Fast R-CNN

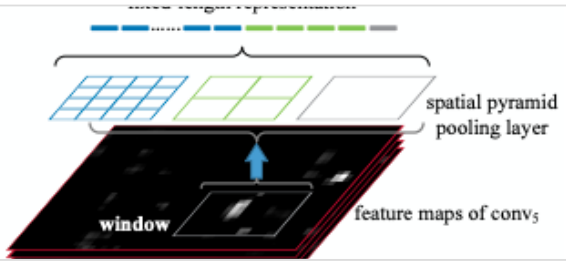
Fast R-CNN 논문은 SPPNet 에서 많은 부분들을 참고한다. Fast R-CNN 을 다루기 전에 SPPNet 을 먼저 살펴본다.

Spatial Pyramid Poolling Network

갈아먹는 Object Detection [2] Spatial Pyramid Pooling Network

갈아먹는 Object Detection [1] R-CNN 지난 시간 R-CNN에 이어서 오늘은 SPP-Net[1]을 리뷰해보도록 하겠습니다. 저역시 그랬고, 많은 분들이 R-CNN 다음으로 Fast R-CNN 논문을 보시는데요, 해당 논문을 보다 보면 SPPNet에서 많은 부분들을 참고한 것을 확인할 수 있습니다. 특히나 핵심인 Spatial Pyramid Pooling은 중요한 개념이므로 리뷰하고 넘어

https://yeomko.tistory.com/14?category=888201



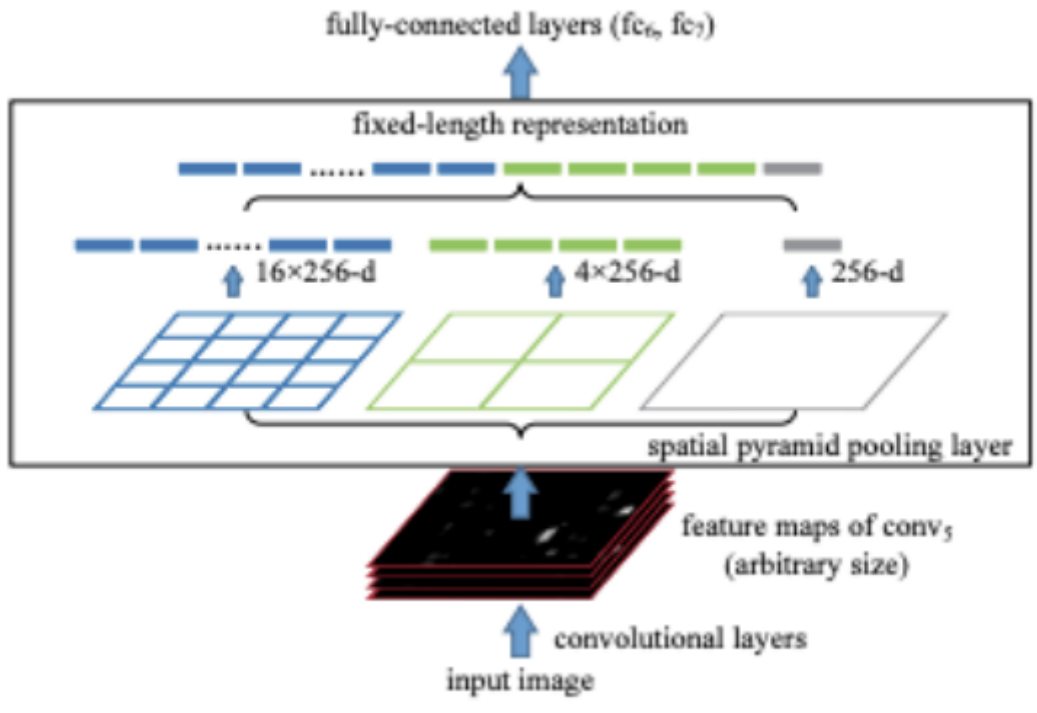
https://arxiv.org/pdf/1406.4729.pdf

논문의 저자들은 R-CNN의 wrap 과정에서 물체의 일부분이 잘리거나, 본래의 생김새와 달라지는 문제점을 지적한다. 이미지를 resize 하는 과정은 fully-connected layer 가 고정된 크기의 입력을 받기 때문인데, Spatial Pyramid Pooling 은 사실 convolution은 입력 이미지가 고정될 필요가 없다는것에 주목하면서 제안되었다.

아이디어는 다음과 같다. “입력 이미지의 크기에 관계없이 conv layer 들을 통과시키고, fully connected layer 통과 전에 feature map 들을 동일한 크기로 조절해주는 pooling 을 적용하자” - 입력 이미지의 크기를 조절하지 않은 채로 convolution 을 진행하면 원본 이미지의 특징을 고스란히 간직한 feature map 을 얻을 수 있다. 또한 사물의 크기 변화에 더 견고한 모델을 얻을 수 있다는 것이 저자들을 주장이다.

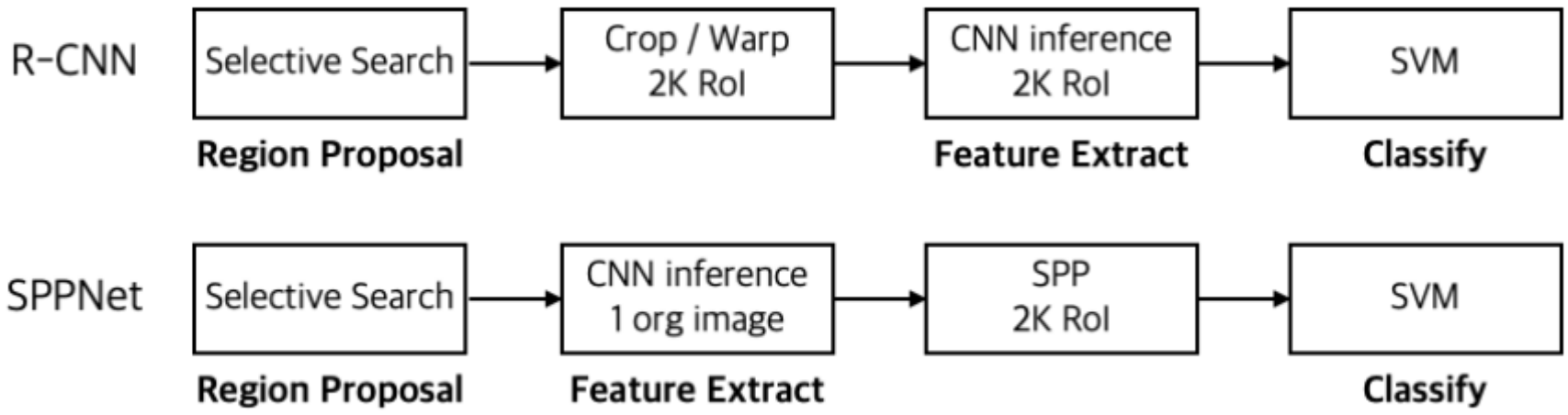
SPP Net 의 구조는 R-CNN 과 크게 다르지 않다. 단지 wrap 을 시키는 부분이 사라지고, conv layer 다음에 SPP를 적용해서 고정된 크기의 feature vector 를 추출하는 부분이 추가되는 것 뿐이다. 이후의 구조 - feature vector 를 SVM classifier 를 통과시키고, bounding box regressor 를 학습 - 는 R-CNN 과 동일하다.

다음의 이미지는 SPP 의 구조를 보여준다.



Conv layer 를 통해 추루된 feature map 을 input 으로 spp layer 는 이를 미리 정해져 있는 영역으로 나누어 준다. 위 이미지에서 spp layer 는 4*4, 2*2, 1*1 세 가지 피라미드를 제공한다. 피라미드 한 칸을 bin 이라고 한다. 각 bin 에서 가장 큰 값만 추출하는 maxpooling 을 수행하고, 그 결과를 concatenate 한다. 입력 피쳐맵의 채널 크기를 k, 이어진 총 bin 의 개수를 m 이라 할 때, SPP layer 의 최종 아웃풋은 k*M 차원의 벡터이다.

다음의 이미지는 R-CNN 과 SPPNet 이 전체적으로 어떻게 다른지를 보여준다.



R-CNN 은 selective search 로 찾은 2,000 개의 물체 영역(RoI; Region of Interest)을 모두 고정 크기로 조절한 다음, 미리 학습된 CNN 모델을 통과시켜 feature 를 추출한다.

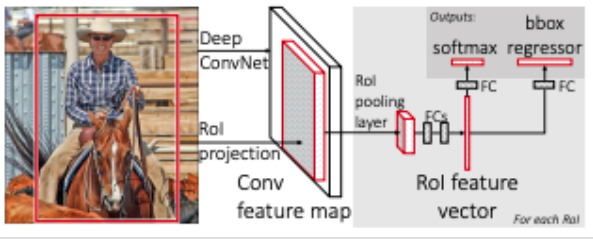
반면 SPPNet 은 입력 이미지를 그대로 CNN에 통과시켜 feature map 을 추출한 다음, 그 feature map 에서 2,000 개의 물체 영역을 찾아 SPP를 적용하여 고정된 크기의 feature vector 를 얻어낸다. 그리고 이를 FC 와 SVM Classifier 에 통과시킨다.

Fast R-CNN

갈아먹는 Object Detection [3] Fast R-CNN

갈아먹는 Object Detection [1] R-CNN 갈아먹는 Object Detection [2] Spatial Pyramid Pooling Network 지난 시간 SPPNet에 이어서 오늘은 Fast R-CNN[1]을 리뷰해보도록 하겠습니다. 저 역시 그랬고, 많은 분들이 R-CNN 다음으로 Fast R-CNN 논문을 보시는데요, 해당 논문을 보다 보면 SPPNet에서 많은 부분들을 참고한 것을 확인할 수 있습니다.

https://yeomko.tistory.com/15?category=888201

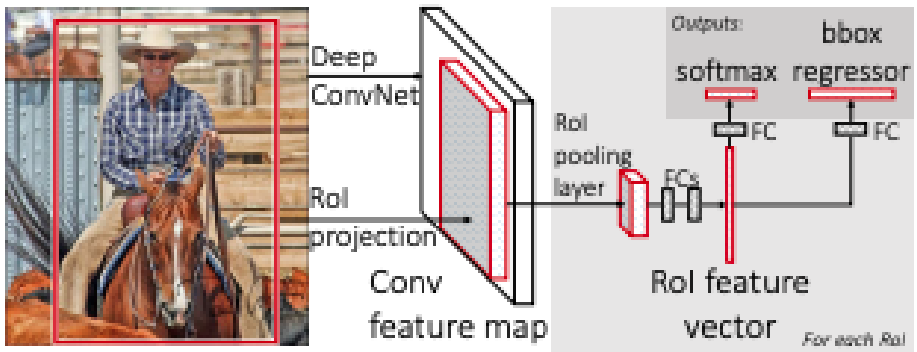


Fast R-CNN 은 SPPNet 에서도 해결하지 못했던 end-to-end 기법을 제시하여 학습 속도와 편의성이 굉장히 증가했다.

SPPNet 은 R-CNN에서 크게 벗어나지 못했는데, 여러 단계를 거쳐야 하는 점과, fully-connected layer 밖에 학습 시키지 못하는 한계점이 있었다. Fast R-CNN 논문 저자는 CNN 특징 추출부터 classification, bounding box regression 까지 모두 하나의 모델에서 학습시키자는 아이디어를 내놓는다.

전체 알고리즘은 다음과 같다.

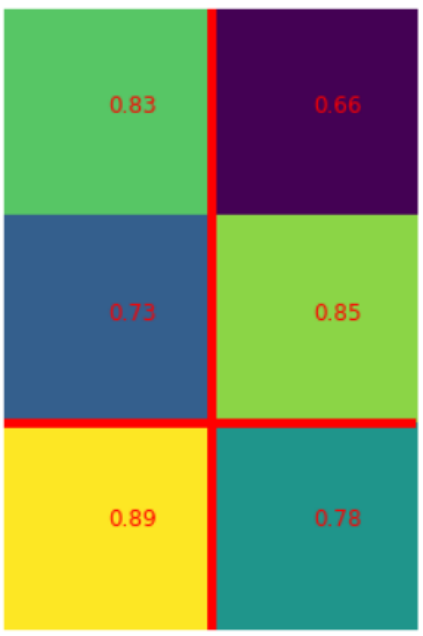
1. 먼저 전체 이미지를 미리 학습된 CNN을 통과시켜 피쳐맵을 추출한다.
2. selective search 를 통해서 찾은 각각의 RoI에 대하여 RoI pooling 을 진행한다. 그 결과로 고정된 크기의 feature vector 를 얻는다.
3. feature vector 는 fully connected layer 들을 통과한 뒤, 두 개의 브랜치로 나뉘게 된다.
4. 하나의 브랜치는 softmax 를 통과하여 해당 RoI가 어떤 물체인지 분류한다. 더이상 SVM 은 사용되지 않는다.
5. 또 다른 브랜치는 bounding box regression 을 통해서 selective search 로 찾은 박스의 위치를 조정한다.

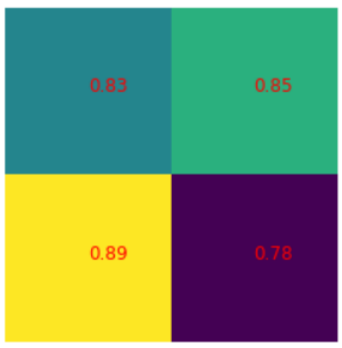


Fast R-CNN 은 CNN 을 통과하는 과정, 그 뒤 feature map 을 공유하는 과정, fully connected layer 를 통과하는 과정과 classification, bounding box regression 과정이 하나의 흐름으로 연결되어 있다. 더이상 스텝별로 쪼개어 학습을 진행하지 않고 end-to-end 로 엮여 있다.

Region of Interest Pooling

fully-connected layer 에 집어넣기 위해 max pooling 을 한다. pooling layer 의 output이 H*W, feature map 위의 RoI 가 h*w 일 때, pool size 를 계산하는 방법으로 h/H, w/W 이렇게 제안했다. 다음의 이미지는 2*2 크기로 pooling 할 때 반올림해서 2*1 만큼의 영역을 우선적으로 잘라서 max pooling 을 수행한다.





input 이미지와 feature map 의 크기가 다를 경우, 위의 방식대로 비율을 구해서 RoI 를 조절한 다음, pooling 을 진행한다.

Loss

Fast R-CNN에서는 classification 과 bounding box regression 을 적절하게 엮어주는 과정이 있고, 이 과정을 multi-task loss 라고 표현한다.

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v),$$

입력으로 p 는 softmax 를 통해서 얻어낸 k+1 개의 확률 값이다. 그다음 u는 해당 RoI 의 ground truth 라벨 값이다.

입력으로 t^u 는 클래스에 대한 x,y,w,h 값의 조정비율이며 ground truth 라벨에 해당하는 값이다. v 는 ground truth bounding box 조절 값에 해당한다.

classification 과 bounding box regression 에 해당하는 구체적인 loss function 은 다음과 같다.

$$L_{\text{cls}}(p, u) = -\log p_u$$

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

L1 norm 이 적용된 smooth 함수는 다음과 같다.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

smooth 함수가 사용된 이유는, 논문의 실험 과정에서 라벨 값과 차이가 많이 나는 이상치 예측 값들이 발생했고, 이들을 그대로 L2 계산하여 적용할 경우 그레디언트 폭주 현상을 관찰했다고 한다. 이를 방지하기 위해 추가주었다고 한다.

Backpropagation through RoI Pooling layer

Fast R-CNN 이전 SPPNet 에서는 CNN 부분은 그대로 놔두고, SPP 이후의 fully-connected layer 들만 fine-tune 하였다. CNN이 학습되지 않는경우 성능 향상에 제약이 있다고 논문의 저자들은 주장한다.

Fast R-CNN 에서는 multi-task loss 를 RoI Pooling layer 를 통고하여 CNN 단까지 fine-tuning 하였고, 실제로 성능 향상에 도움이 되었다는 실험 결과를 보인다. 실험 결과는, CNN 단을 깊이 학습시킬 수록 성능이 향상되었으며, 테스트에 소요되는 시간 변화는 거의 없었다.