1. 코드 구성
   1. **original/cross.py**
      1. 원본 코드
   2. **final/detect\_local.py**
      1. 우리가 구현한 코드
   3. final/detect
      1. TRIP을 구현하기 위해 시도해봤던 코드들
      2. **cross\_detect.py**
         1. 영상처리를 통해 입력 영상에서 횡단보도를 찾아보는 코드(1)
         2. HSV, houghlinep등의 방법을 사용
      3. **cross\_test.py**
         1. 영상처리를 통해 입력 영상에서 횡단보도를 찾아보는 코드(2)
         2. 이미지 화질 변경, find contour등의 함수 사용
      4. **detect.py**
         1. 입력 영상에서 횡단보도 인식을 통해 보행자, 차량에게 안전 정보를 알려주는 코드
         2. 인식 모델로는 횡단보도, 차량 인식 모델을 각각 사용한다
      5. **detect\_show\_split.py**
         1. 입력 영상에서 횡단보도 인식을 통해 보행자, 차량에게 안전 정보를 알려주는 코드
   4. final/train
      1. 모델을 학습할 때 쓰인 코드들
      2. car\_pedestrian/
         1. 차량, 보행자 인식 모델 관련
         2. **YOLO\_train.py**
            1. yolo 모델을 이용하여 모델을 학습하는 코드
         3. **convert.py**
            1. AI-HUB에서 제공한 label파일을 yolo 형식으로 변환하는 코드
         4. **convert\_train.py**
            1. train set의 label을 yolo 형식으로 변환하는 코드
         5. **convert\_valid.py**
            1. valid set의 label을 yolo 형식으로 변환하는 코드
         6. **draw\_box.py**
            1. 변환이 올바르게 되었는지 확인하기 위해 변환된 label 형식으로 사진상에 box를 그려보는 코드
      3. cross/
         1. 횡단보도, 빨간불, 초록불 인식 모델 관련
         2. **YOLO\_train.py**
            1. yolo 모델을 이용하여 모델을 학습하는 코드
         3. **convert\_class.py**
            1. 차량, 보행자 데이터와 횡단보도, 신호등 모델을 합칠 때 클래스 번호를 변환하는 코드
         4. **convert\_train.py**
            1. train set의 label을 yolo 형식으로 변환하는 코드
         5. **convert\_valid.py**
            1. valid set의 label을 yolo 형식으로 변환하는 코드
         6. **preprocess.py**
            1. dataset을 train, valid로 나누고 label을 yolo 형식으로 변환하는 코드
         7. **split.py**
            1. dataset을 train, valid로 나누는 코드
2. 데이터 구성
   1. 데이터의 용량이 너무 커서 구글 드라이브 링크를 첨부했습니다.
   2. <https://drive.google.com/drive/folders/1-kA-24ls4H_zT9m31wy0f8EVAqB3xAbM?usp=share_link>
   3. data/car\_pedestrian
      1. 차량, 보행자 데이터
      2. dataset1, dataset2로 나뉘어져 있습니다.
      3. 초반에 전이학습을 실시할 때 모든 데이터를 메모리에 캐싱 할 수가 없어서 두 부분으로 나누었습니다.
   4. data/cross
      1. 횡단보도, 신호등 데이터
      2. dataset1, dataset2, origin으로 나뉘어져 있습니다.
      3. 실제 학습을 진행한 데이터는 dataset1, dataset2입니다.
      4. Origin은 변환 이전의 label들이 들어있습니다.
3. 원본 코드에 대한 설명

* original/cross.py

‘빵형의 개발도상국’으로부터 참조한 코드입니다. 세계최초? 교차로 우회전 위반 적발하는 인공지능(<https://www.youtube.com/watch?v=kNRo2DryF58>)으로부터 참조했습니다.

코드로부터 참조한 부분은 다음과 같습니다.

1. .pt 파일을 불러와서 설정하는 부분

|  |
| --- |
| device = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')  ckpt = torch.load(MODEL\_PATH, map\_location=device)  model = ckpt['ema' if ckpt.get('ema') else 'model'].float().fuse().eval()  class\_names = ['횡단보도', '빨간불', '초록불'] # model.names  stride = int(model.stride.max())  colors = ((50, 50, 50), (0, 0, 255), (0, 255, 0)) # (gray, red, green) |

1. 이미지를 전처리 하는 부분

|  |
| --- |
| # preprocess  img\_input = letterbox(img, img\_size, stride=stride)[0]  img\_input = img\_input.transpose((2, 0, 1))[::-1]  img\_input = np.ascontiguousarray(img\_input)  img\_input = torch.from\_numpy(img\_input).to(device)  img\_input = img\_input.float()  img\_input /= 255.  img\_input = img\_input.unsqueeze(0)  # inference 횡단보도,신호등  pred = model(img\_input, augment=False, visualize=False)[0]  # postprocess  pred = non\_max\_suppression(pred, conf\_thres, iou\_thres, classes, agnostic\_nms, max\_det=max\_det)[0]  pred = pred.cpu().numpy()  pred[:, :4] = scale\_coords(img\_input.shape[2:], pred[:, :4], img.shape).round() |

1. Bounding box끼리의 좌표를 비교해서 위치관계를 파악하는 아이디어

|  |
| --- |
| if len(idxs) > 0:  for i in idxs.flatten():  class\_name = LABELS[class\_ids[i]]  x1, y1, w, h = boxes[i]  alert\_text = ''  color = (255, 0, 0) # blue  if class\_name == '차량':  if x1 < cw\_x2:  alert\_text = '[!위반]'  color = (0, 0, 255) # red  annotator.box\_label([x1, y1, x1+w, y1+h], '%s %d' % (alert\_text+class\_name, confidences[i] \* 100), color=color) |

* 이 외의 부분은 직접 구현했습니다.