Hidden Soldier Detection

양지현 김근호 김미라
오석준 권형근 우유정

Content

1. Data
2. Model
3. Experiment
4. Test

1. Data

※Object

1. Data – filtering(2000)

※ Desirable

- 목표 적합성
- ‘Person’ 특성 최소화
- 학습 용이성

※ Undesirable

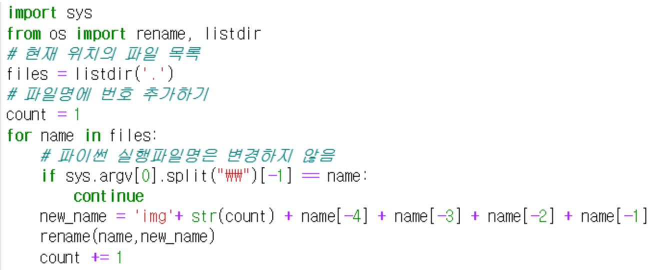
- 지형지물 이용 정도가 큼
- 육안 식별의 어려움
- 성능 저하

1. Data – Preprocessing 1.

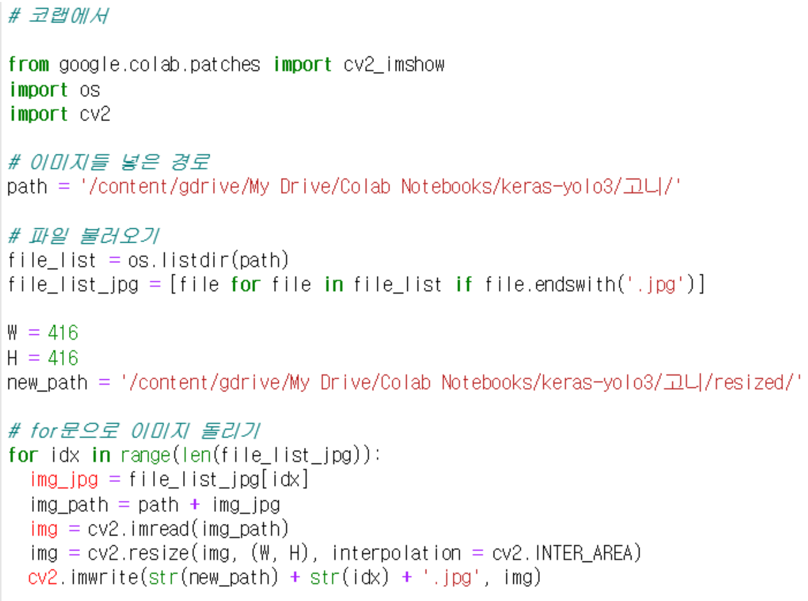
※ Resizing

- 416*416 for YOLO V3 input
- 정보손실 최소화

※ Renaming



```
import sys
from os import rename, listdir
# 현재 위치의 파일 목록
files = listdir('.')
# 파일명에 번호 추가하기
count = 1
for name in files:
    # 파이썬 실행파일명은 변경하지 않음
    if sys.argv[0].split("₩₩")[-1] == name:
        continue
    new_name = 'img'+ str(count) + name[-4] + name[-3] + name[-2] + name[-1]
    rename(name,new_name)
    count += 1
```



```
# 코랩에서

from google.colab.patches import cv2_imshow
import os
import cv2

# 이미지들 넣은 경로
path = '/content/gdrive/My Drive/Colab Notebooks/keras-yolo3/고니/'

# 파일 불러오기
file_list = os.listdir(path)
file_list_jpg = [file for file in file_list if file.endswith('.jpg')]

W = 416
H = 416
new_path = '/content/gdrive/My Drive/Colab Notebooks/keras-yolo3/고니/resized/'

# for문으로 이미지 돌리기
for idx in range(len(file_list_jpg)):
  img_jpg = file_list_jpg[idx]
  img_path = path + img_jpg
  img = cv2.imread(img_path)
  img = cv2.resize(img, (W, H), interpolation = cv2.INTER_AREA)
  cv2.imwrite(str(new_path) + str(idx) + '.jpg', img)
```

1. Data – Preprocessing 2.

Labelme

※ Direct to JSON

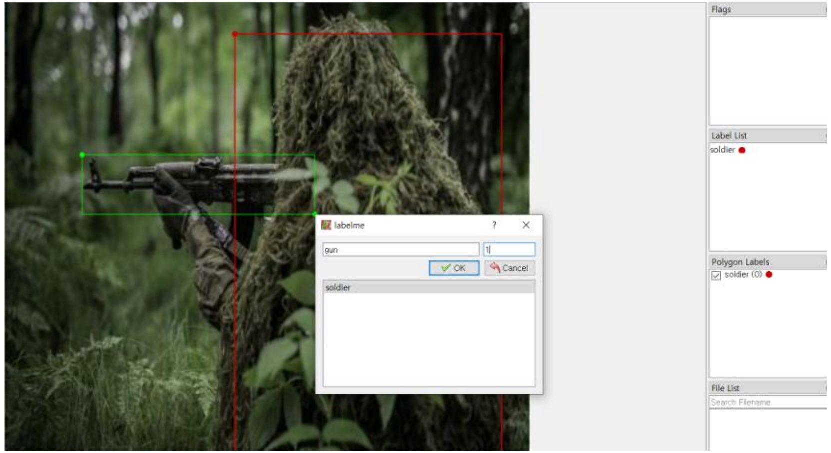
※ Segmentation

※ Polygons or Circle etc..

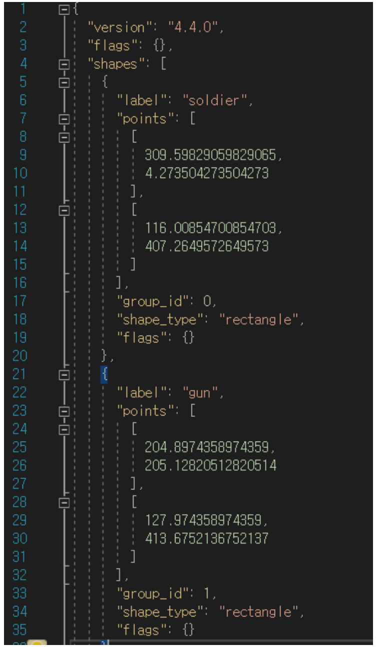
We defined

Soldier , group id =0
Gun,   group id=1

사용 예시



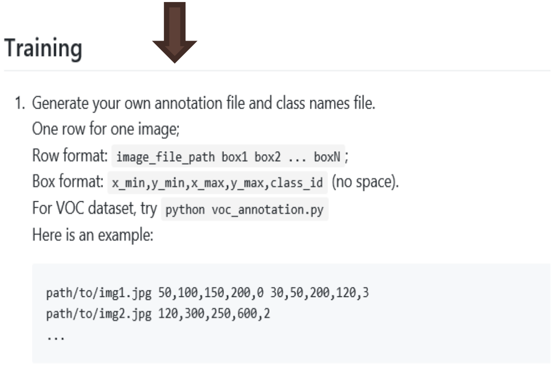
1. Data – Preprocessing 3.



```
{
    "version": "4.4.0",
    "flags": {},
    "shapes": [
      {
        "label": "soldier",
        "points": [
          [
            309.59829059829065,
            4.273504273504273
          ],
          [
            116.00854700854703,
            407.2649572649573
          ]
        ],
        "group_id": 0,
        "shape_type": "rectangle",
        "flags": {}
      },
      {
        "label": "gun",
        "points": [
          [
            204.8974358974359,
            205.12820512820514
          ],
          [
            127.974358974359,
            413.6752136752137
          ]
        ],
        "group_id": 1,
        "shape_type": "rectangle",
        "flags": {}
```

JSON data format

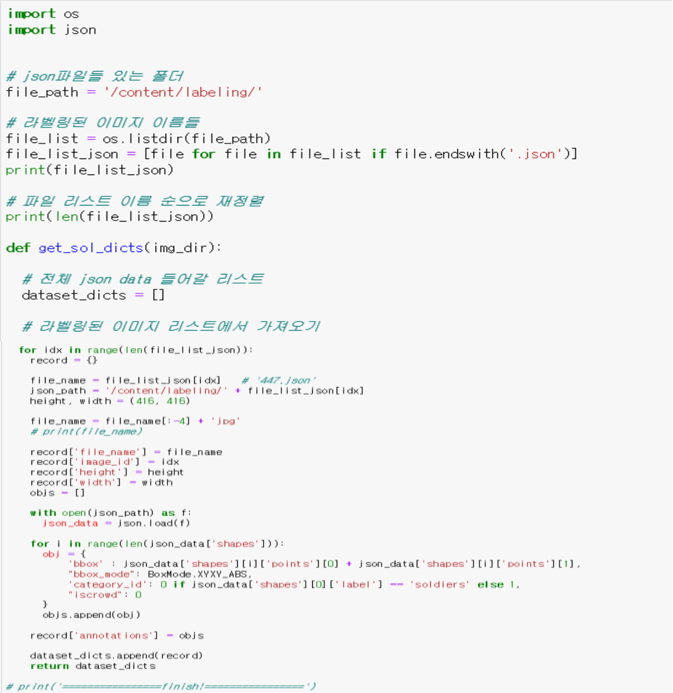
Input data format for YOLO V3



Training

1. Generate your own annotation file and class names file.
   One row for one image;
   Row format: `image_file_path box1 box2 ... boxN`;
   Box format: `x_min,y_min,x_max,y_max,class_id` (no space).
   For VOC dataset, try `python voc_annotation.py`
   Here is an example:

```
path/to/img1.jpg 50,100,150,200,0 30,50,200,120,3
path/to/img2.jpg 120,300,250,600,2
...
```



```python
import os
import json


# json파일들 있는 폴더
file_path = '/content/labeling/'

# 라벨링된 이미지 이름들
file_list = os.listdir(file_path)
file_list_json = [file for file in file_list if file.endswith('.json')]
print(file_list_json)

# 파일 리스트 이름 순으로 재정렬
print(len(file_list_json))

def get_sol_dicts(img_dir):

  # 전체 json data 들어갈 리스트
  dataset_dicts = []

  # 라벨링된 이미지 리스트에서 가져오기
  for idx in range(len(file_list_json)):
    record = {}

    file_name = file_list_json[idx]    # '447.json'
    json_path = '/content/labeling/' + file_list_json[idx]
    height, width = (416, 416)

    file_name = file_name[:-4] + 'jpg'
    # print(file_name)

    record['file_name'] = file_name
    record['image_id'] = idx
    record['height'] = height
    record['width'] = width
    objs = []

    with open(json_path) as f:
      json_data = json.load(f)

    for i in range(len(json_data['shapes'])):
      obj = {
          'bbox' : json_data['shapes'][i]['points'][0] + json_data['shapes'][i]['points'][1],
          "bbox_mode": BoxMode.XYXY_ABS,
          'category_id': 0 if json_data['shapes'][0]['label'] == 'soldiers' else 1,
          "iscrowd": 0
      }
      objs.append(obj)

    record['annotations'] = objs

    dataset_dicts.append(record)
    return dataset_dicts

# print('==============finish!==============')
```

2. Model

2.1 Mask R-CNN

※ Mask R-CNN

※ Advantage

- Object Detection & Image Segmentation
- High AP (Average Precision)

※ Disadvantage

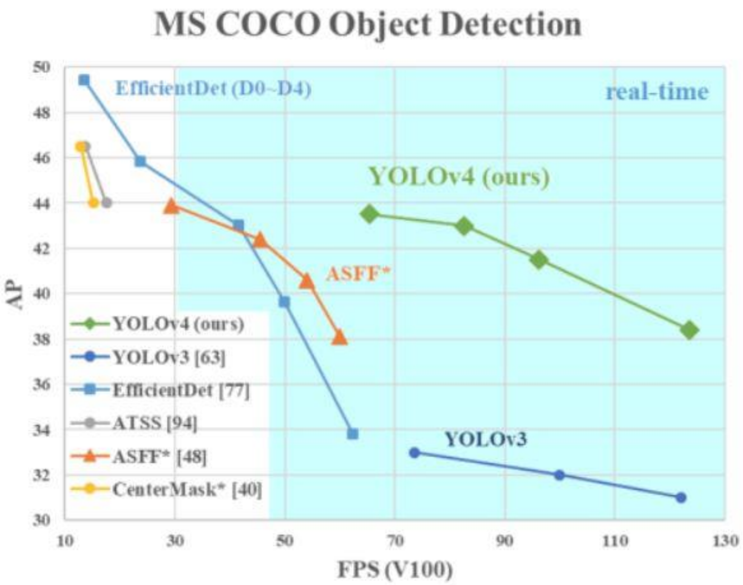
- Labelling
- Time

※ Conclusion

- 적절하지 않다

2.2 EfficientDet

※ EfficientDet



※ Advantage

- BiFPN 구조 고안
- High AP (Average Precision)

※ Disadvantage

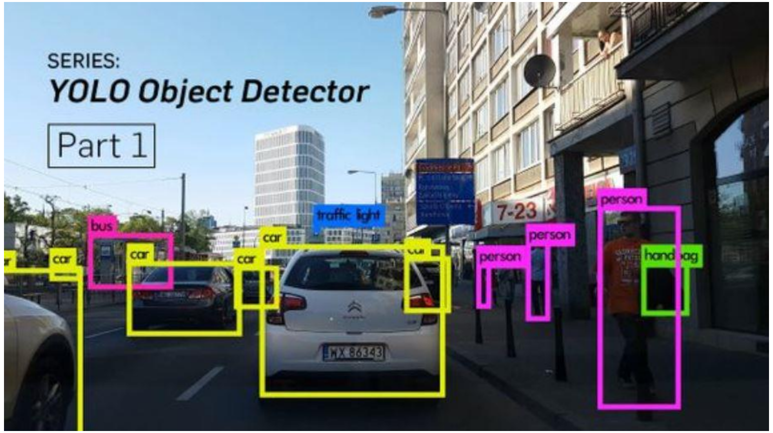
- Code를 찾기 어려움
- Time

※ Conclusion

- 적절하지 않다

2.3 YOLO v3

※ YOLO v3



※ Advantage

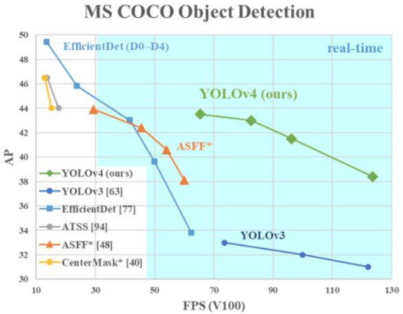
- 매우 빠른 속도 (Regression problem)
- Image의 전역 파악 (Contextual information)
- Generalizable representation 학습

※ Disadvantage

- 높은 localization error

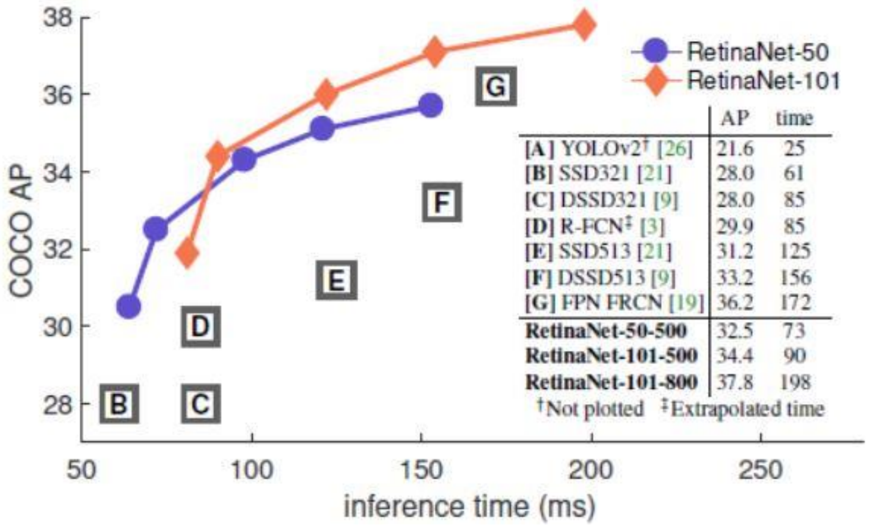
※ Conclusion

- 적절하다



2.4 RetinaNet

※ RetinaNet



	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

※ Advantage

- 빠른 속도 (One-stage)
- High AP
- Focal Loss

※ Disadvantage

- YOLO에 비해 느린 속도

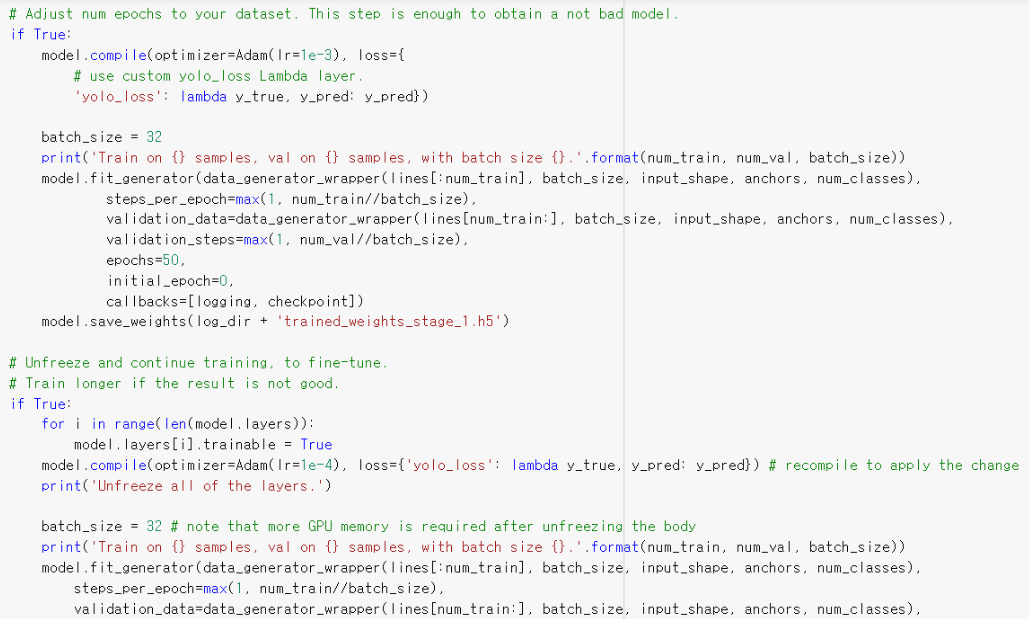
※ Conclusion

- 적절하다

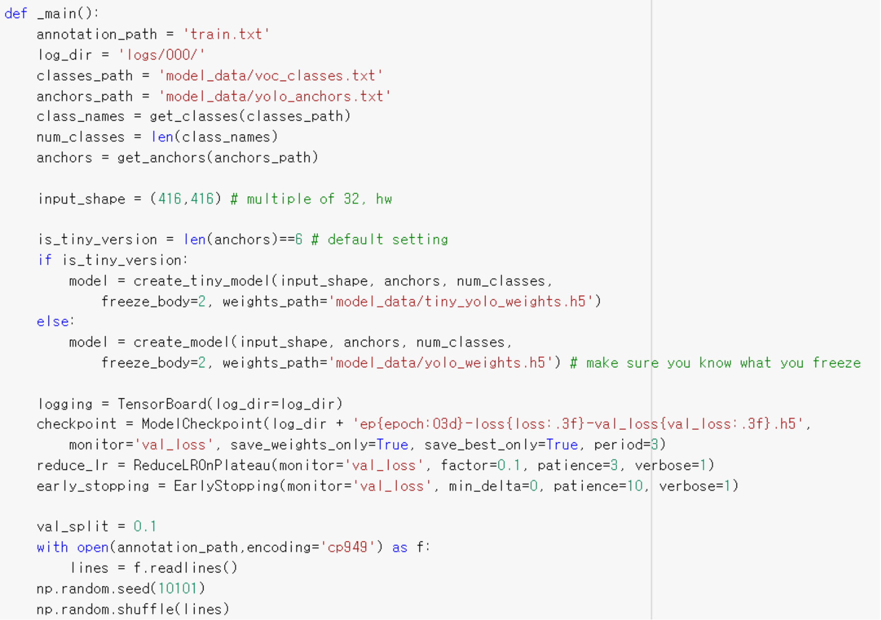
3.Experiment

3.1 YOLO v3

※ Training



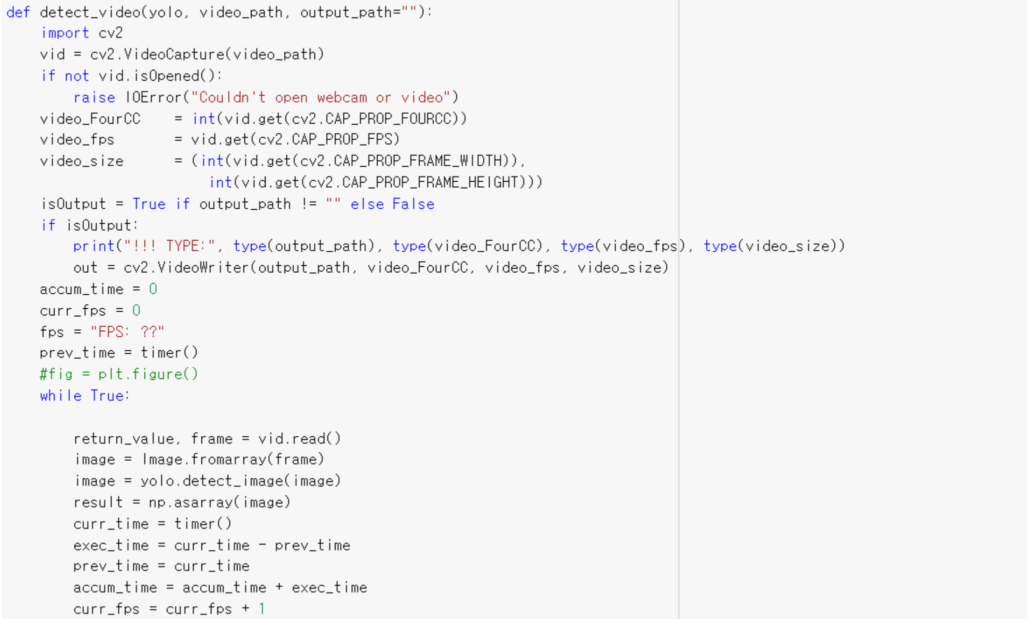
```
# Adjust num epochs to your dataset. This step is enough to obtain a not bad model.
if True:
    model.compile(optimizer=Adam(lr=1e-3), loss={
        # use custom yolo_loss Lambda layer.
        'yolo_loss': lambda y_true, y_pred: y_pred})

    batch_size = 32
    print('Train on {} samples, val on {} samples, with batch size {}.'.format(num_train, num_val, batch_size))
    model.fit_generator(data_generator_wrapper(lines[:num_train], batch_size, input_shape, anchors, num_classes),
            steps_per_epoch=max(1, num_train//batch_size),
            validation_data=data_generator_wrapper(lines[num_train:], batch_size, input_shape, anchors, num_classes),
            validation_steps=max(1, num_val//batch_size),
            epochs=50,
            initial_epoch=0,
            callbacks=[logging, checkpoint])
    model.save_weights(log_dir + 'trained_weights_stage_1.h5')

# Unfreeze and continue training, to fine-tune.
# Train longer if the result is not good.
if True:
    for i in range(len(model.layers)):
        model.layers[i].trainable = True
    model.compile(optimizer=Adam(lr=1e-4), loss={'yolo_loss': lambda y_true, y_pred: y_pred}) # recompile to apply the change
    print('Unfreeze all of the layers.')

    batch_size = 32 # note that more GPU memory is required after unfreezing the body
    print('Train on {} samples, val on {} samples, with batch size {}.'.format(num_train, num_val, batch_size))
    model.fit_generator(data_generator_wrapper(lines[:num_train], batch_size, input_shape, anchors, num_classes),
        steps_per_epoch=max(1, num_train//batch_size),
        validation_data=data_generator_wrapper(lines[num_train:], batch_size, input_shape, anchors, num_classes),
```



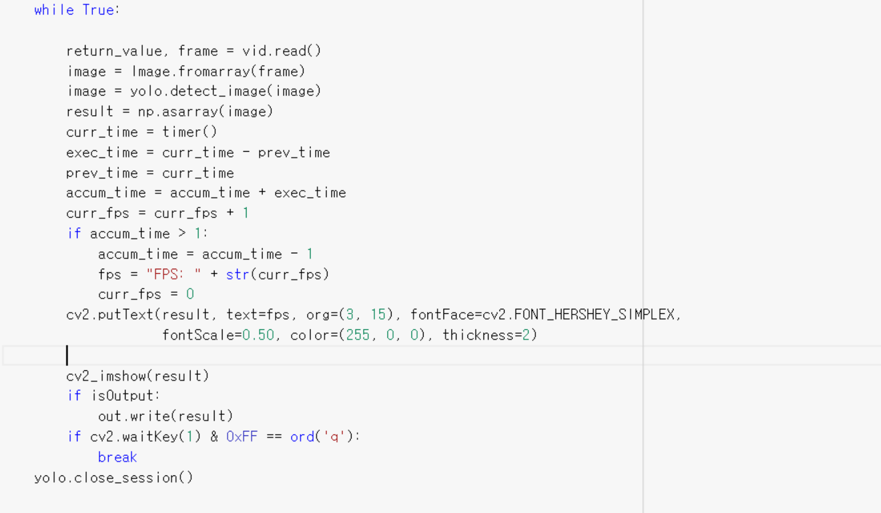
```
def _main():
    annotation_path = 'train.txt'
    log_dir = 'logs/000/'
    classes_path = 'model_data/voc_classes.txt'
    anchors_path = 'model_data/yolo_anchors.txt'
    class_names = get_classes(classes_path)
    num_classes = len(class_names)
    anchors = get_anchors(anchors_path)

    input_shape = (416,416) # multiple of 32, hw

    is_tiny_version = len(anchors)==6 # default setting
    if is_tiny_version:
        model = create_tiny_model(input_shape, anchors, num_classes,
            freeze_body=2, weights_path='model_data/tiny_yolo_weights.h5')
    else:
        model = create_model(input_shape, anchors, num_classes,
            freeze_body=2, weights_path='model_data/yolo_weights.h5') # make sure you know what you freeze

    logging = TensorBoard(log_dir=log_dir)
    checkpoint = ModelCheckpoint(log_dir + 'ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5',
        monitor='val_loss', save_weights_only=True, save_best_only=True, period=3)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, verbose=1)
    early_stopping = EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1)

    val_split = 0.1
    with open(annotation_path,encoding='cp949') as f:
        lines = f.readlines()
    np.random.seed(10101)
    np.random.shuffle(lines)
```

3.1 YOLO v3

※ Detecting image



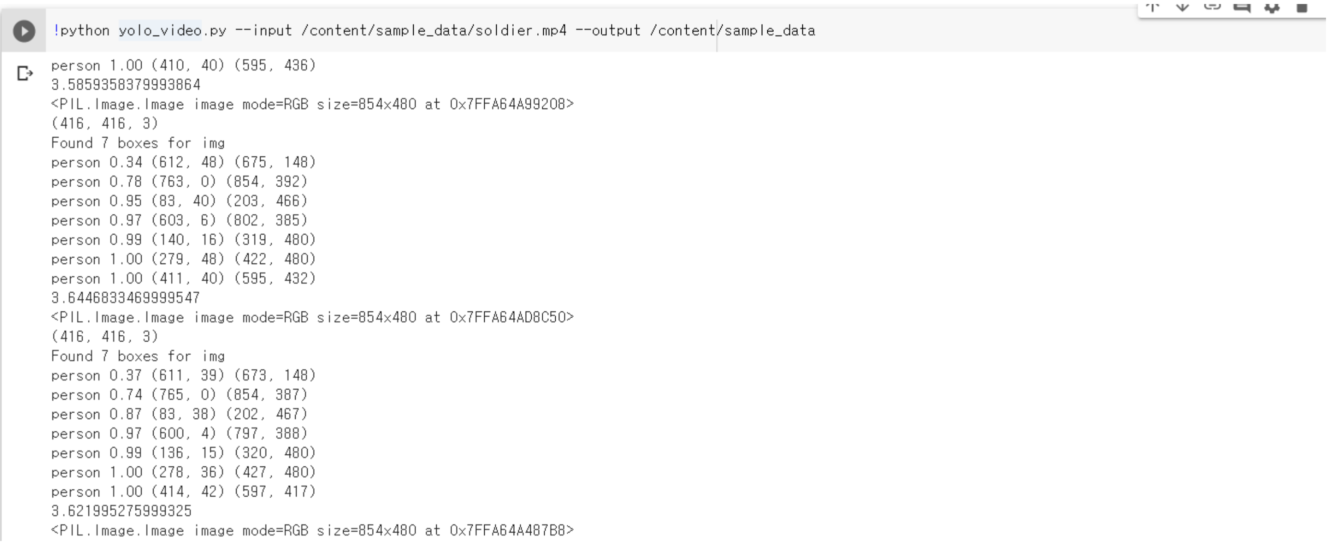
```
def detect_video(yolo, video_path, output_path=""):
    import cv2
    vid = cv2.VideoCapture(video_path)
    if not vid.isOpened():
        raise IOError("Couldn't open webcam or video")
    video_FourCC   = int(vid.get(cv2.CAP_PROP_FOURCC))
    video_fps      = vid.get(cv2.CAP_PROP_FPS)
    video_size     = (int(vid.get(cv2.CAP_PROP_FRAME_WIDTH)),
                       int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT)))
    isOutput = True if output_path != "" else False
    if isOutput:
        print("!!! TYPE:", type(output_path), type(video_FourCC), type(video_fps), type(video_size))
        out = cv2.VideoWriter(output_path, video_FourCC, video_fps, video_size)
    accum_time = 0
    curr_fps = 0
    fps = "FPS: ??"
    prev_time = timer()
    #fig = plt.figure()
    while True:

        return_value, frame = vid.read()
        image = Image.fromarray(frame)
        image = yolo.detect_image(image)
        result = np.asarray(image)
        curr_time = timer()
        exec_time = curr_time - prev_time
        prev_time = curr_time
        accum_time = accum_time + exec_time
        curr_fps = curr_fps + 1
```



```
    while True:

        return_value, frame = vid.read()
        image = Image.fromarray(frame)
        image = yolo.detect_image(image)
        result = np.asarray(image)
        curr_time = timer()
        exec_time = curr_time - prev_time
        prev_time = curr_time
        accum_time = accum_time + exec_time
        curr_fps = curr_fps + 1
        if accum_time > 1:
            accum_time = accum_time - 1
            fps = "FPS: " + str(curr_fps)
            curr_fps = 0
        cv2.putText(result, text=fps, org=(3, 15), fontFace=cv2.FONT_HERSHEY_SIMPLEX,
                    fontScale=0.50, color=(255, 0, 0), thickness=2)

        cv2_imshow(result)
        if isOutput:
            out.write(result)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    yolo.close_session()
```

3.1 YOLO v3

※ Detecting video

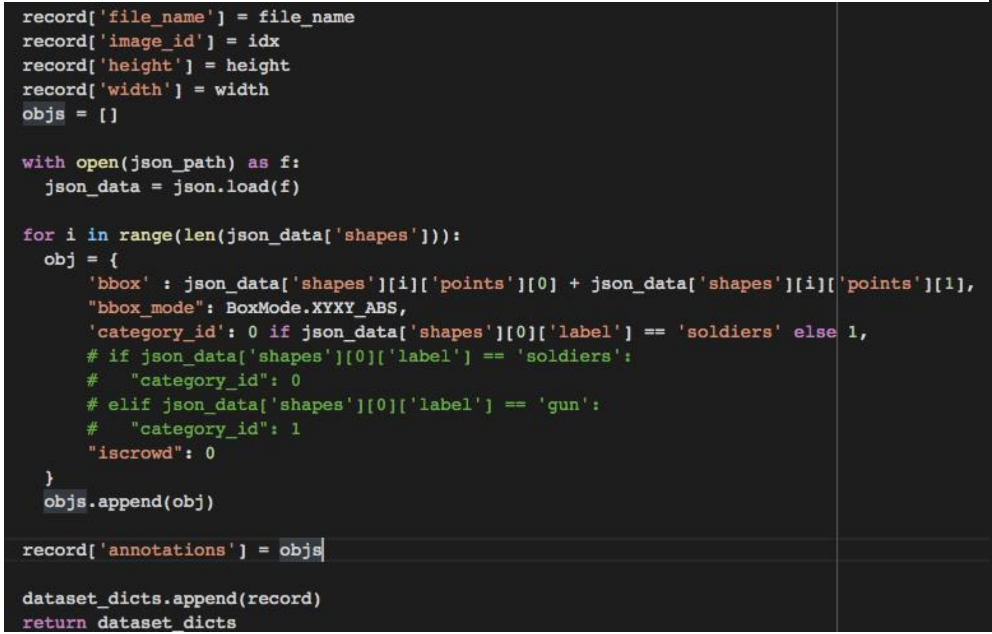


```
!python yolo_video.py --input /content/sample_data/soldier.mp4 --output /content/sample_data
```

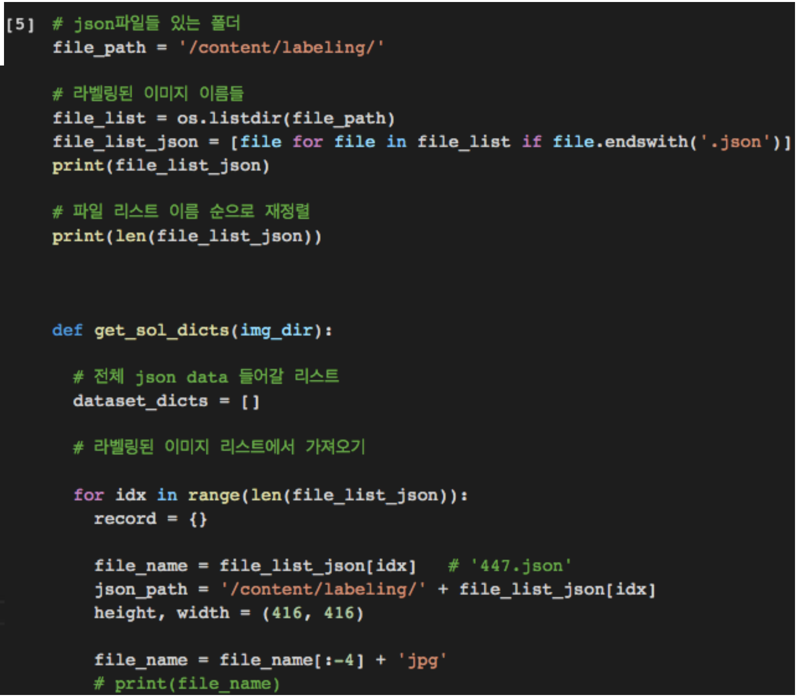
```
person 1.00 (410, 40) (595, 436)
3.5859358379993864
<PIL.Image.Image image mode=RGB size=854x480 at 0x7FFA64A99208>
(416, 416, 3)
Found 7 boxes for img
person 0.34 (612, 48) (675, 148)
person 0.78 (763, 0) (854, 392)
person 0.95 (83, 40) (203, 466)
person 0.97 (603, 6) (802, 385)
person 0.99 (140, 16) (319, 480)
person 1.00 (279, 48) (422, 480)
person 1.00 (411, 40) (595, 432)
3.6446833469999547
<PIL.Image.Image image mode=RGB size=854x480 at 0x7FFA64AD8C50>
(416, 416, 3)
Found 7 boxes for img
person 0.37 (611, 39) (673, 148)
person 0.74 (765, 0) (854, 387)
person 0.87 (83, 38) (202, 467)
person 0.97 (600, 4) (797, 388)
person 0.99 (136, 15) (320, 480)
person 1.00 (278, 36) (427, 480)
person 1.00 (414, 42) (597, 417)
3.621995275999325
<PIL.Image.Image image mode=RGB size=854x480 at 0x7FFA64A487B8>
```

3.2 RetinaNet

※ 요구 형식에 맞게 파일 구조 정리



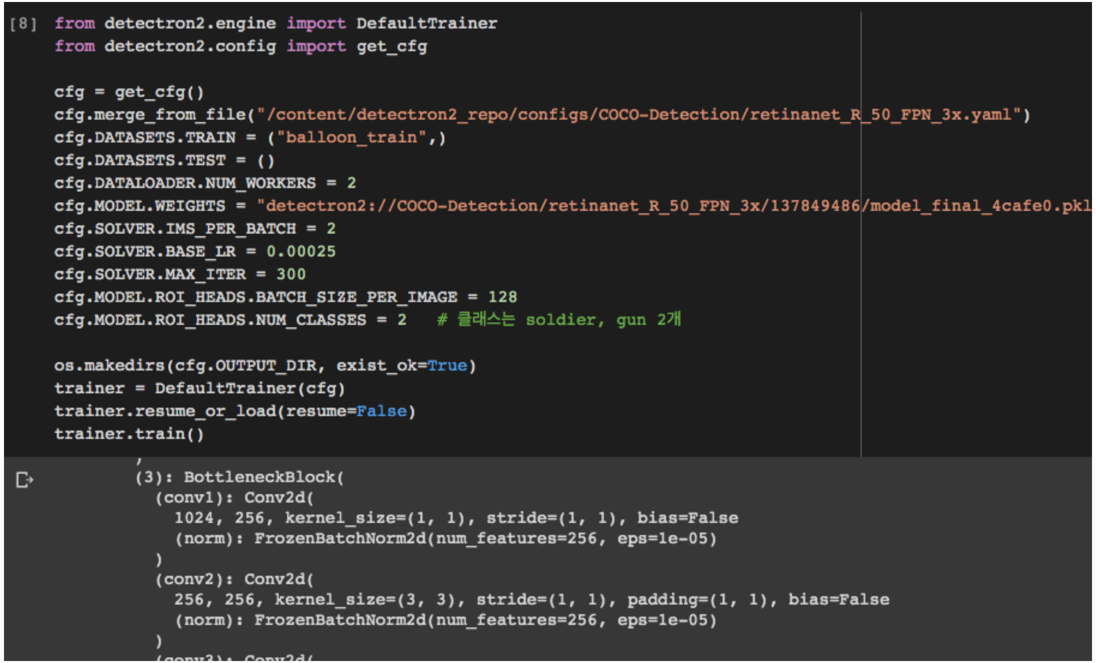
```
record['file_name'] = file_name
record['image_id'] = idx
record['height'] = height
record['width'] = width
objs = []

with open(json_path) as f:
 json_data = json.load(f)

for i in range(len(json_data['shapes'])):
 obj = {
   'bbox' : json_data['shapes'][i]['points'][0] + json_data['shapes'][i]['points'][1],
   "bbox_mode": BoxMode.XYXY_ABS,
   'category_id': 0 if json_data['shapes'][0]['label'] == 'soldiers' else 1,
   # if json_data['shapes'][0]['label'] == 'soldiers':
   #   "category_id": 0
   # elif json_data['shapes'][0]['label'] == 'gun':
   #   "category_id": 1
   "iscrowd": 0
 }
 objs.append(obj)

record['annotations'] = objs

dataset_dicts.append(record)
return dataset_dicts
```



```
[5] # json파일들 있는 폴더
    file_path = '/content/labeling/'

    # 라벨링된 이미지 이름들
    file_list = os.listdir(file_path)
    file_list_json = [file for file in file_list if file.endswith('.json')]
    print(file_list_json)

    # 파일 리스트 이름 순으로 재정렬
    print(len(file_list_json))



    def get_sol_dicts(img_dir):

     # 전체 json data 들어갈 리스트
     dataset_dicts = []

     # 라벨링된 이미지 리스트에서 가져오기

     for idx in range(len(file_list_json)):
      record = {}

      file_name = file_list_json[idx]   # '447.json'
      json_path = '/content/labeling/' + file_list_json[idx]
      height, width = (416, 416)

      file_name = file_name[:-4] + 'jpg'
      # print(file_name)
```

3.2 RetinaNet

※ Training

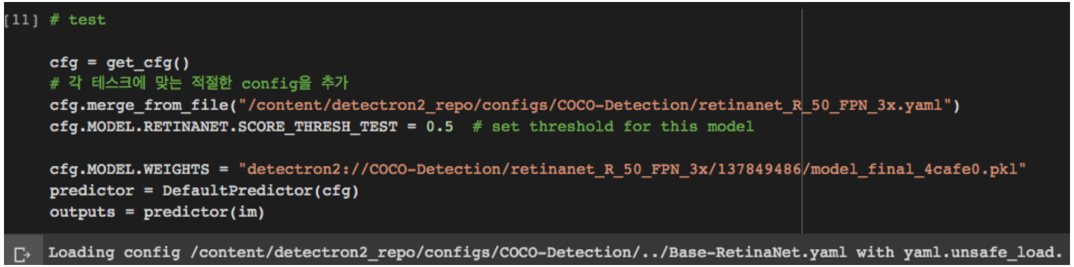


```
[8] from detectron2.engine import DefaultTrainer
    from detectron2.config import get_cfg

    cfg = get_cfg()
    cfg.merge_from_file("/content/detectron2_repo/configs/COCO-Detection/retinanet_R_50_FPN_3x.yaml")
    cfg.DATASETS.TRAIN = ("balloon_train",)
    cfg.DATASETS.TEST = ()
    cfg.DATALOADER.NUM_WORKERS = 2
    cfg.MODEL.WEIGHTS = "detectron2://COCO-Detection/retinanet_R_50_FPN_3x/137849486/model_final_4cafe0.pkl
    cfg.SOLVER.IMS_PER_BATCH = 2
    cfg.SOLVER.BASE_LR = 0.00025
    cfg.SOLVER.MAX_ITER = 300
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2   # 클래스는 soldier, gun 2개

    os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
    trainer = DefaultTrainer(cfg)
    trainer.resume_or_load(resume=False)
    trainer.train()
```

```
        (3): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
```

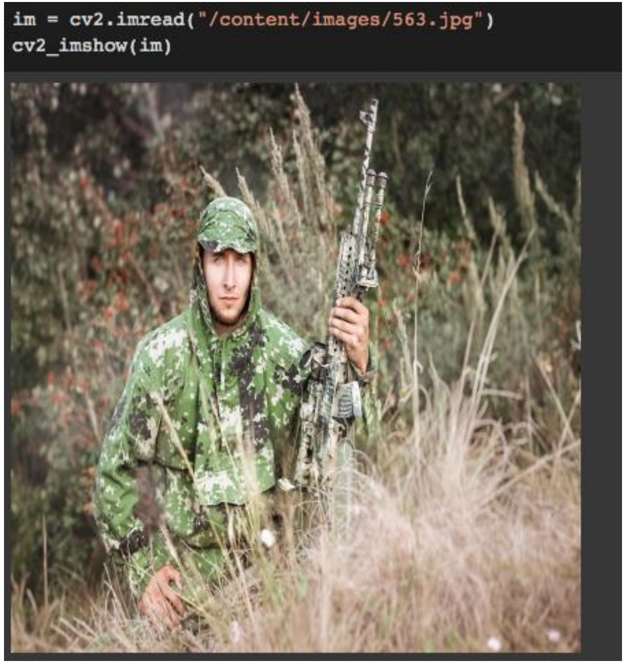
3.2 RetinaNet

※ Test 과정



```
[11] # test

    cfg = get_cfg()
    # 각 테스크에 맞는 적절한 config을 추가
    cfg.merge_from_file("/content/detectron2_repo/configs/COCO-Detection/retinanet_R_50_FPN_3x.yaml")
    cfg.MODEL.RETINANET.SCORE_THRESH_TEST = 0.5  # set threshold for this model

    cfg.MODEL.WEIGHTS = "detectron2://COCO-Detection/retinanet_R_50_FPN_3x/137849486/model_final_4cafe0.pkl"
    predictor = DefaultPredictor(cfg)
    outputs = predictor(im)
```

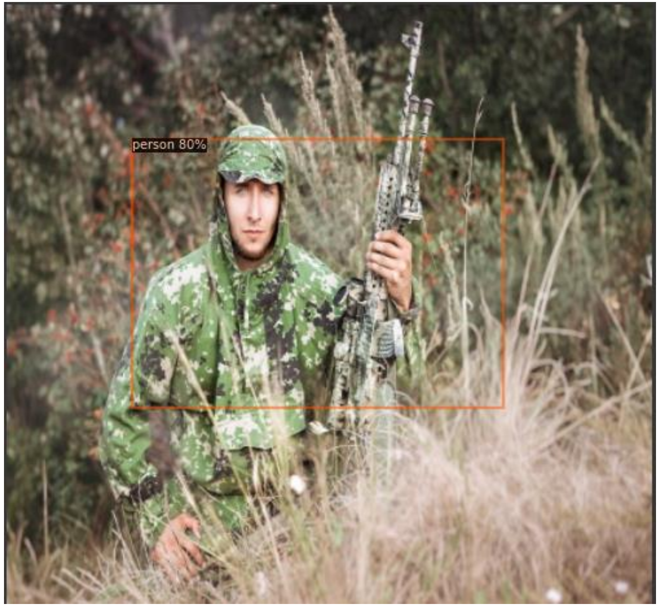
```
Loading config /content/detectron2_repo/configs/COCO-Detection/../Base-RetinaNet.yaml with yaml.unsafe_load.
```

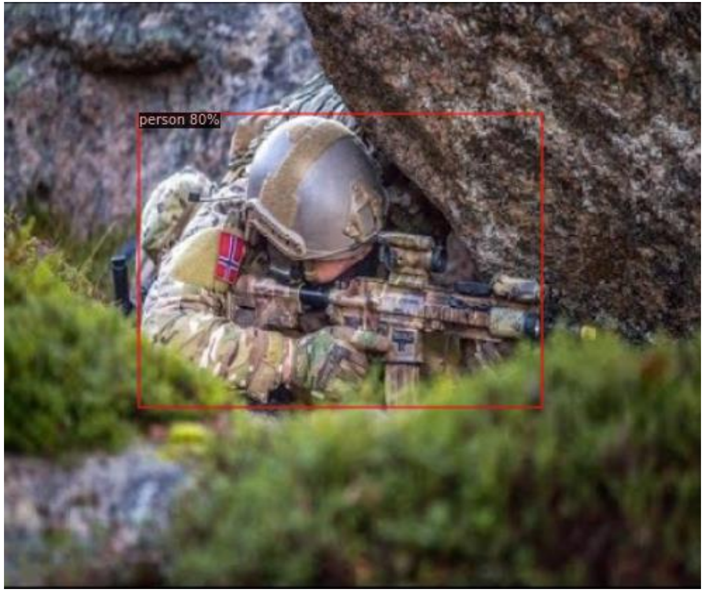


```
im = cv2.imread("/content/images/563.jpg")
cv2_imshow(im)
```

3.2 RetinaNet

※ Test 결과





3.2 RetinaNet

※ 동영상 Test 결과

3.Result

의의

- 고된 수작업 라벨링 과정에서 발생한 동기애
- 다양한 모델에 대해 연구
- Transfer learning 에 대한 이해
- Gpu의 필요성… 코랩의 한계점 파악….

한계점

- Person과 soldier를 구분하지 못함 → COCO dataset으로 pre trained 된 모델을 transfer learning 하다 보니, COCO에서 person으로 많은 학습이 이루어짐
- 종속관계를 파악하지 못함 (person >> soldier)
- Person으로 학습이 이루어지지 않은 weight를 불러오면 해결 될 것이라 생각함

감사합니다!