

올로5

- 2020년 6월 출시했다. v4에 비해 낮은 용량과 빠른 속도를 가지고 있다. (성능은 비슷하다)
- YOLOv4와 같은 CSPNet 기반의 backbone을 설계하여 사용했다.
- YOLOv3를 PyTorch로 implementation한 GlennJocher가 발표했다.
- Darknet(C로 작성한 신경망 공개소스)이 아닌 PyTorch 구현이기 때문에, 이전 버전들과 다르다고 할 수 있다.
- 처음 출시될 때 논문이 함께 출시되지 않았고, 이름을 YOLOv5로 하는것에 대한 논란이 있다.

참고) 올로와 CNN성능비교

<https://koreascience.kr/article/JAKO202011161035249.pdf>

꼭 읽어보세요) 올로모델들 성능비교

<https://github.com/ultralytics/yolov5/releases>

Model	size (pixels)	accuracy top1	accuracy top5	Train time 90 epochs 4x A100 (hours)	Speed ONNX-CPU (ms)	Speed TensorRT-V100 (ms)	params (M)	FLOPs @224 (B)
YOLOv5n-cls	224	64.6	85.4	7:59	3.3	0.5	2.5	0.5
YOLOv5s-cls	224	71.5	90.2	8:09	6.6	0.6	5.4	1.4
YOLOv5m-cls	224	75.9	92.9	10:06	15.5	0.9	12.9	3.9
YOLOv5l-cls	224	78.0	94.0	11:56	26.9	1.4	26.5	8.5
YOLOv5x-cls	224	79.0	94.4	15:04	54.3	1.8	48.1	15.9
ResNet18	224	70.3	89.5	6:47	11.2	0.5	11.7	3.7
ResNet34	224	73.9	91.8	8:33	20.6	0.9	21.8	7.4
ResNet50	224	76.8	93.4	11:10	23.4	1.0	25.6	8.5
ResNet101	224	78.5	94.3	17:10	42.1	1.9	44.5	15.9
EfficientNet_b0	224	75.1	92.4	13:03	12.5	1.3	5.3	1.0
EfficientNet_b1	224	76.4	93.2	17:04	14.9	1.6	7.8	1.5
EfficientNet_b2	224	76.6	93.4	17:10	15.9	1.6	9.1	1.7
EfficientNet_b3	224	77.7	94.0	19:19	18.9	1.9	12.2	2.4

올로는 기본적으로 아래의 COCO dataset(머신러닝을 하기 위해 만들어진 수 많은 데이터셋 중에 하나)으로 학습되어 있음. 학습된 모델은 `model = torch.hub.load('ultralytics/yolov5', 'yolov5s')` 로 불러와서 사용가능함.

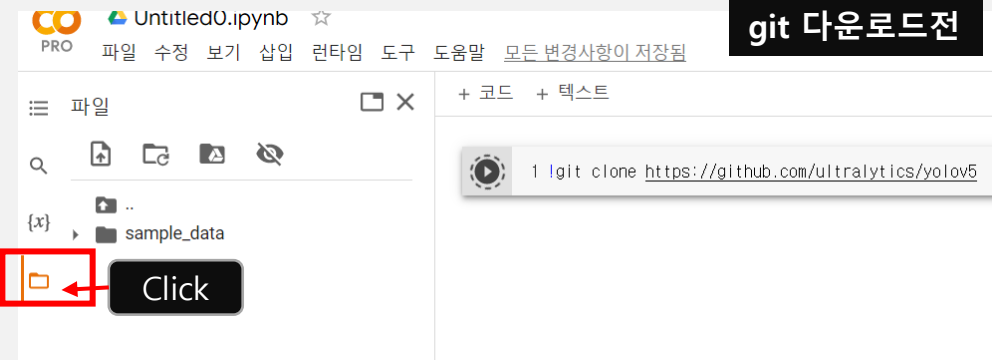
* 2017 coco dataset 기준 객체

"person, bicycle, car, motorbike, aeroplane, 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, fribee, 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, sofa, pottedplant, bed, diningtable, toilet, tvmonitor, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush"

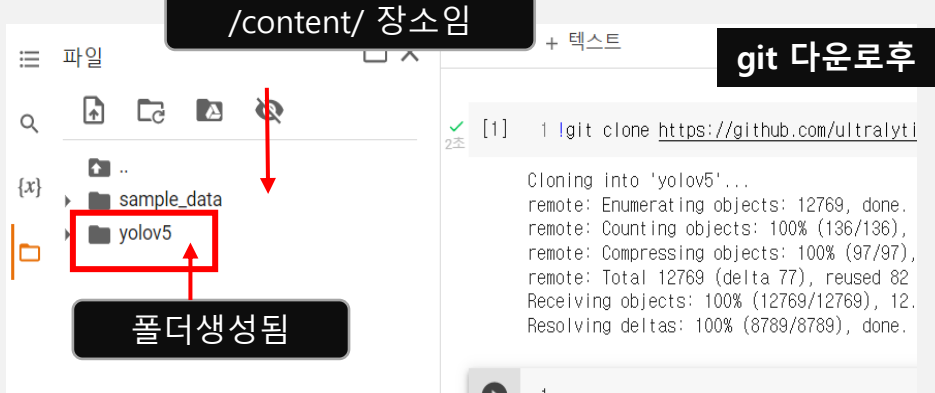
올로모델을 이용한 객체인식

1. 코랩실행, GPU선택 (런타임-유형변경)
2. 올로5의 코드가 있는 git 사이트 폴더 모두 복사함. (다운로드후 압축풀고 코랩에 올려놓는거와 같음)
별도의 드라이브 마운트 과정이 없었기 때문에 /content/yolov5에 생성됨 (코랩종료후 다시 실행하면 없어지는 폴더임)
만약 이 폴더코드가 늘상필요하면 드라이브마운트해서 폴더이동하고 그 폴더에 생성하면됨.(드라이브용량을 차지함, 또는 변경되는 업데이트 자료가 아니어서 문제가 될수도 있음)

Code !git clone <https://github.com/ultralytics/yolov5>



git 다운로드전



/content/ 장소임

git 다운로드후

폴더생성됨

3. 설치된 yolov5폴더로 이동하고 폴더경로 확인함

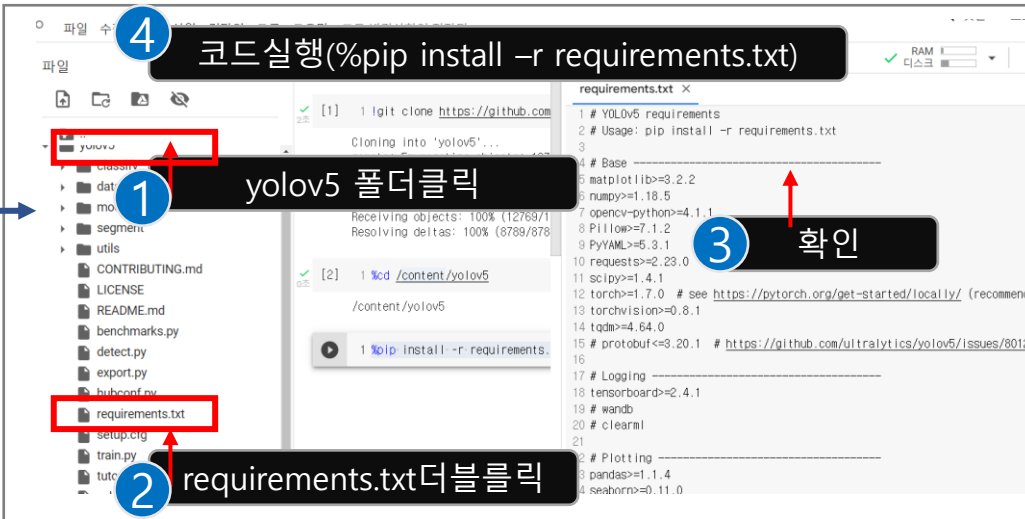
Code %cd /content/yolov5

Code %pwd

4. 작업에 필요한 패키지를 설치함

Code %pip install -r requirements.txt

파이썬설치목록의 모음집, 파일명은 달라도 되지만
일반적으로 requirements.txt 로 지정함
우측작업에서 (1,2,3은 필수 아님), 코드만 작성해도 됨



4 코드실행(%pip install -r requirements.txt)

1 yolov5 폴더클릭

2 requirements.txt더블클릭

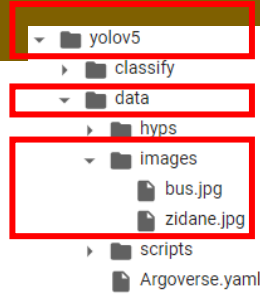
3 확인

윌로모델을 이용한 객체인식

5. yolov5 폴더에서 제공하는 detect.py 를 실행함
사용자 이미지로 하고자 하면 필요한 이미지를 드라이브에 드래그함.

```
Code !python detect.py --source content/yolov5/data/images
```

yolov5를 git에서 복사하면 자동으로 설치되는 data/images폴더의 모든 이미지를 사용하겠다는 의미임
!python detect.py --source content/yolov5/data/images/bus.jpg 로 해도 됨
위의 작업은 detect.py 에서 미리세팅된 weights값이나, img사이즈, conf(신뢰도)값이 자동으로 설정되며
아래 코드와 같이 사용자가 직접 지정해도 됨



```
Code !python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source /content/sample.jpg
```

사용자가 지정하는 옵션값

6. 출력되는 결과물 확인

```
1 !python detect.py --source /content/yolov5/data/images/

detect: weights=yolov5s.pt, source=/content/yolov5/data/images/, data=data/coco128.yaml, imgsz=[640, 640]
YOLOv5 v6.2-169-g959a466 Python-3.7.14 torch-1.12.1+cu113 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, 10.0ms
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties, 10.0ms
Speed: 0.4ms pre-process, 10.0ms inference, 1.1ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp7
```

폴더명은 detect까지는 동일하나 exp 뒤의 번호는 계속 다름

