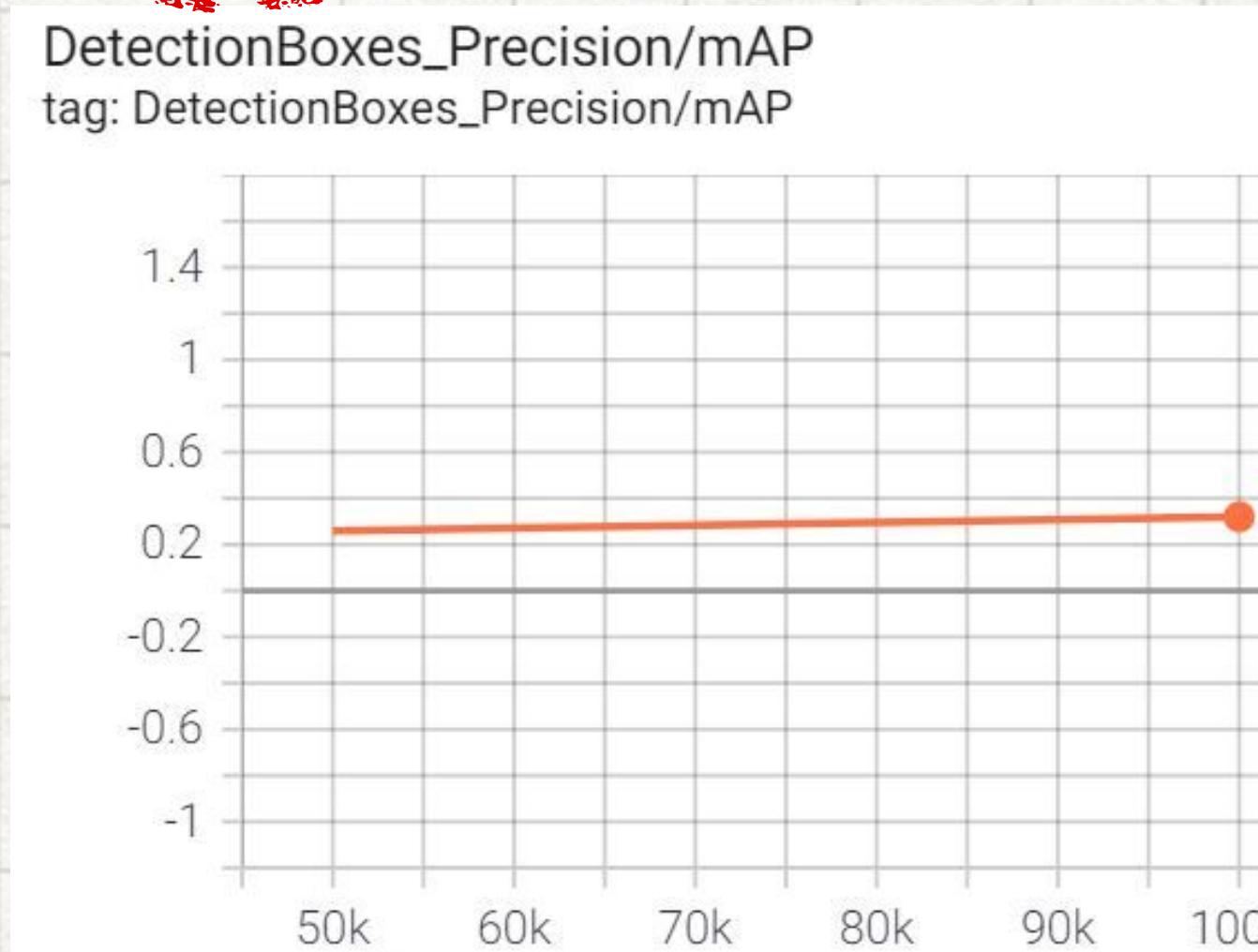
모델 선정

모델 학습

모델 평가

<MS COCO 방식의 평가 실시>

mAP	mAP(large)	mAP(medium)	mAP(small)	mAP@.50IOU	mAP@.75IOU
32.0	13.6	28.0	19.6	60.8	26.4



▶ mAP? (Mean average precision)

Multiple object detection 알고리즘에 대한

성능을 1개의 scalar value로 표현한 것

정밀도(precision)와 재현율(recall)로 산정한

클래스 별 평균 정밀도를 한번 더 평균 낸 값

<Pre-trained COCO 데이터셋>

Model name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes

학습한 모델을 안드로이드에 어떻게 배포할 것인가?

<Tensorflow Lite>

Edge Machine Learning으로,
서버를 통해서 데이터를 주고 받는 것이 아니라
네트워크의 Edge 기기에서 ML을 실행한다

- ◆ 짧은 응답 속도 서버에 접근할 필요 없이 기기에서 수행한다
- ◆ 개인정보 보호 서버에 데이터를 전달하지 않아 개인 정보 노출 위험이 없다
- ◆ 네트워크 불필요 서버에 접근하지 않아 오프라인에서도 사용이 가능하다
- ◆ 낮은 소비 전력 네트워크 연결을 위한 전력 소모가 없고,
모델 사이즈가 작아서 전력 소모도 적다

▶ 학습이 완료된 가중치 파일과 모델 Pipeline을 통해 Tensorflow Lite에 맞게 추출한다

ckpt-102.data-00000-of-00001	2021-03-17 오전 1:36	DATA-00000-OF-...	20,264KB
ckpt-102.index	2021-03-17 오전 1:36	INDEX 파일	47KB
pipeline.config	2021-03-17 오전 7:54	CONFIG 파일	5KB



```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_saved_model('F:/project/ob_project/model/fpn640v1/saved_model')
tflite_model = converter.convert()

with open('F:/project/ob_project/model/fpn640v1/saved_model/model.tflite', 'wb') as f:
    f.write(tflite_model)
f.close()
```



▶ 추출된 모델을 tflite 형식으로 변환한다

assets	2021-03-17 오전 7:49	파일 폴더
variables	2021-03-17 오전 7:49	파일 폴더
model.tflite	2021-03-17 오전 7:50	TFLITE 파일 11,783KB
saved_model.pb	2021-03-17 오전 7:49	PB 파일 11,524KB

```
1  from tflite_support import metadata as _metadata
2  from tflite_support import metadata_schema_py_generated as _metadata_fb
3  import flatbuffers
4  import os
5
6  model_meta = _metadata_fb.ModelMetadataT()
7  model_meta.name = "MobileNetV2 SSD 300x300 Object Detector"
8  model_meta.description = ("Localize the most prominent object in the"
9  "image from a set of N categories such as"
10 "Crosswalk, Green/Red signal, Car")
11 model_meta.version = 'V1'
12 model_meta.author = "Phoenix"
13 model_meta.license = ('Apache License. Version 2.0 '
14 'http://www.apache.org/licenses/LICENSE-2.0. ')
15
16 input_meta = _metadata_fb.TensorMetadataT()
17
18 input_meta.name = 'Image'
19 input_meta.description = (
20     "Input image to be classified. The expected image is {0} x {1}, with "
21     "three channels (red, blue, green) per pixel. Each value in the "
22     "tensor is a single byte between 0 and 255. ".format(640, 640))
23 input_meta.content = _metadata_fb.ContentT()
24 input_meta.content.contentProperties = _metadata_fb.ImagePropertiesT()
25 input_meta.content.contentProperties.colorSpace = _metadata_fb.ColorSpaceType.RGB
26 input_meta.content.contentPropertiesType = _metadata_fb.ContentProperties.ImageProperties
27 input_normalization = _metadata_fb.ProcessUnitT()
28 input_normalization.optionsType = _metadata_fb.ProcessUnitOptions.NormalizationOptions
29 input_normalization.options = _metadata_fb.NormalizationOptionsT()
30 input_normalization.options.mean = [127.5]
31 input_normalization.options.std = [127.5]
32 input_meta.processUnits = [input_normalization]
33 input_stats = _metadata_fb.StatsT()
34 input_stats.max = [255]
35 input_stats.min = [0]
36 input_meta.stats = input_stats
37
38 labelmap_file = "F:/Android/object_detection/android/app/src/main/assets/labelmap.txt"
39 exported_model_path = "F:/Android/object_detection/android/app/src/main/assets/model_fpnlite640.tflite"
40
41 output_tensorGroups = _metadata_fb.TensorGroupT()
42 output_tensorGroups.name = "detection result"
43 output_tensorGroups.tensorNames = ["location", "category", "score"]
44
45 location = _metadata_fb.TensorMetadataT()
46 location.name = "location"
47 location.description = "The location of the detected boxes. "
48 location.content = _metadata_fb.ContentT()
49 location.content.contentProperties = _metadata_fb.BoundingBoxPropertiesT()
```

▶ 최종적으로 안드로이드에서 사용될 수 있도록
tflite 파일에 메타데이터를 삽입한다

안드로이드 APP 개발

▶ SSD-Mobilenet 모델의 경우 구글에서 데모앱을 제공한다

데모앱을 다운받아 모델과 라벨을 학습시킨 것으로 변경했다

데모앱 변경

TTS 코드

시나리오 코드

UI 디자인

(4) > examples-master > lite > examples > object_detection > android > app > src > main > assets

이름	수정한 날짜	유형	크기
detect.tflite	2021-03-18(목) 오후 5:...	TFLITE 파일	4,088KB
labelmap	2021-03-18(목) 오후 5:...	텍스트 문서	1KB
model_fpnlite640.tflite	2021-03-18(목) 오후 5:...	TFLITE 파일	11,786KB

labelmap - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

Crosswalk
Traffic_Light_Green
Traffic_Light_Red
Car

```
// Configuration values for the prepackaged SSD model.  
  
private static final int TF_OD_API_INPUT_SIZE = 640;  
private static final boolean TF_OD_API_IS_QUANTIZED = false;  
private static final String TF_OD_API_MODEL_FILE = "model_fpnlite640.tflite"; //새로 학습한 모델  
private static final String TF_OD_API_LABELS_FILE = "labelmap.txt"; //새로 한 라벨링  
private static final DetectorMode MODE = DetectorMode.TF_OD_API;  
// Minimum detection confidence to track a detection.  
private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.20f;  
private static final boolean MAINTAIN_ASPECT = false;  
private static final Size DESIRED_PREVIEW_SIZE = new Size( width: 1088, height: 1088); // 640, 480  
private static final boolean SAVE_PREVIEW_BITMAP = false;  
private static final float TEXT_SIZE_DIP = 10;  
OverlayView trackingOverlay;  
private Integer sensorOrientation;
```

안드로이드 APP 개발

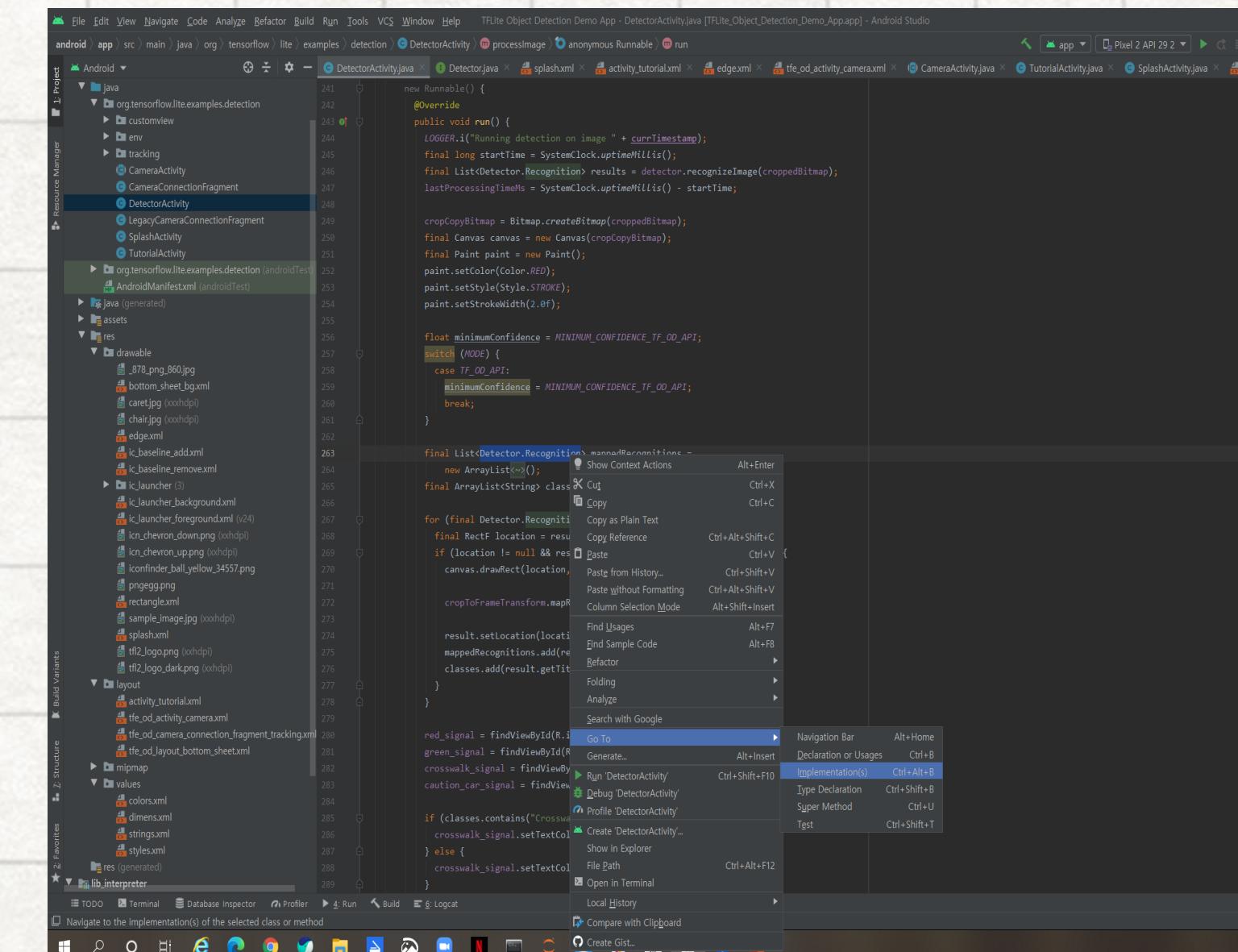
데모앱 변경

TTS 코드

시나리오 코드

UI 디자인

- ▶ 시각장애인을 위한 보행 도움 앱이기 때문에 Text-To-Speech (TTS)를 통해 음성으로 보행 안내를 하고자 했다
- ▶ 데모 앱에서 객체 탐지 후 결과 라벨이 저장되는 위치와 변수를 찾아야 했다



TFLiteObjectDetectionAPIModel.java

There you can add a log statement at line 227 for

recognitions object

for example

```
Log.i("Recognitions", String.valueOf(recognitions.get(0).getTitle()));
```

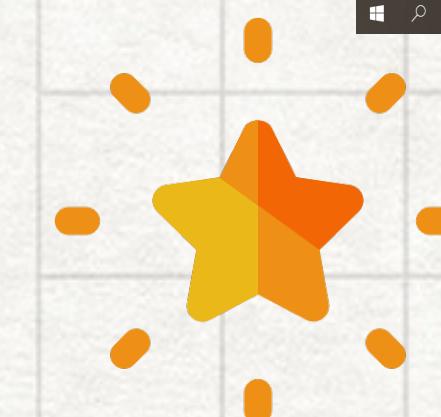
Second inside

DetectorActivity.java

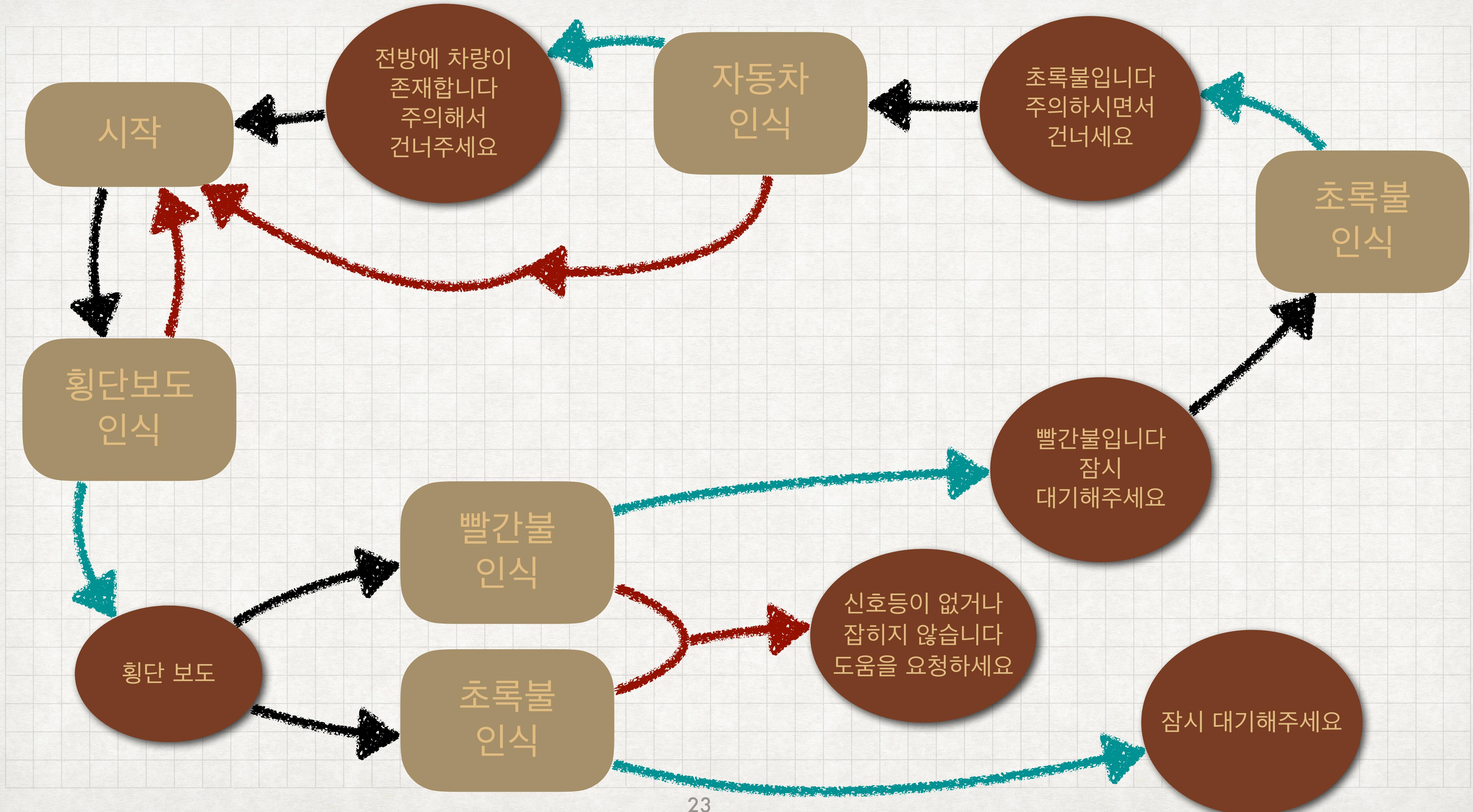
You can log

results object

at line 181.



인터넷 검색과 안드로이드 스튜디오 코드 탐색 기능을 이용해 결과값이 Recognition 클래스에 title 변수로 존재한다는 것을 찾아냈다



문제 발생

1 음성 안내가 밀려서 나옴

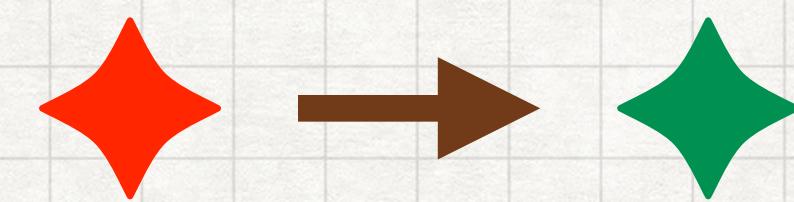


2 [빨->초]에만 건너야 하는데,

[빨->초] 후 초록색이 인식되면

건너라는 안내가 나옴

◆ 4초마다 객체를 새로 인식



◆ [빨->초]에 반응하는 특정 변수 생성

이 변수가 변해야만 초록불 안내 음성이 출력됨

특정 변수는 일정 시간이 지나면

원상태로 되돌아옴

안드로이드 APP 개발

▶ 어플의 용도가 잘 드러나도록 UI 디자인을 했다

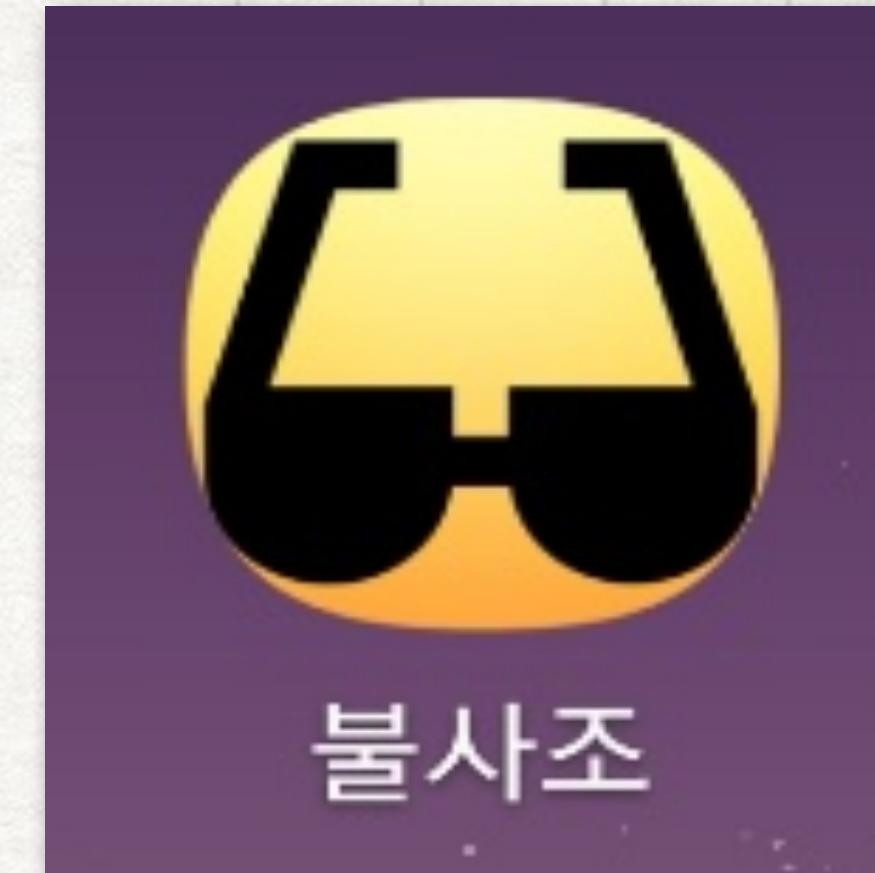
데모앱 변경

TTS 코드

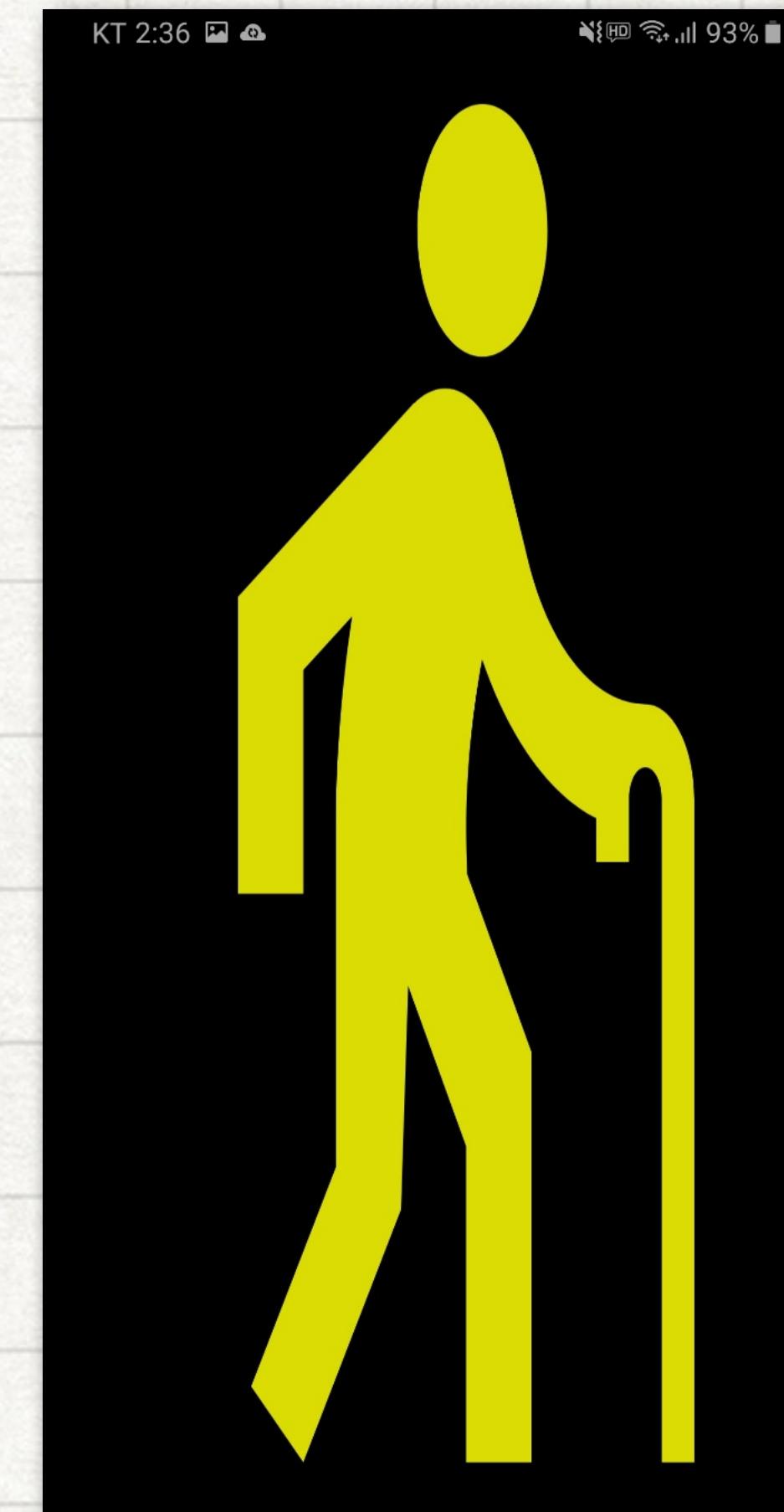
시나리오 코드

UI 디자인

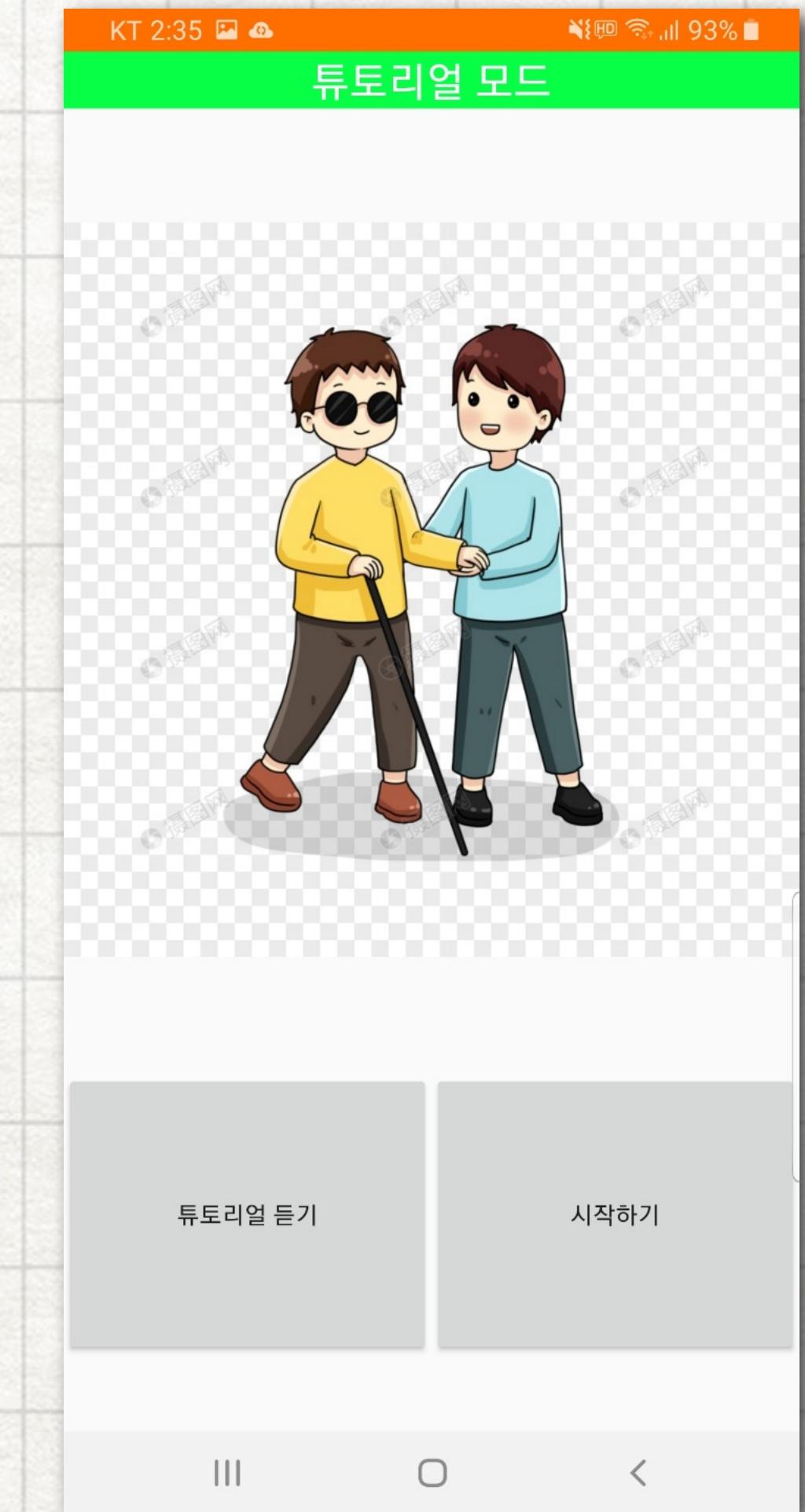
<아이콘과 이름>



<로딩 화면>



<튜토리얼 화면>



안드로이드 APP 개발

▶ 화면 아래부분에 인식된 객체의 라벨을 Bounding BOX의 색과 일치하게
출력하는 코드를 작성했다

데모앱 변경

TTS 코드

시나리오 코드

UI 디자인

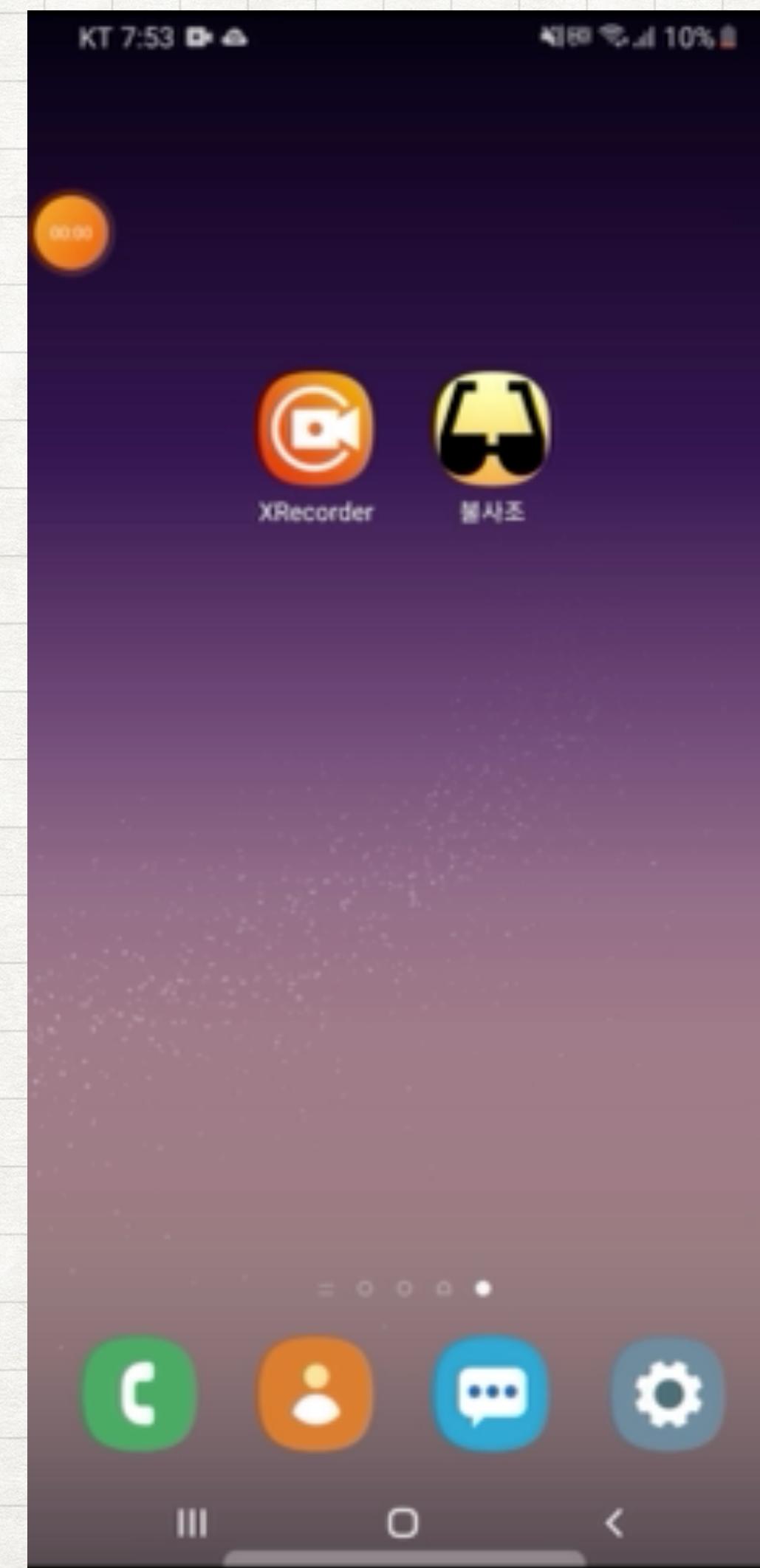


시연 영상

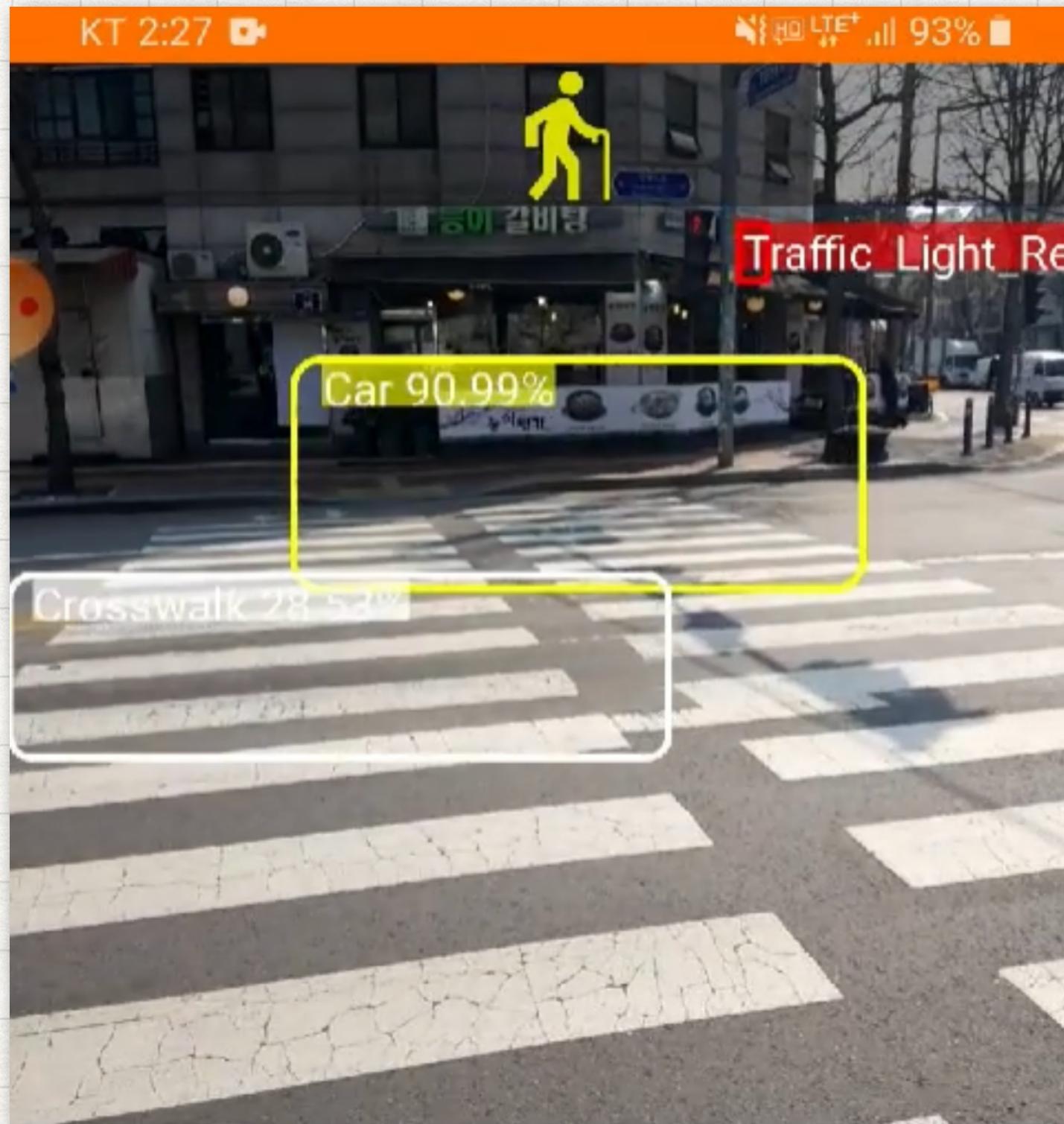
▶ 실제 구동 영상



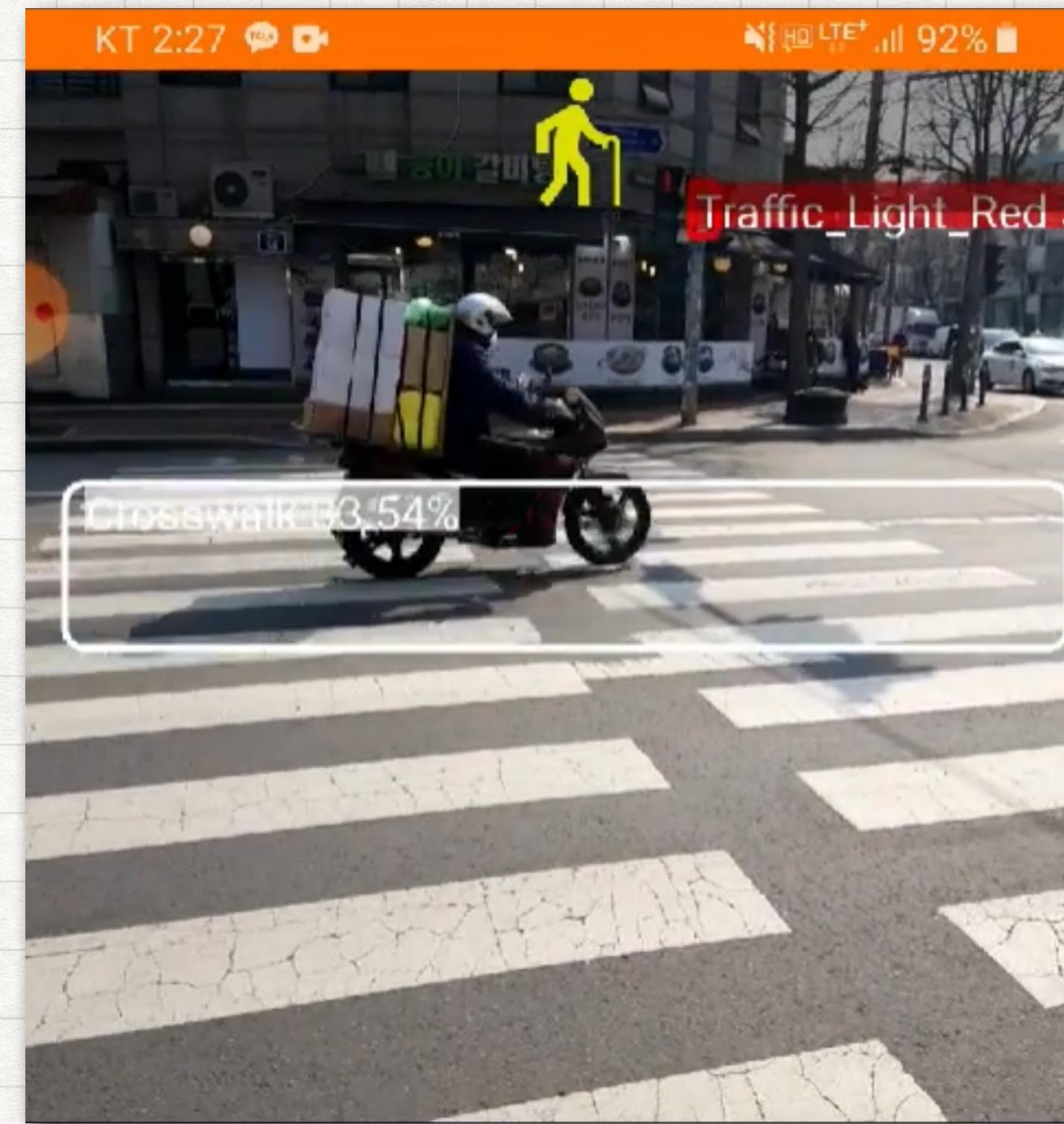
▶ 튜토리얼 영상



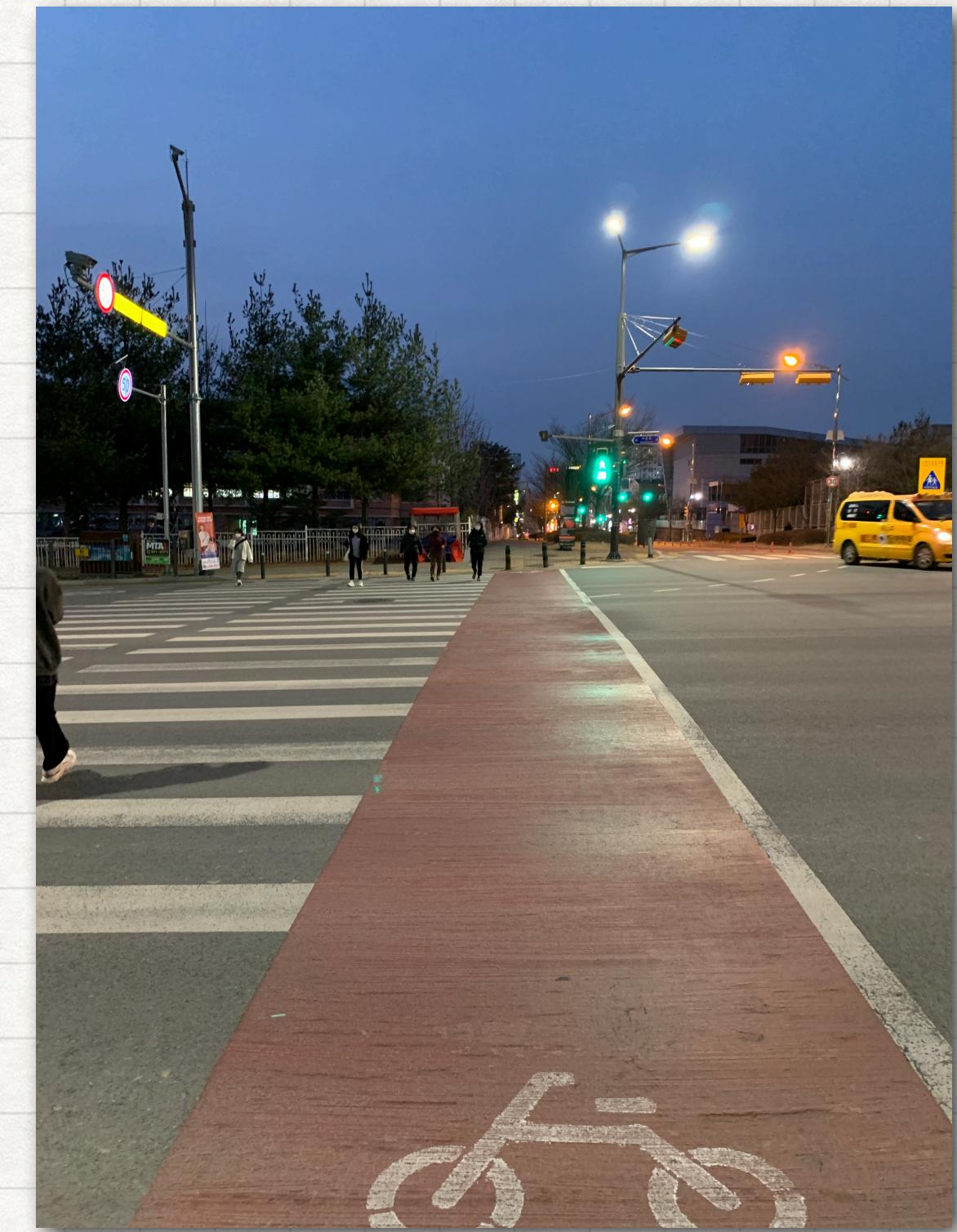
한계점



자동차가 늦게 잡힌다



오토바이를 생각하지 못했다



날씨, 햇빛에 영향을 받는다

프로젝트를 마치며 ...

신승배

프로젝트 주제를 현실성 있는 아이디어로 구체적으로 짜려면 많은 논의와 시간이 필요하다
프로젝트를 진행하면서 무엇보다 기초 개념 공부의 부족함과 중요성을 느꼈다

김종현

로컬로 모델을 학습하다보니 모델 학습에 오랜 시간이 걸렸다
객체탐지와 관련된 기초 이론을 완벽히 공부하지 않은 채로 모델을 학습 시키려다 보니 초반에 헤맸다
또, 잘 정제된 데이터의 중요성을 느꼈다

이다혜

프로젝트 주제 정하는 시기가 가장 어려우면서도 중요한 단계라는 걸 느꼈다
나 혼자서는 하지 못했을 과제를 팀원들과 함께 해서 해결할 수 있었다
모른다고 두지 말고, 적극적으로 배우려는 자세가 중요하다

유정용

수 천장의 라벨링 작업 과정이 오히려 훨씬 수월했다고 느낄 정도로,
프로젝트 초기에 팀원들과 주제 선정을 위해 서로 논의했던 시간이 힘들었다.

KISTI 4차인재 양성과정 팀프로젝트 최종 발표 끝

감사합니다

2020. 10. 05 ~ 2021. 03. 26