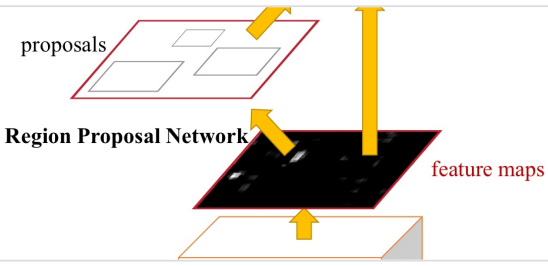
Faster R-CNN

Faster R-CNN 논문 리뷰 및 코드 구현

논문 제목: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks SPPnet과 Fast R-CNN은 region proposal computation에 많은 시간을 사용한다. 이는 병목현상을 일으킨다. Faster R-CNN은 이런 문제를 해결하기 위해 detection network와 convolutional features를 공유하는 Region Proposal Network(RPN)을 제안

https://velog.io/@skhim520/Faster-R-CNN-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-%EB%B0%8F-%EC%BD%94%EB%93%9C-%EA%B5%AC%ED%98%84



Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

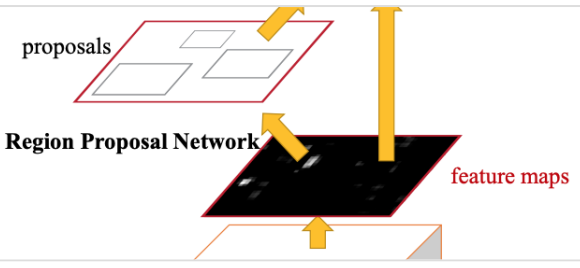
State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a Region Proposal Network

https://arxiv.org/abs/1506.01497?source=post_page

갈아먹는 Object Detection [4] Faster R-CNN

갈아먹는 Object Detection [1] R-CNN 갈아먹는 Object Detection [2]Spatial Pyramid Pooling Network 갈아먹는 Object Detection [3] Fast R-CNN Fast-RCNN에 이어서 오늘은 Faster R-CNN[1]을 리뷰해보도록 하겠습니다. 본격적으로 Real Time Object Detection의 포문을 연 논문이라고 할 수 있겠습니다. Mask R-CNN이 남긴 했지만 그동안 열

https://yeomko.tistory.com/17

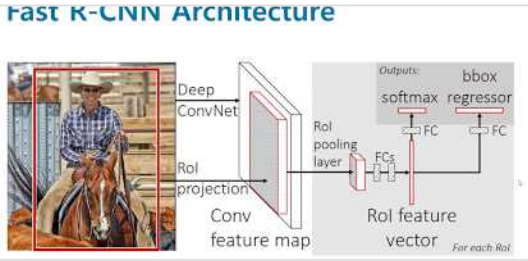


https://github.com/shkim960520/faster-rcnn-for-studying

PR-012: Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks

논문링크 : https://arxiv.org/abs/1506.01497발표자료 : https://www.slideshare.net/JinwonLee9/pr12-faster-rcnn170528

https://www.youtube.com/watch?v=kcPAGIgBGRs

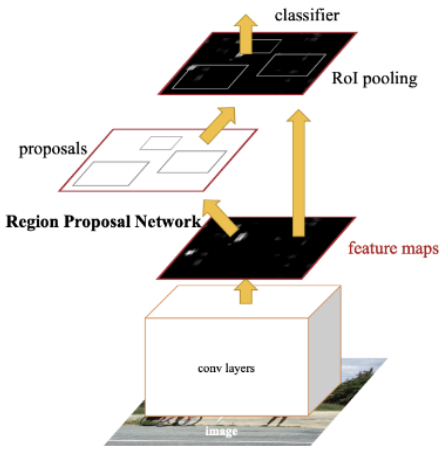


Faster R-CNN 의 주요 기여

그 동안 selective search 를 사용하여 계산해왔던 region proposal 단계를 neural network 안으로 끌어와서 진정한 의미의 end-to-end detection 모델을 제시했다.

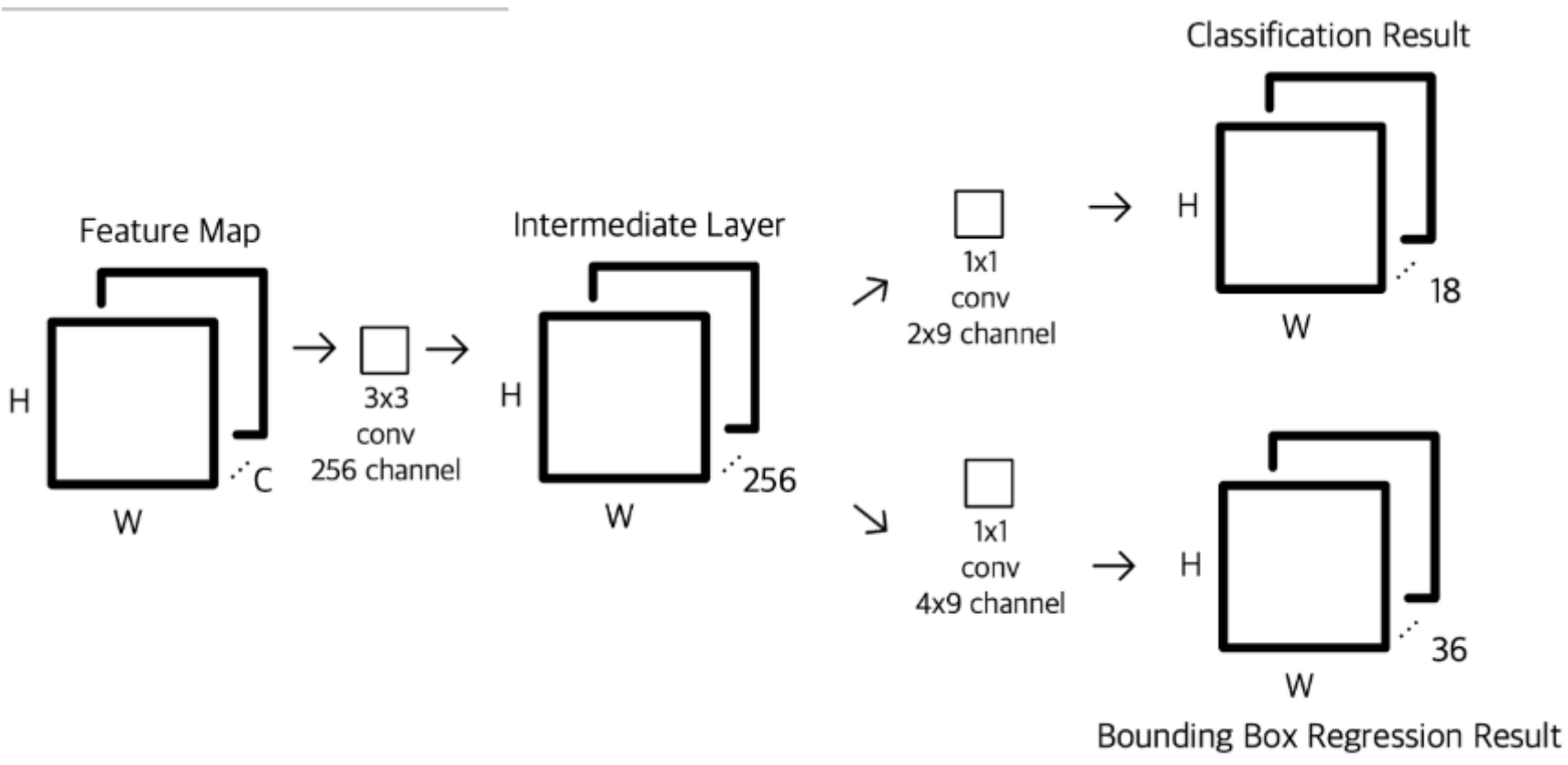
핵심 아이디어 - Region Porposal Network

기존 Fast R-CNN 구조를 그대로 가져오면서 selective search 를 제거하고 RPN 을 통해서 800개 정도의 RoI 를 계산한다.

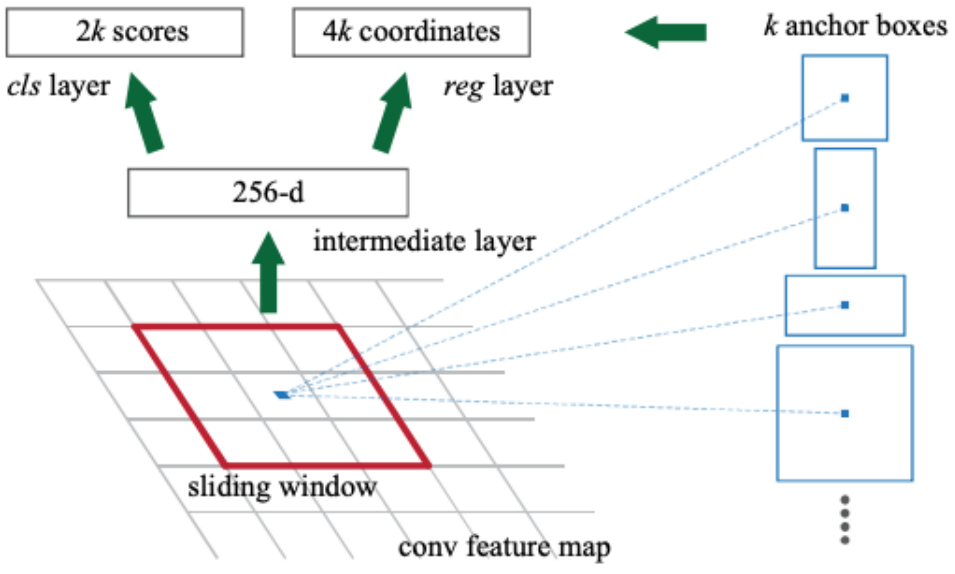


Faster R-CNN structure

다음의 이미지는 RPN 의 구조를 풀어서 표현한다.



RPN Structure



1. CNN을 통해 뽑아낸 feature map 을 입력으로 받는다. feature map 의 크기는 H*W*C 로 잡는다.

1. feature map 에 3*3 convolution 을 256 혹은 512 channel 만큼 수행한다. (intermediate layer 부분) padding을 1로 설정해 주어 channel 을 제외한 나머지 사이즈가 보존될 수 있도록 한다.
2. intermediate layer 의 출력인 feature map 을 classification 과 bounding box regression 을 위한 입력으로 사용한다. 이 때 1*1 convolution 을 이용해 연산하는 fully convolution network 구조를 활용한다. 입력 이미지의 크기에 상관없이 동작할 수 있는 이점이 있다.

3. classification 을 수행하기 위해 1*1 convolution 을 2(object 인지 아닌지 나타내는 지표 수)*9(anchor 개수) = 18 채널 만큼 수행해주며 그 결과로 H*W*18 사이즈의 feature map 을 얻는다. H*W 상의 하나의 인덱스는 feature map 상의 좌표를 의미하고, 그 아래 18개 채널은 각각 해당 좌표를 anchor 로 삼아 k 개의 anchor box 들이 object 인지 아닌지에 대한 예측 값을 담고 있다. 이제 이 예측값들을 적절히 reshape 한 다음 softmax 를 적용해 해당 anchor 가 object 일 확률을 얻는다.

4. 두 번째로 bounding box regression 예측 값을 얻기 위한 1*1*(4*9) convolution 을 수행.

5. classification 을 통해 얻은 물체일 확률 값들을 정렬한 다음, 높은 순으로 k 개의 anchor 만을 추려낸다. 그 다음 k 개의 anchor 들에 각각 bounding box regression 을 적용해준 다음, non-maximum-suppression 을 적용하여 RoI 를 구한다.

Loss Function

RPN 은 classification 과 bounding box regression 을 수행하므로 loss function 은 두 가지 task 에서 얻은 loss 를 엮은 형태를 취하고 있다.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Multi Task Loss

i 는 하나의 anchor 를 말한다. p_i 는 classification 을 통해서 얻은 해당 앵커가 오브젝트일 확률을 의미한다. ti 는 bounding box regression 을 통해서 얻은 박스 조정 값 벡터를 의미한다. pi* 과 ti* 은 ground truth 라벨에 해당한다. classfication 은 log loss 을 통해서 계산하며 regression 은 fast R-CNN 에서 사용한 smoothL1 함수를 사용한다.

N_cls 는 minibatch 사이즈 이며, N_reg 는 앵커 개수에 해당한다. 논문에서는 256*9 = 약 2400 이다.

λ 는 classification loss 와 regression loss 사이에 가중치를 조절해주는 부분인데 논문에서는 10으로 설정되어 있어, 사실상 두 loss 에 동일한 가중치가 매겨진다.

RPN 과 Fast R-CNN 네트워크 의 연결 → Training

RPN 이 제대로 RoI 를 계산하지 못하면 뒤의 classification layer 가 제대로 학습되기 어렵다. 이렇듯 전체 모델을 한 번에 학습시키는 작업은 어렵다. 논문에서는 4단계에 걸쳐 학습이 진행되는 기법을 사용한다.

1. ImageNet pretrained 모델을 불러온 다음, PRN 을 학습시킨다.
2. 1단계에서 학습시킨 RPN 에서 기본 CNN 을 제외한 Reggion Proposal 레이어만 가져온다. 이를 활용하여 Fast RCNN 을 학습시킨다. 이 때 처음 feature map 을 추출하는 CNN 까지 fine tune 시킨다.

3. 앞서 학습시킨 Fast R-CNN 과 RPN 을 불러온 다음, 다른 가중치들을 고정하고 RPN 에 해당하는 레이어들만 fine tune 시킨다. 여기서부터 RPN 과 Fast R-CNN 이 convolution weight 를 공유하게 된다.
4. 마지막으로 공유하는 CNN 과 RPN 은 고정시킨 채, Fast R-CNN 에 해당하는 레이어만 fine tune 시킨다.