3. keras에 있는 retinanet 을 보고 분석

4. RetinaNet의 구조  
keras code 에서는 Backbone 으로 ResNet50을 이용   
특정 layer에 대해 fpn을 만들고 class subnet과 boxsubnet을 추출  
retinanet => object detector

5. 논문과는 다른 learning rate를 초기화 해주고 있다.

아마 학습 반복횟수가 달라서 그런게 아닐까?

6. label encoder의 역할은 raw label을 bounding box와 클래스로 구성된 training에 할수 있는 형태로 바꿔주는 역할

아래는 cocoset을 직접열어보진 않고 검색해서 나온 내용인데

다음 내용이 json형태로 저장 되 있습니다.

전에 했었던 classification 같은 부분은 사진 하나에 class 하나 였다면

지금은 사진 내의 class 마다 정보를 저장합니다.

8. anchor box는 object를 예측위한 bounding box인데 물체의 center와 anchorbox의 width height 보고 object scale을 예측. Retinanet은 9개 anchorbox를 사용.(다른 비율 3, 다른 크기 3

9. compute dims : feature pyramid의 각 레벨의 모든 anchorbox를 compute  
get\_anchors : feature pyramid 에서 anchor boxes를 발생

논문에서 level p3~p7 총 5개의 feature map을 만듬

10. 실제로 사용하는 함수는 encode\_batch  
gtboxes = ground truth box

11. Anchorbox, groundtruth box 와 비교해서 IOU 0.4~0.5 무시  
0.5 이상은 positive - foreground 0.4 이하는 negative – background

12. IOU 계산하는 코드

13. IOU 계산에서 사용 된 전환 함수들

원래는 box x, y width height 형태인데 이를 왼쪽 아래 모서리 오른쪽 위 모서리로 전환하여 쉽게 계산

14. retinanet + focal loss에 해당하는 핵심부분

15. resnet을 불러오는부분인데

(get\_layer로 저 부분을 불러오는게 이해가 조금 안됨)

16. box subnet : smooth l1 loss

Class subnet : focal loss

(그냥 더해버림?)

17. RetinaNet 같은 경우에는 오른쪽의 사진과 같이 init 에서 Resnet을 Feature pyramid를 따오고

각 feature pyramid 에서 class subnet과 boxsubnet을 만든후 concat 하는것을 알 수 있습니다.

18. backbone에서 특정 layer를 뽑아내서 upsampling 을 통해 합쳐줍니다.

19. 그림에 그려진것과 같이 4번 convolution을 진행   
outputfilter도 그림과 같이

20 setting up callbacks

21 load coco2017

22 data efficiently -> preprocess sample

Padded\_batch를 이용해서 다른 개수의 물체나 다른 차원을 패딩  
LabelEndoer sample을 만듬

23.

Jitter : jitter can be [0-1] and used to crop images during training for data augumentation. The larger the value of jitter, the more invariance would neural network to change of size and aspect ratio of the objects

24. Dataset 개수를 제한

25. decodes predictions of the RetinaNet model