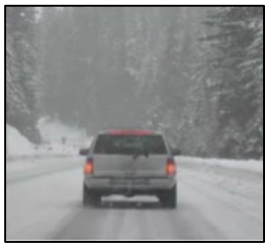
신경망 리뷰

박성호 (neowizard2018@gmail.com)



Neural Network
(ANN, CNN, RNN)

Y

➢ 위와 같은 이미지를 신경망(Neural Network)에 입력으로 주고 신경망 출력 값 Y 를 계산하는 경우라고 가정할 때,

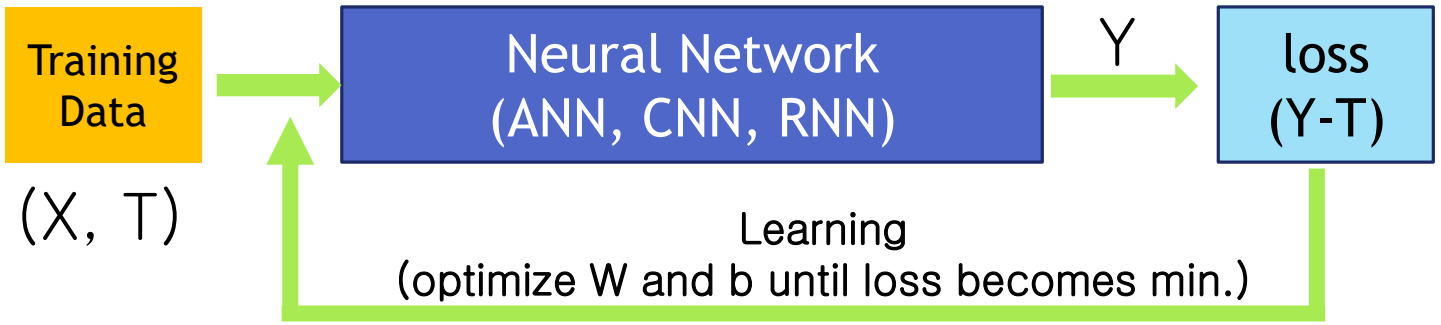
- 입력 이미지는 자동차, 트럭, 비행기 가운데 하나일때, 신경망 출력 노드 개수는 ?
- 출력 층에서의 Y = [0.6, 0.1, 0.3] 으로 나타날 때, 손실함수(loss) 값을 계산하기 위해서는, 정답(label)의 차원(dimension)은 어떻게 표현할 수 있는가 ?
- 입력 데이터의 정답 T = [1, 0, 0] 으로 표현할 때, 출력 층에서의 Y 값은 어떤 차원(dimension)을 가져야만 손실함수 값을 계산할 수 있는가 ?

Neural Network
(ANN, CNN, RNN)

Y

➢ 이미지 내의 객체가 무엇인지 알아내고 그러한 객체가 존재하는 시작 위치 (좌표) 정보를 신경망을 통해서 예측하고 하는 경우,

- 입력 이미지는 자동차, 트럭, 비행기 가운데 하나이며, 해당 객체(object)가 존재하는 좌측 상단 (x, y) 값이 정답(label)으로 주어질 때, 신경망 출력 노드의 개수는 ?
- 입력 이미지의 정답 T 값의 차원(dimension)은 ?
- 신경망 출력 값 Y 값의 차원(dimension)은 ?



손실(loss) 값을 계산하기 위해서는

정답 T와 신경망의 출력 값 Y 차원이 동일해야 함

딥러닝
컴퓨터 비전

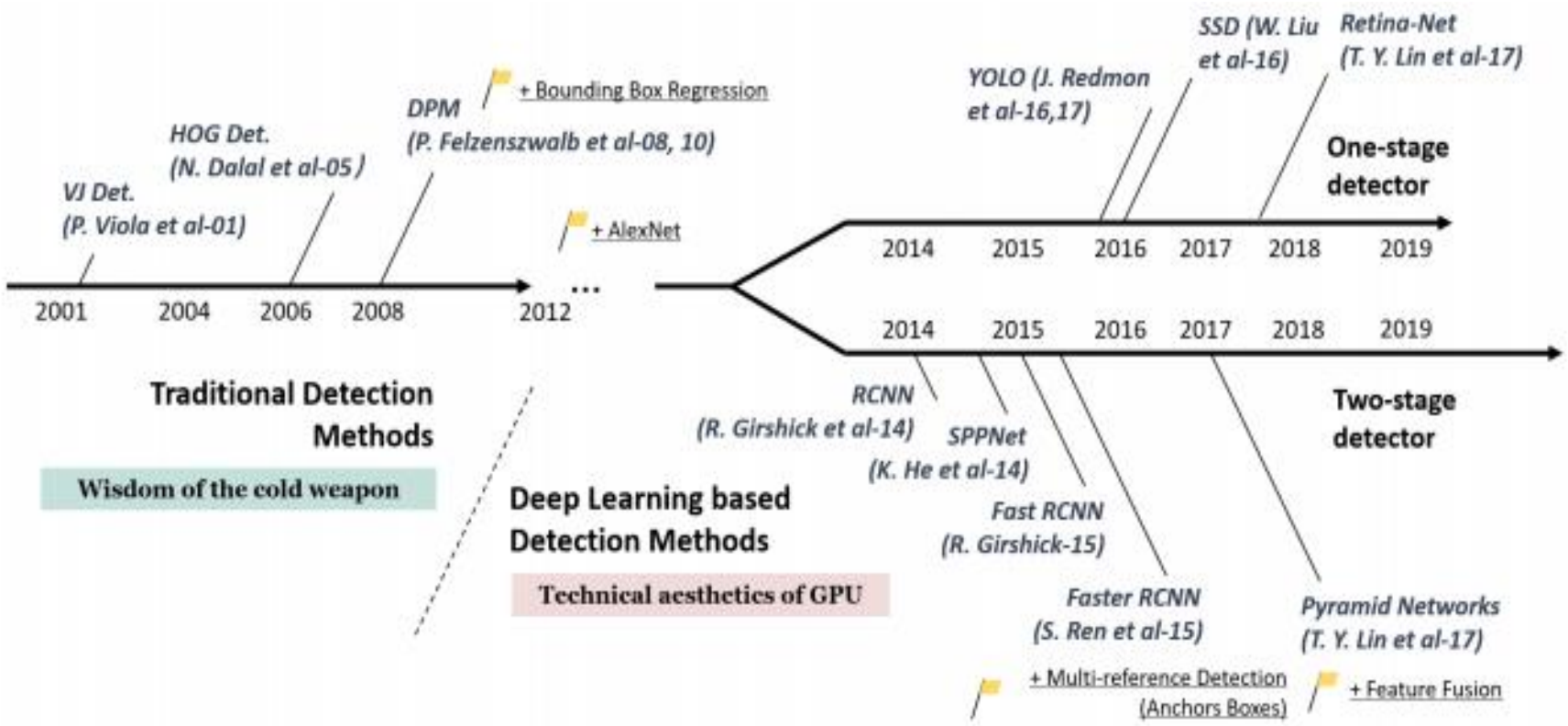
- Object Detection 개요 -

박성호 (neowizard2018@gmail.com)

Object Detection 개요

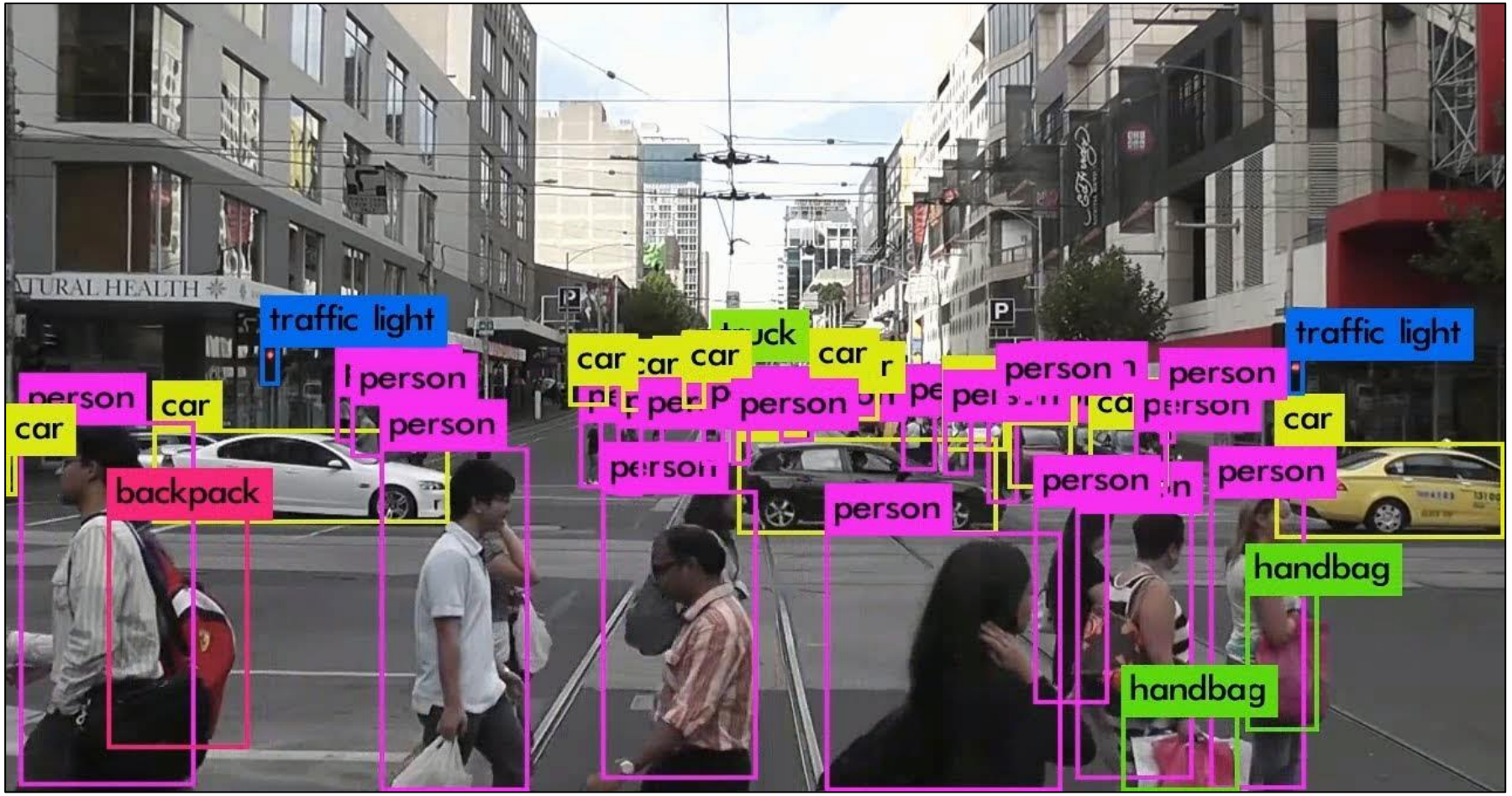
- Object Detection은 이미지나 비디오 내의 자동차, 사람, 동물, 물건 등의 위치와 종류를 알아내는 것을 Object Detection이라고 함

  - ✓ Object Detection은 2012년 이전에는 모두 영상 처리 알고리즘으로 해결했으나, 2012년 AlexNet이 나타나고 부터는 딥러닝을 활용하여 문제를 처리하고 있음



Object Detection = Object Classification + Object Localization

= Object Classification with Localization



Object Classification with Localization – concept

Object Classification

car

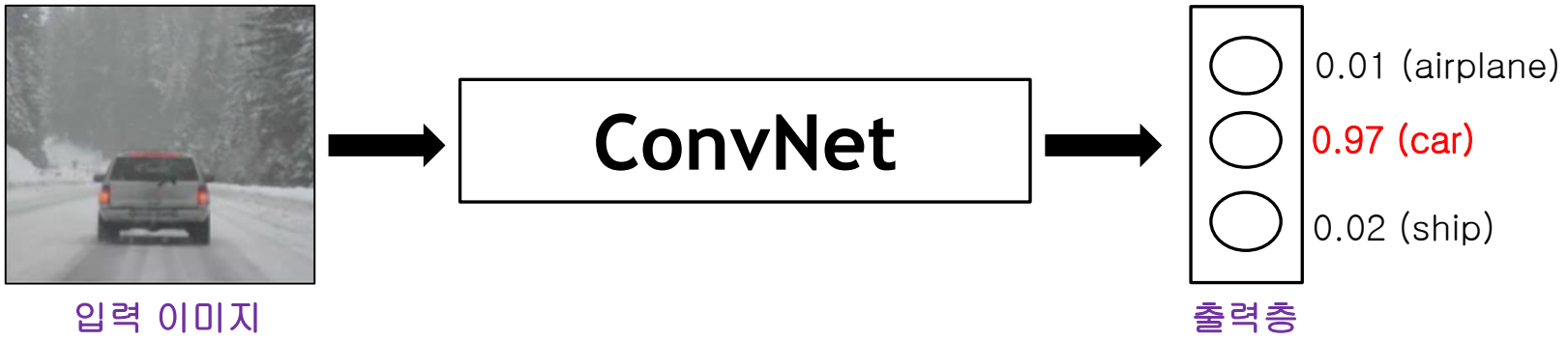
Classification with Localization

car with bounding box

※ 사진 속 물체가 여러 개라면, 당연히 모든 물체를 분류하고 동시에 bounding box를 이용해서 정확한 위치 또한 알아내야하지만, 지금은 개념 확립을 위해서 1개의 물체 만을 가정함

Object Classification using softmax

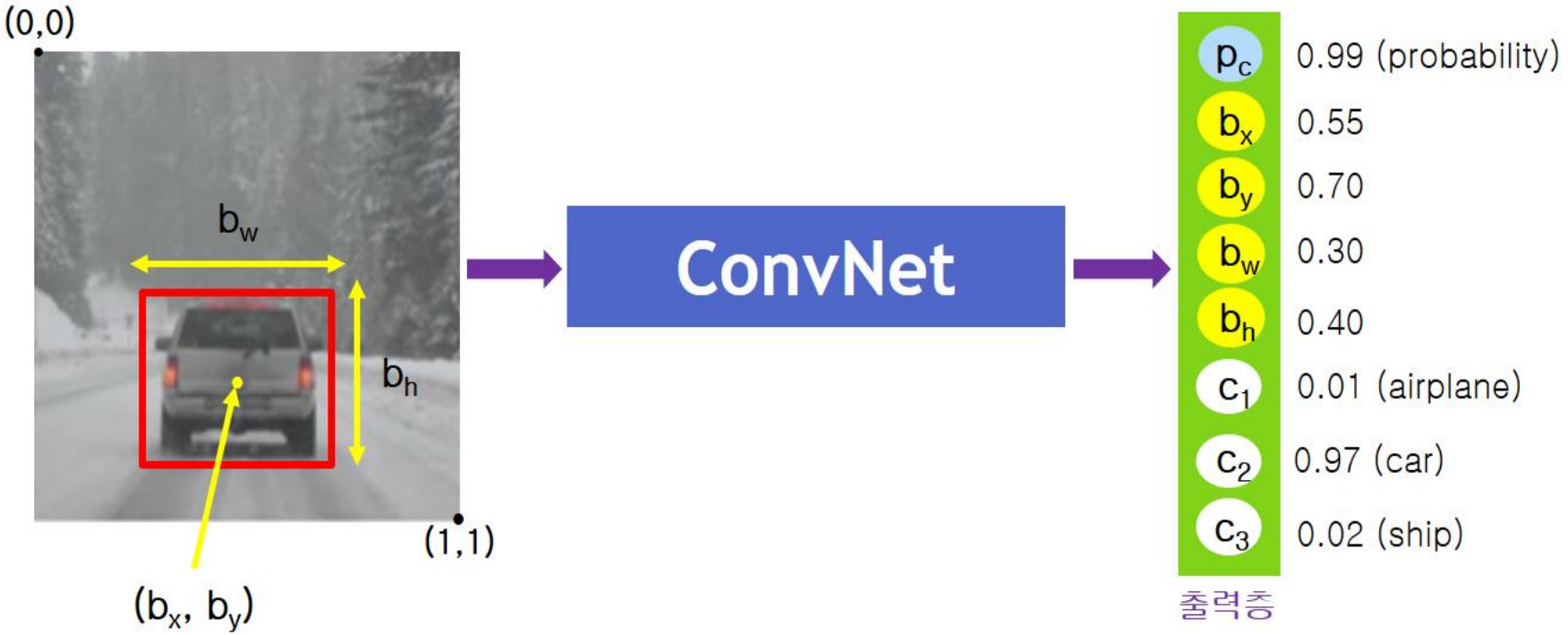
- 이미지에서 물체를 예측하기 위해서는 이미지를 ConvNet에 넣어주고, 출력 층에서 softmax 함수를 사용하면 이미지내의 물체가 무엇인지 알아 낼 수 있음

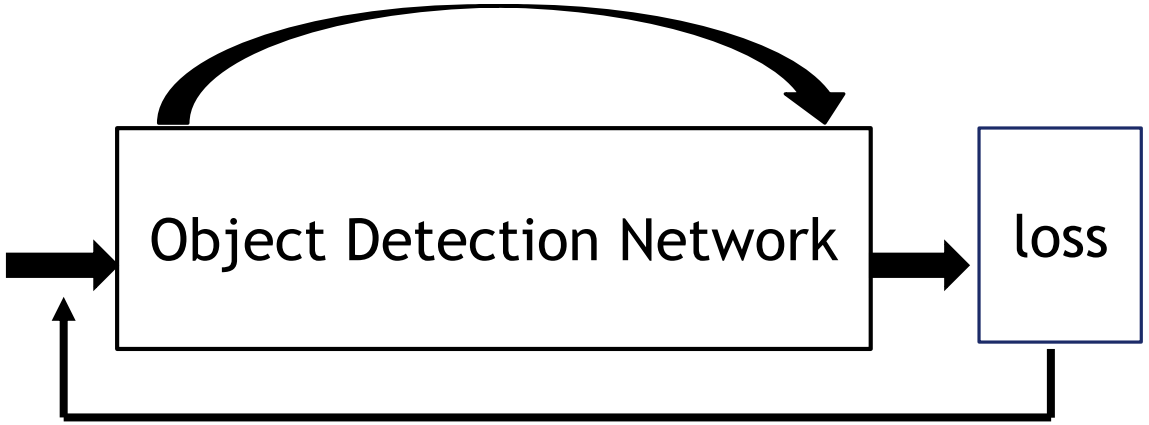


Object Localization using bounding box

- Object Localization은 물체의 주변에 bounding box를 표시하고, 이러한 bounding box의 위치(좌표)를 신경망 출력에 포함시키면 물체의 위치를 쉽게 파악할 수 있음

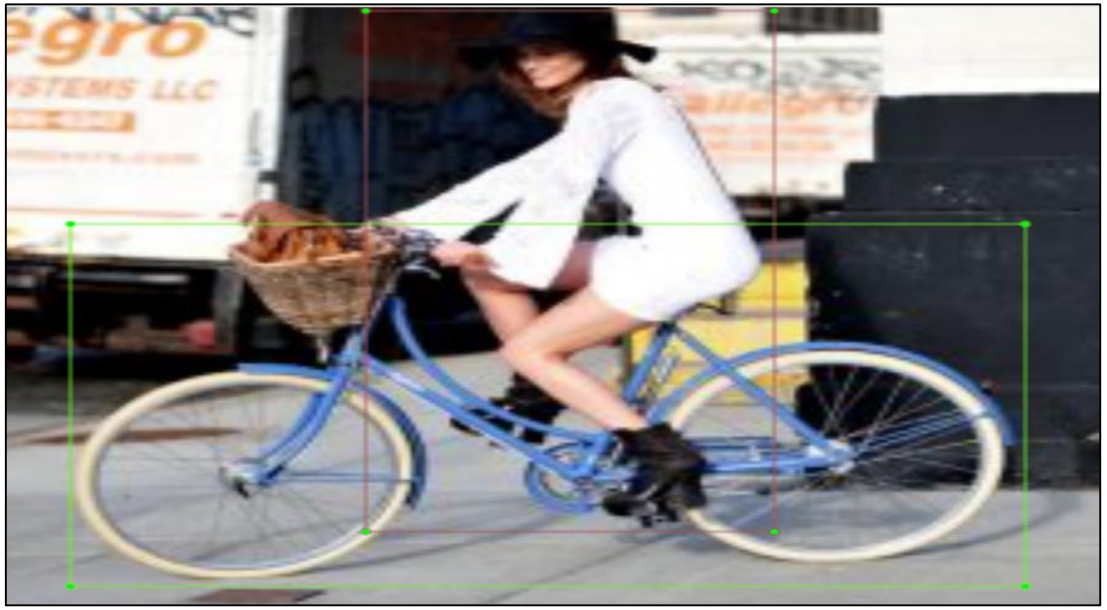
신경망 출력 $y = [\ p_c, b_x, b_y, b_w, b_h, c_1, c_2, c_3\ ]$






```
0 0.510101 0.432500 0.279461 0.855000
1 0.494949 0.652500 0.653199 0.595000
```

bounding box 를 labeling 한 이미지



YOLO
(You Only Look Once)

– Grid Cell · Bounding Box · IoU –

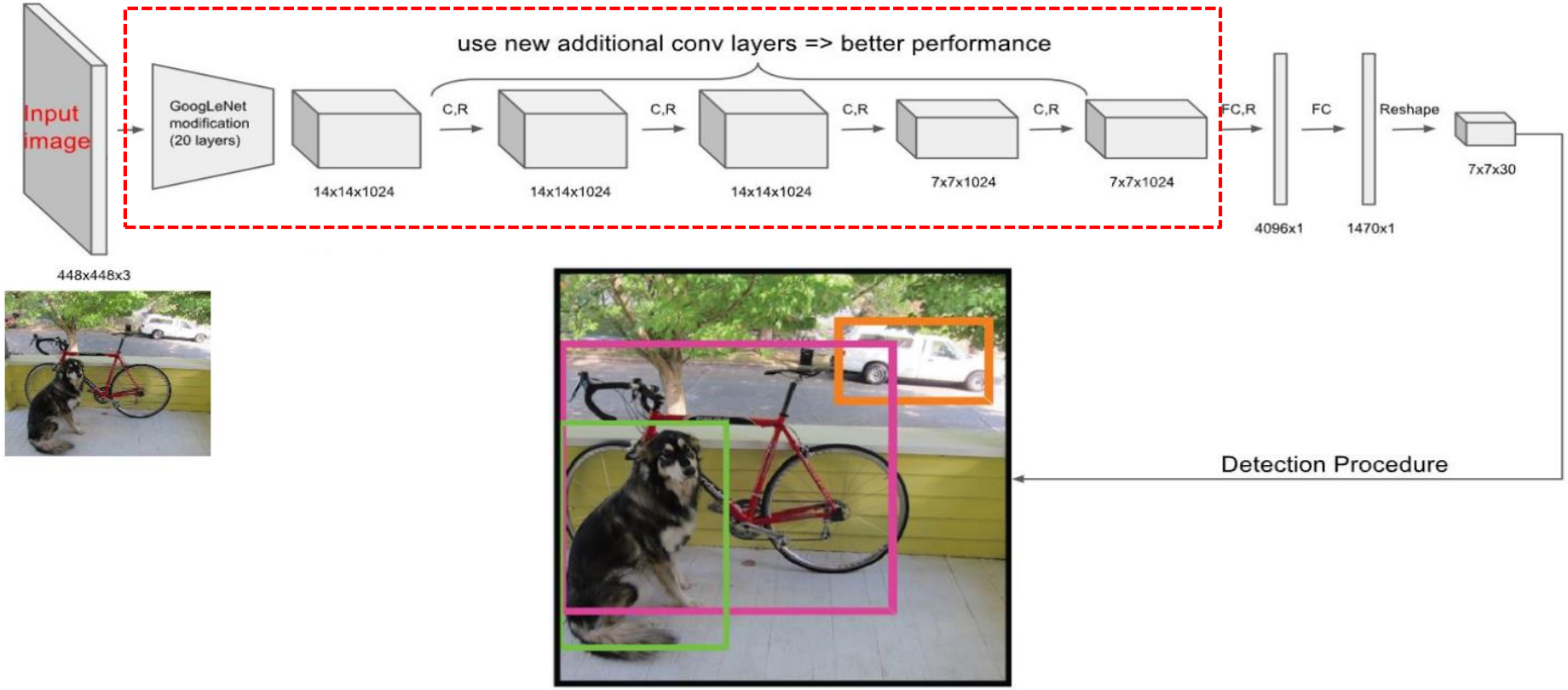
박성호 (neowizard2018@gmail.com)

YOLO (You Only Look Once)

- YOLO는 You Only Look Once 약어로서, 2015년 Joseph Redmon이 워싱턴 대학교에서 여러 친구들과 함께 YOLO 아키텍처를 논문과 함께 발표함

  - ✓ 당시만 해도 two-shot-detection 방식인 Faster R-CNN (Region with CNN)가 가장 좋은 성능을 내지만 실시간 성이 굉장히 부족함 (7 FPS 최대)

  - ✓ 이때 one-shot-detection 방식으로 동작하는 YOLO 가 등장하여 평균 45 FPS을 보여주었고 빠른 버전의 경우 최대 155 FPS을 기록하며 사람들을 놀라게 함.

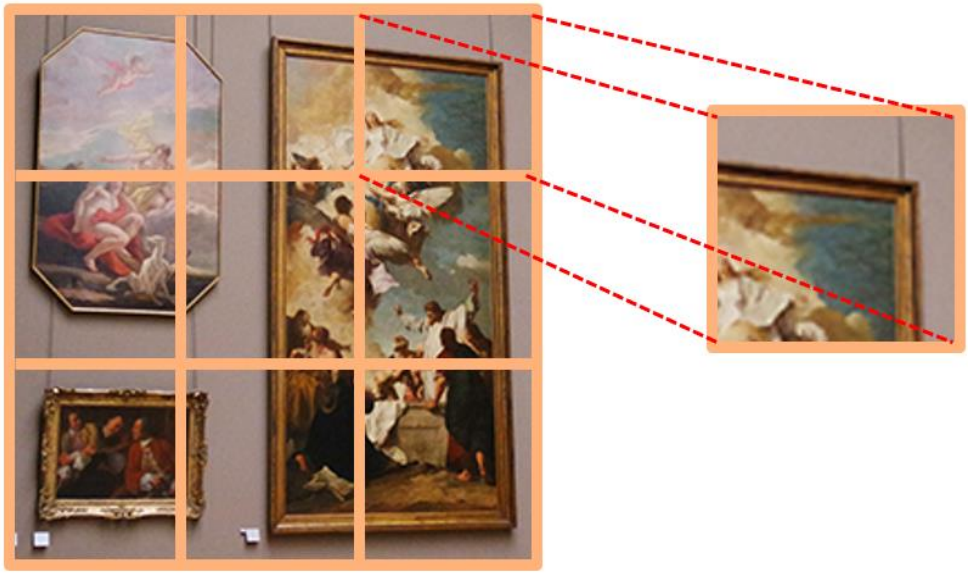


- YOLO 아키텍처는 feature extractor 로서 Google LeNet + 5 개의 new layer 를 사용하며, Flatten layer 를 거쳐서 최종 출력층에서는 7x7x30 텐서를 만들어내는 아키텍처이다. 즉 YOLO 아키텍처는 기본적으로 yolov5s.pt 같은 pre-trained model 로서 제공되며, 사용자 데이터 이용해서 Transfer Learning 에서 fine-tuning 하는 과정임

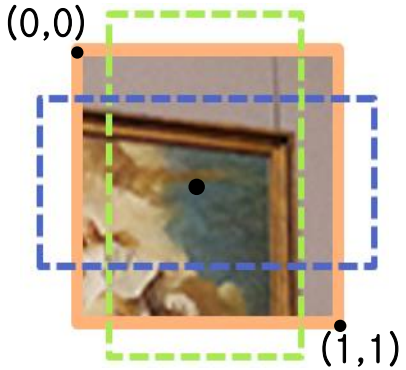
- YOLO는 입력 이미지에 대해서, Bounding Box와 classification 작업을 동시에 수행

✓ YOLO 아키텍처는, 먼저 입력 이미지를 S x S grid로 나눔.

(다음 이미지는 3 x 3 grid로 나누었으나, 실제 YOLO 논문에서는 7 x 7 grid로 분할함)



✓ YOLO 아키텍처를 학습할 경우, 각각의 그리드 셀은 B 개의 bounding box 를 가지고 있으며, 각각의 그리드 셀(grid cell)은 B개의 bounding box와 그 bounding box에 대한 confidence score를 가짐 . 즉 각각의 bounding box는 (x, y, w, h, confidence score) 같은 5개의 값으로 구성됨



(x, y) 는 bounding box의 중심점

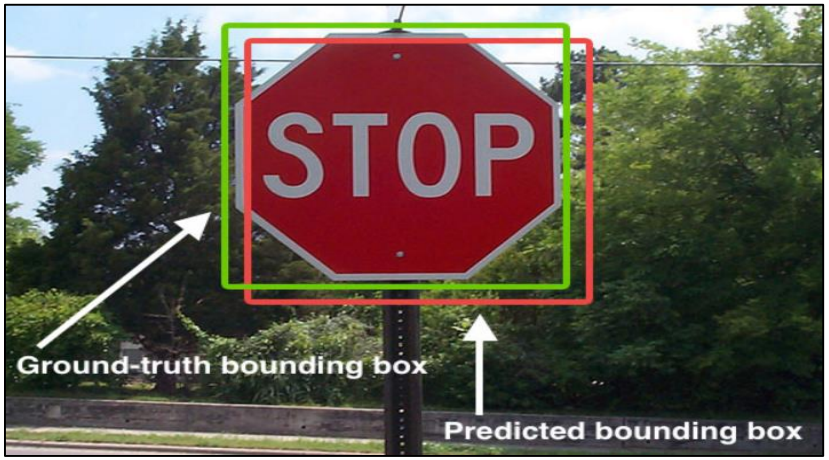
(w, h) 는 해당 셀에서 bounding box가 차지하는 width, height

$$\text{confidence score} = \Pr(\text{Object}) * \text{IoU}_{\text{pred}}^{\text{truth}}$$

Confidence: 해당 그리드 셀에 물체가 존재할 확률을 나타냄

[참고] Ground-truth, IoU

✓ 딥러닝에서 Ground-truth는 학습 데이터의 실제 값을 표현할 때 사용되는 개념. 즉 학습데이터에서 주어지는 정답(label)으로 생각해도 무방함.



✓ 딥러닝 모델이 얼마나 예측을 잘 했는지를 알기 위해 IoU(Intersection over Union) 사용

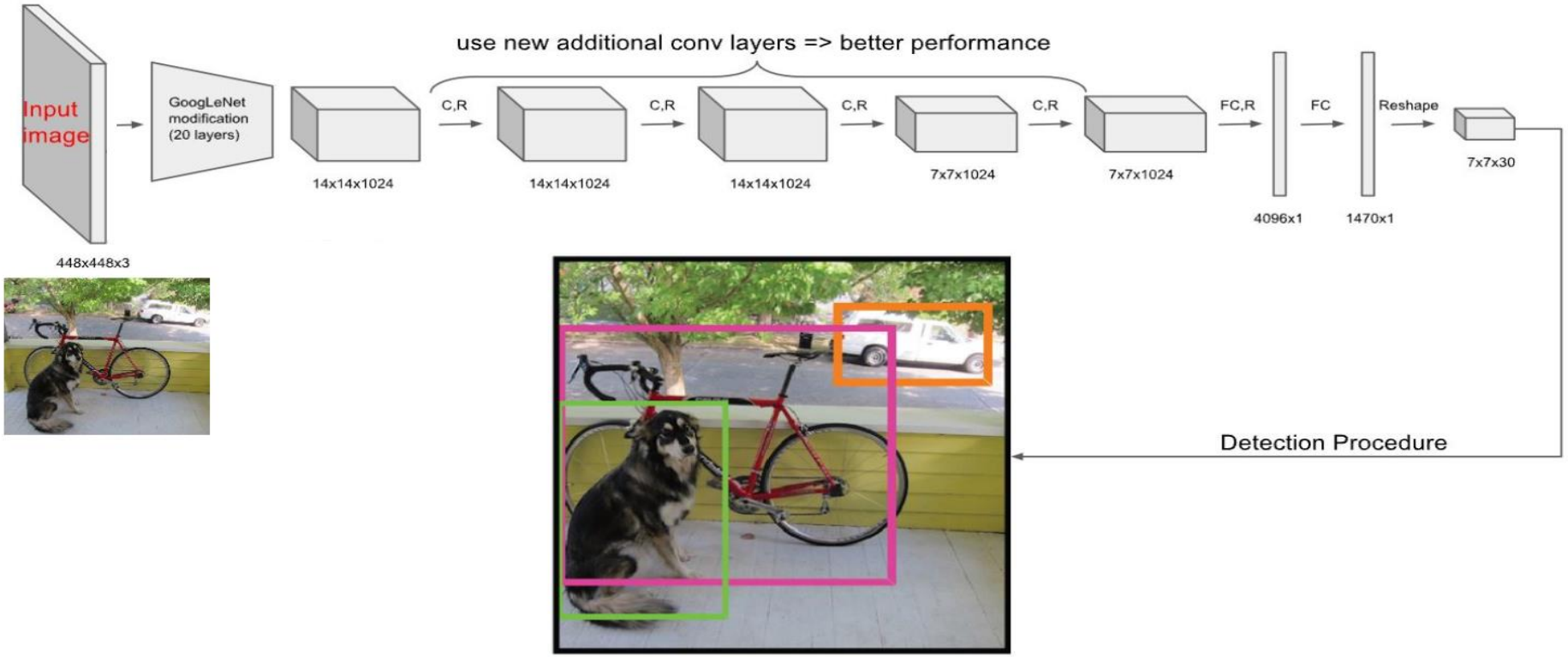
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

YOLO
Inference Process

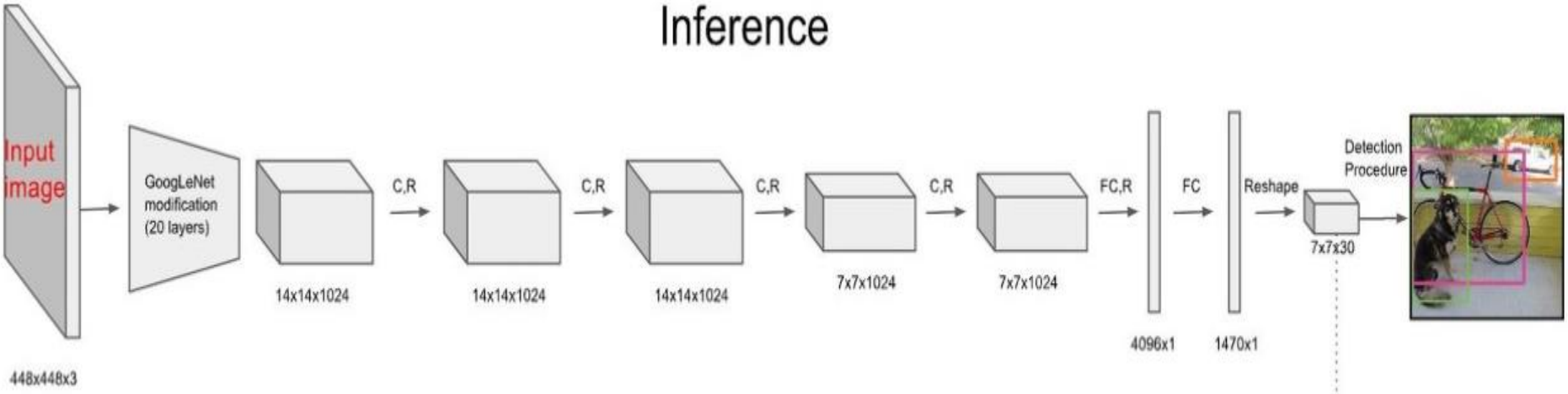
박성호 (neowizard2018@gmail.com)

YOLO Inference Process

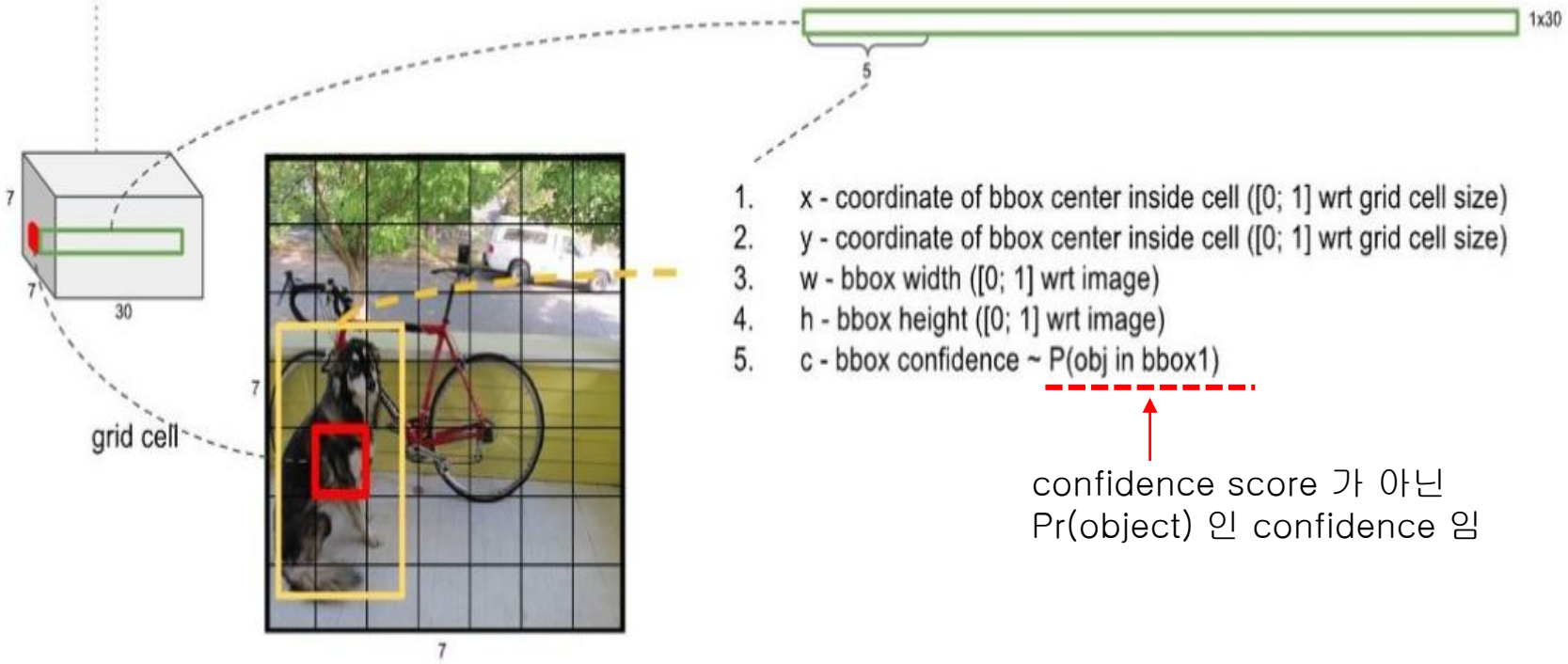
✓ 입력 이미지를 7 x 7 그리드 셀로 나누며, 각각의 그리드 셀에 들어있는 2개의 bounding box 정보와 물체의 클래스 정보가 들어있는 7x7x (5+5+20) 데이터가 YOLO 예측 결과임



1st bounding box of a grid cell



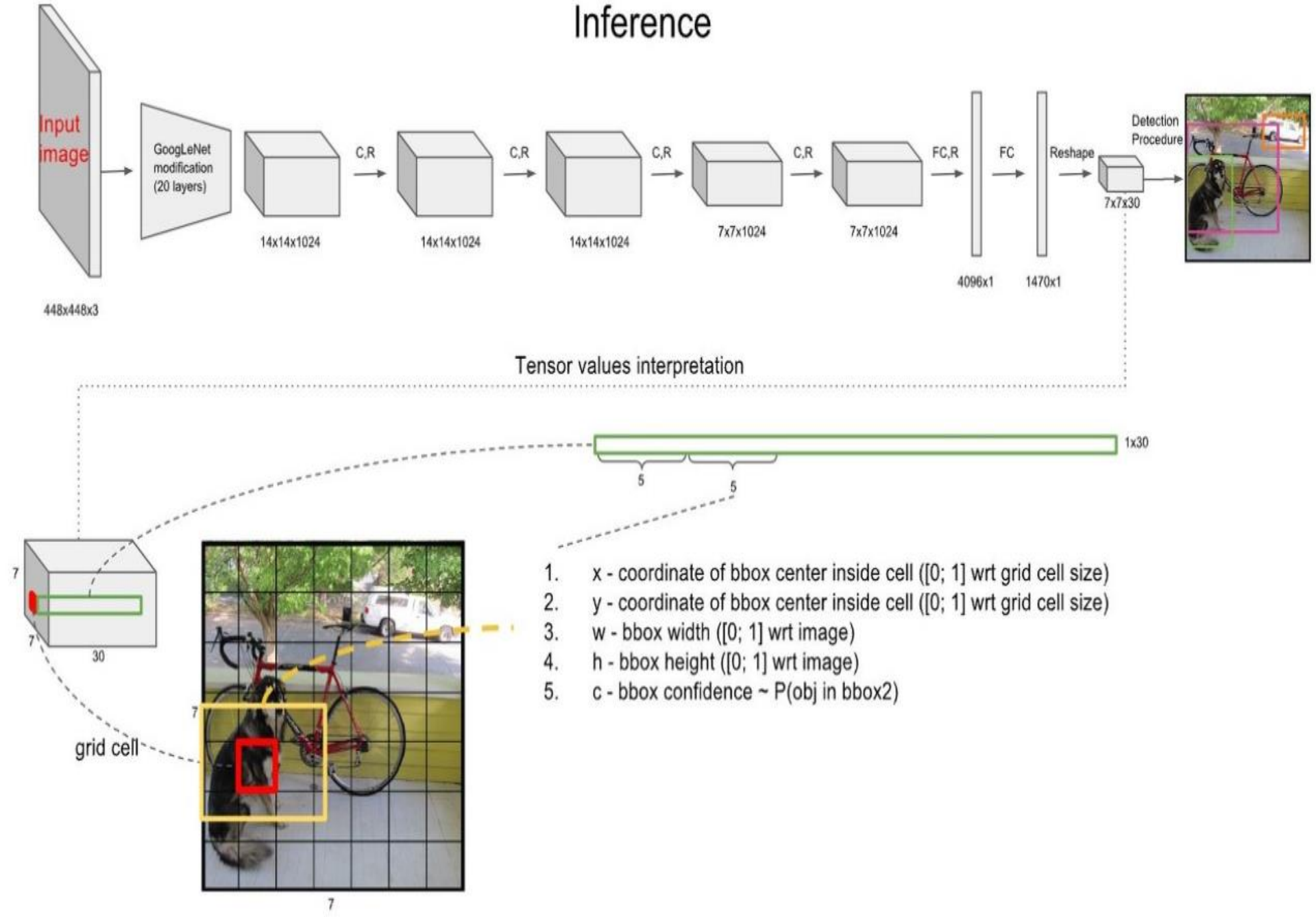
Tensor values interpretation



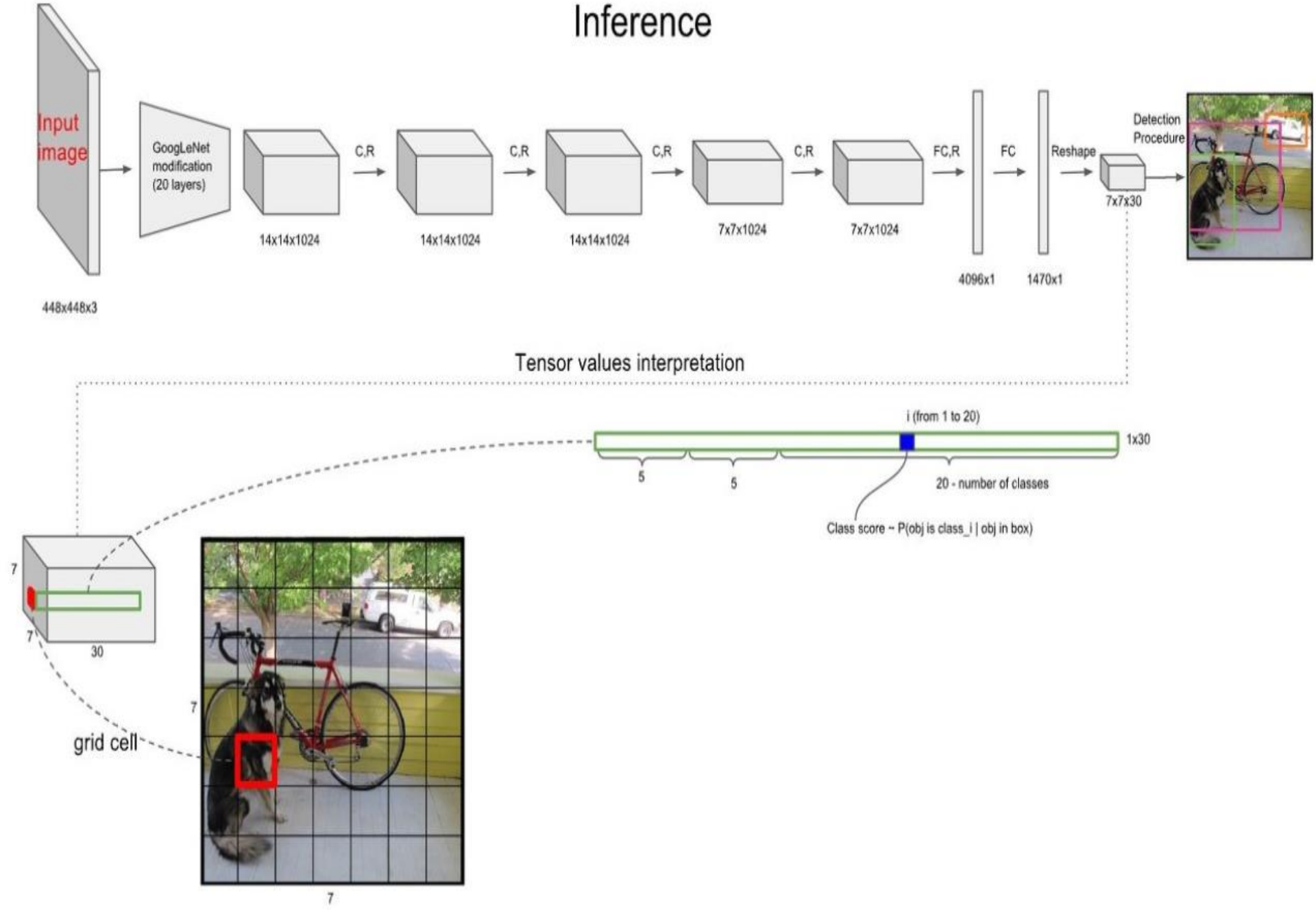
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

confidence score 가 아닌
Pr(object) 인 confidence 임

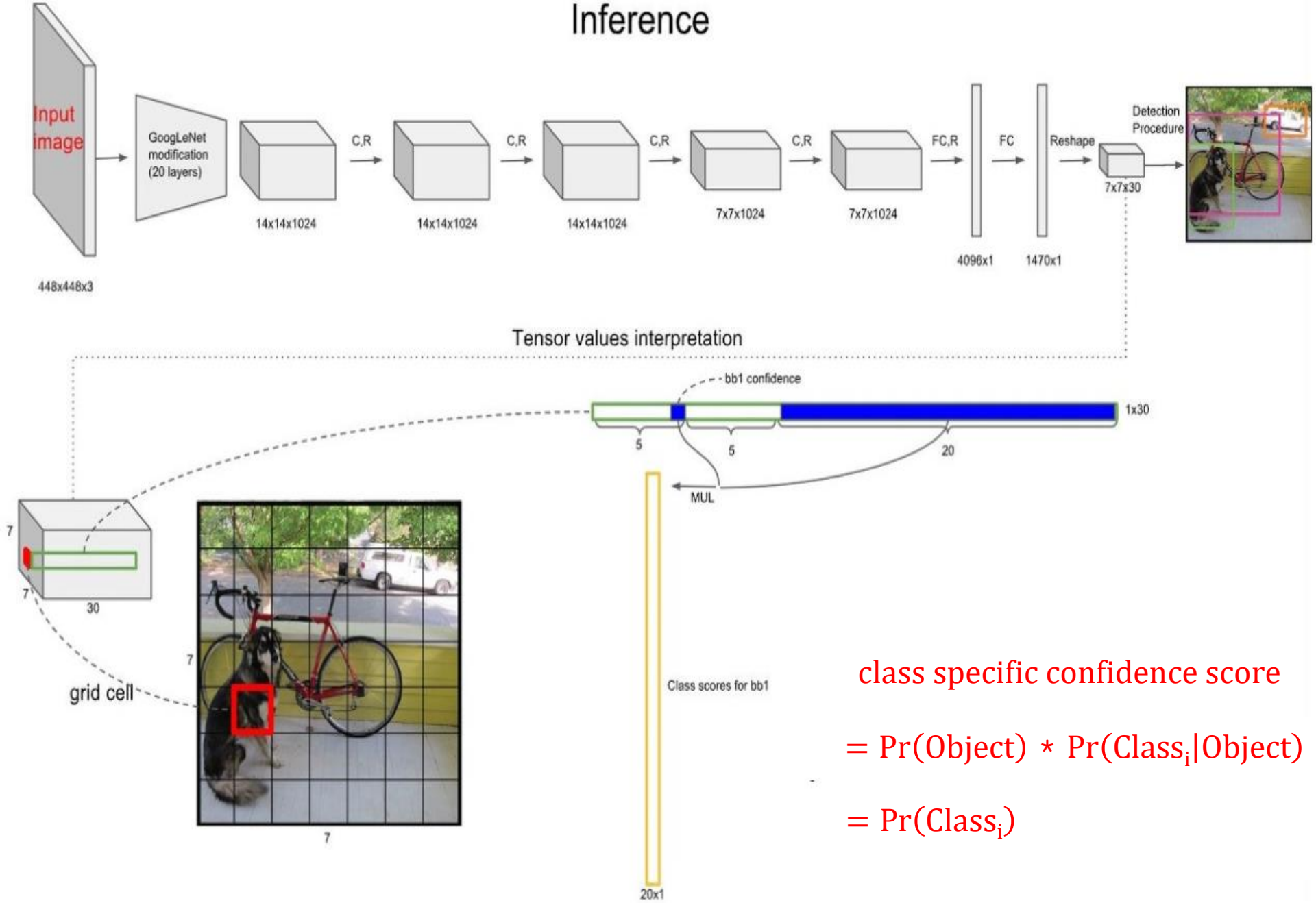
2nd bounding box of a grid cell



Class score


Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
i (from 1 to 20)
1x30
5
5
20 - number of classes
Class score ~ P(obj is class_i | obj in box)
7
7
30
7
grid cell
7

각 grid cell 마다 2개의 class-specific confidence score 계산

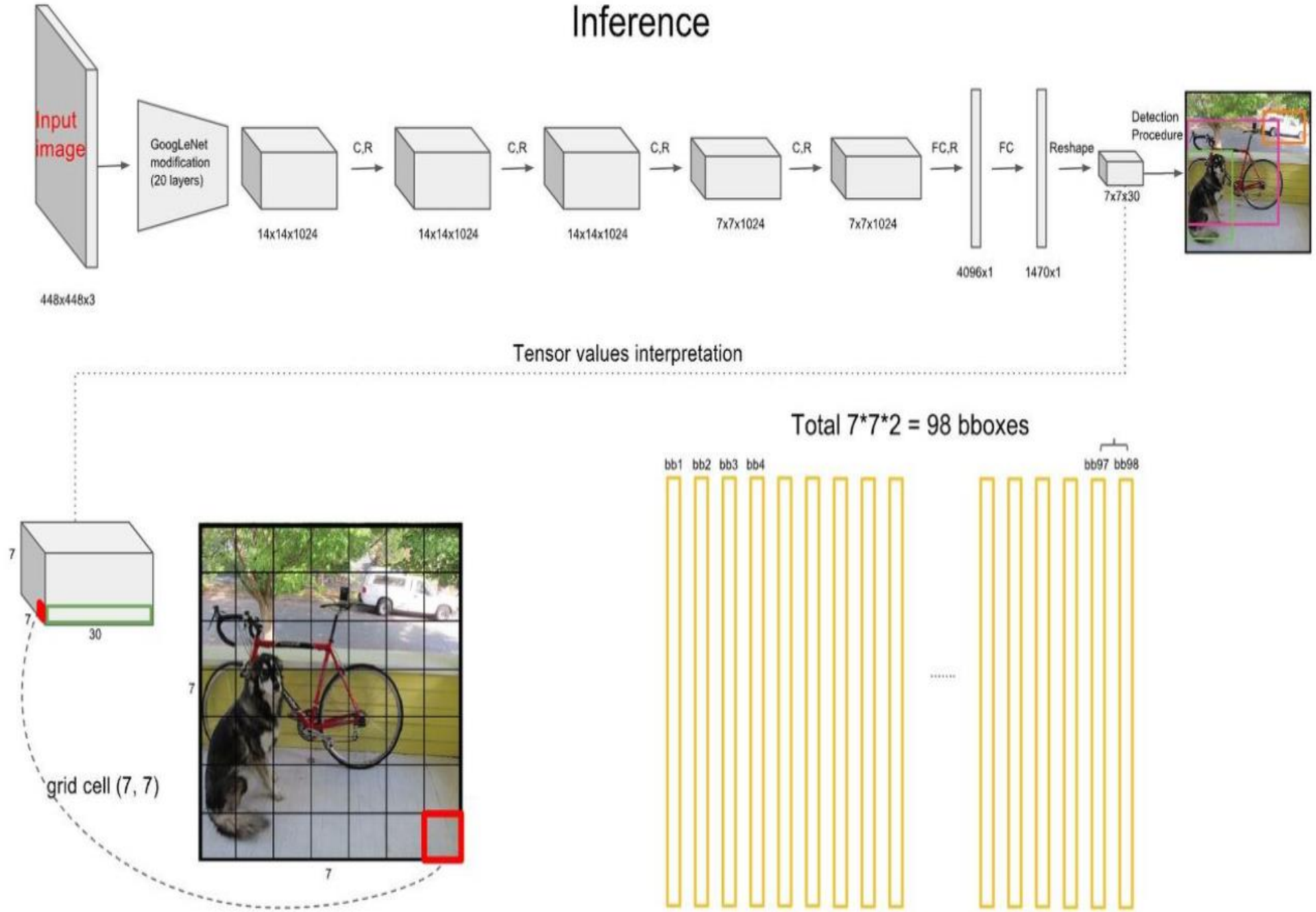


class specific confidence score

$= \Pr(\text{Object}) * \Pr(\text{Class}_i|\text{Object})$

$= \Pr(\text{Class}_i)$

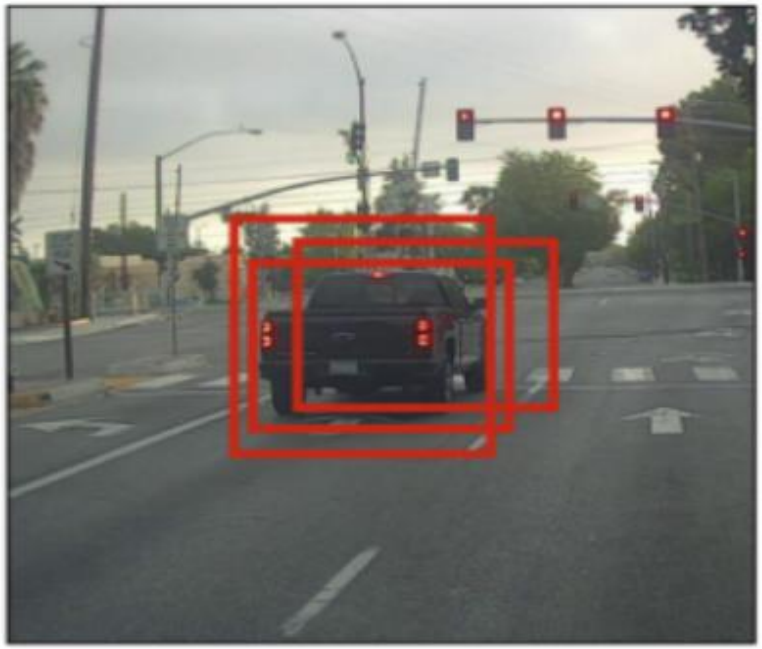
총 98개의 바운딩 박스의 class-specific confidence score 계산



➢ YOLO에서는 동일한 객체에 대하여 많은 bounding box가 잡힐 수 있음

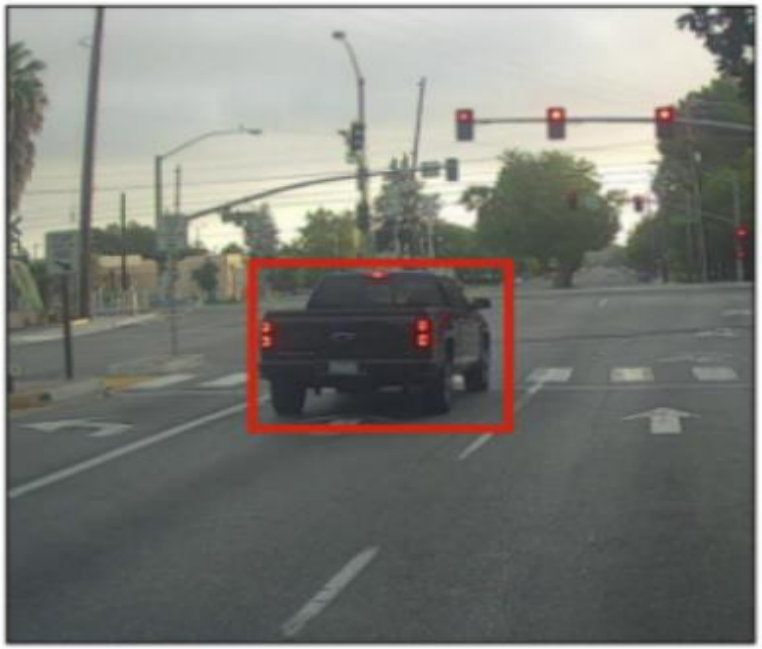
- 98개 bbox 각각이 가지고 있는 class specific confidence score에 대해서 각 20개의 클래스에 대해 신뢰도가 가장 높은 bbox만 남기고 나머지 bbox를 없애는 NMS (non-maximum suppression) 알고리즘을 이용하면, 객체에 대한 확률과 객체를 둘러싸고 있는 bbox 위치를 알아낼 수 있음.

before NMS



NMS

after NMS



※ YOLO 및 NMS 알고리즘 참고 사이트: https://goo.gl/byNZTn

deepsystems.io

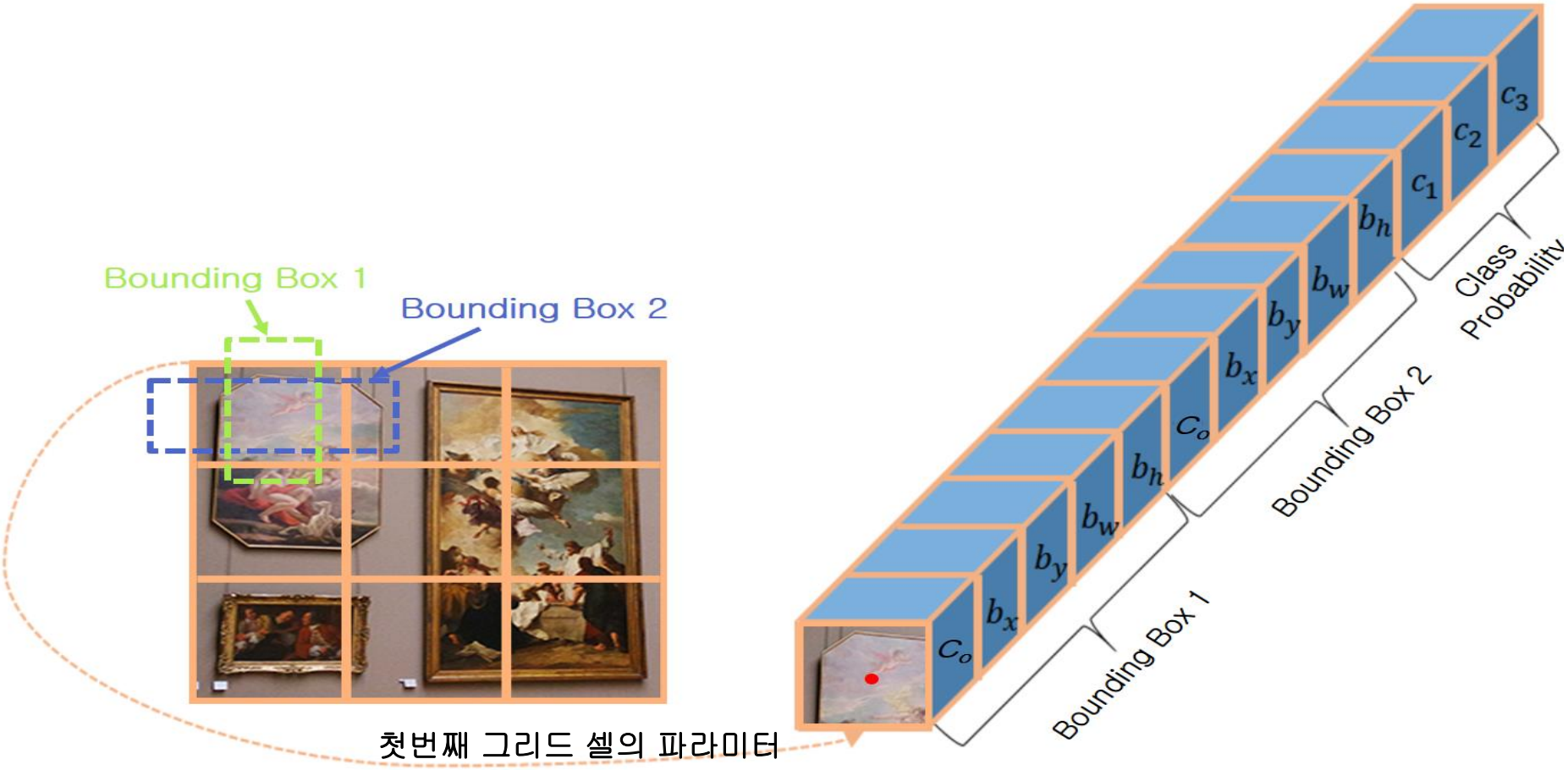
입력 이미지를 3 x 3 그리드로 나누고, 각각의 그리드 셀에는 2개의 Bounding Box 를 가진다. 이러한 그리드 셀은 3개의 물체를 분류(classification) 할 수 있다고 가정 할 때,

(1) 1개의 그리드 셀이 가지는 데이터 개수와 (2) 전체 그리드 셀이 가는지 데이터 개수를 계산 하시오

입력 이미지를 7 x 7 그리드로 나누고, 각각의 그리드 셀에는 2개의 Bounding Box 를 가진다. 이러한 그리드 셀은 20개의 물체를 분류(classification) 할 수 있다고 가정 할 때,

(1) 1개의 그리드 셀이 가지는 데이터 개수와 (2) 전체 그리드 셀이 가는지 데이터 개수를 계산 하시오

[예] 입력 이미지를 3 x 3 그리드로 나누면, 1개의 그리드 셀이 가지는 파라미터는 2 x 5 + 3 = 13 개 이고, 모든 그리드 셀이 가지는 총 데이터 개수는 3 x 3 x (2 x 5 + 3) = 117 개임



[예] 입력 이미지를 7 x 7 그리드로 나누고, 1개의 그리드 셀에는 2개의 바운딩 박스가 있으며, 이러한 각각의 그리드 셀에 포함되어 있는 물체를 클래스 확률 20개로 예측한다면,

1개의 그리드 셀이 가지는 파라미터는 2 x 5 + 20 = 30 개 이고, 모든 그리드 셀이 가지는 총 데이터 개수는 7 x 7 x (2 x 5 + 20) = 1470 개임

(위 조건은 YOLO 논문에서 제안한 스펙임)

Computer Vision Datasets

https://public.roboflow.com

Roboflow hosts free public computer vision datasets in many popular formats (including CreateML JSON, COCO JSON, Pascal VOC XML, YOLO v3, and Tensorflow TFRecords). For your convenience, we also have downsized and augmented versions available.

If you'd like us to host your dataset, please get in touch.

Open Poetry Vision Dataset

Object Detection (Bounding Box)

2000 images | 7 exports | Last updated 2 days ago

Hard Hat Workers Dataset

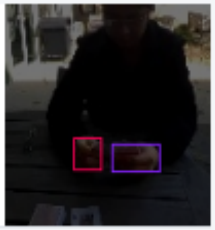
Object Detection (Bounding Box)

7035 images | 3 exports | Last updated 9 days ago

EgoHands Dataset

Object Detection (Bounding Box)

4800 images | 8 exports | Last updated 17 days ago

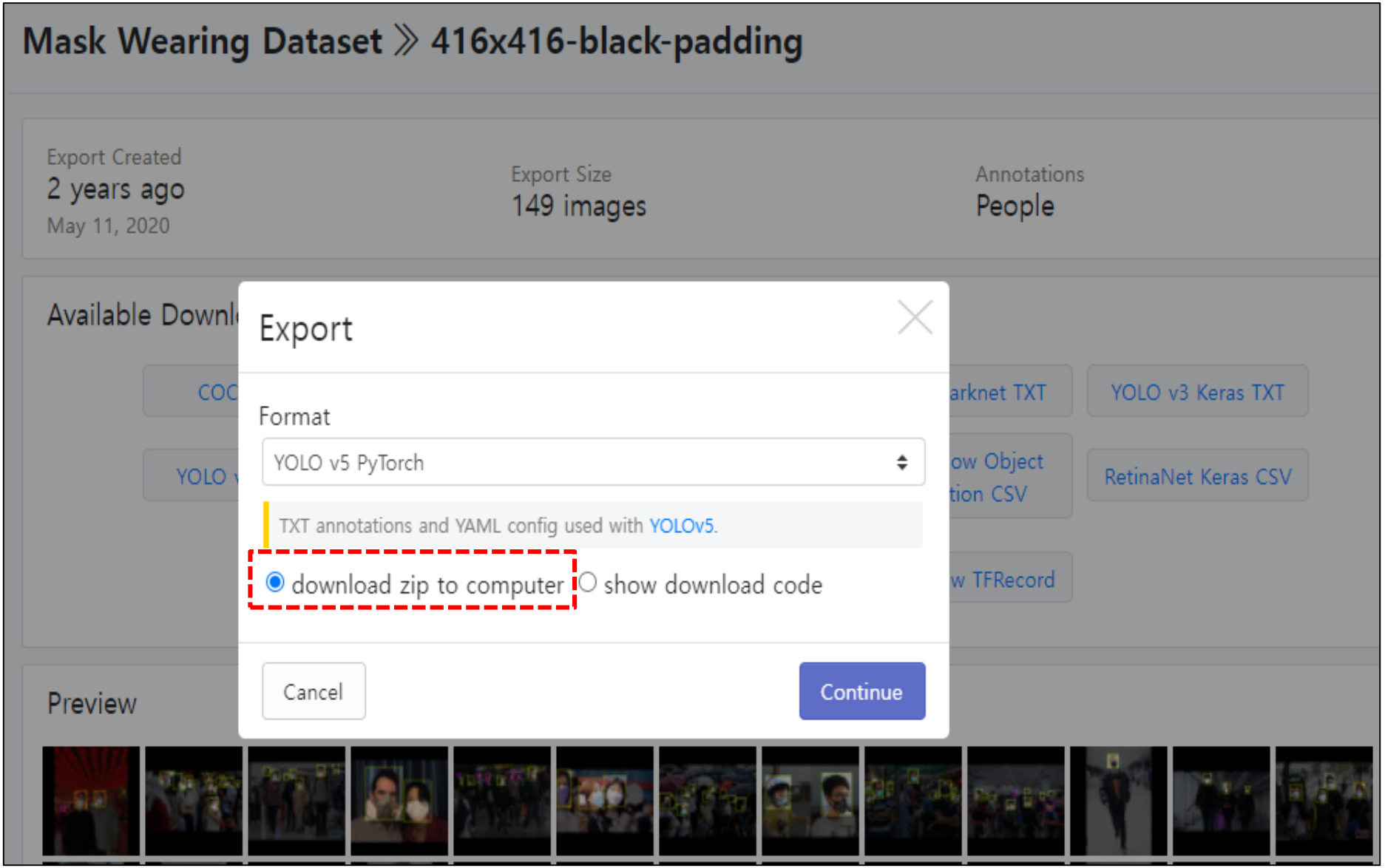


Mask Wearing Dataset

Object Detection (Bounding Box)

901 images | 11 exports | Last updated a month ago

download zip to computer 선택 후에, continue 통해서 Local PC 에 저장한 후, Mask_Data.zip 으로 이름을 변경하시오



[1] Mask_Data.zip 을 Colab 으로 업로드 한 후에,

/content/Mask_Data/train/images 파일 개수와 /content/Mask_Data/train/labels 파일 개수와 파일 이름을 비교하는 파이썬 코드,

/content/Mask_Data/valid/images 파일 개수와 /content/Mask_Data/valid/labels 파일 개수와 파일 이름을 비교하는 파이썬 코드를 작성하시오

[2]

os.remove(..) 함수를 이용해서 임의의 train 이미지 파일을 2개 / 임의의 train 정답 파일을 3개 삭제 한 후, 동일한 이름을 가지는 파일에 대해서 이미지는 /content/Mask_Data/new_train/images 에 저장하고, 정답은 /content/Mask_Data/new_train/labels 디렉토리에 각각 저장하는 파이썬 코드를 작성하시오,

마찬가지로 os.remove(..) 함수를 이용해서 임의의 valid 이미지 파일을 3개 / 임의의 valid 정답 파일을 1개 삭제 한 후, 동일한 이름을 가지는 파일에 대해서 이미지는 /content/Mask_Data/new_valid/images 에 저장하고, 정답은 /content/Mask_Data/new_valid/labels 디렉토리에 각각 저장하는 파이썬 코드를 작성하시오,

[3]

YOLOv5_File_Check_Example_2 코드에서 생성된 new_train, new_valid 데이터를 가지고서, YOLOv5 기반의 Detection 코드를 구현하시오

[YOLOv5 Exercise] Vehicles-OpenImages Dataset

https://public.roboflow.com/object-detection/vehicles-openimages

다음과 같은 roboflow dataset을 이용하여 Yolov5 학습한 후에,

[1] 주어진 test data 에 대해서 detect 실행 후 결과를 확인 하시오

[2] 본인이 인터넷등에서 다운로드 한 이미지를 대상으로 detect 실행 후 결과를 확인하시오

YOLOv5

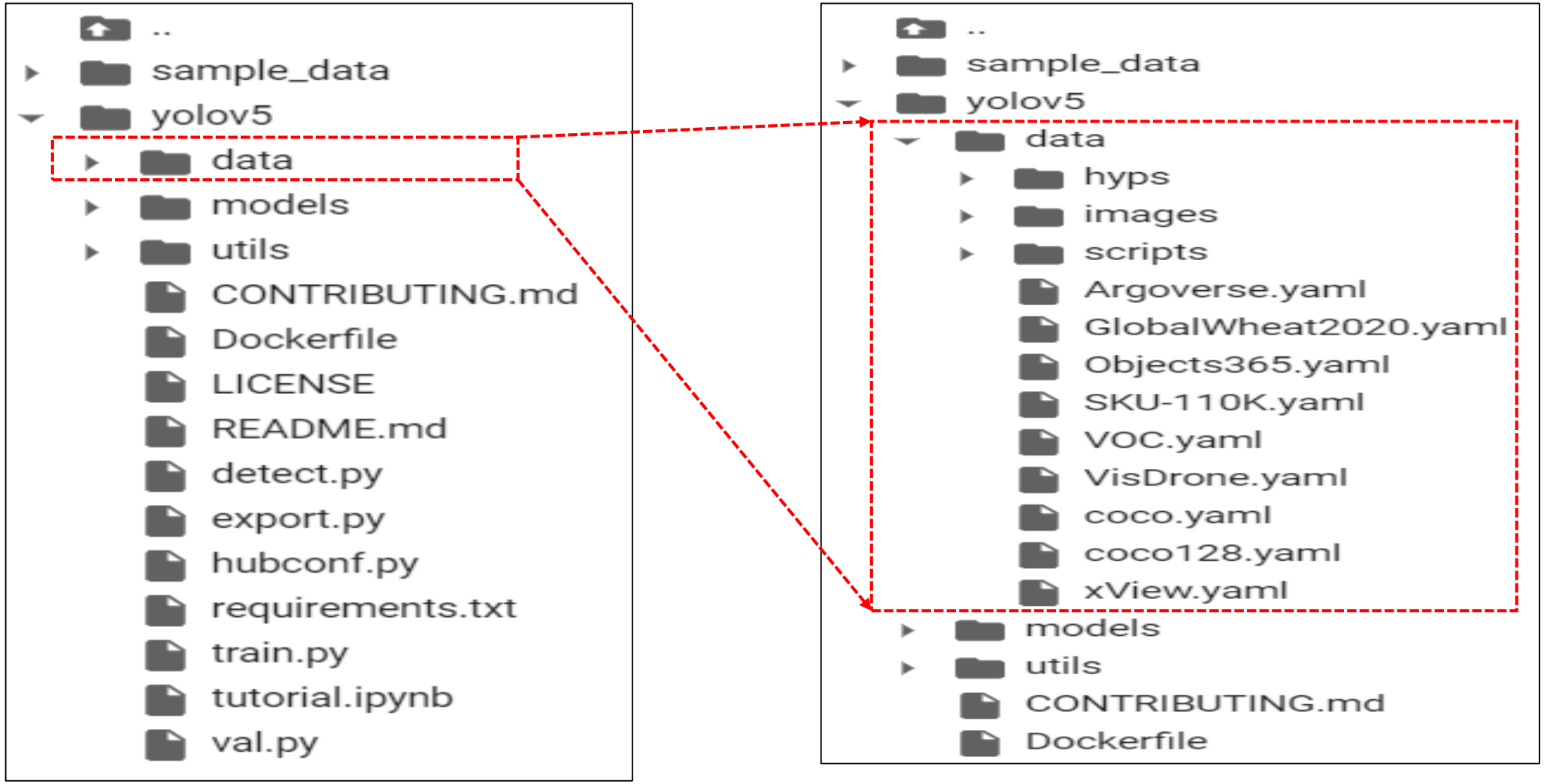
Basic Detection Process

(using pre-trained model)

박성호 (neowizard2018@gmail.com)

YOLOv5 git clone

```
%cd /content

!git clone https://github.com/ultralytics/yolov5.git
```



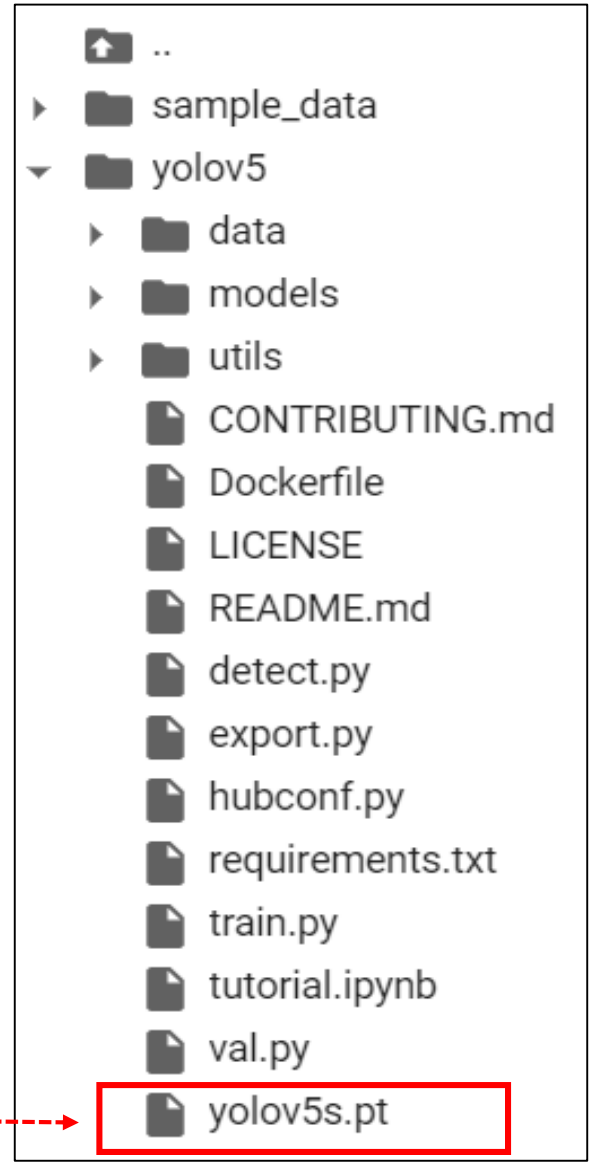
필수 라이브러리 설치

```
# 필수 라이브러리 설치

!pip install -r /content/yolov5/requirements.txt
```

```
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.7/dist-packages (from -r /conten
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.7/dist-packages (from -r /con
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Collecting PyYAML>=5.3.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2
     |████████████████████████████████| 596 kB 13.3 MB/s
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.7/dist-packages (from -r /content
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from -r /conte
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: tensorboard>=2.4.1 in /usr/local/lib/python3.7/dist-packages (from -r /conte
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/
Collecting thop
  Downloading thop-0.0.31.post2005241907-py3-none-any.whl (8.7 kB)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib
```

pre-trained model 다운로드 (option)

```
!wget -P /content/yolov5/  https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt

--2022-11-23 04:18:51--  https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/76
--2022-11-23 04:18:51--  https://objects.githubusercontent.com/github-production-release-asset-2e65
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443...
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to:  '/content/yolov5/yolov5s.pt'

yolov5s.pt          100%[===================>]  14.12M  --.-KB/s    in 0.1s

2022-11-23 04:18:52 (132 MB/s) -  '/content/yolov5/yolov5s.pt'  saved [14808437/14808437]
```



detect 실행 시에, 자동으로 다운로드 된 pre-trained model
(default=yolo5s.pt). COCO Dataset으로 이미 학습되어 있음

학습데이터 정보 (nc, names, path 등) 가지고 있는 yaml 확인 및 수정 (현재는 확인)

```
# yaml 파일 확인 및 필요시 설정 (데이터셋 위치를 알려주는 config file)

%cat /content/yolov5/data/coco128.yaml
```

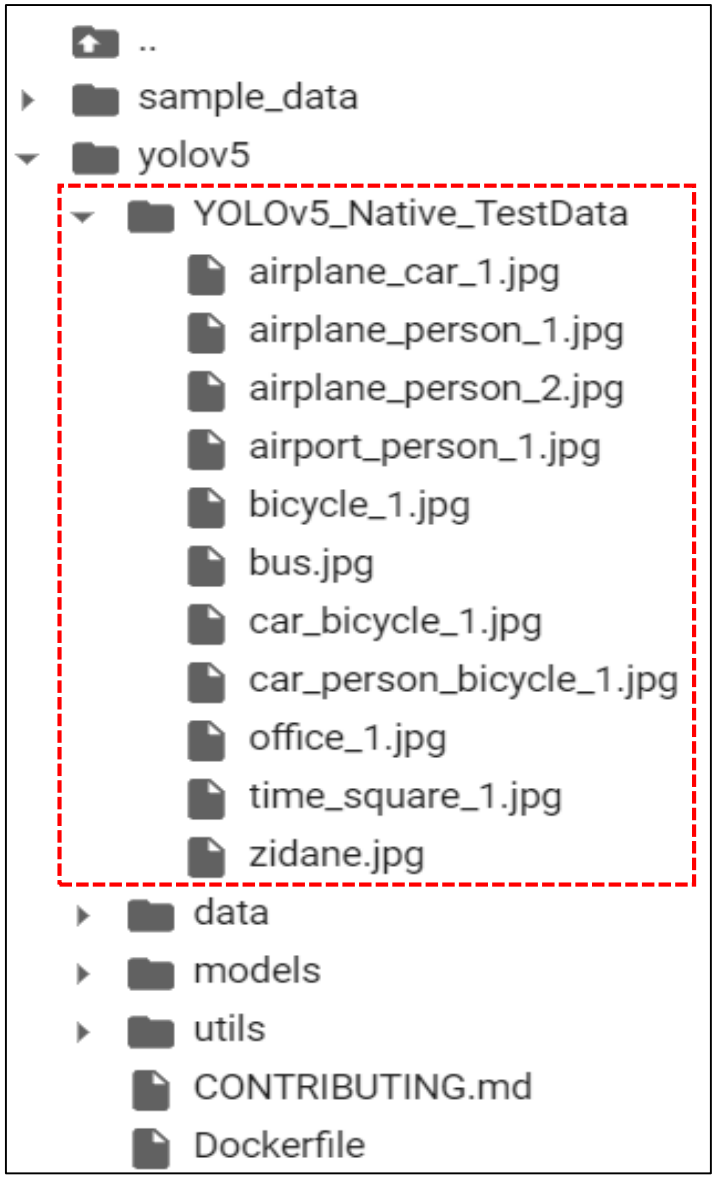
```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
# COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017)
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128  ← downloads here


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/coco128  # dataset root dir
train: images/train2017  # train images (relative to 'path') 128 images
val: images/train2017  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
nc: 80  # number of classes
names: ['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']  # class names


# Download script/URL (optional)
download: https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip
```

테스트 데이터 업로드 및 확인

```
import zipfile

with zipfile.ZipFile('/content/YOLOv5_Native_TestData.zip', 'r') as target_file:

    target_file.extractall('/content/yolov5/YOLOv5_Native_TestData/')
```

User-Defined Directory

```
# 테스트 이미지

import glob

test_image_list = glob.glob('/content/yolov5/YOLOv5_Native_TestData/*.jpg')

print(len(test_image_list))

test_image_list.sort()

for i in range(len(test_image_list)):

    print('i = ',i, test_image_list[i])
```



COCO Dataset으로 사전학습된 모델 이용하여 테스트 실행

```
# detect 실행

!python3 /content/yolov5/detect.py --weights /content/yolov5/yolov5s.pt  --source /content/yolov5/YOLOv5_Native_TestData/
```

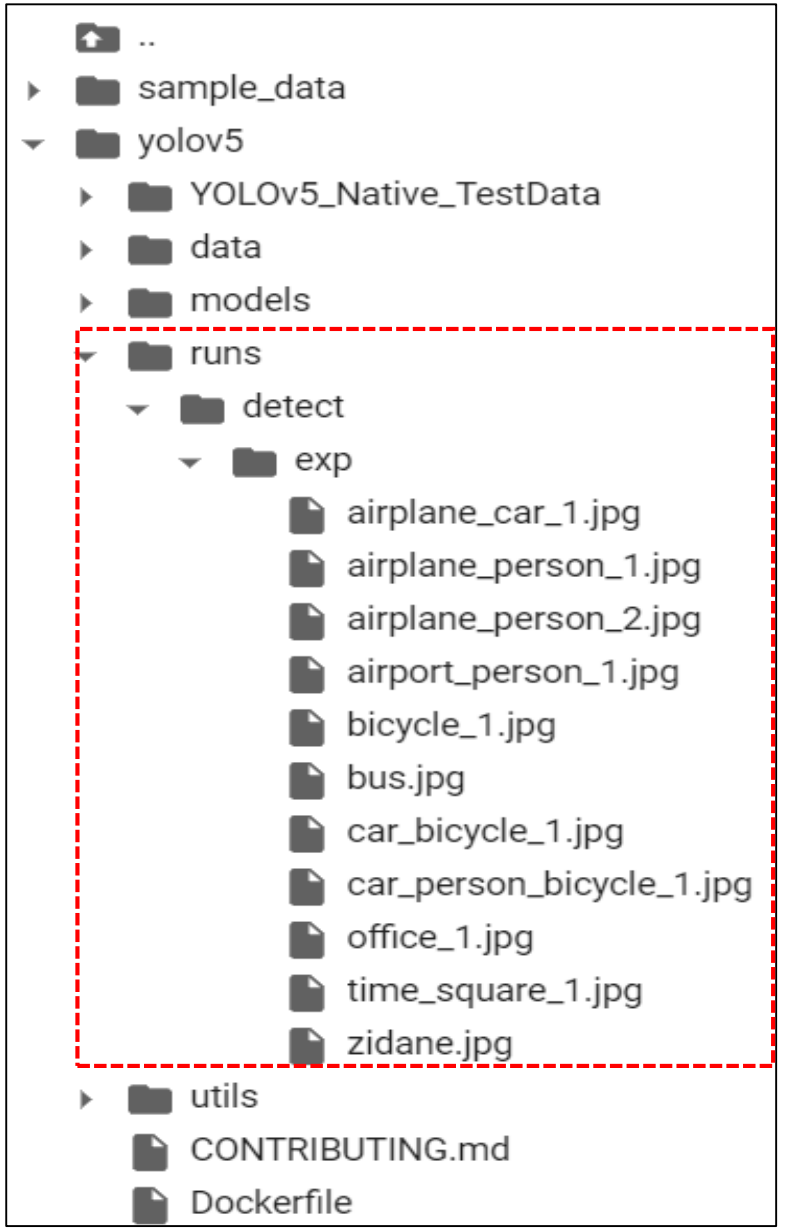
```
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
detect: weights=['/content/yolov5/yolov5s.pt'], source=/content/yolov5/YOLOv5_Native_TestData/, imgsz=[640, 640], conf_thres=0.25, iou
YOLOv5 🚀 v6.0-25-g15e8c4c torch 1.9.0+cu111 CUDA:0 (Tesla K80, 11441.1875MB)

Fusing layers...
Model Summary: 213 layers, 7225885 parameters, 0 gradients
image 1/11 /content/yolov5/YOLOv5_Native_TestData/airplane_car_1.jpg: 512x640 2 persons, 1 car, 3 airplanes, 7 trucks, Done. (0.035s)
image 2/11 /content/yolov5/YOLOv5_Native_TestData/airplane_person_1.jpg: 448x640 1 person, 1 airplane, Done. (0.031s)
image 3/11 /content/yolov5/YOLOv5_Native_TestData/airplane_person_2.jpg: 448x640 1 person, 6 airplanes, Done. (0.029s)
image 4/11 /content/yolov5/YOLOv5_Native_TestData/airport_person_1.jpg: 384x640 8 persons, 1 handbag, 1 suitcase, 1 cell phone, Done. (
image 5/11 /content/yolov5/YOLOv5_Native_TestData/bicycle_1.jpg: 416x640 9 persons, 2 bicycles, 1 car, Done. (0.031s)
image 6/11 /content/yolov5/YOLOv5_Native_TestData/bus.jpg: 640x480 4 persons, 1 bus, Done. (0.033s)
image 7/11 /content/yolov5/YOLOv5_Native_TestData/car_bicycle_1.jpg: 480x640 6 persons, 3 cars, 6 motorcycles, 2 umbrellas, Done. (0.03
image 8/11 /content/yolov5/YOLOv5_Native_TestData/car_person_bicycle_1.jpg: 352x640 2 persons, 1 bicycle, 7 cars, 1 bus, 1 traffic ligh
image 9/11 /content/yolov5/YOLOv5_Native_TestData/office_1.jpg: 320x640 1 person, 1 cup, 1 potted plant, 1 laptop, 1 vase, Done. (0.027
image 10/11 /content/yolov5/YOLOv5_Native_TestData/time_square_1.jpg: 448x640 18 persons, 3 cars, 1 truck, 2 traffic lights, Done. (0.0
image 11/11 /content/yolov5/YOLOv5_Native_TestData/zidane.jpg: 384x640 2 persons, 1 tie, Done. (0.030s)
Speed: 0.5ms pre-process, 30.7ms inference, 5.7ms NMS per image at shape (1, 3, 640, 640)
Results saved to yolov5/runs/detect/exp
```

result directory

결과 확인

```
import glob

detetced_image_list = glob.glob(('/content/yolov5/runs/detect/exp/*.jpg'))

detected_image_nums = len(detetced_image_list)

print(detected_image_nums)

print(detetced_image_list)
```

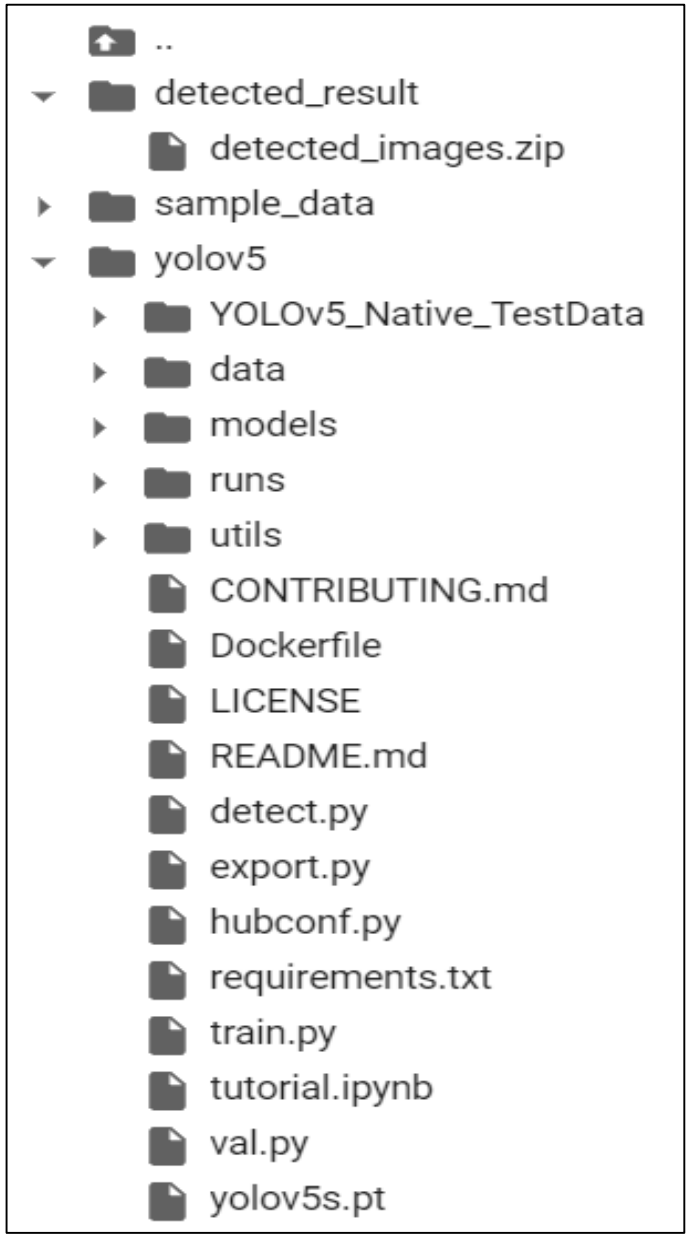
```
11
['/content/yolov5/runs/detect/exp/car_bicycle_1.jpg', '/content/yolov5/runs/detect/exp/bicycle_1.jpg', '/c
```



결과 다운로드

```
# 다운로드를 위한 inference image 압축

import zipfile
import os

if not os.path.exists('/content/detected_result/'):
    os.mkdir('/content/detected_result/')
    print('detected_result dir is created !!!')


with zipfile.ZipFile('/content/detected_result/detected_images.zip', 'w') as detected_images:
    for idx in range(detected_image_nums):
        detected_images.write(detetced_image_list[idx])
```

```
# 압축파일 다운로드

from google.colab import files

files.download('/content/detected_result/detected_images.zip')
```



[참고] detect..py 실행 옵션 (--source, --weights, --conf,...)

```
python detect.py --source 0   # webcam
                         file.jpg   # image
                         file.mp4   # video
                         path/   # directory
                         path/*.jpg   # glob
                         'https://youtu.be/NUsoVlDFqZg'   # YouTube video
                         'rtsp://example.com/media.mp4'   # RTSP, RTMP, HTTP stream
```

```
python detect.py --weights /content/yolo5/weights/best.pt  --conf 0.5  --source ....
```

YOLOv5

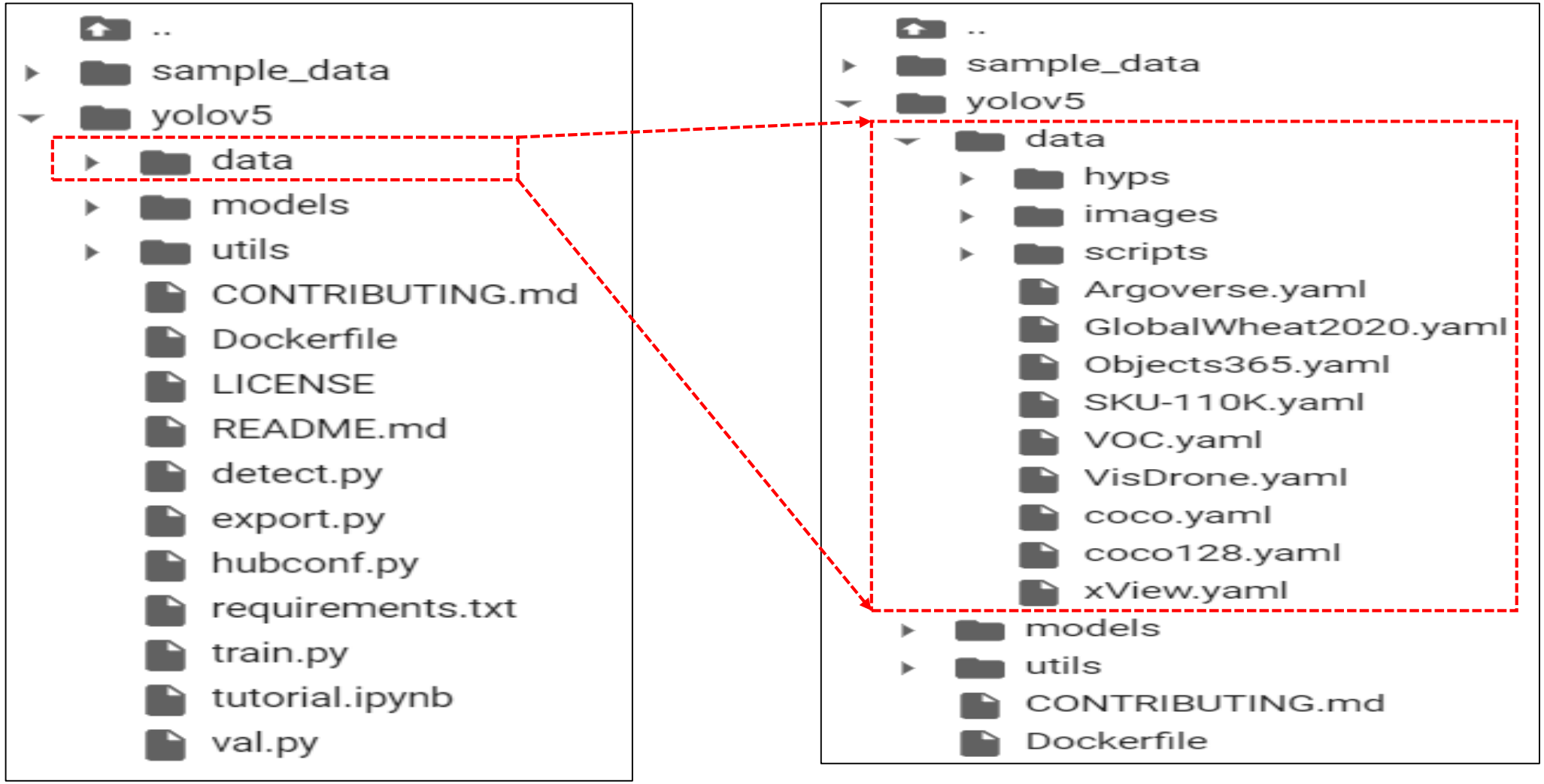
Basic Detection Process

(using roboflow)

박성호 (neowizard2018@gmail.com)

YOLOv5 git clone

```
%cd /content

!git clone https://github.com/ultralytics/yolov5.git
```



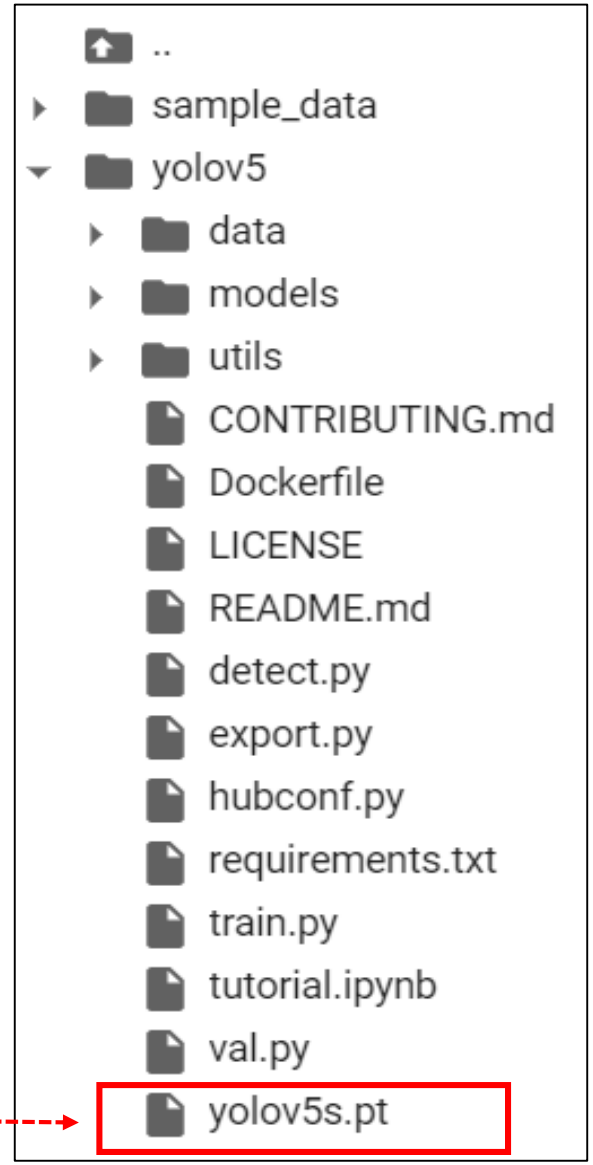
필수 라이브러리 설치

```
# 필수 라이브러리 설치

!pip install -r /content/yolov5/requirements.txt
```

```
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.7/dist-packages (from -r /conten
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.7/dist-packages (from -r /con
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Collecting PyYAML>=5.3.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2
     |████████████████████████████████| 596 kB 13.3 MB/s
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.7/dist-packages (from -r /content
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from -r /conte
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/yol
Requirement already satisfied: tensorboard>=2.4.1 in /usr/local/lib/python3.7/dist-packages (from -r /conte
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.7/dist-packages (from -r /content/yo
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/
Collecting thop
  Downloading thop-0.0.31.post2005241907-py3-none-any.whl (8.7 kB)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib
```

pre-trained model 다운로드 (option)

```
!wget -P /content/yolov5/  https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt

--2022-11-23 04:18:51--  https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/76
--2022-11-23 04:18:51--  https://objects.githubusercontent.com/github-production-release-asset-2e65
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443...
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to:  '/content/yolov5/yolov5s.pt'

yolov5s.pt          100%[===================>]  14.12M  --.-KB/s    in 0.1s

2022-11-23 04:18:52 (132 MB/s) -  '/content/yolov5/yolov5s.pt'  saved [14808437/14808437]
```



detect 실행 시에, 자동으로 다운로드 된 pre-trained model
(default=yolo5s.pt). COCO Dataset으로 이미 학습되어 있음

roboflow – computer vision datasets (https://public.roboflow.com/)

roboflow

Blog | Public Datasets | Model Zoo | Docs

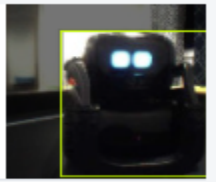
DATASET TYPE

- All Datasets 39
- Object Detection 35
- Classification 4

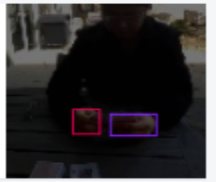
Computer Vision Datasets

Roboflow hosts free public computer vision datasets in many popular formats (including CreateML JSON, COCO JSON, Pascal VOC XML, YOLO v3, and Tensorflow TFRecords). For your convenience, we also have downsized and augmented versions available.

If you'd like us to host your dataset, please get in touch.

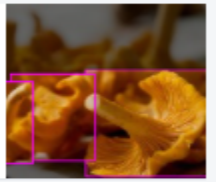


Anki Vector Robot Dataset Dataset
Object Detection (Bounding Box)
417 images | 8 exports | Last updated a day ago



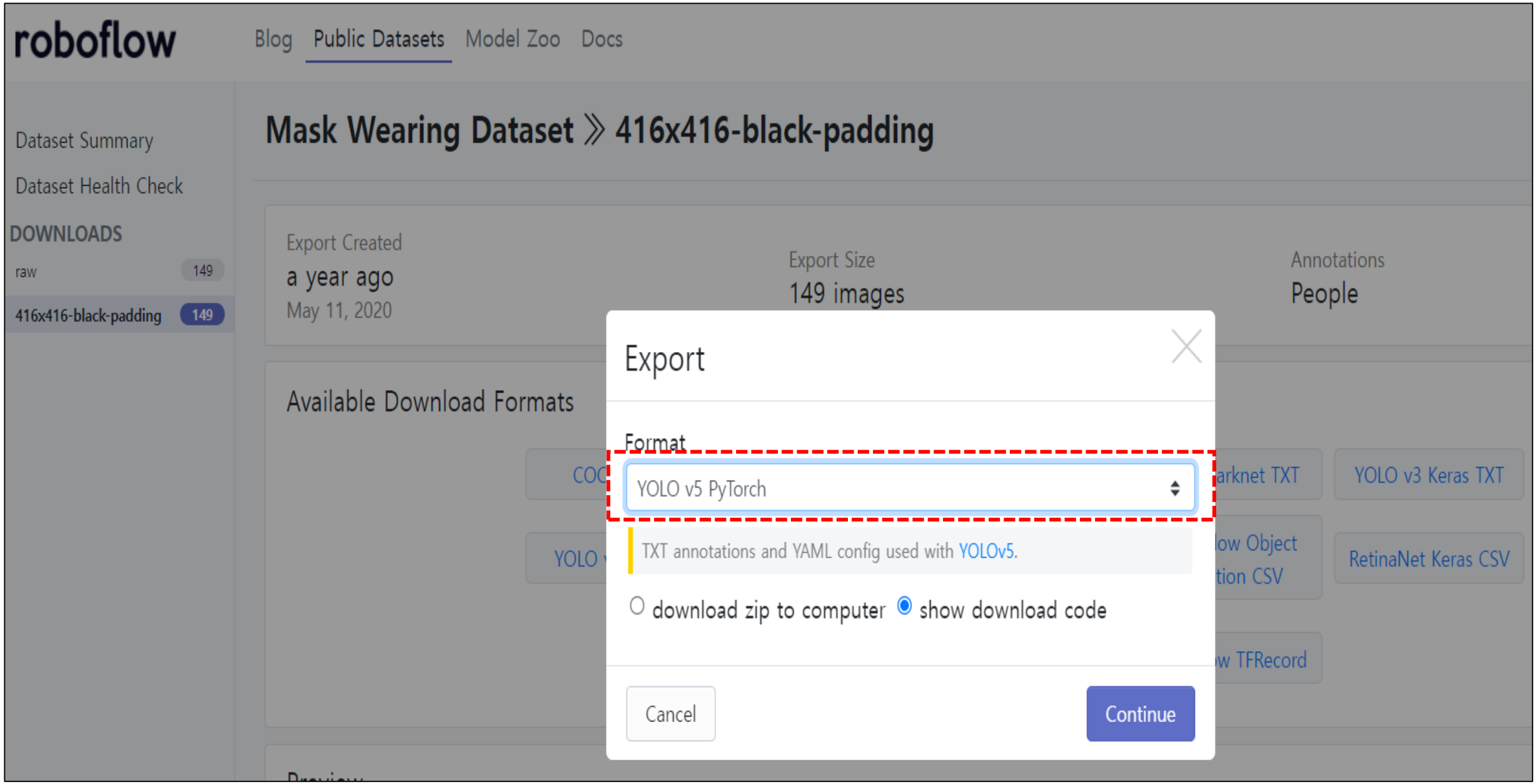
EgoHands Dataset
Object Detection (Bounding Box)
4800 images | 5 exports | Last updated 5 days ago

Microsoft COCO 2017 Dataset
Object Detection (Bounding Box)
121448 images | 9 exports | Last updated a month ago

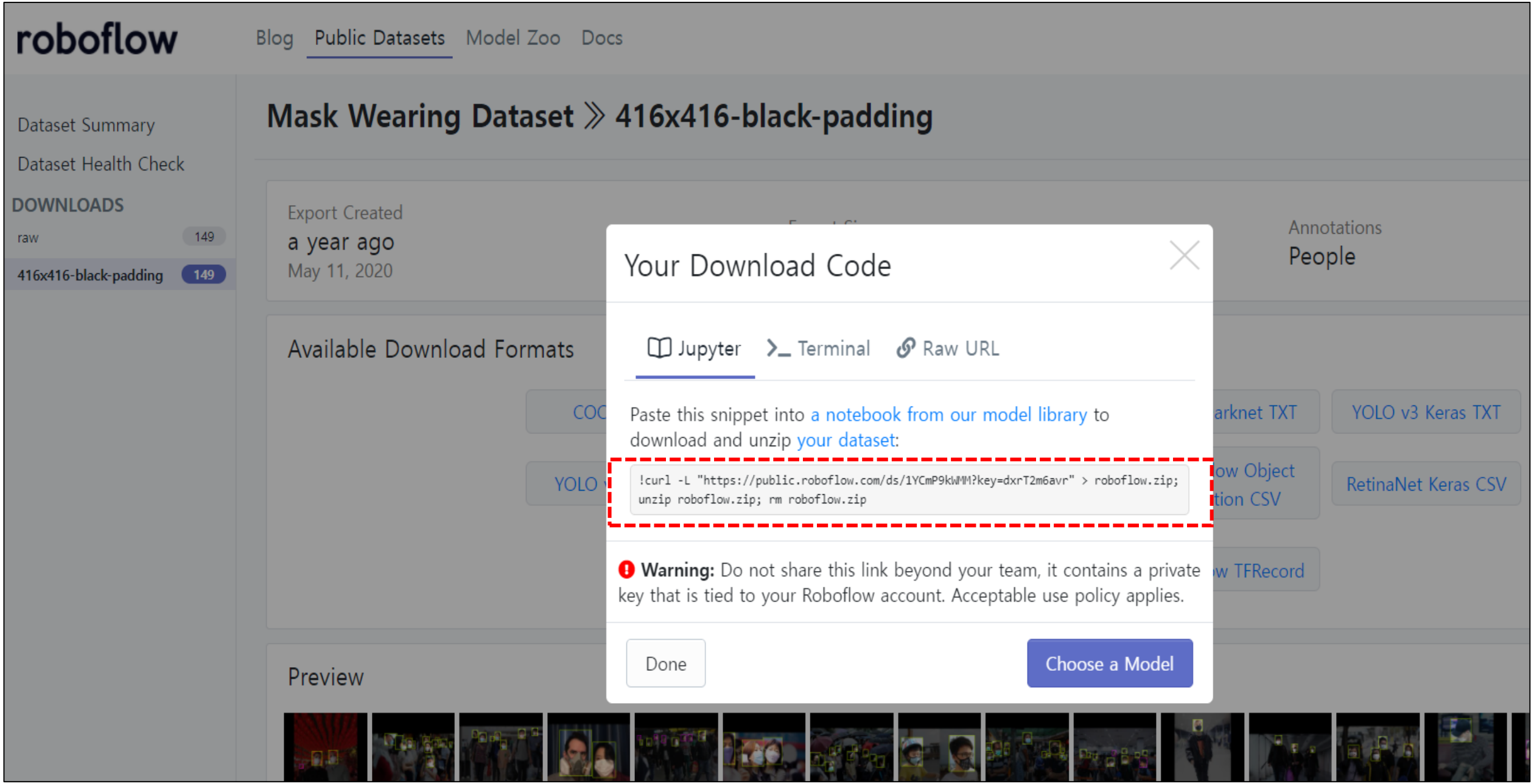


North American Mushrooms Dataset
Object Detection (Bounding Box)
51 images | 3 exports | Last updated 2 months ago

roboflow – computer vision datasets (https://public.roboflow.com/)



roboflow – computer vision datasets (https://public.roboflow.com/)



YOLO5 - Mask Wearing Dataset (custom data) 다운로드

Public Dataset : https://public.roboflow.com/object-detection/mask-wearing

```
[ ]  # custom 데이터 디렉토리 생성

     %mkdir /content/Mask_Data

     %cd /content/Mask_Data/
```

```
     /content/Mask_Data
```

```
[ ]  # custom 데이터 다운로드 (Mask Wearing Dataset)

     !curl -L "https://public.roboflow.com/ds/1YCmP9kWMM?key=dxrT2m6avr" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
       % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                      Dload  Upload   Total   Spent    Left  Speed
     100   896  100   896    0     0   1866      0 --:--:-- --:--:-- --:--:--  1862
     100 3791k  100 3791k    0     0  6276k      0 --:--:-- --:--:-- --:--:-- 6276k
     Archive:  roboflow.zip
      extracting: test/images/w1240-p16x9-0e48e0098f6e832f27d8b581b33bbc72b9967a63_jpg.rf.34ed1e8f70eebdabaf43ab9d40dc1c9b.jp
      extracting: test/images/RTX7CCFN_jpg.rf.66ed5c5054f30d933d19ab3d56ace004.jpg
      extracting: test/images/the-first-day-of-wuhan-s-closure-some-people-fled-some-panicked_jpg.rf.0302fefb0879eb37736a704c
      extracting: test/images/1288126-10255706714jpg_jpg.rf.95f7324cbfd48e0386e0660b5e932223.jpg
      extracting: test/images/126202-untitled-design-13_jpg.rf.56b50d413464989bb2232448a8fbb915.jpg
      extracting: test/images/r1p00017o8171png407_jpg.rf.6fd25b7219a249e97f54fcabf2b52726.jpg
```

yaml 파일 설정 (데이터셋 위치 알려주는 config file)

```
%cat /content/Mask_Data/data.yaml
```

```
train: ../train/images
val: ../valid/images

nc: 2
names: ['mask', 'no-mask']
```

```
# yaml 파일을 학습이 가능하도록 경로 설정.
# cat 결과 내용을 data['train'], data['val'], data['nc'], data['names'] 에 넣어주는데,
# 가장 중요한 부분은 데이터 경로 설정임.
# 즉 train: ../train/images 에서 .. 부분은 절대경로인 /content/Maks_Data 바꾸어 주는것이 중요
# https://stackoverflow.com/questions/69564817/typeerror-load-missing-1-required-positional-argument-loader-in-google-col
# yaml.load() => yaml.safe_load()

import yaml

with open('/content/Mask_Data/data.yaml', 'r') as f:
  data = yaml.safe_load(f)

print(data)

data['train'] = '/content/Mask_Data/train/images/'
data['val'] = '/content/Mask_Data/valid/images/'

data['nc'] = 2
data['names'] = ['mask','no-mask']

with open('/content/Mask_Data/data.yaml', 'w') as f:
  yaml.dump(data, f)

print(data)
```

학습 데이터 디렉토리는 다음과 같이 구성되야함

- data/train/images
- data/train/labels
- data/valid/images
- data/valid/labels

```
{'train': '../train/images', 'val': '../valid/images', 'nc': 2, 'names': ['mask', 'no-mask']}
{'train': '/content/Mask_Data/train/images/', 'val': '/content/Mask_Data/valid/images/', 'nc': 2, 'names': ['mask', 'no-mask']}
```

하이퍼 파라미터 설정 및 train

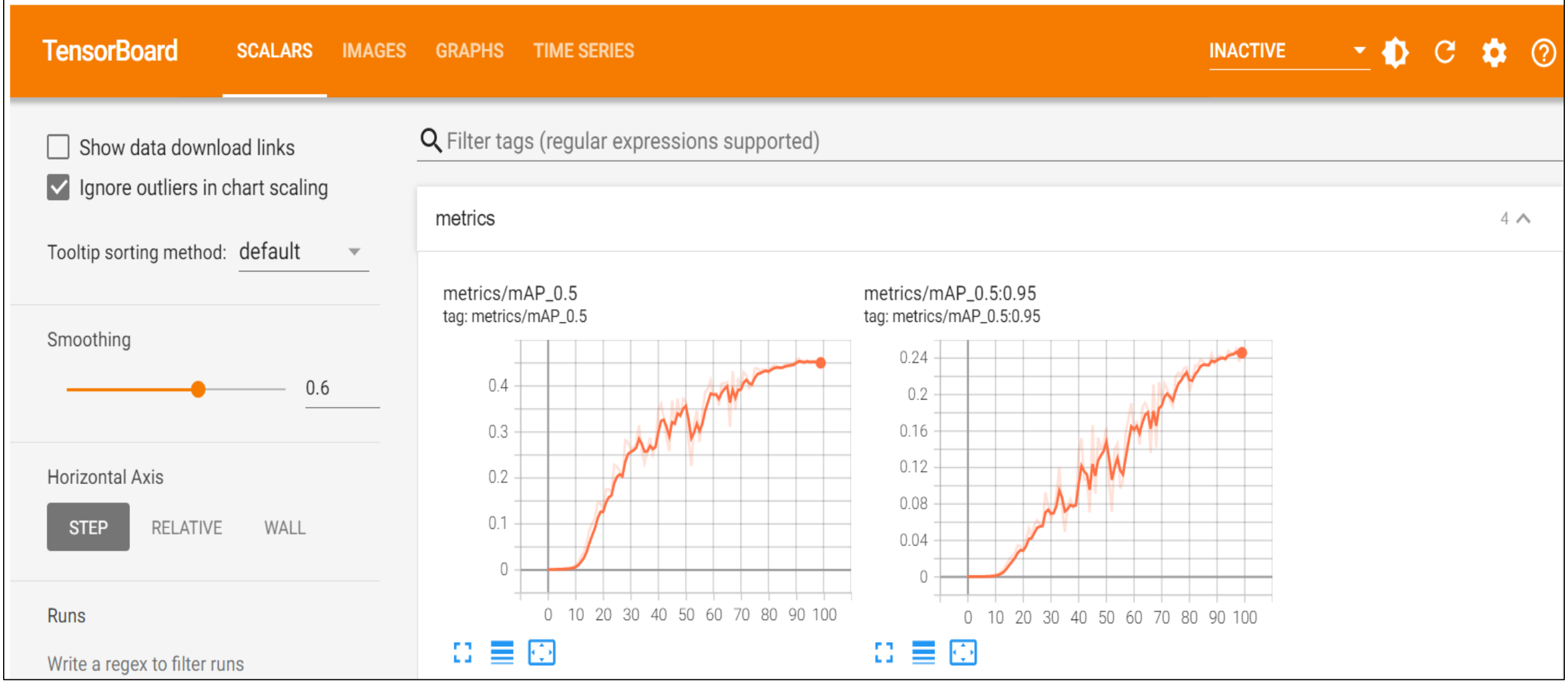
```
# 하이퍼 파라미터 설정

img_size = 416
bacth_size = 32
epochs = 100

data_path = '/content/mask_data/data.yaml'
yalm_path = '/content/yolov5/models/yolov5s.yaml'
weights_path = '/content/yolov5/yolov5s.pt'


# train

!python3 /content/yolov5/train.py --img {img_size} --batch {bacth_size} --epochs {epochs} --data {data_path} ₩
--cfg {yalm_path} --weights {weights_path}
```

```
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
train: weights=/content/yolov5/yolov5s.pt, cfg=/content/yolov5/models/yolov5s.yaml, data=/content/mask_data/data.yaml, hyp
github: skipping check (not a git repository), for updates see https://github.com/ultralytics/yolov5
YOLOv5 🚀 v6.0-30-gfee83c1 torch 1.9.0+cu111 CUDA:0 (Tesla K80, 11441.1875MB)

hyperparameters: lr0=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_b
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 🚀 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir yolov5/runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
```

텐서보드를 이용한 학습과정 출력

train 명령어 실행할 경우, 학습과정 저장 디렉토리가 출력되므로 --logdir 다음에 해당 디렉토리를 적어주면 된다

```
%load_ext tensorboard
%tensorboard --logdir /content/yolov5/runs/train/exp/
```



테스트 이미지 데이터 생성 및 확인

```python
# 테스트 이미지

from glob import glob

test_image_list = glob('/content/mask_data/test/images/*.jpg')

print(len(test_image_list))

test_image_list.sort()

for i in range(len(test_image_list)):

    print('i = ',i, test_image_list[i])
```

```
15
i =  0 /content/mask_data/test/images/0_Concern-In-China-As-Mystery-Virus-Spreads_jpg.rf.3135dfc5feab288d76a4ccfd22dfc5bf.jpg
i =  1 /content/mask_data/test/images/1224331650_g_400-w_g_jpg.rf.b816f49e2d84044fc997a8cbd55c347d.jpg
i =  2 /content/mask_data/test/images/126202-untitled-design-13_jpg.rf.56b50d413464989bb2232448a8fbb915.jpg
i =  3 /content/mask_data/test/images/1288126-10255706714jpg_jpg.rf.95f7324cbfd48e0386e0660b5e932223.jpg
i =  4 /content/mask_data/test/images/15391513324714o1n0r10n6_jpg.rf.a91fbc7be8a94ed3c48d2e4b35bd53bb.jpg
i =  5 /content/mask_data/test/images/15391513329330sooq10859_jpg.rf.89c8524c2096175fa2c728e5d73f1c28.jpg
i =  6 /content/mask_data/test/images/1579924271_jpg.rf.be5b27c2b2801bccc191e6dbd9bfccca.jpg
i =  7 /content/mask_data/test/images/RTX7CCFN_jpg.rf.66ed5c5054f30d933d19ab3d56ace004.jpg
i =  8 /content/mask_data/test/images/phplpE73q_jpg.rf.bd81cab9f8ff2674ce2e58278f7d37fa.jpg
i =  9 /content/mask_data/test/images/r1p00017o8171pnq407_jpg.rf.6fd25b7219a249e97f54fcabf2b52726.jpg
i =  10 /content/mask_data/test/images/shutterstock_1627199179_jpg.rf.8432d033a37b3d142ec4ffcede508c7d.jpg
i =  11 /content/mask_data/test/images/the-first-day-of-wuhan-s-closure-some-people-fled-some-panicked_jpg.rf.0302fefb0879eb37736a704ca5
i =  12 /content/mask_data/test/images/w1240-p16x9-0e48e0098f6e832f27d8b581b33bbc72b9967a63_jpg.rf.34ed1e8f70eebdabaf43ab9d40dc1c9b.jpg
i =  13 /content/mask_data/test/images/w1240-p16x9-2019-10-04t075956z_1862636027_rc15d4d49d00_rtrmadp_3_hongkong-protests_jpg.rf.061f2c7
i =  14 /content/mask_data/test/images/w1240-p16x9-fa978043deff83fed485af12d16e39c61398fc30_jpg.rf.185d01b7e55e049c6661b8ecd49679fc.jpg
```

YOLOv5 이용한 이미지내의 객체 검출 (Inference)

```
[ ]  # inference

     !python3 /content/yolov5/detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --img 416 --conf 0.5 --source /content/mask_data/test/images/
```

```
detect: weights=['/content/yolov5/runs/train/exp/weights/best.pt'], source=/content/mask_data/test/images/, imgsz=[416, 416], conf_thres=0.5, iou_thres=0
YOLOv5 🚀 v6.0-30-gfee83c1 torch 1.9.0+cu111 CUDA:0 (Tesla K80, 11441.1875MB)

Fusing layers...
Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
image 1/15 /content/mask_data/test/images/0_Concern-In-China-As-Mystery-Virus-Spreads_jpg.rf.3135dfc5feab288d76a4ccfd22dfc5bf.jpg: 416x416 2 masks, Done.
image 2/15 /content/mask_data/test/images/1224331650_g_400-w_g_jpg.rf.b816f49e2d84044fc997a8cbd55c347d.jpg: 416x416 4 masks, Done. (0.027s)
image 3/15 /content/mask_data/test/images/126202-untitled-design-13_jpg.rf.56b50d413464989bb2232448a8fbb915.jpg: 416x416 5 masks, Done. (0.028s)
image 4/15 /content/mask_data/test/images/1288126-10255706714jpg_jpg.rf.95f7324cbfd48e0386e0660b5e932223.jpg: 416x416 2 masks, Done. (0.028s)
image 5/15 /content/mask_data/test/images/15391513324714o1n0r10n6_jpg.rf.a91fbc7be8a94ed3c48d2e4b35bd53bb.jpg: 416x416 1 mask, Done. (0.028s)
image 6/15 /content/mask_data/test/images/15391513329330sooq10859_jpg.rf.89c8524c2096175fa2c728e5d73f1c28.jpg: 416x416 1 mask, Done. (0.028s)
image 7/15 /content/mask_data/test/images/1579924271_jpg.rf.be5b27c2b2801bccc191e6dbd9bfccca.jpg: 416x416 7 masks, Done. (0.028s)
image 8/15 /content/mask_data/test/images/RTX7CCFN_jpg.rf.66ed5c5054f30d933d19ab3d56ace004.jpg: 416x416 2 masks, Done. (0.028s)
image 9/15 /content/mask_data/test/images/phplpE73q_jpg.rf.bd81cab9f8ff2674ce2e58278f7d37fa.jpg: 416x416 29 masks, Done. (0.028s)
image 10/15 /content/mask_data/test/images/r1p00017o8171pnq407_jpg.rf.6fd25b7219a249e97f54fcabf2b52726.jpg: 416x416 1 mask, Done. (0.028s)
image 11/15 /content/mask_data/test/images/shutterstock_1627199179_jpg.rf.8432d033a37b3d142ec4ffcede508c7d.jpg: 416x416 5 masks, Done. (0.028s)
image 12/15 /content/mask_data/test/images/the-first-day-of-wuhan-s-closure-some-people-fled-some-panicked_jpg.rf.0302fefb0879eb37736a704ca5d070ff.jpg: 41
image 13/15 /content/mask_data/test/images/w1240-p16x9-0e48e0098f6e832f27d8b581b33bbc72b9967a63_jpg.rf.34ed1e8f70eebdabaf43ab9d40dc1c9b.jpg: 416x416 2 mas
image 14/15 /content/mask_data/test/images/w1240-p16x9-2019-10-04t075956z_1862636027_rc15d4d49d00_rtrmadp_3_hongkong-protests_jpg.rf.061f2c7f7d17a0b472510
image 15/15 /content/mask_data/test/images/w1240-p16x9-fa978043deff83fed485af12d16e39c61398fc30_jpg.rf.185d01b7e55e049c6661b8ecd49679fc.jpg: 416x416 1 mas
Speed: 0.4ms pre-process, 28.0ms inference, 1.5ms NMS per image at shape (1, 3, 416, 416)
Results saved to yolov5/runs/detect/exp
```

결과 확인 및 다운로드

```
[ ]  import glob

     detetced_image_list = glob.glob(('/content/yolov5/runs/detect/exp/*.jpg'))

     detected_image_nums = len(detetced_image_list)

     print(detected_image_nums)

     print(detetced_image_list)
```

```
15
['/content/yolov5/runs/detect/exp/phplpE73q_jpg.rf.bd81cab9f8ff2674ce2e58278f7d37fa.jpg', '/content/yolov5/runs/
```

```
import zipfile
import os

if not os.path.exists('/content/detected_result/'):
    os.mkdir('/content/detected_result/')
    print('detected_result dir is created !!!')


with zipfile.ZipFile('/content/detected_result/Mask_detected_images.zip', 'w') as detected_images:

    for idx in range(detected_image_nums):
        detected_images.write(detetced_image_list[idx])
```

```
# 파일 다운로드

import shutil
from google.colab import files

shutil.copy('/content/yolov5/runs/train/exp/weights/best.pt', '/content/yolov5/Mask_best.pt')

files.download('/content/detected_result/Mask_detected_images.zip')

files.download('/content/yolov5/Mask_best.pt')
```

YOLO 버전 비교

- YOLOv3, YOLOv4, YOLOv5 -

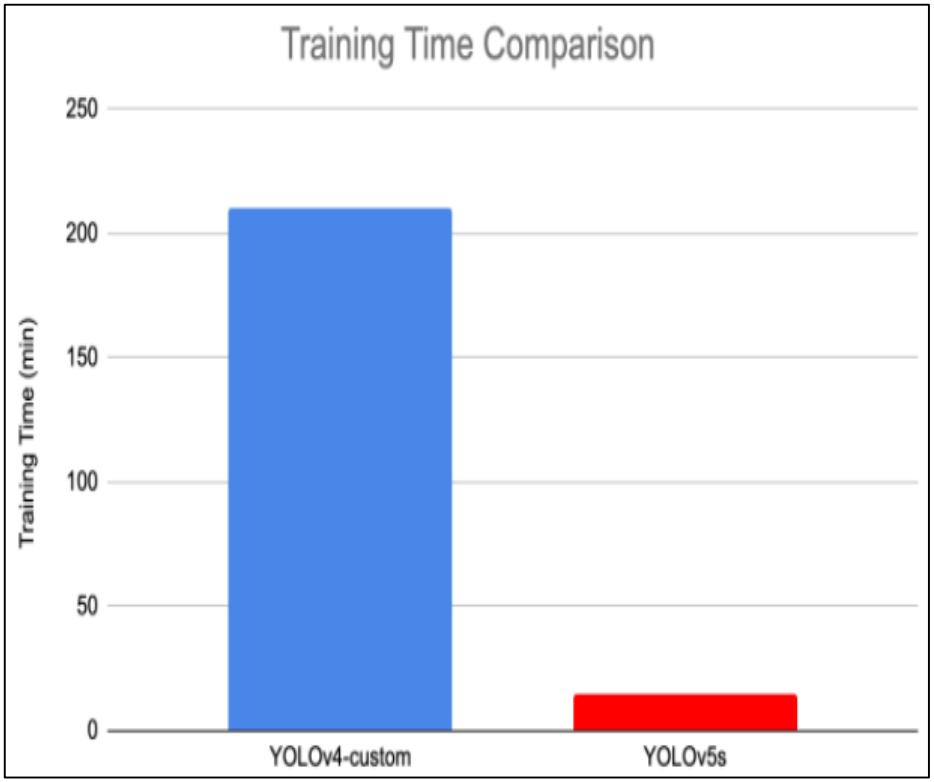
박성호 (neowizard2018@gmail.com)

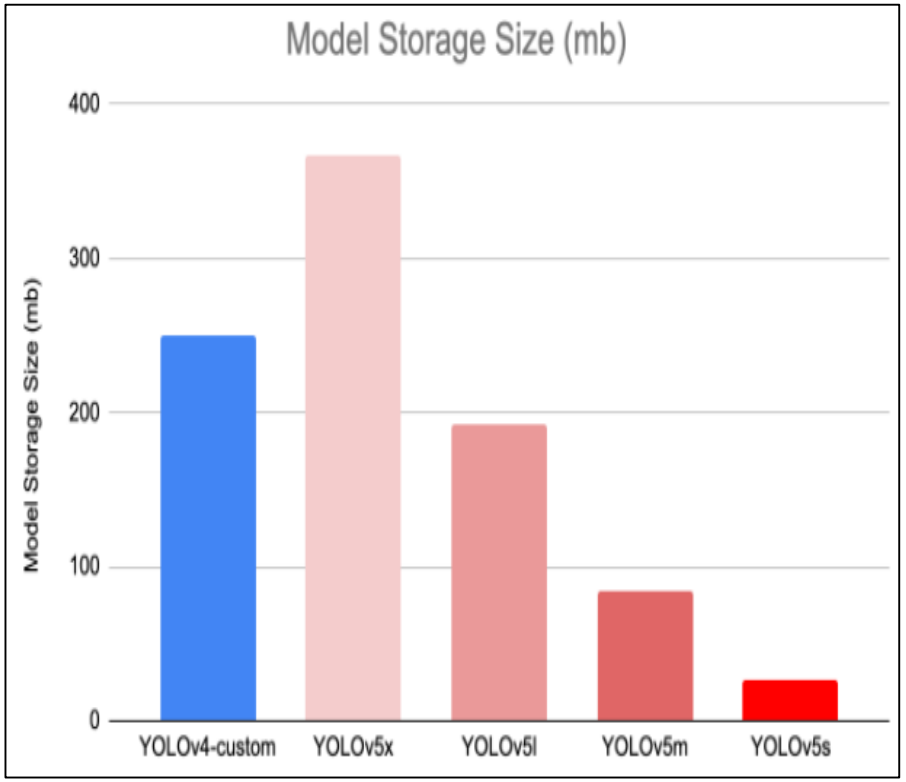
YOLOv3 YOLOv4 YOLOv5

- YOLOv3 (https://github.com/pjreddie/darknet)
  - ✓ 2018년 4월, YOLO를 만든 Josept Redmon 에 의해 발표됨. 백본 아키텍처 Darknet 기반으로 만들어졌음 (Darknet은 C, CUDA로 작성된 오픈소스 신경망 프레임워크)
  - ✓ 그러나 Josept Redmon은 YOLOv3 마지막으로 더이상 개발하지 않는다고 밝힘 (2020.2)

- YOLOv4 (https://github.com/AlexeyAB/darknet)
  - ✓ YOLOv3와 다른 개발자인 AlexeyBochkousky가 2020년 4월 발표함.
  - ✓ v3에 비해 AP, FPS가 각각 10%, 12% 증가함

- YOLOv5 (https://github.com/ultralytics/yolov5)
  - ✓ YOLOv3를 PyTorch로 implementation한 GlennJocher가 2020년 6월 발표함
  - ✓ v4에 비해 낮은 용량과 빠른 속도를 가지고 있음. Darknet이 아닌 PyTorch 구현이므로 이전 버전들과 다르다고 할 수 있으며, 처음 출시될 때 논문이 함께 출시되지 않았고, 이름을 YOLOv5로 하는 것에 대한 논란이 있었음





➢ YOLOv4는 v5에 비해 느리게 동작하지만 FPS성능을 최적화 할 수 있고, YOLOv5는 v4에 비해 더 쉽게 환경을 구성하고, 구현할 수 있다는 특징이 있음.

✓ 결과적으로 아키텍처 연구 및 분석 용도로는 Darknet의 YOLOv4가 좋을 것 같고, 실무적으로 빠르게 구현하고 deploy 하기에는 YOLOv5가 더 좋을 것이라고 (개인적으로) 생각됨

※ 이미지출처: https://yong0810.tistory.com/30

나만의 Object Detection System

[프로젝트] 자신만의 Object Detection System 만들기

[1] 먼저 Detection 하고자 하는 class 를 정의한다. 즉 Person, Car, Truck, Airplane, Bird 등을 detection 하고자 한다면, 주어진 class 개수는 5개임 (class 최소 개수는 4개 이상 정의)

[2] 크롤링이나 다운로드 등을 통해서 해당되는 class 이미지를 디렉토리에 저장한다. 학습의 정확도를 위해 최소 300 개의 이미지가 있어야 함 (1개의 이미지내에 다양한 class 가 있어도 300 장 정도 확보하는것이 필요함)

[3] labeling tool 을 이용해서 주어진 이미지에 대해 labeling 수행

[4] YOLOv5 이용해서 학습

[5] 본인이 촬영한 이미지나 새롭게 다운받은 이미지 등을 이용해서 테스트 수행

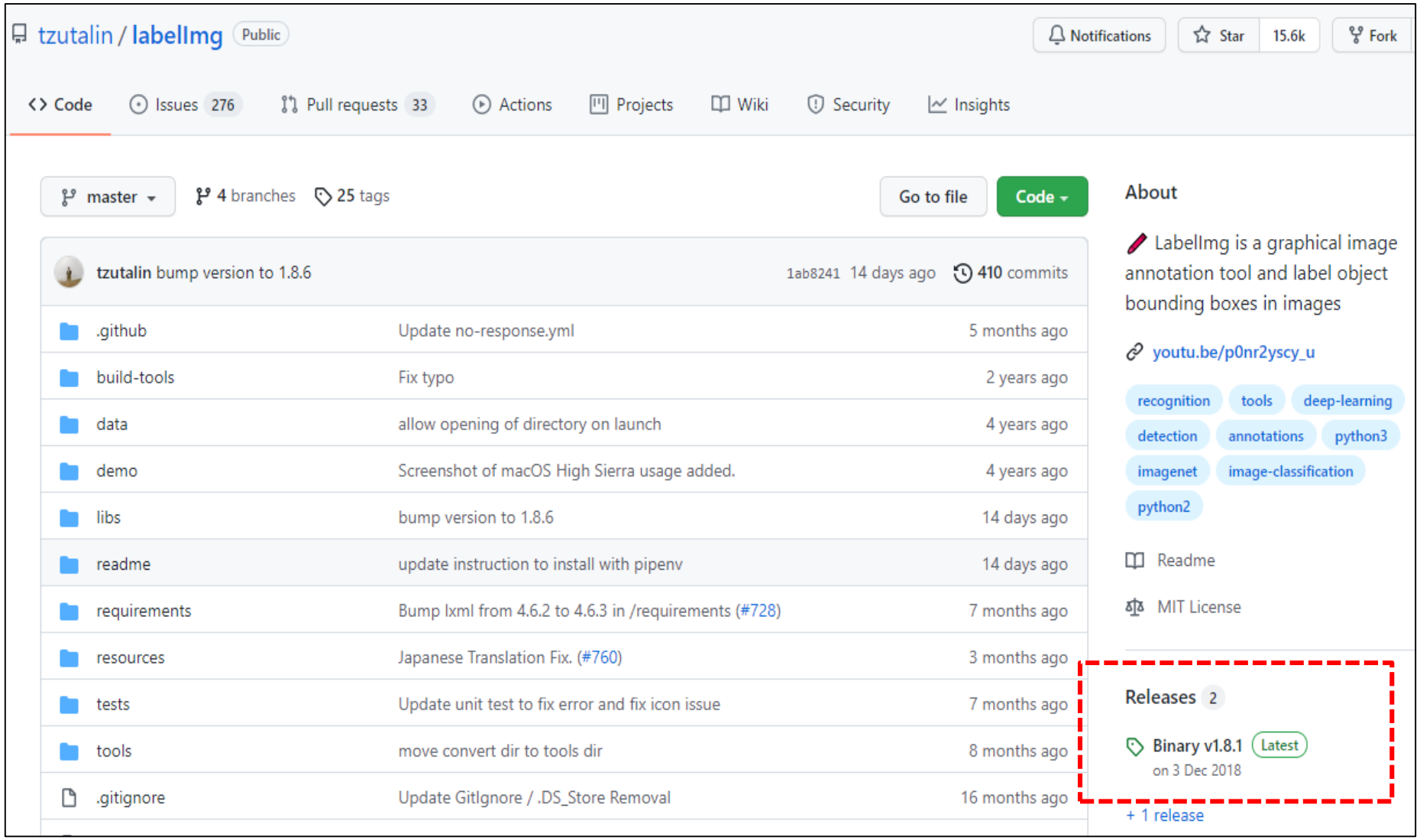
[6] 테스트 결과를 바탕으로 문제점, 이슈사항 및 해결 방안 등을 논의하시오

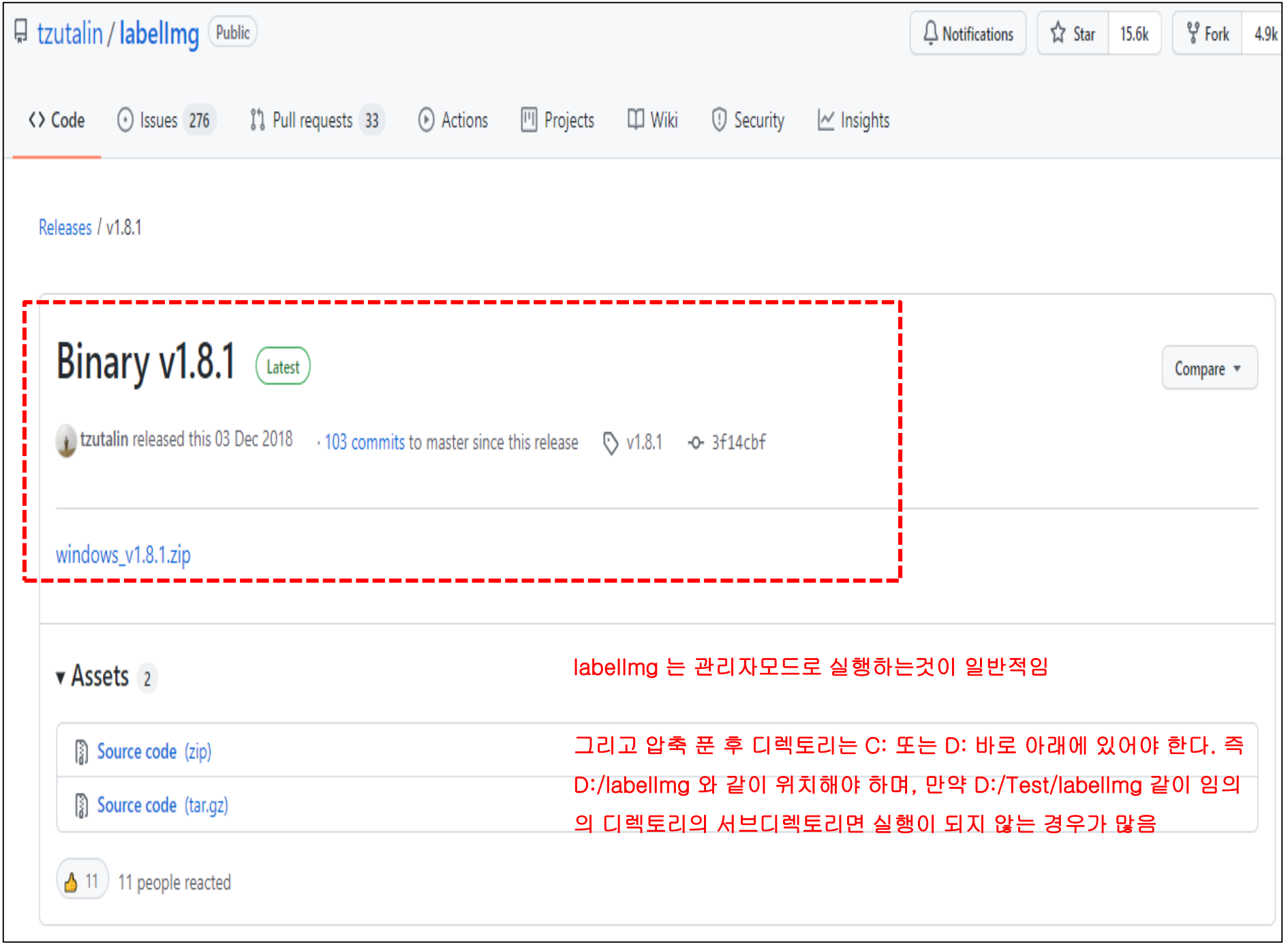
YOLOv5 Labeling using labelImg

박성호 (neowizard2018@gmail.com)

➢ 윈도우 OS에서 labelImg 설치 (https://github.com/tzutalin/labelImg)

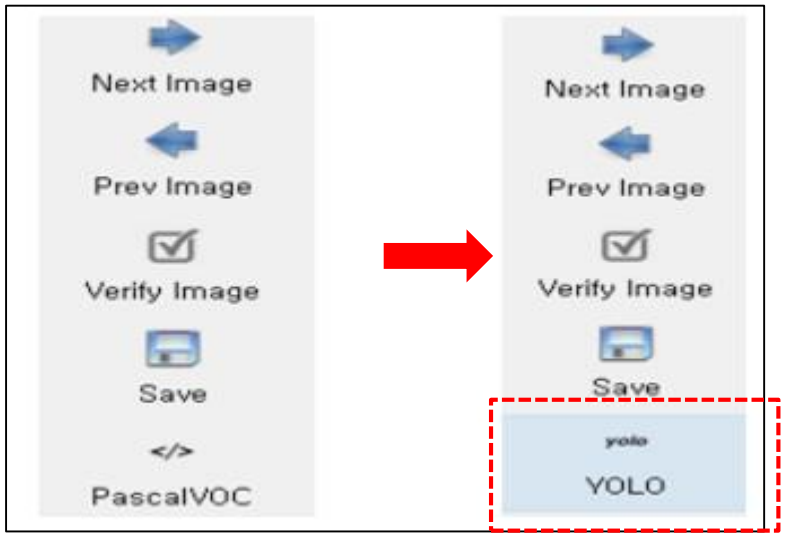
✓ pre-built 버전 다운로드 받은 후에, 압축을 푼 후 관리자 모드로 실행. 실행 error 발생시 C:/사용자/계정/.labelImgSettings.pkl 삭제후 실행하면 대부분 해결됨.

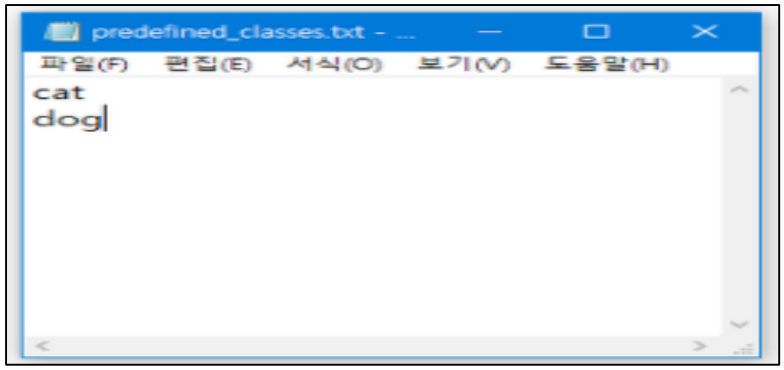


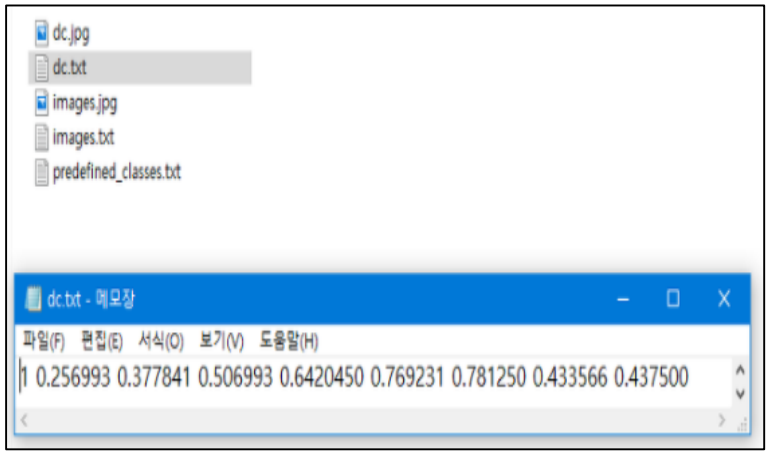

tzutalin / labelImg Public
Notifications
Star 15.6k
Fork 4.9k
Code
Issues 276
Pull requests 33
Actions
Projects
Wiki
Security
Insights
Releases / v1.8.1
Binary v1.8.1
Latest
Compare
tzutalin released this 03 Dec 2018
· 103 commits to master since this release
v1.8.1
3f14cbf
windows_v1.8.1.zip
Assets 2
Source code (zip)
Source code (tar.gz)
11 people reacted
labelImg 는 관리자모드로 실행하는것이 일반적임
그리고 압축 푼 후 디렉토리는 C: 또는 D: 바로 아래에 있어야 한다. 즉 D:/labelImg 와 같이 위치해야 하며, 만약 D:/Test/labelImg 같이 임의의 디렉토리의 서브디렉토리면 실행이 되지 않는 경우가 많음

➢ YOLOv5 학습 데이터 만들기

- ✓ 메뉴에서 형식을 YOLO로 반드시 바꾸어야함
- ✓ 학습할 데이터의 영문 클래스명을 predefined_classes.txt파일에 추가하고, Open Dir 버튼을 클릭해 데이터셋이 저장된 폴더 선택
- ✓ w키를 눌러 학습 이미지 범위 선택후, 해당 범위 클래스를 선택한 뒤 ctrl+s를 눌러 저장함
- ✓ a키는 이전 이미지, d키는 다음 이미지 선택
- ✓ 이렇게 저장하고 하고 나면 원본 이미지 파일과, 그 이미지 파일의 이름과 같은 txt 파일이 생성되는데, 이 txt 파일에는 YOLO 학습에 필요한 라벨 정보(클래스, 바운딩 박스 정보가 저장되어 있음) .
- ✓ 중요한건 학습할 이미지에 대해 하나하나 전부 수작업으로 해줘야함.







[1] labelImg 이용해서 3개의 class (예를 들면 치킨, 맥주, 사람)에 대해 5장의 이미지를 temp 디렉토리에 다운 받은 후에, **Open Dir 지정해서 labelImg 이용해서 라벨링 작업을 수행하는 예제**. 이때 라벨링된 파일 저장 디렉토리와 이때 생성되는 파일은 총 몇개인지 확인하시오

[2] labelImg 이용해서 3개의 class (예를 들면 치킨, 맥주, 사람)에 대해 5장의 이미지를 temp/images 디렉토리에 다운 받은 후에, **Open Dir 은 temp/images 지정하고 Change Save Dir 는 temp/labels 지정하고, labelImg 이용해서 라벨링 작업을 수행하는 예제**. 이때 Change Save Dir 지정후에 저장하면, 라벨링된 파일 저장 디렉토리와 이때 생성되는 파일은 총 몇개인지 확인하시오