# 교통표지판을 instance segmentation 하기

작성자: 유지수 국민대 명예교수
작성일자:2019.11.16

## Acknowledgment

본 문서의 내용은 Adam Kelly의 Udemy 강의 Complete Guide to Creating COCO Datasets를 참조해서 만든 것임. 본 문서의 Mask-RCNN이외의 python program은 모두 Adam Kelly의 강의에 나온 것을 그대로 사용한 것임. Adam Kelly의 강의를 저는 적극 추천함. COCO data format을 만드는 방법을 아주 자세하게 설명하고 있음. 그가 만든 python program도 프로그램의 정석이라고 해도 과언이 아닐 정도로 아주 compact하고 clear하게 되어 있음. 그의 코딩 스타일도 배울 수 있는 장점이 있음.

## 학습목표

**- coco data format에 대한 이해**
**- Mask R-CNN의 모델에 대한 이해**
**- GIMP를 사용하여 이미지에 mask를 처리하는 방법에 관한 학습**
**- Data Augmentation을 위한 이미지 합성 방법에 관한 학습**

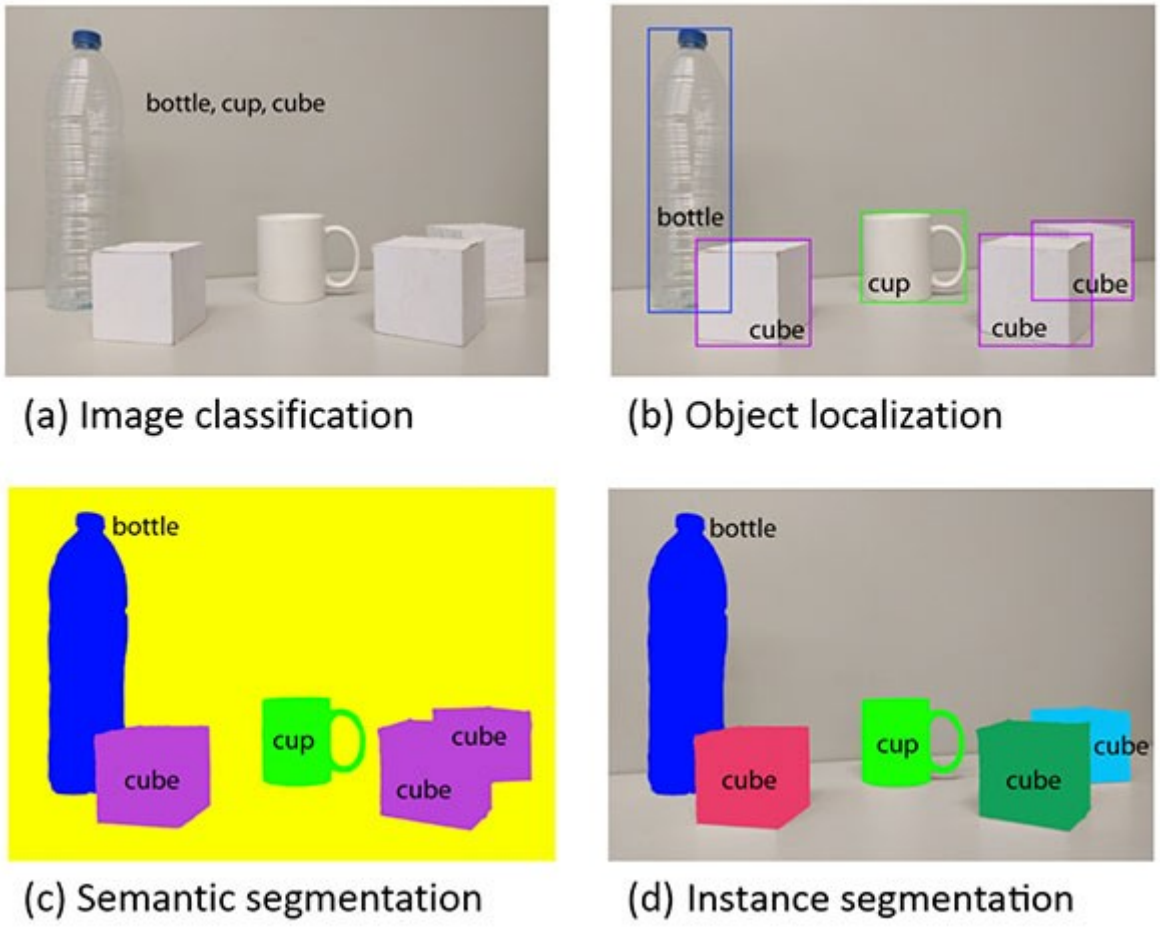
## 본 문서의 위치

https://github.com/jisooyu/coco_instance_segmentation 에서 이 문서를 clone할 수 있음.

## Deep Learning 의 object recognition의 다양한 방법

우선 object detection에 나오는 다양한 방법의 구별.
Image classification, object detection, semantic segmentation, instance segementation의 차이점을 visually 보여주는 블로그.
그리고 OpenCV에서 mask-RCNN을  어떻게 작동하는 가를 설명

Source: https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/



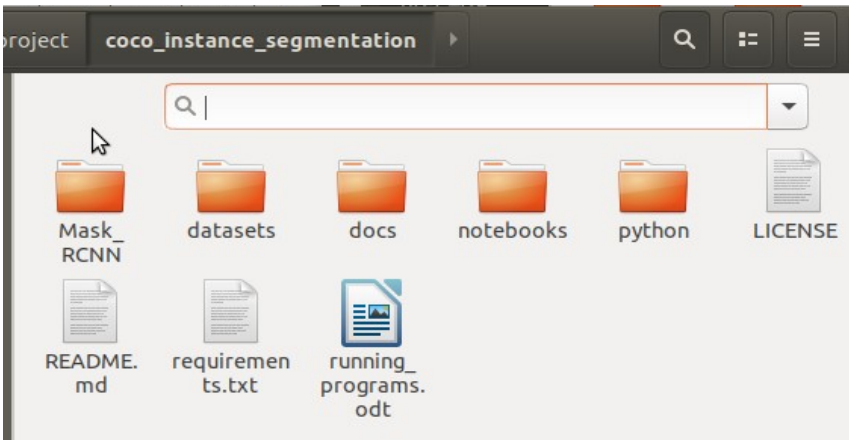

source: A Review on Deep Learning Techniques Applied to Semantic Segmentation
Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Jose Garcia-Rodriguez

## File Structure

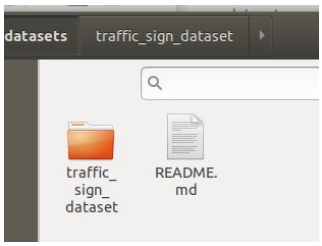
Adam Kelly 는 본인의 github에 프로그램을 만들어 놓았으니 git clone을 하셔서 위와 같은 file structure를 만들기 바람.

## https://github.com/akTwelve/cocosynth

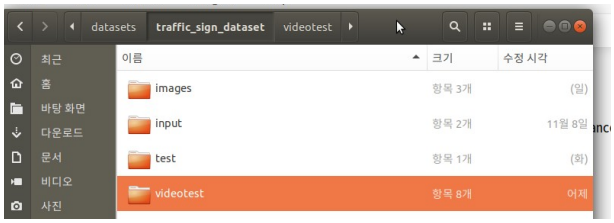
inside coco_instance_segmentation




Inside datasets




inside traffic_sign_dataset



본 강의에서는 directory 이름을  coco_instance_segmentation으로 하겠음.

여러분의 컴퓨터에 다음가 같이 실행하여 directory를 만드시고 그 directory로 이동하시기 바람. 물론 directory name을 coco_instance_segmentation이 아닌 다른 이름으로 하셔도 무방함.

**mkdir coco_instance_segmentation && cd coco_instance_segmentation**

그리고 우리가 사용할 mask_rcnn모델을 다음에서 clone하셔서 여러분의 directory (우리 강의의 경우 coco_instance_segmentation) 에 저장하시기 바람.

https://github.com/matterport/Mask_RCNN

위의 모델을 우리 directory notebooks에 있는  "train_mask_rcnn.ipynb" Jupyter notebook 프로그램에서 사용할 것임.

## Tensorflow GPU 설치 유의점

https://www.tensorflow.org/install/gpu

instance segmentation model인 mask_rcnn을 사용하려면
1. Google colab을 사용
2. GPU 설치가 되어 있는 PC를 사용

사실 가장 편한 방법은 1번인 Google colab를 사용하는 것임. gpu/tpu 사용시 시간 당 약 4000원 정도 인 것으로 알고 있음.  Google gmail 이 있어야 하고 google drive에 우리의 datasets을 올려 놓아야 사용이 가능함.  사용 방법은 YouTube 나 여러 blog에 나와 있음.

2번의 방법을 택하면 여러분이 tensorflow-gpu를 설치해야 하는 데 만만치 않은 작업임.
GPU driver, CUDA, Cudnn, tensorflow, tensorflow-gpu, keras 등이 서로 정합성 (compatibility)가 있어야 함. 쉽지 않은 작업.

2번의 방법을 선택하면

1. 시스템 level (conda virtual env에서가 아님)에서 Cuda driver 설치
   ① 이미 설치된 경우가 많겠지만 system level에서 Cuda driver를 설치해야 함. 이곳을 참조할 것.
   https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#pre-installation-actions 반드시 7. Post-installation Actions을 따라 .bashrc에 CUDA 설정을 해야 함.

2. tensorflow-gpu, cudatoolkit, cudnn, 그리고 other libraries 설치
   ① 제가 시행착오 끝에 찾은 가장 믿을 만한 설치 블로그는 https://medium.com/@redowan/no-bullshit-guide-on-installing-tensorflow-gpu-ubuntu-18-04-18-10-238924cc4a6a 임 참조할 것. 단, Nvidia Graphic Card의 driver 가 이미 설치되어 있다면 이 부분은 skip하고 tensorflow-gpu와 cudnn을 설치 하시기 바람.

   **② conda create -n tfcoco-gpu python=3.6**

   **③ conda activate tfcoco-gpu**

   **④ conda install **
   **tensorflow-gpu=1.12 **
   **cudatoolkit=9.0 **
   **cudnn=7.1.2 **
   **h5py**
   ----그리고 mask-rcnn model을 사용하면 다음을 더 설치---

   **⑤ conda install -c conda-forge imgaug**

   **⑥ conda install -c conda-forge jupyter**

   **⑦ conda install -c conda-forge tqdm**

   **⑧ conda install -c conda-forge ipykernel**

   **⑨ python -m ipykernel install --user --name  tfcoco_gpu --display-name "tfcoco_gpu"**

**(주요 사항) 나중에 jupyter notebook을 실행할 때 menu에서 kernel → change kernel → tfcoco_gpu를 선택한 후 menu에서 cell → run all을 실행 할 것.**

Note: 다음 조합으로 virtual env를 만들어도 무난히 돌아가는 것 같음.

**conda install **
**tensorflow-gpu=1.12 **
**cudatoolkit=9.0 **
**cudnn=7.3.1 **
**h5py**

## GIMP 프로그램 다운 로드

본 강의에서는 instance segmentation에 필요한 COCO data format을 만들기 위해 GIMP를 사용할 것임. 이 강의의 핵심 중 하나는  GIMP를 사용하는 방법을 터득하는 것임. 다음에서 GIMP를 다운 받을 것. GOOD NEWS: GIMP 는 무료.

https://www.gimp.org/downloads/

GIMP를 사용하여 object에 색칠을 함으로써 instance를 구분할 수 있도록 하는 것임. Pascal VOC format을 만드는 것보다는 수작업이 더 필요 하기는 하지만 각 instance별로 형체를 구분하도록 함으로써 Pascal VOC 데이터 포멧보다 더 정확한 인식이 가능함.

## GIMP 사용법

- open file
  - File → Open → datasets/traffici_sign_dataset/images에서 image file 선택
- check
  - windows→Dockable Dialogs → Channels
  - 만약 channels가  indexed 로 되어 있다면 Image → Mode → RGB로 변경
  - 이미 RGB로 되어 있다면 그대로 유지 할 것
- 왼쪽 상단에 있는 아이콘 중 Foreground Select Tool을 선택(왼쪽 상단 첫번째 줄에서 7번째 아이콘)
  - 주의 : 반드시 하단에서 feather edges 를 check
  - contiguous 가 있다면 이는 uncheck
- 이미지 크기를 필요하다면 변경
  - image → scale image → 255 X 255
- 필요한 이미지 부분을 추출해서 이를 coco_instance_segmentation의 datasets로 내보냄 ( **여기에 기술한 mask 처리 방법이 가장 좋은 방법인지는 확신이 없음. 많은 제언을 바람**)
  - 문서로 설명하기 어려운 내용이므로 유투브 강의를 참조하시기 바람.
  - (https://www.youtube.com/watch?v=-_eOBXnx0k8)

## 프로그램의 실행

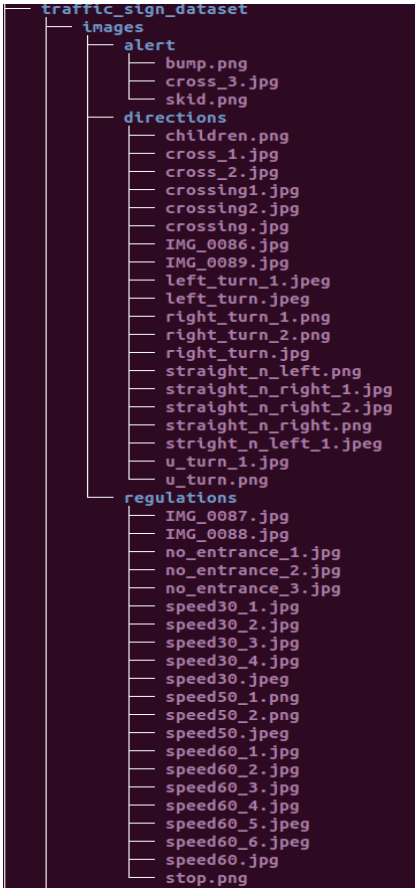
train에 사용할 coco_instances.json format data를 만드는 것이 궁극적인 목표임.

- **Directory와 image 준비 사항**

이를 위해 우선 우리가 준비한 image와 mask를 사용하여 train 용과 validation 용의 데이터를 만들어야 함. direcory구조를 보면 다음과 같음.

우리는 이미 traffic_sign_dataset에 input directory를 만들어 놓았음. 앞으로 image_composition.py를 돌려 여기에 train, val을 만들 것임. test와 videotest는 여러분이 직접 만들어 test에는 이미지 파일을 videotest에는 여러분이 촬영한 비디오를 저장해 놓아야 함.

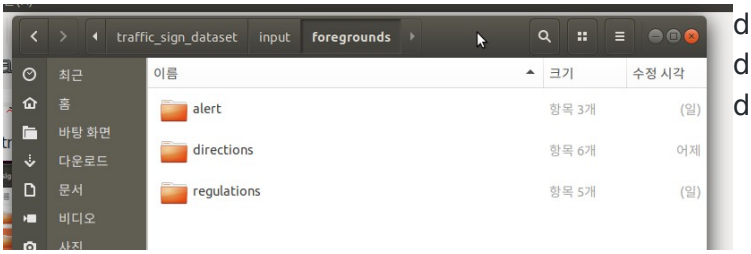
우리가 이미 만들어 놓은 dirctory에 대해 다시 복습을 하여 보면..
우선 traffic_sign_dataset directory안에는 images가 있고 이 안에 alert, directions, regulations 가 있음. ....




그리고 input, test, videotest가 있고
input 안에는 foreground와 background를 만들어 놓은 것을 알 수 있음.

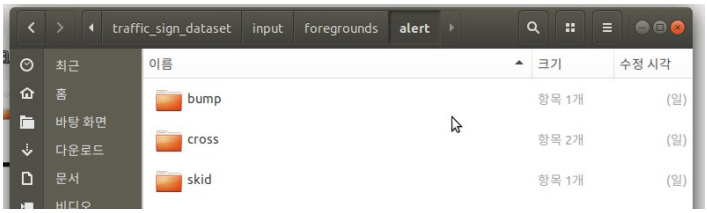
다시 foreground 안을 보면…

alert, directions, regulations 의 대분류 교통표지판 directory를 만들어 놓았고 그리고 다시 이

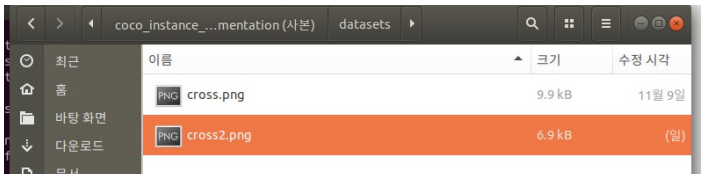



directory 내부에는 소분류 교통표지판 directory가 있음. 예를 들어 alert 대분류 directory내부를 보면…

bump, cross, skid와 같은 소분류 교통표지판 directory를 우리가 만들어 놓았음. 그리고 이 안에 각 소분류에 GIMP로 만든 mask layer를 저장해 놓아야 함 .




Cross 소분류 directory안에는 GIMP로 만든 mask layer 이미지가 만들어져 있어야 함. 만약 이 준비가 안되어 있다면 여러분이 이와 같은 directory structure를 만들고 해당 이미지를 GIMP로 만들어 저장해 놓아야 함.

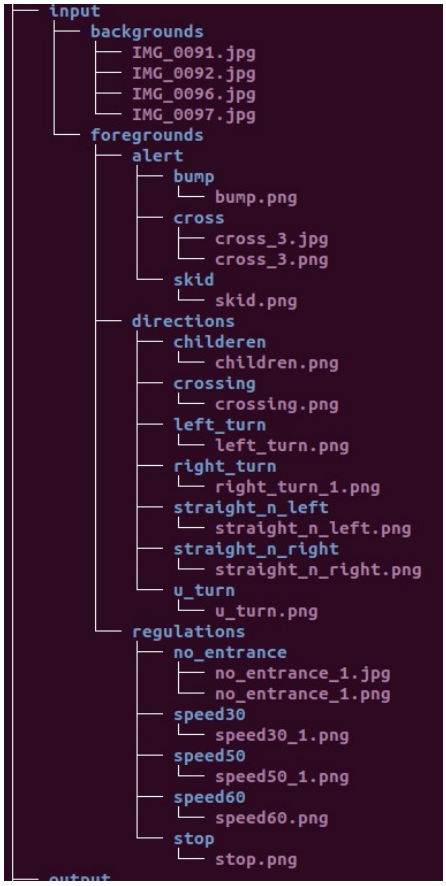



1.

마지막으로 traffic_sign_dataset 내에 videotest directory를 만들고 여러분이 거리에서 찍은 거리의 교통표지판 비디오를 이곳에 저장함. 나중에 우리가 훈련시킨 mask_rcnn이 정말로 교통표지판에 instance segmentation을 할 수 있는 지를 테스트 하기 위한 것임.

정리하면 ….

datasets → traffic_sign_datasets → input → foreground → alert/directions/regulations → 소분류로 구성되어 있어야 함.
소분류는
alert → bump/cross/skid
directions → children/crossing/left_turn/right_turn/straight_n_left/straight_n_right/u_turn
regulations → no_entrance/speed30/speed50/speed60/stop
으로 나누어져 있음.




그리고 videotest directory에 여러분이 찍은 거리의 교통표지판 비디오가 저장되어 있어야 함.

Adam Kelly 가 만든 프로그램은 directory 이름으로 대분류/소분류 구분을 하게 되어 있음. 따라서 이런 structure만드는 것이 중요함.

- **image_composition.py 프로그램의 실행**

**(note) 여기 나오는 image_composition.py와 coco_json_utils.py 는 https://github.com/akTwelve/cocosynth 에서 git clone을 하셨다면 python directory내에 이미 이 프로그램들이 있을 것임.**

1. 일단 테스트로 실행 (선택사항)

```
python ./python/image_composition.py --input_dir ./datasets/traffic_sign_dataset/input \
    --output_dir ./datasets/traffic_sign_dataset/output --count 20 --width 710 --height 710
```

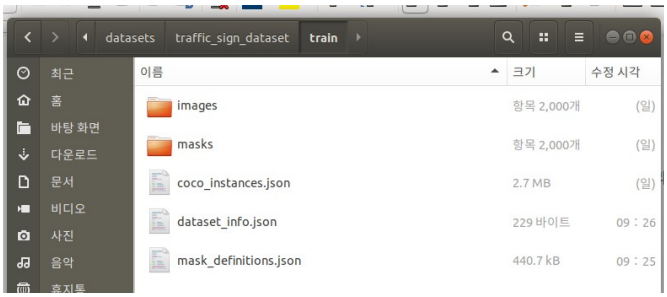
2. train data set을 만들기 위한 실행

```
python ./python/image_composition.py --input_dir ./datasets/traffic_sign_dataset/input \
    --output_dir ./datasets/traffic_sign_dataset/train --count 2000 --width 710 --height 710
```

3. validation data set을 만들기 위한 실행

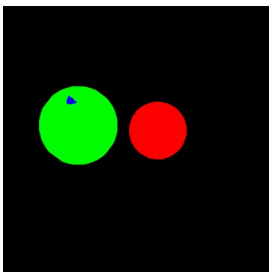
**python ./python/image_composition.py --input_dir ./datasets/traffic_sign_dataset/input **
**--output_dir ./datasets/traffic_sign_dataset/val --count 200 --width 710 --height 710**

1번의 실행은 우선 데이터에 문제가 없는 가를 확인 하기 위한 것임. 실행 안 해도 상관은 없음.
2번을 실행하면 traffic_datasets 내에 train directory 가 생성되고 그 안에 images와 masks directory가 생성됨. images와 masks안에 있는 이미지는 우리가 만든 foreground와 background의 이미지와 mask layer를 사용하여 random으로 합성파일을 만들어 놓은 것임.
images에는 합성 이미지, 그리고 masks에는 합성 mask가 만들어 졌음. 그리고 dataset_info.json, mask_definitions.json도 생성됨.
이 json file은 이미지 정보와 우리가 GIMP 로 만든 mask layer에 관한 정보를 수록하고 있음. 이 파일은 coco_json_utils.py에서 사용될 것임.




3번을 실행하면 2번과 같이 val directory안에 파일과 directory가 생성됨.

train directory내의 images와 masks directory 에 생성된 합성image와 합성mask의 예

- **coco_json_utils.py의 실행**

coco_json_utils.py은 전 단계에서 만든 파일을 사용하여 mask_rcnn 모델에서 사용할 수 있는 json 포멧(coco_instances.json)의 파일을 만듬. 실행 후에는 coco_instances.json 파일이 생김. 이 파일은 mask_rcnn model 을 training 시킬 때 필요한 image 정보과 mask정보가 수록되어 있음.

1. mask_rcnn을 train 시키기 위해 coco_instances.json을 만드는 작업

cd ~/coco_instance_segmentation 으로 이동… python directory의 상위 directory에 위치할 것.

```
python ./python/coco_json_utils.py -md
./datasets/traffic_sign_dataset/train/mask_definitions.json \
   -di ./datasets/traffic_sign_dataset/train/dataset_info.json
```

2. train 시킨 mask_rcnn 모델의 정확도를 확인(validation)하는 데 사용할 coco_instances.json을 만드는 작업

```
python ./python/coco_json_utils.py -md
./datasets/traffic_sign_dataset/val/mask_definitions.json \
   -di ./datasets/traffic_sign_dataset/val/dataset_info.json
```

## 모델의 훈련

터미널에서 coco_instance_segmentation directory로 이동하여 **jupyter notebook**을 실행

1. **jupyter notebook**

2. broweser로 가서 notebooks 선택

3. train_mask_rcnn.ipynb 선택

4. kernel에서 Change kernel을 선택, 그리고 여러분의 conda virtual env (여기에서는 tfcoco-gpu)를 선택

5. jupyter notebook에 여러분의 data directory를 알려 주어야 함. 예를 들어 train directory를
dataset_train.load_data('../datasets/traffic_sign_dataset/train/coco_instances.json',
'**../datasets/traffic_sign_dataset/train/images**') 로 고쳐 주어야 함.
여러분이 jupyter notebook을 살펴보면서 다른 곳에 있는 directory도 변경하시기 바람.

6. 우리의 교통표지판의 소분류 숫자에 맞게 NUM_CLASSES를 변경하여야 함.
우리의 경우는 다음과 같이 변경
**NUM_CLASSES = 1 + 14**
총 소분류는 14 이고 여기에 1을 더하는 이유는 background를 추가하는 것임.

7. Cell 에서 Run All을 선택하면 train 이 시작됨.

8. train이 종료되면 videotest directory의 save directory로 가서 video를 틀어 봄. Instance segmentation이 작동되는 지를 확인 해 볼 것.

## 참고 사항

PASCAL VOC 형태의 파일로 object detection (localization)을 IoT device (Raspberry Pi 4 + Coral TPU Accelerator)에서 ssd_mobilenet_v1을 구현 방법을 YouTube에 올려 놓았음. 참조 바람.

https://www.youtube.com/watch?v=_02_0oS1NgM