Week 3. 합성곱 신경망2 (6.2,6.3)

21기 분석 이건하

목차

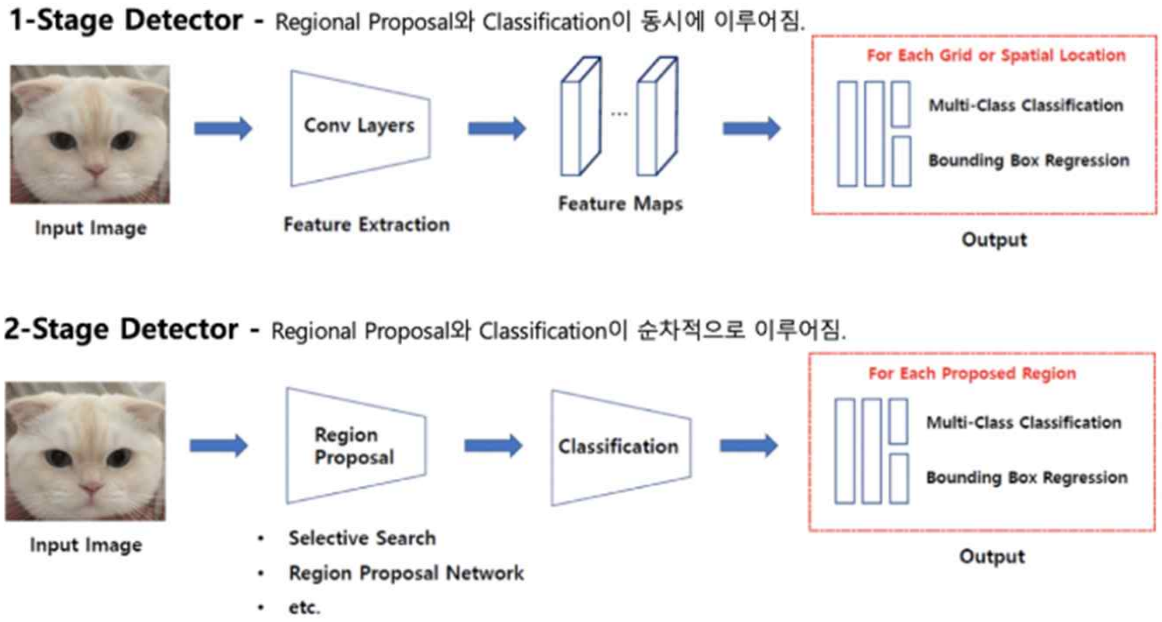
6.2 객체 인식을 위한 신경망 (object detection)

- 객체인식(Object Detection)

객체 인식 = 여러 가지 객체에 대한 분류 + 객체의 위치 정보를 파악하는 위치 검출

  - 이미지나 영상 내에 있는 객체를 식별하는 컴퓨터비전 기술
    - (Regional Proposal) 객체가 어디에 위치하는지: bounding box
    - (Classification) 각 개체가 무엇인지

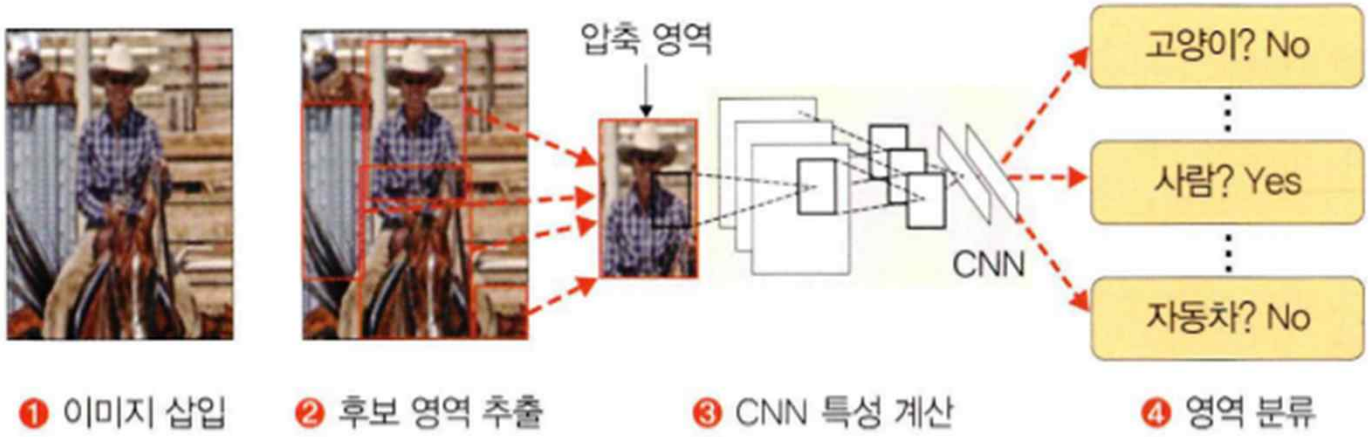
  - 1단계 객체인식 VS 2단계 객체인식
    - 1단계: 빠르지만, 성능이 떨어짐 (YOLO)
    - 2단계: 느리지만, 성능이 뛰어남 (R-CNN)



R-CNN

- R-CNN 학습 절차
  - 1) 이미지 입력
  - 2) 2000개의 bounding box를 **선택적 탐색 알고리즘**으로 추출한 후 자르고(Cropping), CNN 모델에 넣기 위해 같은 크기(227X227)로 통일(Warping)
  - 3) 크기가 동일한 이미지 **2000개**에 각각 **CNN모델을 적용**(결국 R-CNN은 객체탐색+CNN)
  - 4) 각각 **분류**를 진행하여 결과도출

그림 6-36 R-CNN 학습 절차



R-CNN

- 선택적 탐색(Selective search) 알고리즘
  - 1단계: 초기 영역 생성
    - 각각의 객체가 영역 한 개에 할당될 수 있도록 많은 초기 영역을 생성. 즉, 입력된 이미지를 영역 다수 개로 분할하는 과정
  - 2단계: 작은 영역의 통합
    - 1단계에서 영역 여러 개로 나눈 것들을 비슷한 영역으로 통합하는데, 이때 탐욕(greedy) 알고리즘을 사용하여 비슷한 영역이 하나로 통합될 때까지 반복
    - 탐욕(greedy) 알고리즘: 여러가지 경우 중 하나를 결정해야 할 때마다 그 순간에 최적이라고 생각되는 것을 선택해 나가는 방식

그림 6-37 R-CNN 학습 1단계

그림 6-38 R-CNN 학습 2단계

R-CNN

- 선택적 탐색(Selective search) 알고리즘
  - 3단계: 후보 영역 생성
    - 2단계에서 통합된 이미지들을 기반으로 다음 그림과 같이 후보 영역(bounding box)을 추출
  - 슬라이딩 윈도우 알고리즘 대체 (주의: CNN에서는 그대로 사용)
    - 이미지의 객체를 탐색하고자 이미지 왼쪽 위부터 일정 크기의 bounding box를 만들고, 그 안에 객체를 탐색하는 과정을 반복
    - 모든 영역을 탐색해야 하기 때문에 시간이 많이 소요되어 비효율적

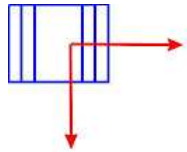
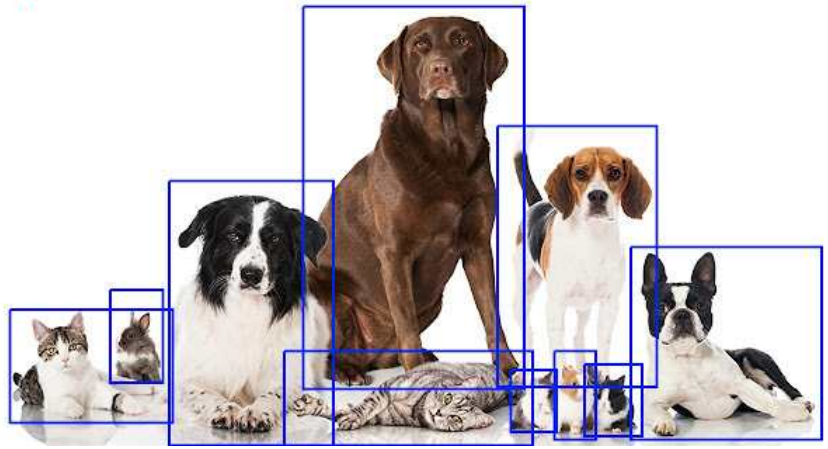


그림 6-39 R-CNN 학습 3단계

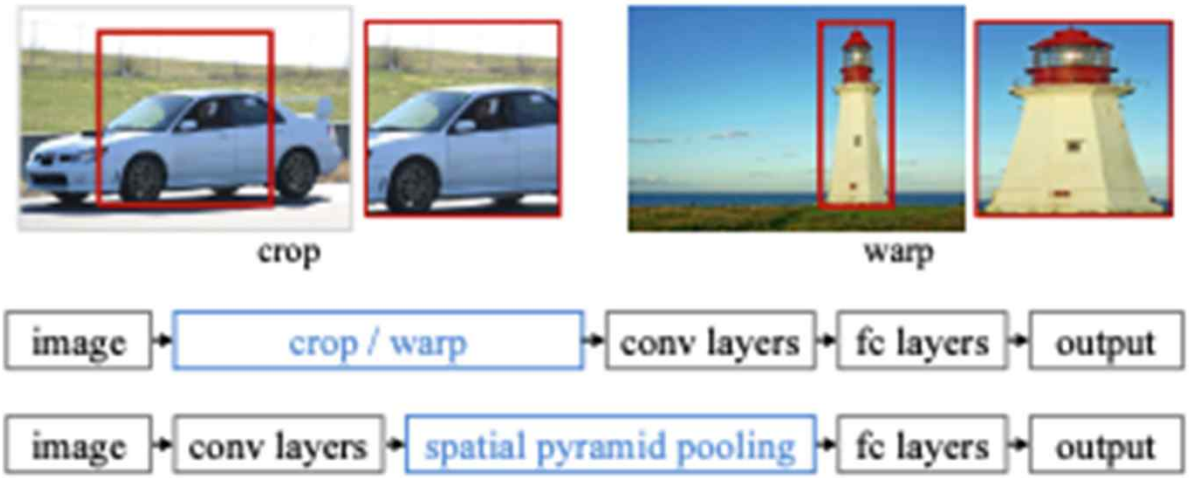
입력 이미지 비슷한 영역을 통합하여 후보 영역 생성



R-CNN

- R-CNN의 한계
  - 1) 선택적 탐색(Selective search) 알고리즘의 복잡한 학습과정
  - 2) 긴 학습시간과 대용량 저장공간 (분할된 영역단위로 CNN)
  - 3) 객체검출 속도 문제 (성능은 뛰어남)
  - 4) (CNN의 한계) 입력 이미지 크기 고정으로 인한 이미지 변형(일부분이 잘리거나, 본래 형태와 달라짐)

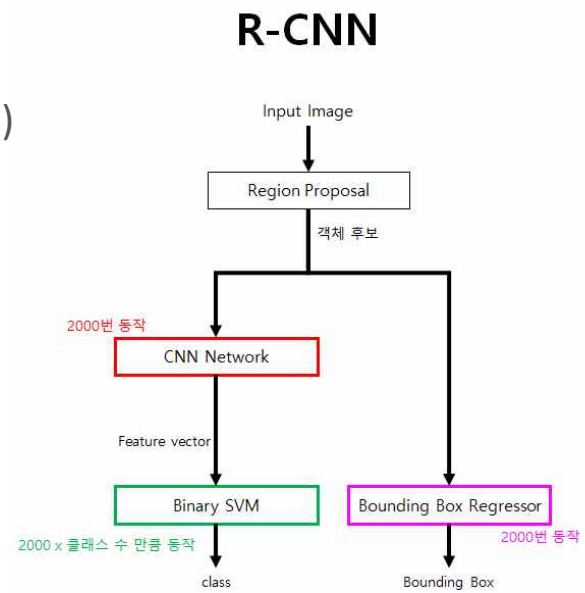
    -> 공간 피라미드 풀링(SPPNet), Fast R-CNN

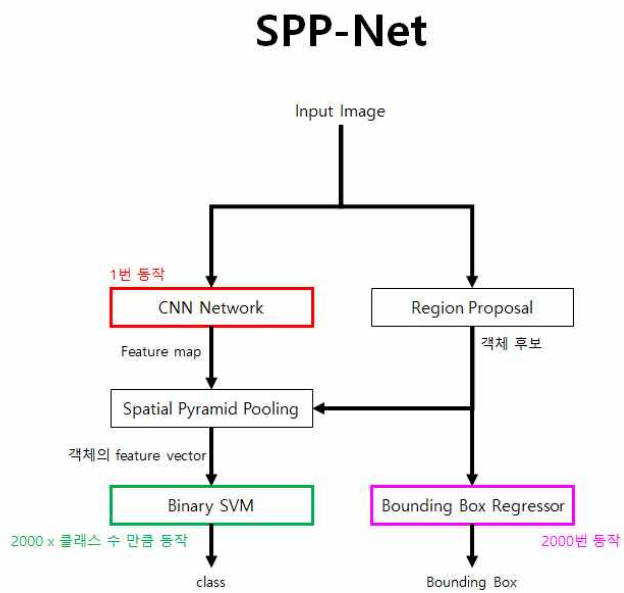


SPPNet

- 공간 피라미드 풀링 (SPPNet에서 제안된 아이디어. 풀링 방법은 중요X 크기를 나중에 고정시킨다는 개념이 중요)
  - R-CNN의 원본 이미지 훼손 문제 해결, 속도 문제 해결
    - 원본 이미지 그대로 합성곱층(CNN)을 통과시킴 -> Region proposal 정보를 featuremap으로 전달 -> 해당 Region proposa에 대해 공간 피라미드 풀링으로 크기를 고정(완전연결층에 전달될 때는 Input의 크기가 고정되어야 하기 때문) -> 완전연결층 통과 -> 분류)
    - R-CNN은 Region proposal 된 단위로 CNN이 진행됐지만, SPPNet은 원본이미지 단위로 CNN을 진행 -> 속도향상

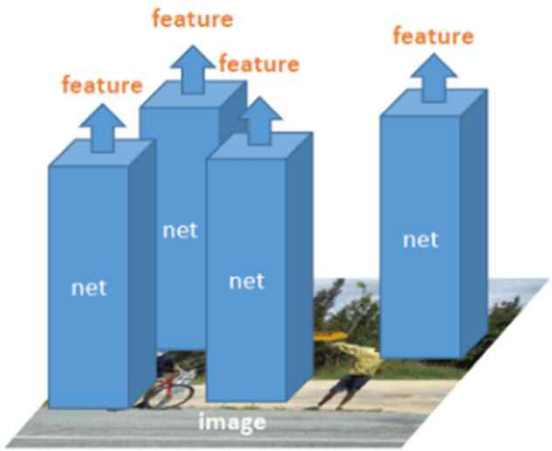
window size = ceiling(feature map size / pooling size)
stride = floor(feature map size / pooling size)



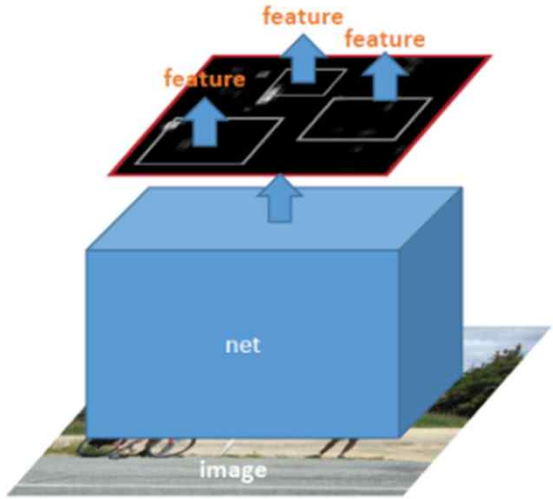


SPPNet

- 공간 피라미드 풀링 (SPPNet에서 제안된 아이디어. 풀링 방법은 중요X 크기를 나중에 고정시킨다는 개념이 중요)
  - R-CNN의 원본 이미지 훼손 문제 해결, 속도 문제 해결
    - 원본 이미지 그대로 합성곱층(CNN)을 통과시킴 -> Region proposa정보를 featuremap으로 전달 -> 해당 Region proposa에 대해 공간 피라미드 풀링으로 크기를 고정(완전연결층에 전달될 때는 Input의 크기가 고정되어야 하기 때문) -> 완전연결층 통과 -> 분류)
    - R-CNN은 Region proposal된 단위로 CNN이 진행됐지만, SPPNet은 원본이미지 단위로 CNN을 진행 -> 속도향상



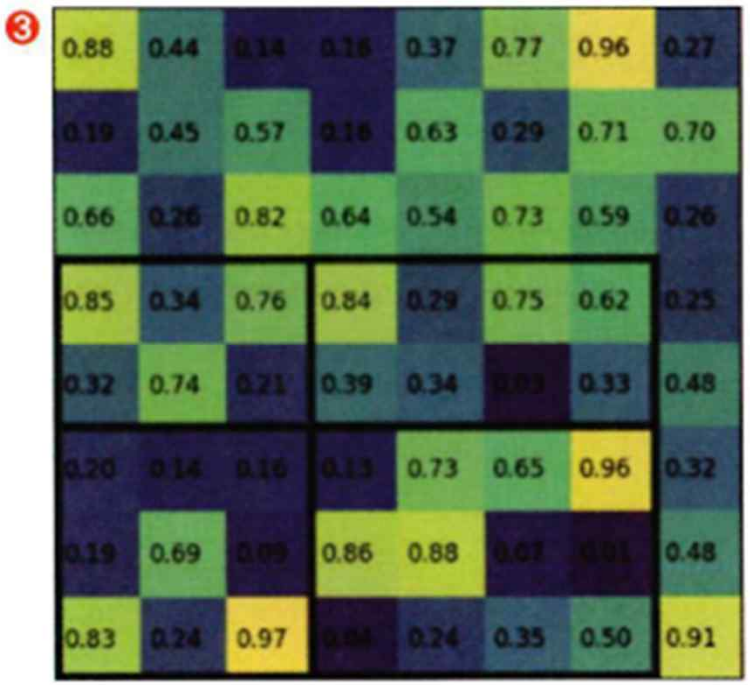
R-CNN
2000 nets on image regions

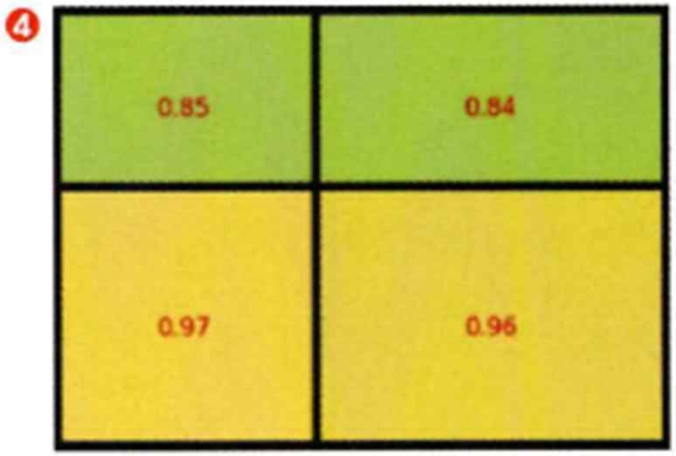


SPP-net
1 net on full image

Fast R-CNN

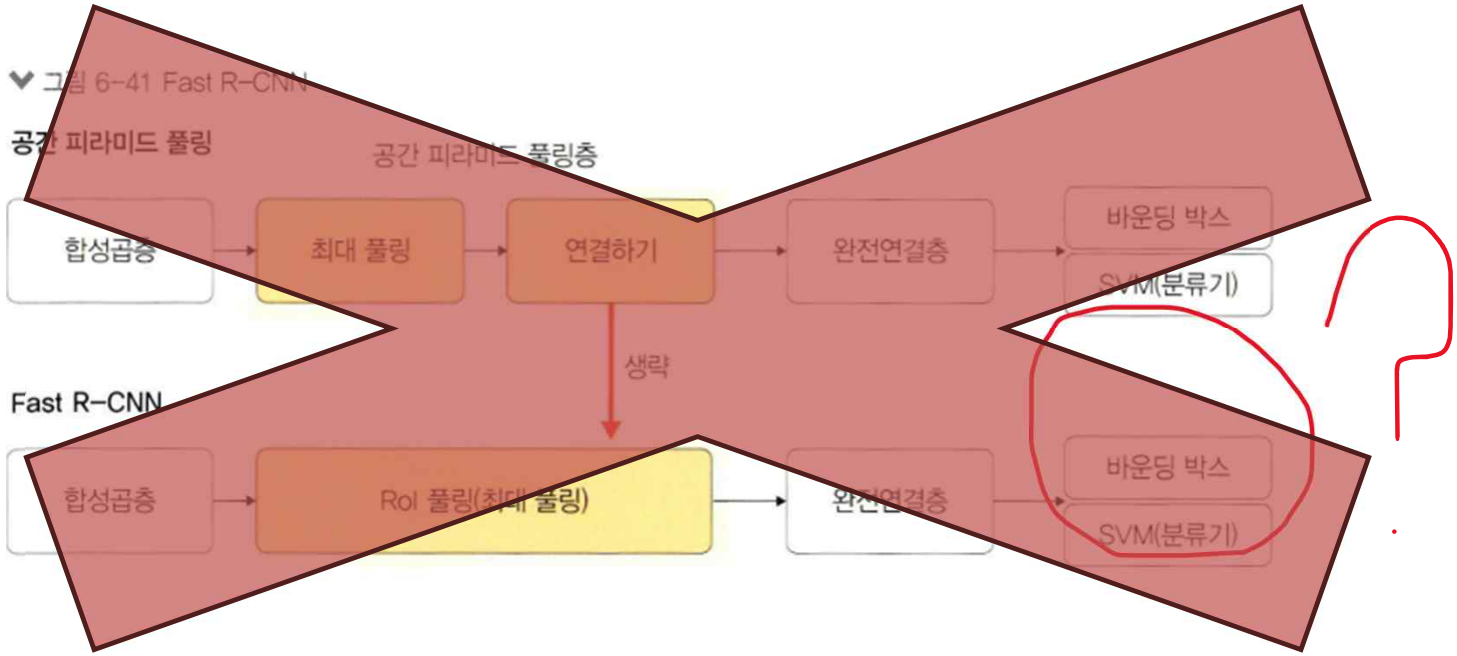
- Fast R-CNN
  - RoI Pooling
    - CNN Network를 통과한 Feature Map에서 객체 후보영역을 Max풀링
    - 크기가 다른 Feature Map의 영역마다 스트라이드를 다르게 하여 완전연결층에 입력되는 Feature Map의 크기 조절





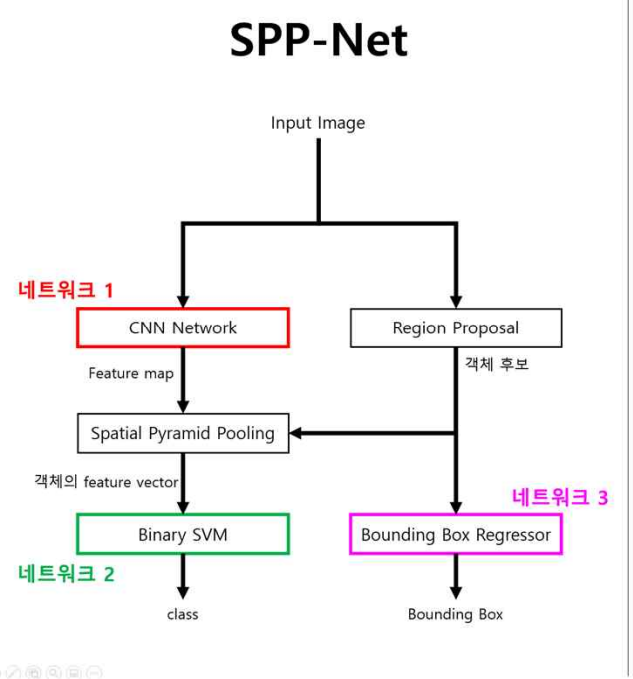
Fast R-CNN

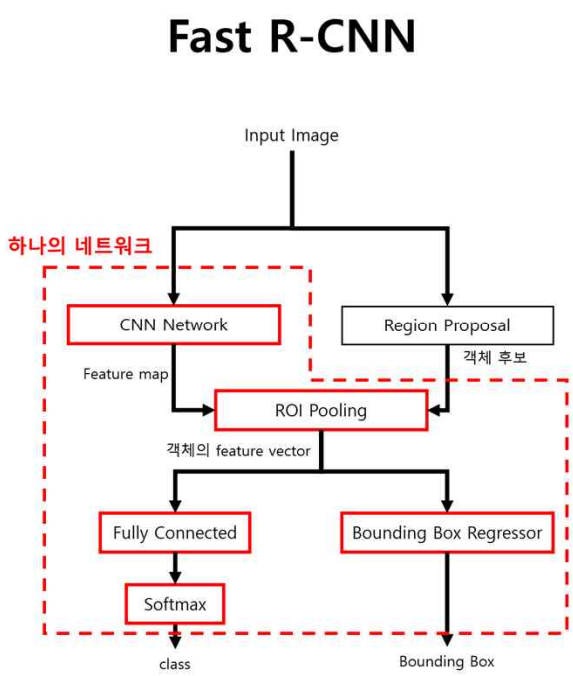
- Fast R-CNN
  - R-CNN과 SPPNet의 단점을 개선
    - Class 분류에서 Binary SVM 대신 FC layer와 softmax를 활용 -SVM은 이진 분류만 가능
    - 하나의 모델에서 end-to-end 학습을 할 수 있기 때문에 학습이 간단해지며, 최적화가 쉬워짐(CNN Network로 추출한 feature vector를 bounding box regressor의 입력으로 사용 -> loss가 한번에 계산)



Fast R-CNN

- Fast R-CNN
  - R-CNN과 SPPNet의 단점을 개선
    - Binary SVM 대신 FC layer와 softmax를 활용하여 Class Classification 진행 -SVM은 이진 분류만 가능
    - 하나의 모델에서 end-to-end 학습을 할 수 있기 때문에 학습이 간단해지며, 최적화가 쉬워짐(CNN Network로 추출한 feature vector를 bounding box regressor의 입력으로 사용 -> loss가 한번에 계산)





Loss Function

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

Classification loss

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

Localization loss

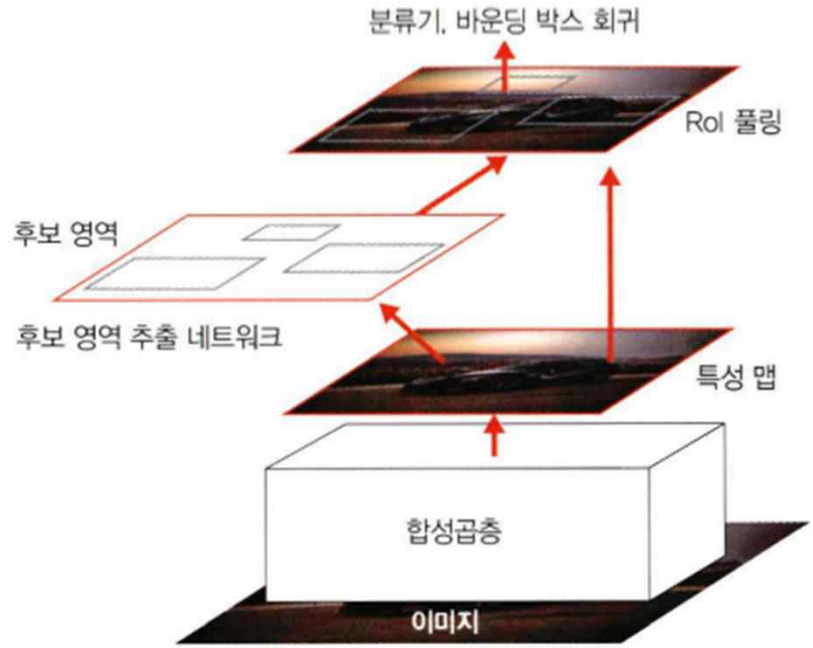
$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

<그림 10> Fast R-CNN loss function

Faster R-CNN

- Faster R-CNN (**RPN + Fast R-CNN**)
  - Fast R-CNN의 Region Proposal(후보 영역 생성)을 개선
    - 선택적탐색(selective search)를 사용하지 않고, **후보 영역 추출 네트워크(Region Proposal Network)**를 통해서 RoI를 계산
    - 후보 영역 생성을 CNN 내부 네트워크에서 진행하도록 하여 GPU를 통해 RoI 연산이 가능해짐

그림 6-43 Faster R-CNN



Faster R-CNN

- Faster R-CNN (**RPN + Fast R-CNN**)
  - 후보 영역 추출 네트워크(Region Proposal Network)의 구조
    - 해당영역에 객체의 존재 유무 판단을 위해 이진분류를 수행하는 작은 네트워크 (슬라이딩 윈도우)
    - 앵커를 기준으로 모든 슬라이딩 윈도우마다 k개의 래퍼런스 박스에 대해 객체유무 판별
    - 값이 높은 앵커들을 기반으로 객체의 위치를 보정과 객체 분류를 수행

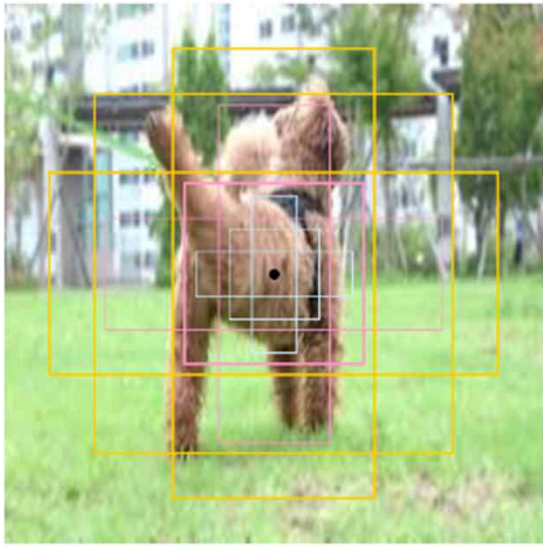


그림 6-44 후보 영역 추출 네트워크

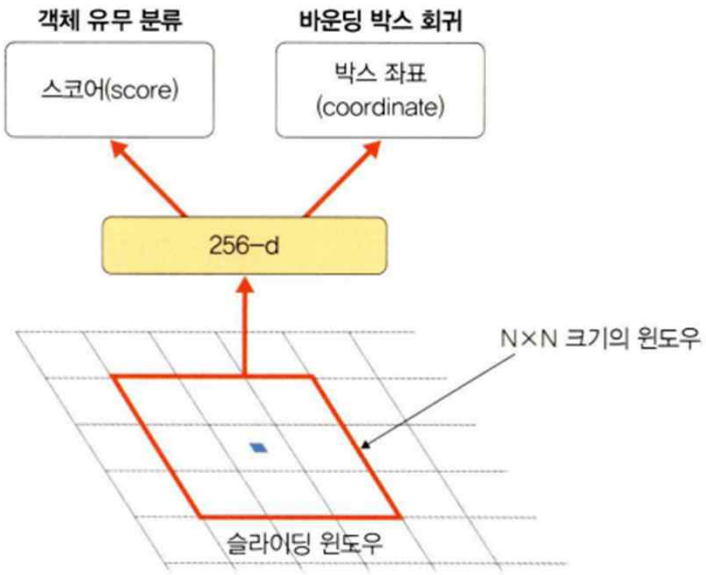
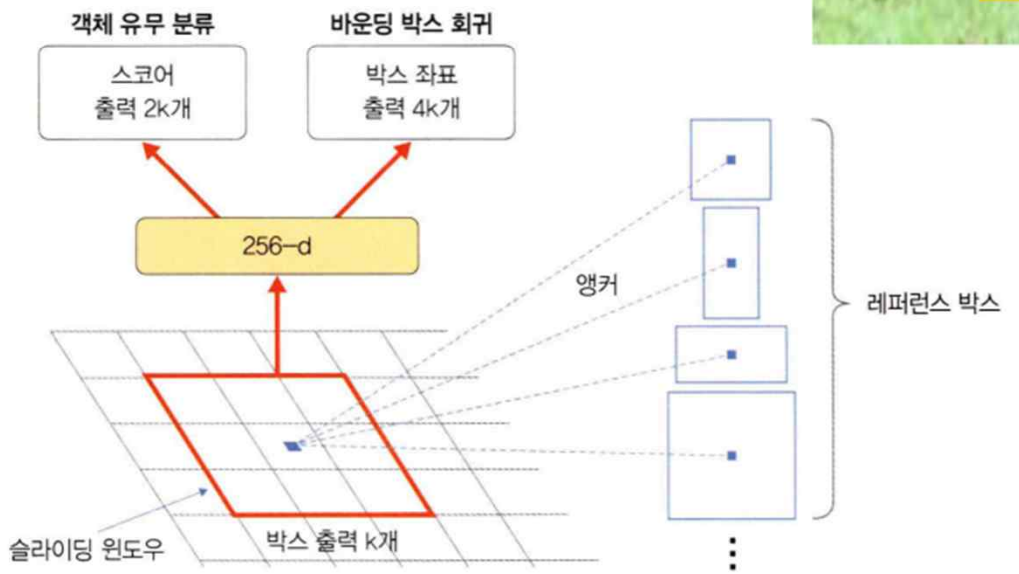


그림 6-45 앵커



6.3 이미지 분할을 위한 신경망

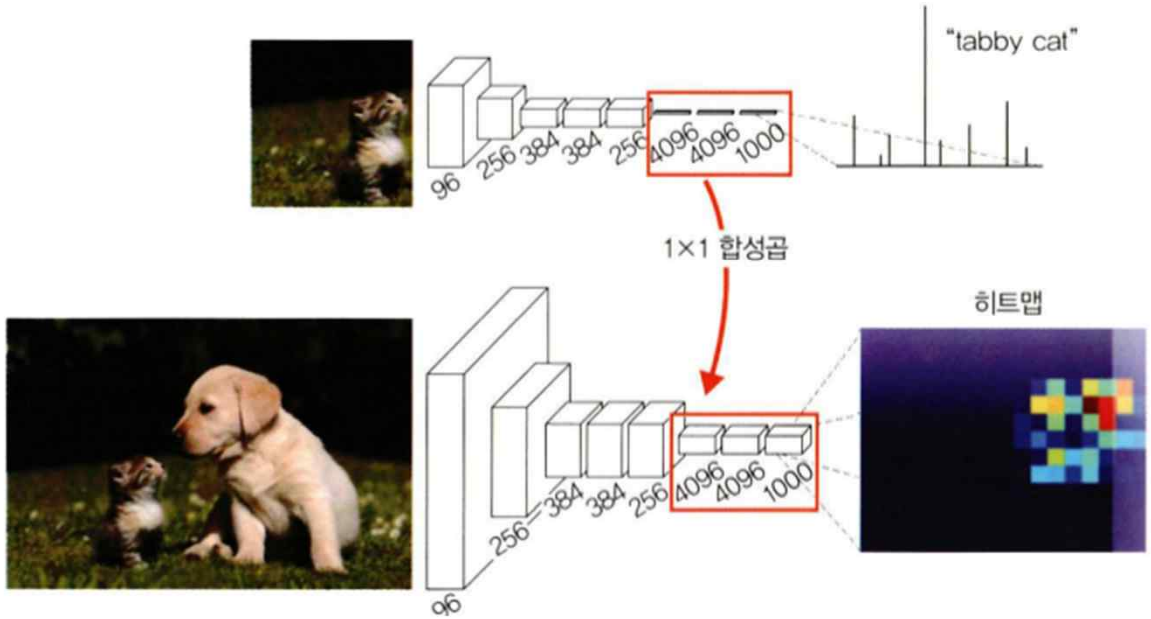
- 이미지 분할(Segmentation)
  - 신경망을 훈련시켜 이미지를 픽셀 단위로 분할 하는 것
  - 이미지 내 정보의 분류와 더불어 이미지 속 픽셀 수준에서 무엇이 있는지 이해하는 데에 사용하는 컴퓨터 비전 기술

BOAZ

완전 합성곱 네트워크

- 완전 합성곱 네트워크 (FCN)
  - 완전연결층을 거친 후에는 위치 정보가 사라지는 문제를 해결하기 완전연결층을 1x1 합성곱으로 대체하는 것
    - 여러단계의 합성곱과 풀링층을 거치면서 해상도가 낮아짐
    - 해상도를 복원하기 위해 업 샘플링(원본 크기로 복원) 방식을 사용하기 때문에 이미지의 세부 정보를 잃어버리는 문제(분할에서 중요)

그림 6-46 완전 합성곱 네트워크



완전 합성곱 네트워크

- 합성곱 & 역합성곱 네트워크 (convolutional l fdcconvolutional neiwork)
  - 역합성곱은 CNN의 최종 출력 결과를 원래의 입력이미지와 같은 크기로 만들고 싶을 때 사용(분류, 객체인식과 다르게 return값이 이미지이기 때문)
    - 1) 각각의 픽셀 주위에 제로패딩을 추가
    - 2) 패딩된 것에 합성곱 연산을 수행

그림 6-47 합성곱 & 역합성곱 네트워크

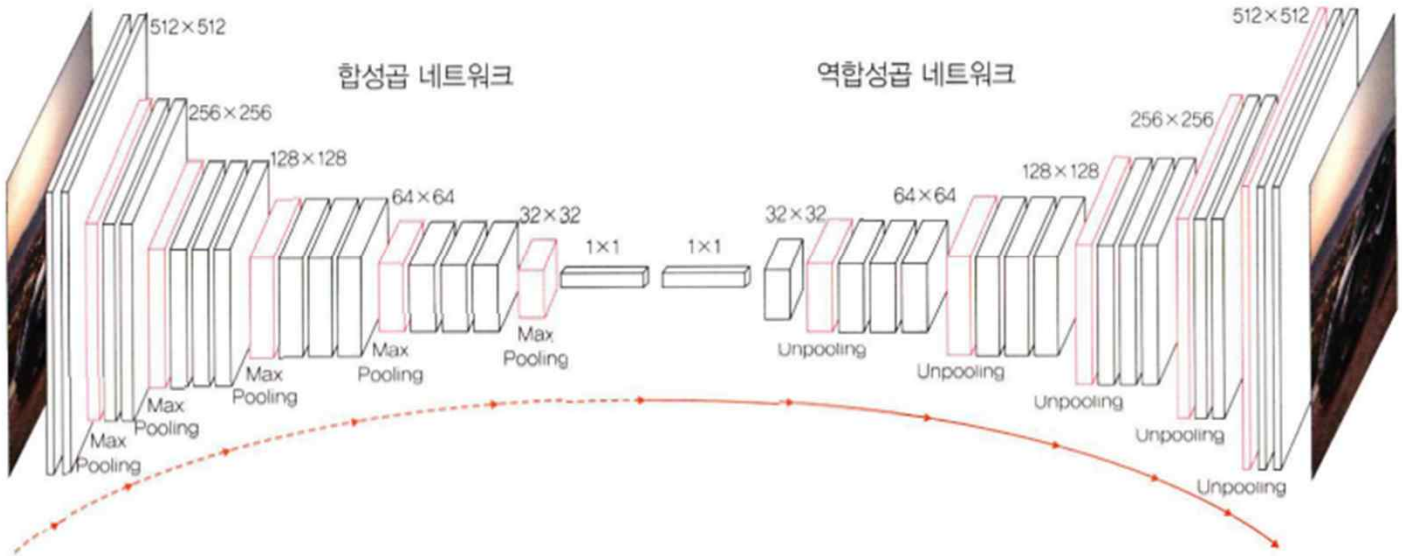
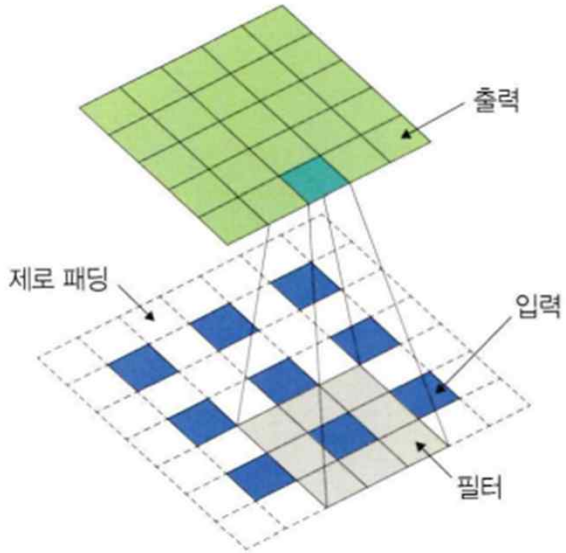


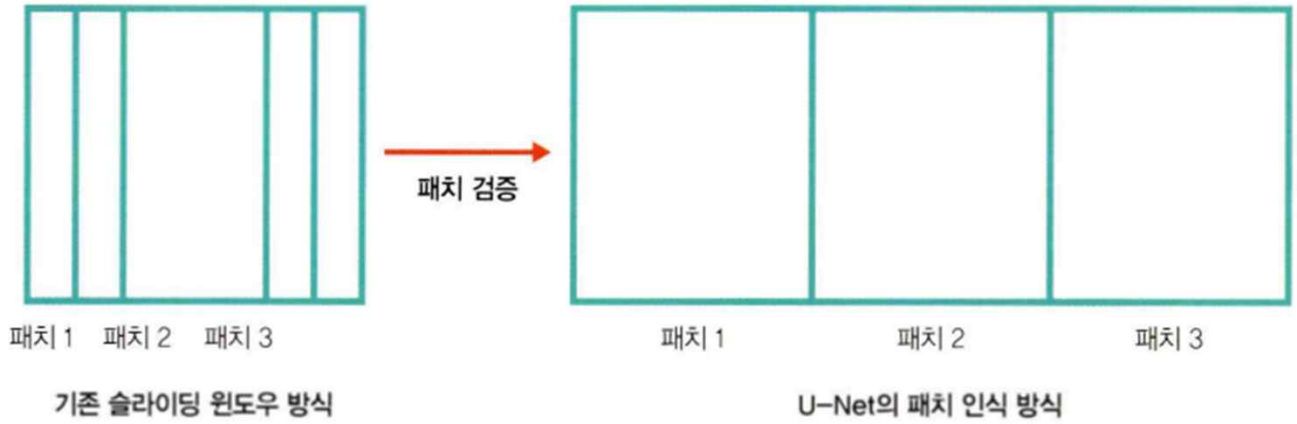
그림 6-48 역합성곱 진행 방식



U-Net

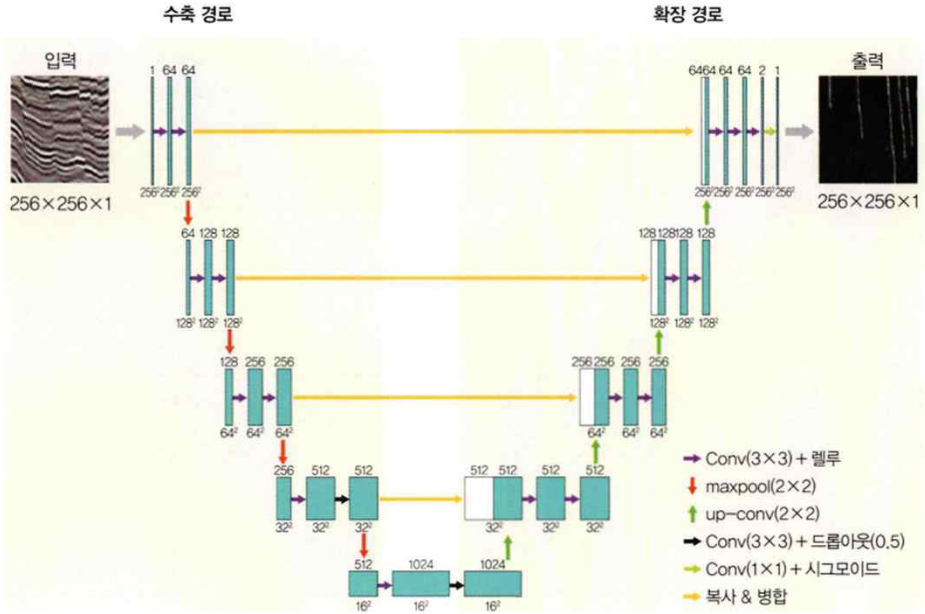
- U-Net
  - 바이오 메디컬 이미지 분할을 위한 합성곱 신경망
    - 이미 검증이 끝난 패치는 건너뛰기 때문에 속도가 빠름 (스트라이드가 W와 동일한 개념)
    - 컨텍스트 인식과 지역화 트레이드오프 문제 개선

그림 6-49 슬라이딩 윈도우 방식



U-Net

- U-Net의 구조
  - FCN을 기반으로 구축되었으며, 수축경로와 확장경로로 구성
    - 수축경로는 컨텍스트를 포착하며, 확장경로는 특성맵을 업샘플링하고 수축경로에서 포착한 특성맵의 컨텍스트와 결합하여 정확한 지역화를 수행
    - 각 합성곱 블록은 3x3 합성곱 2개로 구성되어 있으며, 그 사이에 드롭아웃이 존재
    - 각 블록은 Max 풀링을 이용하여 크기를 줄이면서 다음 블록으로 넘어감



그 외 이미지분할 모델

- 이미지 분할(Segmentation)
  - PSPNet
    - 완전연결층의 한계를 극복하기 위해 피라미드 풀링 모듈을 추가
    - 양선형 보간법
  - DeepLabv3/DeepLabv3+
    - Atrous 합성곱을 사용하는 네트워크
    - 인코더-디코더 구조

BOAZ

Any Questions?