

# 오픈소트 SW 실습 final project

## mmdetection 기반 K-fashion 데이터 학습

5 조

피선우, 한상범, 홍찬의

### 프로젝트 주제

#### K-fashion 데이터를 활용한 Mmdetection MASK-RCNN 학습 및 실행 환경 배포

- K-fashion 데이터에 맞춰 mmlab mask-rcnn 학습
- 깃허브 및 레드마인 활용하여 협업
- 모델 실행 환경 도커 이미지로 배포

#### 1. mmdetection 학습을 위한 환경 구축

- conda 가상환경 생성

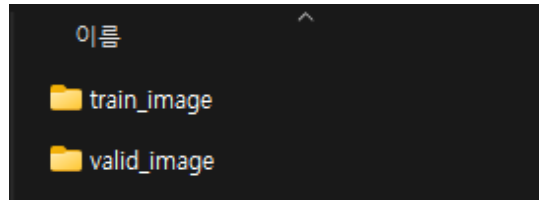
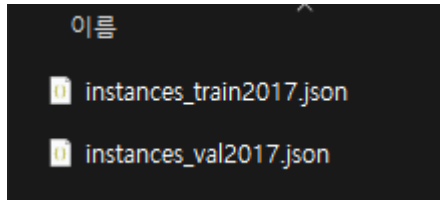
```
conda create -n openSW python=3.7
```

- mmdetection 환경에 맞는 라이브러리 설치

- python 3.7.13
- pytorch 1.6.0
- torchvision 0.7.0
- cuda 10.0 - rtx 2080 super
- mmcv(mmcv-full) 1.7.0
- mmdet 2.26.0

## 2. 데이터 형식 및 전처리

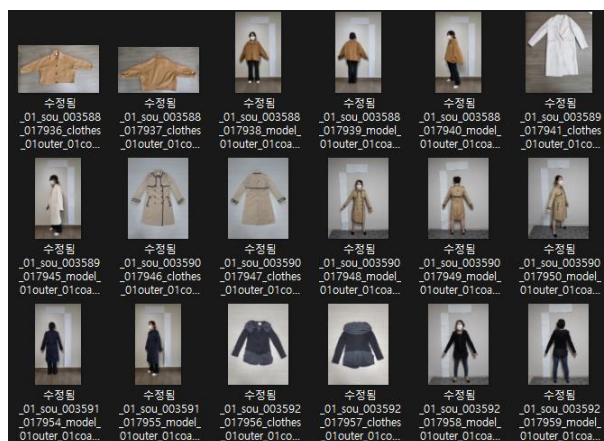
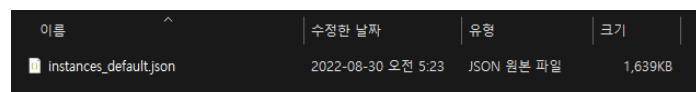
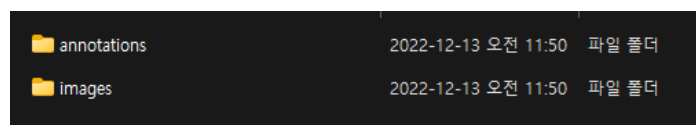
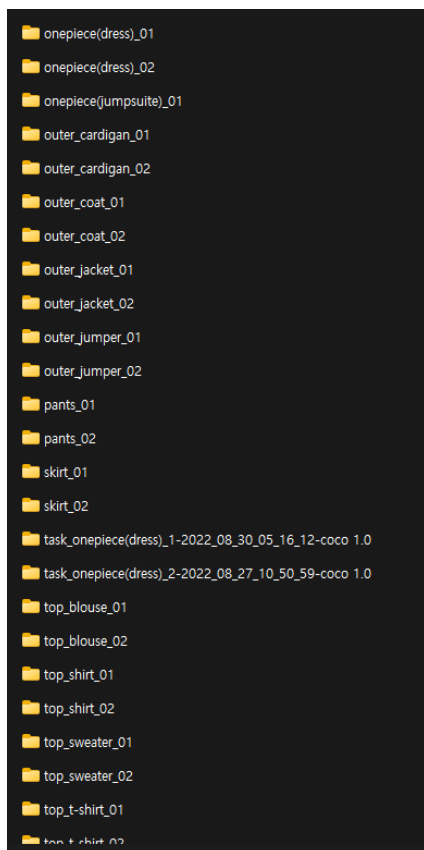
- 기존 데이터 클래스 분류가 coco데이터 형식에 맞지 않게 되어있었음 (coco 데이터는 train.json파일과 val.json파일 그리고 이미지가 train valid 두가지로 나누어져 있음)



위 사진과 같은 coco data 형식이 존재하지만

우리가 가지고 있는 fashion data는 다음과 같다.

클래스 별로 두가지 이상 파일이 존재하고 각 파일은 json과 이미지가 각자 저장되어 존재함



따라서 클래스 별로 분류된 json 파일을 coco data 형식에 맞게 알고리즘을 활용하여 id를 순서대로 만들어서 모두 합쳐주고자 함.

```

file_path_1 = 'C:/Users/PiSunWoo-RTOS/Desktop/Mask_RCNN_실습데이터/onepiece(dress)_02/annotations/instances_default.json'
file_path_2 = 'C:/Users/PiSunWoo-RTOS/Desktop/Mask_RCNN_실습데이터/onepiece(dress)_03/annotations/instances_default.json'

with open(file_path_1, 'r') as f:
    json_data_1 = json.load(f)

with open(file_path_2, 'r') as f:
    json_data_2 = json.load(f)

print(len(json_data_1['images']))
print(len(json_data_2['images']))
print(len(json_data_1['annotations']))
print(len(json_data_2['annotations']))

json_1_id_len = len(json_data_1['images'])
json_2_id_len = len(json_data_2['images'])
for image_id in range(json_2_id_len):
    json_data_2['images'][image_id]['id'] += json_1_id_len

json_1_id_len = len(json_data_1['annotations'])
json_2_id_len = len(json_data_2['annotations'])
for image_id in range(json_2_id_len):
    json_data_2['annotations'][image_id]['id'] += json_1_id_len
    json_data_2['annotations'][image_id]['image_id'] += json_1_id_len

with open(file_path_2, 'w', encoding='utf-8') as file:
    json.dump(json_data_2, file, indent="\t")

```

change\_json\_numbering.py를 이용하여 같은 class(ex. 400장)의 json 파일을 += len(json\_data\_1[])를 이용하여 (id : 0~199, id : 0~199) >>> (id : 0~199, id 200~399)으로 만들어준 후

```

import json

file_path_1 = 'C:/Users/PiSunWoo-RTOS/Desktop/Mask_RCNN_실습데이터/11. top_t-shirt_01/annotations/instances_default.json'
file_path_2 = 'C:/Users/PiSunWoo-RTOS/Desktop/Mask_RCNN_실습데이터/11. top_t-shirt_02/annotations/instances_default.json'

with open(file_path_1, 'r') as f:
    json_data_1 = json.load(f)

with open(file_path_2, 'r') as f:
    json_data_2 = json.load(f)

json_1_images_id_len = len(json_data_1['images'])
json_2_images_id_len = len(json_data_2['images'])
json_1_annotations_id_len = len(json_data_1['annotations'])
json_2_annotations_id_len = len(json_data_2['annotations'])

for i in range(json_2_images_id_len):
    json_data_1['images'].append(json_data_2['images'][i])

for i in range(json_2_annotations_id_len):
    json_data_1['annotations'].append(json_data_2['annotations'][i])

with open(file_path_1, 'w', encoding='utf-8') as file:
    json.dump(json_data_1, file, indent="\t")

```

combine\_json.py를 이용하여 (id : 0~199, id : 200~399)로 구성된 두 개의 json 파일을 append()를 이용하여 (id : 0~399)로 구성된 한 개의 json 파일로 만들어준다.

그 후 train\_valid\_json.py를 이용하여

```
categories = [
    {
        "id": 0,
        "name": "onepiece(dress)",
        "supercategory": "onepiece(dress)"
    },
    {
        "id": 1,
        "name": "jumpsuite",
        "supercategory": "jumpsuite"
    },
    {
        "id": 2,
        "name": "cardigan",
        "supercategory": "cardigan"
    },
    {
        "id": 3,
        "name": "coat",
        "supercategory": "coat"
    },
    {
        "id": 4,
        "name": "jacket",
        "supercategory": "jacket"
    },
    {
        "id": 5,
        "name": "jumper",
        "supercategory": "jumper"
    },
    {
        "id": 6,
        "name": "pants",
        "supercategory": "pants"
    },
    {
        "id": 7,
        "name": "skirt",
        "supercategory": "skirt"
    },
    {
        "id": 8,
        "name": "blouse",
        "supercategory": "blouse"
    },
    {
        "id": 9,
        "name": "shirt",
        "supercategory": "shirt"
    },
    {
        "id": 10,
        "name": "sweater",
        "supercategory": "sweater"
    },
    {
        "id": 11,
        "name": "t-shirt",
        "supercategory": "t-shirt"
    }
]
```

category id와 name을 설정해준 후

```
divide = [860, 44, 389, 480, 480, 385, 270, 329, 460, 293, 297, 453] # Train : Valid = 8 : 2
```

Train : Valid 이미지 수 비율을 설정 그리고

```

for i in class_json:
    print(i)

    with open(rootDir + i, 'r') as f:
        json_data = json.load(f)

        for combined_json_categories in json_data["categories"]:
            combined_json_categories["id"] = categoryID

        for images in json_data["images"]:
            if(id < divide[categoryID]):
                images["file_name"] = os.path.basename(images["file_name"])
                images["id"] = train_images_id
                train_images.append(images)
                train_images_id += 1

            else:
                images["file_name"] = os.path.basename(images["file_name"])
                images["id"] = val_images_id
                val_images.append(images)
                val_images_id += 1

            id += 1

        id = 0

        for anno in json_data["annotations"]:
            anno["category_id"] = categoryID

            if(id < divide[categoryID]):
                anno["id"] = train_anno_id
                train_anno.append(anno)

                if anno["image_id"] != train_pre_img :
                    train_img_id += 1
                    train_pre_img = anno["image_id"]
                    anno["image_id"] = train_img_id

                train_anno_id += 1

            else:
                anno["id"] = val_anno_id
                val_anno.append(anno)

                if anno["image_id"] != val_pre_img :
                    val_img_id += 1
                    val_pre_img = anno["image_id"]
                    anno["image_id"] = val_img_id

                val_anno_id += 1

            id += 1

        id = 0

        categoryID += 1

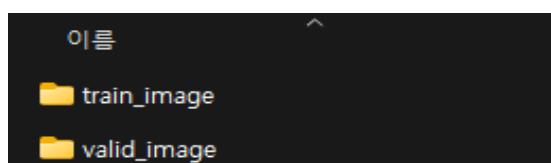
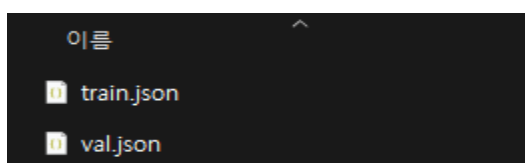
train_object["licenses"] = licenses
train_object["info"] = info
train_object["categories"] = categories
train_object["images"] = train_images
train_object["annotations"] = train_anno

val_object["licenses"] = licenses
val_object["info"] = info
val_object["categories"] = categories
val_object["images"] = val_images
val_object["annotations"] = val_anno

```

append()를 이용하여

해당 비율만큼의 Train 4740장 Valid 1186장의 Image 그리고 train.json, val.json을 생성하여  
다음과 같이 K-fashion data를 coco 형식으로 맞추어 줌



### 3. mmdetection 코드 수정

- 이제 데이터는 다 만들어 주었으니 mask\_rcnn의 코드 일부분을 수정해 주어야 한다.  
( 기존 코드는 coco data형식으로 되어 있음 )
- 우리는 mask\_rcnn\_r50\_fpn\_1x\_coco.py와 mask\_rcnn\_r101\_fpn\_2x\_coco.py 두가지 모델을 학습시킬 예정이다.
- 먼저 mmdetection/configs/mask\_rcnn/ 경로로 이동하여 파일을 확인해보았다.

mask_rcnn_r101_fpn_1x_coco.py	[Refactor] move model.pretrained to model.backbone.init_cfg (#5370)
mask_rcnn_r101_fpn_2x_coco.py	[Refactor] move model.pretrained to model.backbone.init_cfg (#5370)
mask_rcnn_r101_fpn_mstrain-poly_3x_coco.py	[Fix] Fix some backbone init from pretrained checkpoint error in conf...
mask_rcnn_r50_caffe_c4_1x_coco.py	Refactor (training cfgs): change default training settings for better... (
mask_rcnn_r50_caffe_fpn_1x_coco.py	[Refactor] move model.pretrained to model.backbone.init_cfg (#5370)
mask_rcnn_r50_caffe_fpn_mstrain-poly_1x_coco.py	[Refactor] move model.pretrained to model.backbone.init_cfg (#5370)
mask_rcnn_r50_caffe_fpn_mstrain-poly_2x_coco.py	[Fix]: Add types of runner in configs (#4669)
mask_rcnn_r50_caffe_fpn_mstrain-poly_3x_coco.py	[Fix]: Add types of runner in configs (#4669)
mask_rcnn_r50_caffe_fpn_mstrain_1x_coco.py	[Refactor] move model.pretrained to model.backbone.init_cfg (#5370)
mask_rcnn_r50_caffe_fpn_poly_1x_coco_v1.py	[Fix] Fix some backbone init from pretrained checkpoint error in conf...
mask_rcnn_r50_fpn_1x_coco.py	delete work_dir in all configs (#2315)
mask_rcnn_r50_fpn_1x_wandb_coco.py	[Feature] Dedicated WandbLogger for MMDetection (#7459)
mask_rcnn_r50_fpn_2x_coco.py	Add configs and benchmarks. (#2446)

```
1 _base_ = [  
2     '../_base_/models/mask_rcnn_r50_fpn.py',  
3     '../_base_/datasets/coco_instance.py',  
4     '../_base_/schedules/schedule_1x.py', '../_base_/default_runtime.py'  
5 ]  
6
```

위 와 같이 파일이 존재하였고 경로에 따라가 파일 일부분을 수정하였다.

mmdetection/configs/\_base\_/models/ 경로에 찾아가

mask\_rcnn\_r50\_fpn.py 파일을 수정하였다. ( model 사용 )



mask\_rcnn\_r50\_fpn.py에서 원래 coco 데이터 80개에서 12개로 클래스 개수 맞추어줌

```

5         fc_out_channels=1024,
6         roi_feat_size=7,
7         num_classes=12, # 클래스 개수
8         bbox_coder=dict(
9             type='DeltaXYWHBBoxCoder',
10            target_means=[0., 0., 0., 0.],
11            target_std=[0.1, 0.1, 0.2, 0.2]),
12            reg_class_agnostic=False,
13            loss_cls=dict(
14                type='CrossEntropyLoss', use_sigmoid=False, loss_weight=1.0),
15            loss_bbox=dict(type='L1Loss', loss_weight=1.0)),
16            mask_roi_extractor=dict(
17                type='SingleRoIExtractor',
18                roi_layer=dict(type='RoIAlign', output_size=14, sampling_ratio=0),
19                out_channels=256,
20                featmap_strides=[4, 8, 16, 32]),
21            mask_head=dict(
22                type='FCNMaskHead',
23                num_convs=4,
24                in_channels=256,
25                conv_out_channels=256,
26                num_classes=12, # 클래스 개수
27                loss_mask=dict(
28                    type='CrossEntropyLoss', use_mask=True, loss_weight=1.0))),
29            # model training and testing settings

```

다음으로는 라벨 값을 기존에 있던 coco data에서 K-fashion data로 바꾸어 주어야 한다.

여기서 중요한 점은 mmdetection은 가상환경을 실행시키고 난 후 mmdet 2.26.0 버전을 설치하게 되면 가상환경에서 mmdetection 코드가 실행된다. 따라서 coco.py는 가상환경 경로

에 찾아가 datasets 폴더 안에 있는 coco.py를 찾아야 한다.

Chanui > anaconda3 > envs > openSW > Lib > site-packages > mmdet > datasets

아래 그림이 기존의 coco.py 파일의 기존에 있던 coco data 클래스

```
... (0, 0, 192), (250, 170, 30), (100, 170, 30), (220, 220, 0)]
...
CLASSES = ('person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
            'train', 'truck', 'boat', 'traffic light', 'fire hydrant',
            'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog',
            'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe',
            'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
            'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',
            'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
            'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',
            'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot',
            'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
            'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop',
            'mouse', 'remote', 'keyboard', 'cell phone', 'microwave',
            'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock',
            'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush')

PALETTE = [(220, 20, 60), (119, 11, 32), (0, 0, 142), (0, 0, 230),
            (106, 0, 228), (0, 60, 100), (0, 80, 100), (0, 0, 70),
            (0, 0, 192), (250, 170, 30), (100, 170, 30), (220, 220, 0),
            (175, 116, 175), (250, 0, 30), (165, 42, 42), (255, 77, 255),
            (0, 226, 252), (182, 182, 255), (0, 82, 0), (120, 166, 157),
            (110, 76, 0), (174, 57, 255), (199, 100, 0), (72, 0, 118),
            (255, 179, 240), (0, 125, 92), (209, 0, 151), (188, 208, 182),
            (0, 220, 176), (255, 90, 164), (0, 0, 73), (123, 120, 255)]
```

coco의 라벨값을 우리 데이터인 fashion에 맞추어 라벨 맞추어 주기

```
CLASSES = ('coat', 'jacket', 'jumper', 'cardigan', 'blouse', 't-shirt',
            'sweater', 'shirt', 'onepiece(dress)', 'jumpsuite', 'pants',
            'skirt',)
PALETTE = [(220, 20, 60), (119, 11, 32), (0, 0, 142), (0, 0, 230),
            (106, 0, 228), (0, 60, 100), (0, 80, 100), (0, 0, 70),
            (0, 0, 192), (250, 170, 30), (100, 170, 30), (220, 220, 0)]
```



- 그리고 제일 중요한 데이터의 경로를 맞추어 주어야 한다.

mmdetection/configs/\_base\_/datasets 경로에 coco\_instance.py를 찾아 K-fashion 데이터와 json파일 경로를 맞추어 준다.

```
# dataset settings
dataset_type = 'CocoDataset'
data_root = '../datafinal/' #경로 맞춰줌
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True, with_mask=True),
    dict(type='Resize', img_scale=(1333, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels', 'gt_masks']),
]
```

이미지 데이터와 json파일 경로 맞추어 주기

```
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type=dataset_type,
        ann_file=data_root + 'annotations/train.json', # train json 파일 경로 맞추기
        img_prefix=data_root + 'images/train_image/', # train 이미지 경로 맞추기
        pipeline=train_pipeline),
    val=dict(
        type=dataset_type,
        ann_file=data_root + 'annotations/val.json', # val json 파일 경로 맞추기
        img_prefix=data_root + 'images/val_image/', # val 이미지 경로 맞추기
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + 'annotations/val.json', # val json 파일 경로 맞추기
        img_prefix=data_root + 'images/val_image/', # test 이미지 경로 맞추기
        pipeline=test_pipeline))
evaluation = dict(metric=['bbox', 'segm'])
```

- 추가적으로 학습 완료시 텔레그램으로 알림을 받기 위해 텔레그램 알람 코드를 넣었다.

```
# - 텔레그램 알람 만들기
import telegram
import schedule
import time
# import datetime
# import pytz

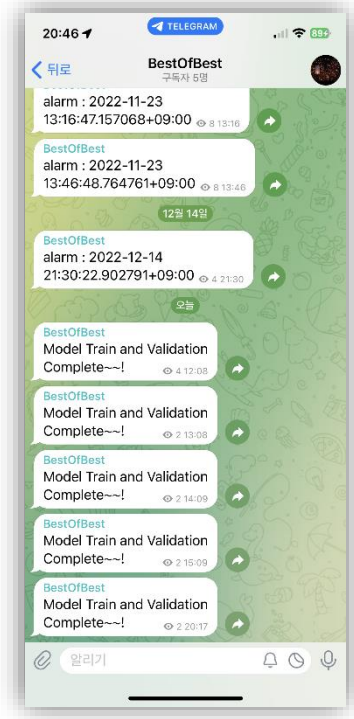
#토큰
token = "5759317578:AAHB10nZxKvOzsBjBw8aBlWamsMWC7A1Fnk"
bot = telegram.Bot(token)
public_chat_name = "@ktest2022"

# 학습 완료되면 알람이 오도록 설정
def job():
    now = ("Model Train and Validation Complete~~!")

    text=(str(now))
    bot.sendMessage(chat_id = public_chat_name, text = text).chat_id
    #print("current time = ", str(now))
```

```
if __name__ == '__main__':
    main()
    # 학습 완료시 1시간 주기로 알려줌
    schedule.every(60).minutes.do(job)
    while True:
        schedule.run_pending()
        time.sleep(60)
```

학습 완료시 1시간 마다 알람이 오며  
학습이 완료되었음을 알려줍니다.



실제 학습 완료 후 모델 학습 완료를 알려주는 코드  
(한시간마다 계속 반복됨)

## 4. 학습 실행

(1) 학습 실행 코드 실행

```
(openSW) C:\Users\Chanui\Desktop\OpenSW_Team5>python ./tools/train.py ./configs/mask_rcnn/mask_rcnn_r50_fpn_1x_coco.py
```

(2) 데이터가 잘 들어갔는지 확인

```
CocoDataset Train dataset with number of images 4723, and instance counts:
```

category	count	category	count	category	count	category	count	category	count
0 [onepiece(dress)]	860	1 [jumpsuite]	44	2 [cardigan]	389	3 [coat]	480	4 [jacket]	480
5 [jumper]	385	6 [pants]	270	7 [skirt]	329	8 [blouse]	460	9 [shirt]	293
10 [sweater]	297	11 [t-shirt]	453						

### (3) 학습 시작

```
acc: 96.6367, loss_bbox: 0.1133, loss_mask: 0.5339, loss: 0.8556
2022-12-14 11:17:11.135 - mmdet - INFO - Epoch [1][1158/2357] lr: 2.000e-02, eta: 0:58:25, time: 0.123, data_time: 0.001, memory: 864, loss_rpn_cls: 0.0351, loss_rpn_bbox: 0.0069, loss_cls: 0.1682,
acc: 96.6599, loss_bbox: 0.1133, loss_mask: 0.5372, loss: 0.8796
2022-12-14 11:17:17.187 - mmdet - INFO - Epoch [1][1200/2357] lr: 2.000e-02, eta: 0:58:09, time: 0.121, data_time: 0.001, memory: 864, loss_rpn_cls: 0.0335, loss_rpn_bbox: 0.0064, loss_cls: 0.1748,
acc: 96.7125, loss_bbox: 0.1113, loss_mask: 0.5380, loss: 0.8890
2022-12-14 11:17:27.200 - mmdet - INFO - Epoch [1][1250/2357] lr: 2.000e-02, eta: 0:57:54, time: 0.120, data_time: 0.001, memory: 865, loss_rpn_cls: 0.0333, loss_rpn_bbox: 0.0063, loss_cls: 0.1819,
acc: 96.3574, loss_bbox: 0.1150, loss_mask: 0.5321, loss: 0.8731
2022-12-14 11:17:29.165 - mmdet - INFO - Epoch [1][1300/2357] lr: 2.000e-02, eta: 0:57:37, time: 0.119, data_time: 0.001, memory: 865, loss_rpn_cls: 0.0555, loss_rpn_bbox: 0.0099, loss_cls: 0.1809,
acc: 96.7520, loss_bbox: 0.1119, loss_mask: 0.5701, loss: 0.9390
2022-12-14 11:17:35.062 - mmdet - INFO - Epoch [1][1350/2357] lr: 2.000e-02, eta: 0:57:21, time: 0.118, data_time: 0.001, memory: 865, loss_rpn_cls: 0.0365, loss_rpn_bbox: 0.0099, loss_cls: 0.1513,
acc: 97.0430, loss_bbox: 0.0971, loss_mask: 0.5409, loss: 0.9437
2022-12-14 11:17:40.947 - mmdet - INFO - Epoch [1][1400/2357] lr: 2.000e-02, eta: 0:57:05, time: 0.118, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0452, loss_rpn_bbox: 0.0087, loss_cls: 0.1602,
acc: 97.4080, loss_bbox: 0.1005, loss_mask: 0.5203, loss: 0.8600
2022-12-14 11:17:46.907 - mmdet - INFO - Epoch [1][1450/2357] lr: 2.000e-02, eta: 0:56:51, time: 0.119, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0386, loss_rpn_bbox: 0.0086, loss_cls: 0.1604,
acc: 96.8320, loss_bbox: 0.1050, loss_mask: 0.5023, loss: 0.8130
2022-12-14 11:17:52.950 - mmdet - INFO - Epoch [1][1500/2357] lr: 2.000e-02, eta: 0:56:39, time: 0.121, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0322, loss_rpn_bbox: 0.0077, loss_cls: 0.1707,
acc: 96.5176, loss_bbox: 0.1155, loss_mask: 0.5331, loss: 0.8671
2022-12-14 11:17:59.008 - mmdet - INFO - Epoch [1][1550/2357] lr: 2.000e-02, eta: 0:56:28, time: 0.121, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0361, loss_rpn_bbox: 0.0084, loss_cls: 0.1717,
acc: 96.6838, loss_bbox: 0.1060, loss_mask: 0.5069, loss: 0.8380
2022-12-14 11:18:05.406 - mmdet - INFO - Epoch [1][1600/2357] lr: 2.000e-02, eta: 0:56:16, time: 0.120, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0416, loss_rpn_bbox: 0.0099, loss_cls: 0.1703,
acc: 96.7949, loss_bbox: 0.1105, loss_mask: 0.5425, loss: 0.8747
2022-12-14 11:18:11.714 - mmdet - INFO - Epoch [1][1650/2357] lr: 2.000e-02, eta: 0:56:11, time: 0.129, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0388, loss_rpn_bbox: 0.0072, loss_cls: 0.1659,
acc: 96.7773, loss_bbox: 0.1066, loss_mask: 0.5317, loss: 0.8410
2022-12-14 11:18:18.144 - mmdet - INFO - Epoch [1][1700/2357] lr: 2.000e-02, eta: 0:56:02, time: 0.123, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0411, loss_rpn_bbox: 0.0080, loss_cls: 0.1485,
acc: 97.1855, loss_bbox: 0.0943, loss_mask: 0.5242, loss: 0.8161
2022-12-14 11:18:30.011 - mmdet - INFO - Epoch [1][1800/2357] lr: 2.000e-02, eta: 0:55:55, time: 0.126, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0321, loss_rpn_bbox: 0.0064, loss_cls: 0.1721,
acc: 96.5480, loss_bbox: 0.1119, loss_mask: 0.4951, loss: 0.8177
2022-12-14 11:18:36.885 - mmdet - INFO - Epoch [1][1850/2357] lr: 2.000e-02, eta: 0:55:48, time: 0.125, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0277, loss_rpn_bbox: 0.0068, loss_cls: 0.1821,
acc: 96.3965, loss_bbox: 0.1174, loss_mask: 0.5220, loss: 0.8739
2022-12-14 11:18:43.313 - mmdet - INFO - Epoch [1][1900/2357] lr: 2.000e-02, eta: 0:55:43, time: 0.129, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0282, loss_rpn_bbox: 0.0063, loss_cls: 0.1632,
acc: 96.6426, loss_bbox: 0.1045, loss_mask: 0.5126, loss: 0.8149
2022-12-14 11:18:49.461 - mmdet - INFO - Epoch [1][1950/2357] lr: 2.000e-02, eta: 0:55:34, time: 0.123, data_time: 0.001, memory: 881, loss_rpn_cls: 0.0318, loss_rpn_bbox: 0.0077, loss_cls: 0.1802,
acc: 96.7018, loss_bbox: 0.1240, loss_mask: 0.5105, loss: 0.8602
```

(4)

mask\_rcnn\_r50\_fpn\_1x\_coco.py은 12 epoch 학습

mask\_rcnn\_r101\_fpn\_2x\_coco.py은 24 epoch 학습

이름	수정된 날짜
mask_rcnn_r50_fpn_1x_coco	2022-12-15 오후 7:12
mask_rcnn_r101_fpn_2x_coco	2022-12-15 오전 11:02

resnet50모델, resnet101모델 가중치 파일과 로그 파일 생성됨

이름	수정된 날짜	유형	크기
20221215_152841.log	2022-12-15 오후 7:17	텍스트 문서	215KB
20221215_152841.log.json	2022-12-15 오후 7:17	JSON 원본 파일	160KB
epoch_1.pth	2022-12-15 오후 3:43	PTH 파일	343.503KB
epoch_2.pth	2022-12-15 오후 4:02	PTH 파일	343.503KB
epoch_3.pth	2022-12-15 오후 4:21	PTH 파일	343.503KB
epoch_4.pth	2022-12-15 오후 4:40	PTH 파일	343.503KB
epoch_5.pth	2022-12-15 오후 4:58	PTH 파일	343.503KB
epoch_6.pth	2022-12-15 오후 5:17	PTH 파일	343.503KB
epoch_7.pth	2022-12-15 오후 5:36	PTH 파일	343.503KB
epoch_8.pth	2022-12-15 오후 5:55	PTH 파일	343.503KB
epoch_9.pth	2022-12-15 오후 6:14	PTH 파일	343.503KB
epoch_10.pth	2022-12-15 오후 6:32	PTH 파일	343.503KB
epoch_11.pth	2022-12-15 오후 6:52	PTH 파일	343.503KB
epoch_12.pth	2022-12-15 오후 7:12	PTH 파일	343.503KB
latest.pth	2022-12-15 오후 7:12	PTH 파일	343.503KB
mask_rcnn_r50_fpn_1x_coco.py	2022-12-15 오후 3:28	Python 원본 파일	9KB

이름	수정된 날짜	유형	크기
20221214_215831.log	2022-12-15 오전 11:08	텍스트 문서	378KB
20221214_215831.log.json	2022-12-15 오전 11:08	JSON 원본 파일	297KB
latest.pth	2022-12-15 오전 11:02	PTH 파일	492.235KB
epoch_24.pth	2022-12-15 오전 11:02	PTH 파일	492.235KB
epoch_23.pth	2022-12-15 오전 10:32	PTH 파일	492.235KB
epoch_22.pth	2022-12-15 오전 9:59	PTH 파일	492.235KB
epoch_21.pth	2022-12-15 오전 9:26	PTH 파일	492.235KB
epoch_20.pth	2022-12-15 오전 8:53	PTH 파일	492.235KB
epoch_19.pth	2022-12-15 오전 8:19	PTH 파일	492.235KB
epoch_18.pth	2022-12-15 오전 7:46	PTH 파일	492.235KB
epoch_17.pth	2022-12-15 오전 7:13	PTH 파일	492.235KB
epoch_16.pth	2022-12-15 오전 6:39	PTH 파일	492.235KB
epoch_15.pth	2022-12-15 오전 6:07	PTH 파일	492.235KB
epoch_14.pth	2022-12-15 오전 5:34	PTH 파일	492.235KB
epoch_13.pth	2022-12-15 오전 4:59	PTH 파일	492.235KB
epoch_12.pth	2022-12-15 오전 4:24	PTH 파일	492.235KB
epoch_11.pth	2022-12-15 오전 3:49	PTH 파일	492.235KB
epoch_10.pth	2022-12-15 오전 3:14	PTH 파일	492.235KB

## (5) 학습 결과 및 테스트

mask\_rcnn\_r50\_fpn\_1x\_coco.py 모델 학습 결과

```
2022-12-15 19:16:59.766 - mmdet - INFO -
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.080
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=1000 ] = 0.137
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=1000 ] = 0.077
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.195
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=300 ] = 0.195
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=1000 ] = 0.195
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = -1.000
```

```
2022-12-15 19:17:04.655 - mmdet - INFO - Exp name: mask_rcnn_r50_fpn_1x_coco.py
2022-12-15 19:17:04.655 - mmdet - INFO - Epoch(val) [12]/[2362] bbox_mAP: 0.0800, bbox_mAP_50: 0.1370, bbox_mAP_75: 0.0770, bbox_mAP_s: 0.0000, bbox_mAP_m: -1.0000, bbox_mAP_l: 0.0800, bbox_mAP_copypaste: 0.080 0.137 0.077
0.000 -1.000 0.080, segm_mAP: 0.0850, segm_mAP_50: 0.1250, segm_mAP_75: 0.0850, segm_mAP_s: 0.0000, segm_mAP_m: -1.0000, segm_mAP_l: 0.0850, segm_mAP_copypaste: 0.085 0.125 0.085 0.000 -1.000 0.085
```

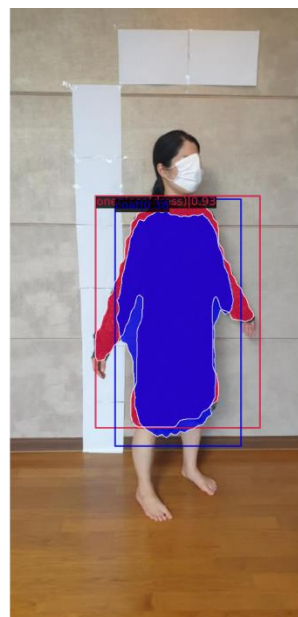
## 이미지 data 결과

### (1)

좌측이 resnet50

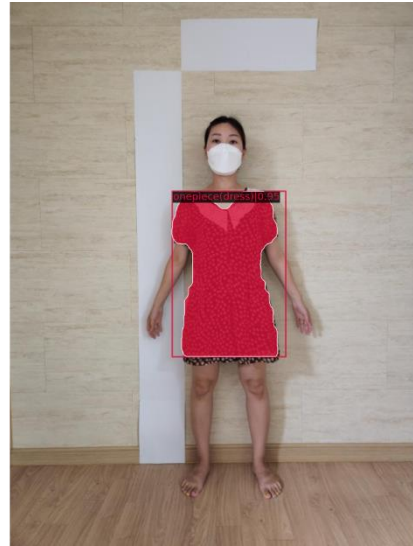


우측이 resnet101



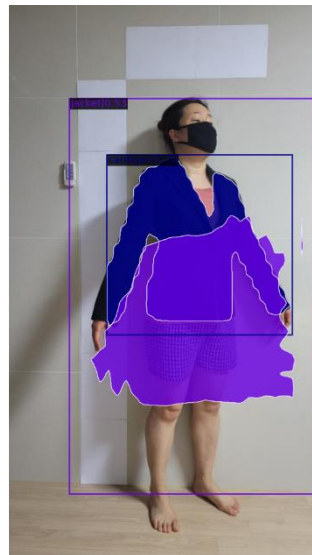
여기서는 resnet50 모델이 더 잘 맞추었다.

(2)



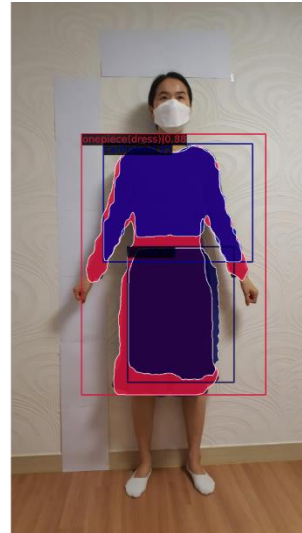
두 모델 모두 비슷하게 예측을 하였다.

(3)



여기서는 resnet101 모델이 jacket을 더 잘 예측하였다.

(4)



원래 skirt 이미지이지만 여기서는 onepiece나 cardigan으로 예측하는 모습을 볼 수 있었다.

(5)



여기서도 resnet101 모델이 pants를 좀 더 정교하게 예측하였다.



(6)



이 image는 원래 pants이지만 resnet101 모델에서 상의까지 예측하는 모습을 볼 수 있었다.

(7)



resnet50모델은 아예 예측하지 못하였지만 resnet101 모델은 정교하진 않지만 어느 정도 예측하는 것을 볼 수 있었다. 좀 더 많은 image 데이터가 많았다면 더 정교하게 예측할 수 있었을 것이다.

이상으로 test image를 확인해 보았으며

확실히 train data가 많지 않고 class간에 불균형이 심하며 학습 진행도 12epoch나 13epoch밖에 학습하지 못하여 높은 성능에 도달하지는 않았다고 생각하였다.

## (6) 모델 개발 환경 docker image 저장 및 배포

### - 베이스 도커 이미지 선택

```
1 ARG PYTORCH="1.6.0"
2 ARG CUDA="10.1"
3 ARG CUDNN="7"
4
5 FROM pytorch/pytorch:${PYTORCH}-cuda${CUDA}-cudnn${CUDNN}-devel
6
7 ENV TORCH_CUDA_ARCH_LIST="6.0 6.1 7.0+PTX"
8 ENV TORCH_NVCC_FLAGS="-Xfatbin -compress-all"
9 ENV CMAKE_PREFIX_PATH="$(dirname $(which conda))/../"
10
11 # To fix GPG key error when running apt-get update
12 RUN apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/3bf863cc.pub
13 RUN apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/7fa2af80.pub
14
15 RUN apt-get update && apt-get install -y ffmpeg libsm6 libxext6 git ninja-build libglib2.0-0 libsm6 libxrender-dev libxext6 \
16     && apt-get clean \
17     && rm -rf /var/lib/apt/lists/*
18
19 # Install MMCV
20 RUN pip install --no-cache-dir --upgrade pip wheel setuptools
21 RUN pip install --no-cache-dir mcv-full==1.3.17 -f https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/index.html
22
23 # Install MMDetection
24 RUN conda clean --all
25 RUN git clone https://github.com/open-mmlab/mmdetection.git /mmdetection
26 WORKDIR /mmdetection
27 ENV FORCE_CUDA="1"
28 RUN pip install --no-cache-dir -r requirements/build.txt
29 RUN pip install --no-cache-dir -e .
```

### Mmdetection 공식 깃허브의 mmdetection dockerfile 선택

### - 베이스 이미지 build

```
# build an image with PyTorch 1.6, CUDA 10.1
# If you prefer other versions, just modified the Dockerfile
docker build -t mmdetection docker/
```

제공되는 dockerfile을 이미지로 변환

### - 컨테이너 가동

```
see docker run --help .
root@hsb-VirtualBox:/home/hsb/mmlab/mmdetection# docker run -it --cpus=1 --shm-size=8gb -v mmdet
ection_volume:/mmdetection/data mmdetection
```

V 옵션으로 우분투 로컬 볼륨과 컨테이너 볼륨 연결



- 컨테이너 내부

```
root@003fe5ec3745:/mmdetection# ls
CITATION.cff  README_zh-CN.md  docker          model-index.yml  resources  tools
LICENSE       configs          docs           pytest.ini       setup.cfg
MANIFEST.in   data            mmdet         requirements     setup.py
README.md     demo           mmdet.egg-info requirements.txt  tests
root@003fe5ec3745:/mmdetection#
```

Github를 commit 한 결과와 동일

- 학습한 모델 test

1) python파일 실행

```
root@003fe5ec3745:/mmdetection/demo# python image_demo.py
No CUDA runtime is found, using CUDA_HOME='/usr/local/cuda'
usage: image_demo.py [-h] [--out-file OUT_FILE] [--device DEVICE]
                    [--palette {coco,voc,citys,random}]
                    [--score-thr SCORE_THR] [--async-test]
                    img config checkpoint
image_demo.py: error: the following arguments are required: img, config, checkpoint
root@003fe5ec3745:/mmdetection/demo#
```

실행 시 에러 발생 img, config, checkpoint 명령행 인자 필요

2) 연결한 볼륨에 가중치 파일 삽입

```
root@hsb-VirtualBox:/var/lib/docker/volumes/mmdetection_volume/_data# ls
mask_rcnn_r50_fpn_1x_coco_20200205-d4b0c5d6.pth
```

3) 컨테이너 볼륨 확인

```
root@29ceae58569c:/mmdetection/data# ls
mask_rcnn_r50_fpn_1x_coco_20200205-d4b0c5d6.pth
```

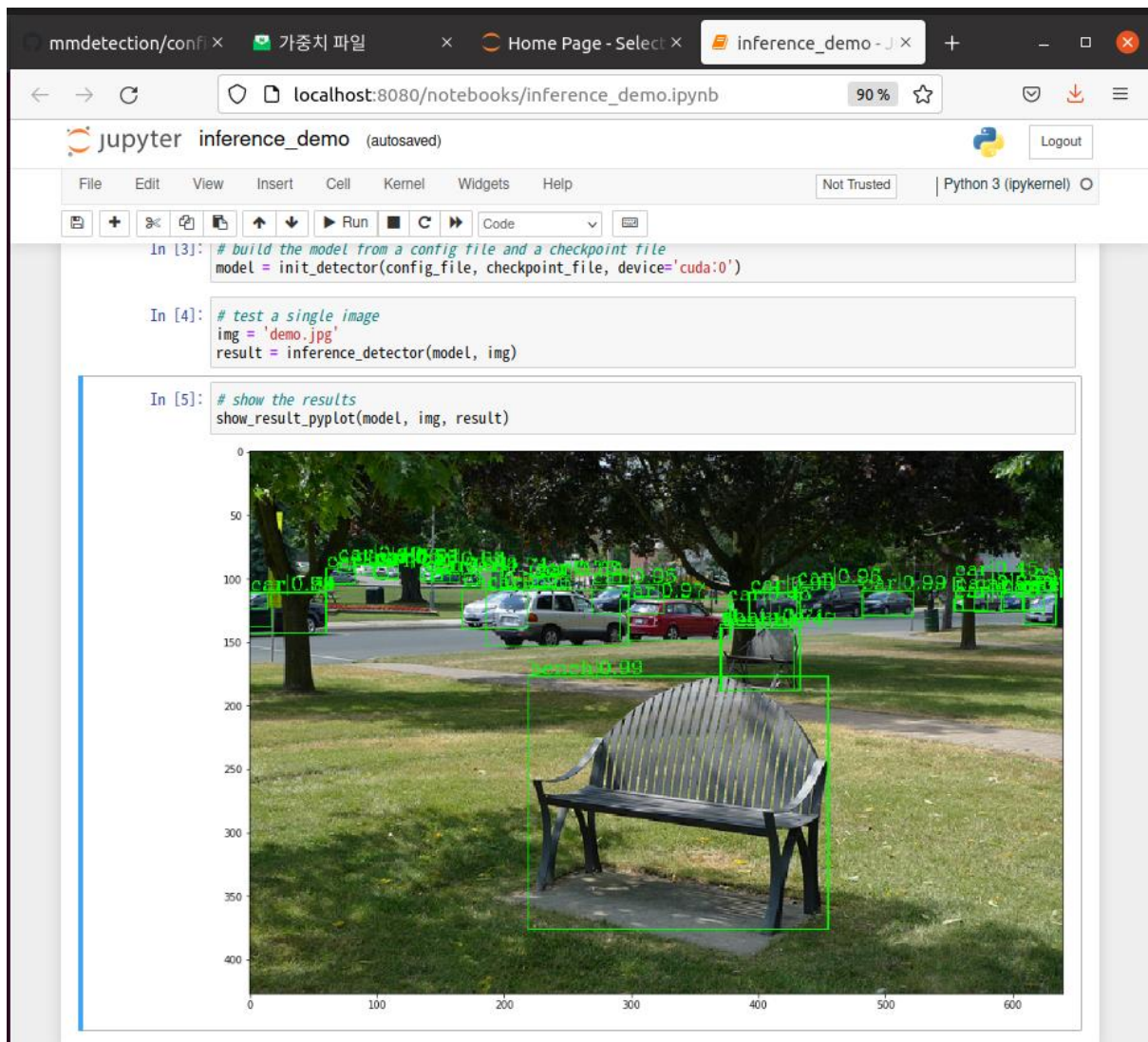
결과 가중치 파일 생성 확인

#### 4) 실행

```
root@29ceae58569c:/mmdetection# python demo/image_demo.py demo/demo.jpg \
> configs/mask_rcnn/mask_rcnn_r50_fpn_1x_coco.py checkpoints/mask_rcnn_r50_fpn_1x_coco_20200205-
d4b0c5d6.pth --device cpu
No CUDA runtime is found, using CUDA_HOME='/usr/local/cuda'
load checkpoint from local path: checkpoints/mask_rcnn_r50_fpn_1x_coco_20200205-d4b0c5d6.pth
/mmdetection/mmdet/datasets/utils.py:70: UserWarning: "ImageToTensor" pipeline is replaced by "D
efaultFormatBundle" for batch inference. It is recommended to manually replace it in the test da
ta pipeline in your config file.
  'data pipeline in your config file.', UserWarning)
Illegal instruction (core dumped)
```

하지만 cli 환경이라서 core dumped 발생

#### 5) 주피터 노트북으로 test



The screenshot shows a Jupyter Notebook titled 'inference\_demo' running on a local host. The notebook contains three code cells:

- In [3]:** `# build the model from a config file and a checkpoint file`  
`model = init_detector(config_file, checkpoint_file, device='cuda:0')`
- In [4]:** `# test a single image`  
`img = 'demo.jpg'`  
`result = inference_detector(model, img)`
- In [5]:** `# show the results`  
`show_result_pyplot(model, img, result)`

The output of the third cell is a plot of the image 'demo.jpg' with green bounding boxes around detected objects. The plot includes a coordinate grid on the left and bottom axes. The detected objects are labeled with their class names and confidence scores: 'car 0.95', 'person 0.95', and 'bench 0.99'.

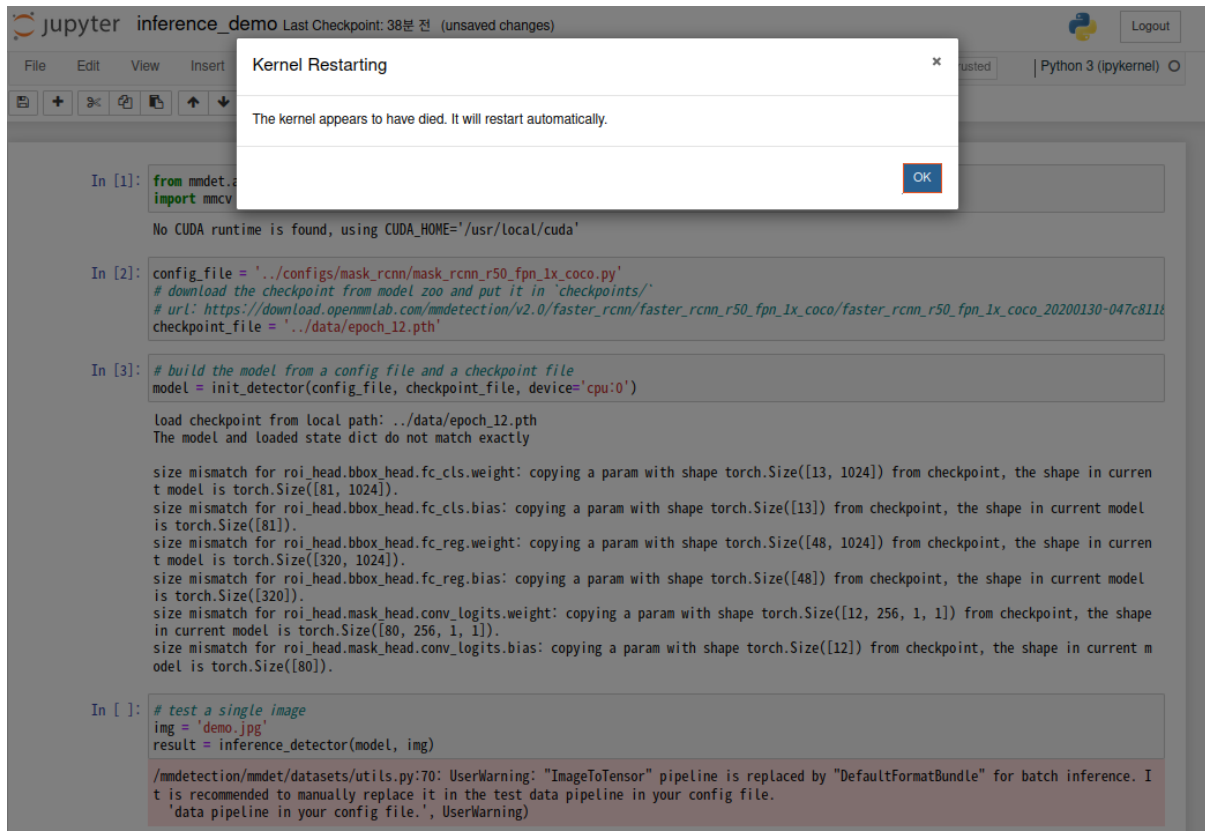
demo.py 실행 성공

## 6) 커스텀 모델 업로드

```
root@12743e928f34:/mmdetection# ls data/  
epoch_12.pth mask_rcnn_r50_fpn_1x_coco_20200205-d4b0c5d6.pth
```

Epoch\_12는 resnet50 모델 mask\_rcnn\_r50\_fpn\_1x\_coco.py 형식 모델

## 7) 모델 테스트



Kernel died

발생 원인

1. 컨테이너 환경은 최소한의 자원을 가지고 있음 따라서 메모리 초과 발생
2. 기본 제공 mask\_rcnn\_r50\_fpn\_1x\_coco.py 모델보다 커스텀 모델의 용량이 큰 것

## 8) docker image commit

```
root@hsb-VirtualBox:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
12743e928f34   mmdetection    "/bin/bash"             2 hours ago   Up 2 hours    0.0.0.0:8080->
8888/tcp, :::8080->8888/tcp
funny_torvalds
be7908faf390   nginx:latest   "/docker-entrypoint...." 5 hours ago   Up 5 hours    80/tcp
crazy_visvesvaraya
root@hsb-VirtualBox:~# docker commit 12743e928f34 kFashion
invalid reference format: repository name must be lowercase
root@hsb-VirtualBox:~# docker commit 12743e928f34 k_fashion
sha256:bc6cf2d7d9f9567ddcccec17298745eb89fcd59bfb477a10fca5c43752de85f4
root@hsb-VirtualBox:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
k_fashion      latest    bc6cf2d7d9f9   7 seconds ago  8.2GB
mmdetection    latest    390310c0870c   6 hours ago    7.98GB
```

## 9) 실행

```
root@hsb-VirtualBox:~# docker run -it k_fashion /bin/bash
root@9691e972ee94:/mmdetection# ls
CITATION.cff  README_zh-CN.md  docker      model-index.yml  resources  tools
LICENSE       configs          docs        pytest.ini       setup.cfg
MANIFEST.in   data            mmdet      requirements      setup.py
README.md     demo            mmdet.egg-info  requirements.txt  tests
```

## K\_fashion 이미지 실행

## 내용 확인

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {},
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "No CUDA runtime is found, using CUDA_HOME='/usr/local/cuda'\n"
          ]
        }
      ],
      "source": [
        "from mmdet.apis import init_detector, inference_detector, show_result_pyplot\n",
        "import mmcv"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {},
      "outputs": [],
      "source": [
        "config_file = '../configs/mask_rcnn/mask_rcnn_r50_fpn_1x_coco.py'\n",
        "# download the checkpoint from model zoo and put it in 'checkpoints/'\n",
        "# url: https://download.openmmlab.com/mmdetection/v2.0/faster_rcnn/faster_rcnn_r50_fpn_1x_coco/faster_rcnn_r50_fpn_1x_coco_20200130-047c8118.pth\n",
        "checkpoint_file = '../data/epoch_12.pth'"
      ]
    }
  ]
}
```

Text: "No CUDA runtime is found" 와 Checkpoint\_file 경로 유지 확인

#### 10) v 옵션으로 실행

```
root@hsb-VirtualBox:/var/lib/docker/volumes# docker run -it -v mmdetection_volume:/mmdetection/data k_fashion
root@2e60626920f0:/mmdetection# ls data/
epoch_12.pth  mask_rcnn_r50_fpn_1x_coco_20200205-d4b0c5d6.pth
root@2e60626920f0:/mmdetection#
```

#### v 옵션 지정 성공 확인

#### 11) 이미지 도커 허브 배포

hsb422 / **k\_fashion**

Contains: Image | Last pushed: 6 minutes ago

 Not Scanned

 0

 0

 Public