YOLO version3 구현

2020 FALL ESC CV3조 이승준 박상재 임선우 김관석 강은서

목차

1. Task 소개

Task 정의 : YOLO version3으로 이미지 + 영상 내의 물체를 `실시간 Classification + Detection`

✓논문은 YOLO v2, YOLO 9000을 다뤘으나 구현이 원활한 YOLO v3을 다룸

준비물

1) data/coco.names

✓coco 데이터셋은 Object detection / Segmentation 등을 위한 데이터셋.

✓330만여개의 이미지, 150만개 가량 object instance, **80개 레이블**

✓레이블은 person , bicycle, car, motorbike ... , hair drier, toothbrush까지 80가지이며 data/coco.names는 label을 단지 나열한 파일

2) dog-cycle-car.png : Test image, 416×416 sized

+ darknet.py + cfg + weights + detect.py

2. YOLO v3 간단소개

2.1 darknet.py : YOLO v3의 레이어를 만든 파일!

✓YOLO v3은 C언어로 작성되었으며 object detection DNN 오픈소스인 darknet에 속해 있다.

✓Alexnet, YOLO v4, Resnet 등등 다양한 신경망을 담음.

✓여기서 darknet.py파일은 YOLO v3의 레이어를 만든 중요파일!

2.2 cfg : YOLO v3의 레이어에 대한 설명자료!

✓CFG 확장자는 configuration (설정)을 의미하며 응용프로그램의 설정 값을 담는다 한다.

✓YOLO v3 cfg파일에서는 모든 layer의 filter수, size, stride, padding, activation (linear / leaky ReLU 등)의 정보를 담음.

✓주의) 기본적으로 이미지 크기가 608×608로 설정되어 있어 Test image의 사이즈에 맞게 416×416로 조정 필요!

✓마지막 `yolo` 레이어의 설정은 바로 아래 이미지

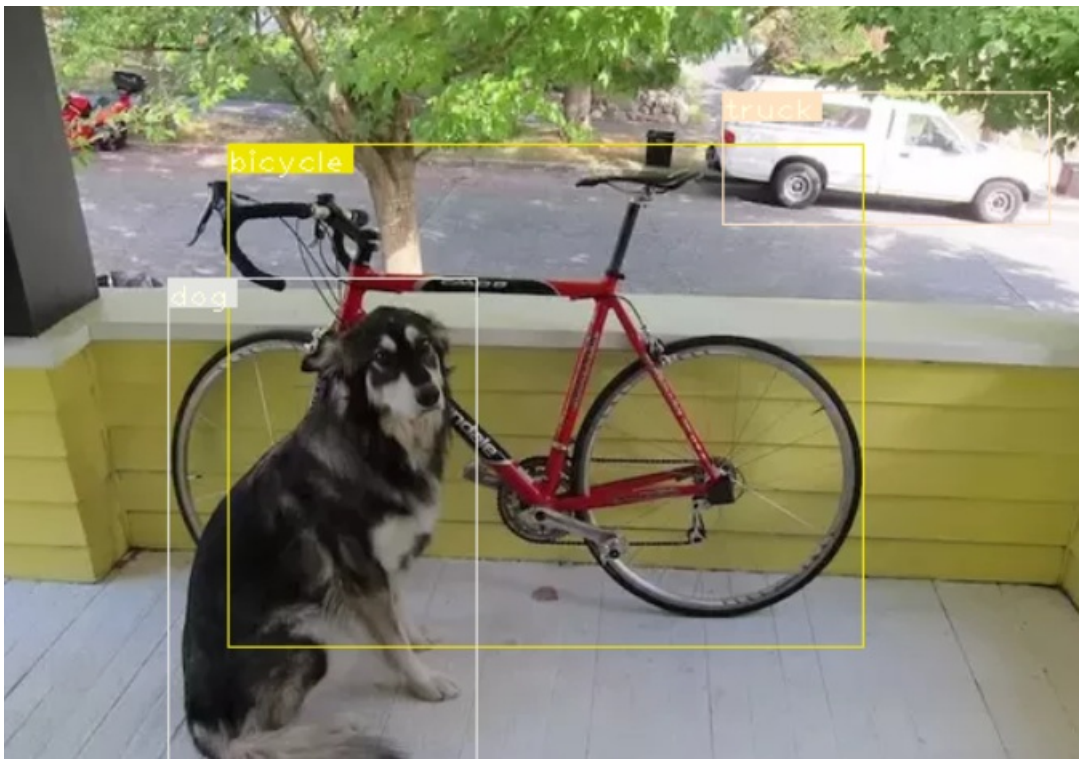
```
[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .5
truth_thresh = 1
random=1
```

2.3 weights : 이미 학습된 weights 불러오기!

✓Weight를 훈련하는 것은 지나치게 오랜 시간이 걸려 이미 학습된 weights를 불러와서 사용

2.4 detect.py : detector를 실행하기 위한 파일!

✓darknet.py를 import하여 쓰며 이미지에 다음과 같이 boxing된 채로 저장시킨다.



3. 코드 구현

3.1 darknet.py 코드순서

> YOLO v3의 레이어를 만든 파일

1st) package import : `In[12]`

2nd) 이미지를 읽어오고 사이즈를 $608 \times 608 \rightarrow 416 \times 416$로 바꾼다. normalize, float로 변환, 변수로 바꾸기까지. : In[18]

3rd) configuration 파일 parsing 후 모든 layer = block의 리스트를 dictionary로 저장한다. : `In[19]`

4th) layer 설정 : `In[20]`

5th) 3번째 과정에서 만든 `parse_cfg` 함수를 통해서 만든 dictionary를 이용하여 PyTorch 모듈을 만든다 (convolutional layer / upsample / route / detection layer) : `In[21]~ [22]`

6th) darknet에서 forward학습하는 함수정의, 학습된 weight를 가져오는 함수정의 : `In [23]`

7th) CFG파일 내용을 가져온 후 darknet을 통해 모든 레이어를 거쳐 [1,10647,85]짜리 tensor를 출력함 : `In[30]`

코드 이해에 도움 되는 링크

3.2 detect.py 코드순서

> ipynb로 수행이 불가능하며 (argparse 함수가 command prompt에서 시행하는 것이라 한다) shell로 실행하여 이미지에 위와 같이 boxing된 채로 저장시킨다.

1st) arg_parse 함수 `In [19]` : argparse는 파이썬 library에 이미 내재된 함수라 import하면 된다. 함수에서 필요한 parameter를 쉽게 작성하기 위한 library라고 한다. (Compsci 지식이 필요한 듯함)

image파일의 경로, detection결과 저장경로, input image의 해상도 등의 argument를 계속 add_argument를 통해 추가한다.

2nd) 네트워크 로딩 : `In [20] ~[23]`

작업 디렉토리를 만들며 + 클래스 파일 loading + network초기화에 weight loading

3rd) Input 이미지를 불러들인다 : `In [24] ~ im_dim_list = torch.FloatTensor(im_dim_list).repeat(1,2)` 까지

4th) Batch 설정 : `leftover = 0 ~ line 130`

5th) Batch마다 detection 작업을 진행 : `write = 0 ~ if CUDA : torch.cuda.synchronize`

detection에 걸리는 시간 (즉각적이길 원함), detected object가 무엇인지를 loop을 통하여 print한다.

6th) Bounding box를 이미지에 덧입히고 저장하기 : `try : output ~ end = time.time()` 까지.

7th) 코드 실행은 끝났으며 마지막으로 detection에 걸리는 시간 등 학습과정을 summary를 printing : `print("SUMMARY")` 부터.

4. 시각화 예시

> dog-cycle-car.png 뿐만 아니라 다른 영상 / 이미지에도 YOLO v3 적용

4.1 테넷 예고편

: 영상이 너무 길어 프로그램이 튕기는 문제

4.2 ESC 서울 MT

: 사진. 꼬깔콘을 썼는데도 사람이라고 옳게 판단

4.3 연세-네이버클라우드 데이터사이언스 교육과정 □COMING SOON!□

위의 링크는 원본 유튜브 링크이며 아래 링크가 3조에서 만든 YOLO v3영상.

연세-네이버클라우드 YOLOv3

: 영상, 역시 성능이 매우 훌륭하다.

Citing :

1) https://blog.paperspace.com/how-to-implement-a-yolo-v3-object-detector-from-scratch-in-pytorch-part-2/

2) https://darkpgmr.tistory.com/170

3) https://jayeon8282.tistory.com/6

4) https://devjin-blog.com/yolo-part5/