11 강. Detection & Segmentation

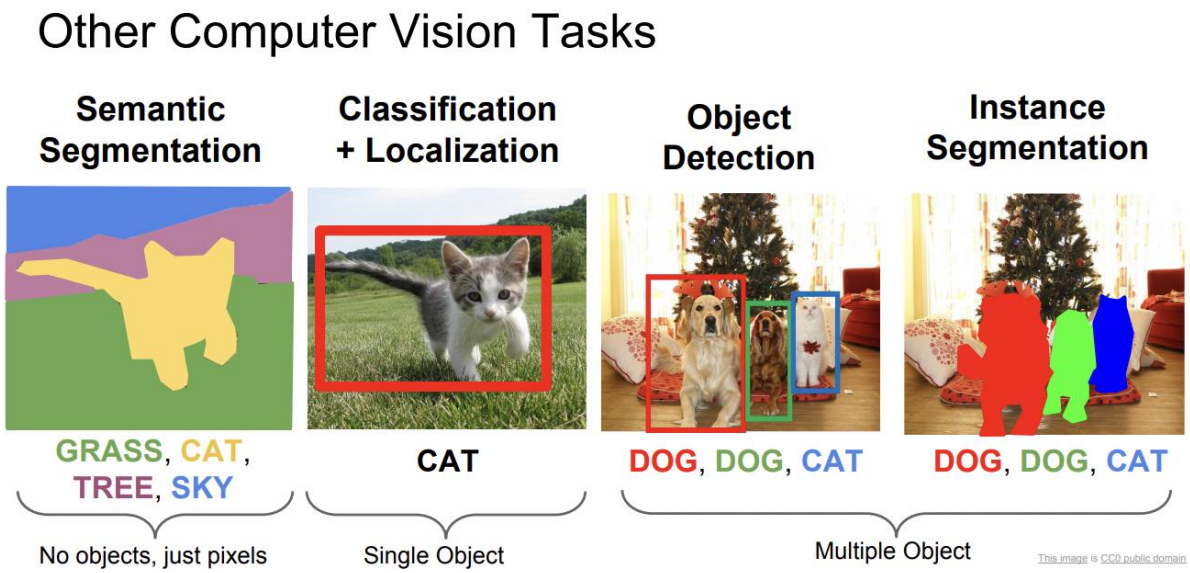
개요

1. Sementic Segmentation
    - 1) sliding window
    - 2) fully convolutional
    - 3) fully convolutional with downsampling/upsampling
    - upsampling 방법
    - 1) Nearest Neighbor
    - 2) Bed of Nails
    - 3) Max Unpooling
    - 4) Transpose Convolution : Learnable Upsampling
2. Classification + Localization
3. Object Detection
4. Instance Segmentation

여태까지는 Deep-Learning 의 basic 과 Image classification task 를 중점적으로 다뤘다.

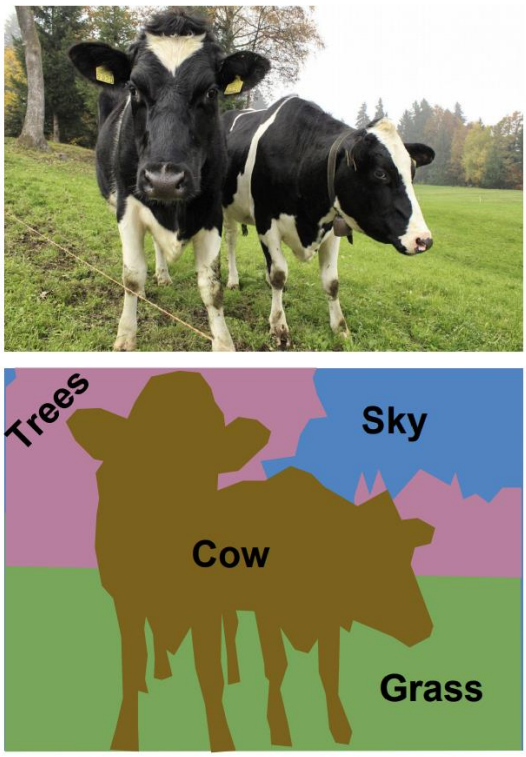
하지만 Computer Vision field 에는 Image Classification 외에도 다양한 task 들이 존재한다.

오늘은 그 중에도 Segmantation, Localization, Detection 에 대해 다뤄보겠다.



1. Semantic Segmentation

: 이미지의 픽셀들이 어떤 클래스에 속하는지 예측하는 과정



Classification: 객체단위

Semantic Segmentation: 픽셀단위, 고양이 1 고양이 2 구분못함. 그냥 고양이.

Instance Segmentation: 픽셀단위, 고양이 1 고양이 2 구분함.

Classification 이 객체단위로 분류했다면,

Semantic Segmentation 은 픽셀 단위로 분류한다.

모든 픽셀은 각자가 속해 있는 클래스 값을 가진다.

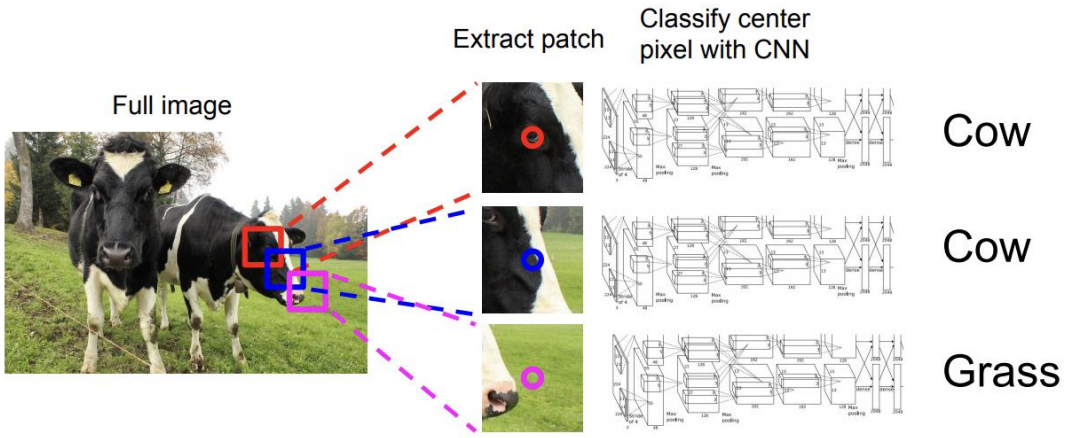
SS 도 Classification 과 마찬가지로 미리 클래스의 수와 그 종류를 정해 놓아야 한다.

SS 는 객체끼리는 분류하지는 않는다.

위의 그림에서 소가 두마리이지만 소 1, 소 2 를 나누지 않고, 둘다 소라고만 인식한다.

만약 같은 분류에 속하는 객체끼리도 구분하고 싶다면, instance segmentaion.

Semantic Segmentation 방법



1) Sliding Window ❌

Classification 처럼 이미지에 sliding window 기법을 사용하면 가능.

하지만 이 방법은 아래와 같은 이유로 사용되지 않는다.

단점 1) Computational Cost 가 매우 높다.

각 pixel 마다 pixel 중심으로 crop 을 진행한다.

→ CNN 에 crop 한 이미지들을 전부 넣는다? 연산이 너무 많다.

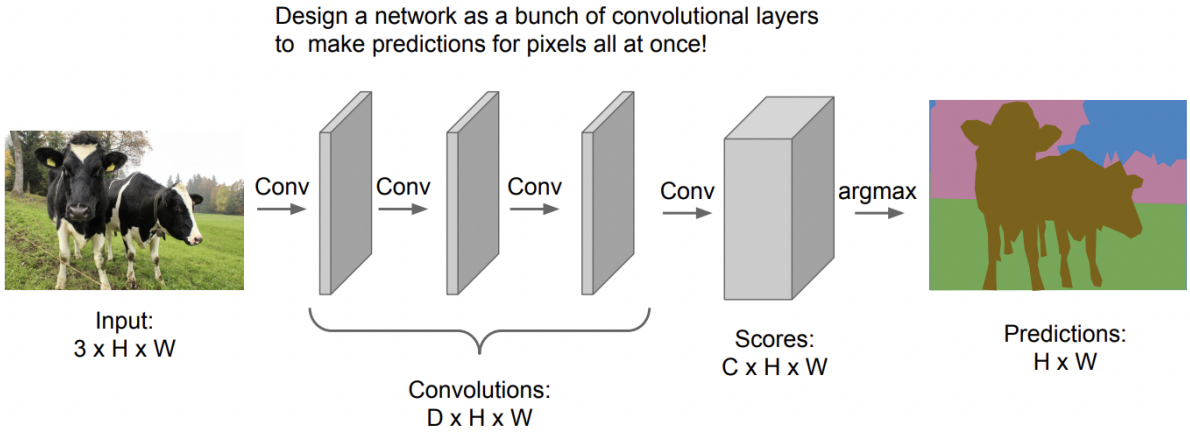
단점 2) Sharing computation 이 고려되지 않았다.

각 pixel 마다 crop 을 진행했을때, 이웃 pixel 들은 crop 한 영역이 필연적으로 겹치게 된다.

하지만 이 중복되는 computation 에 대한 처리가 없어 효율성이 떨어진다.

단점 3) 그 외에도 국소적으로 부분부분 탐색하기 때문에 이미지 전체의 context 정보를 간과한다는 단점이 있다.

2) Fully Convolutional ❌



이미지 한장을 CNN 에 통째로 통과시키는 방법.

spatial resolution 을 유지하면서 레이어들을 통과시키다 보면 final output 으로 CxHxW vector 를 얻을 수 있다.

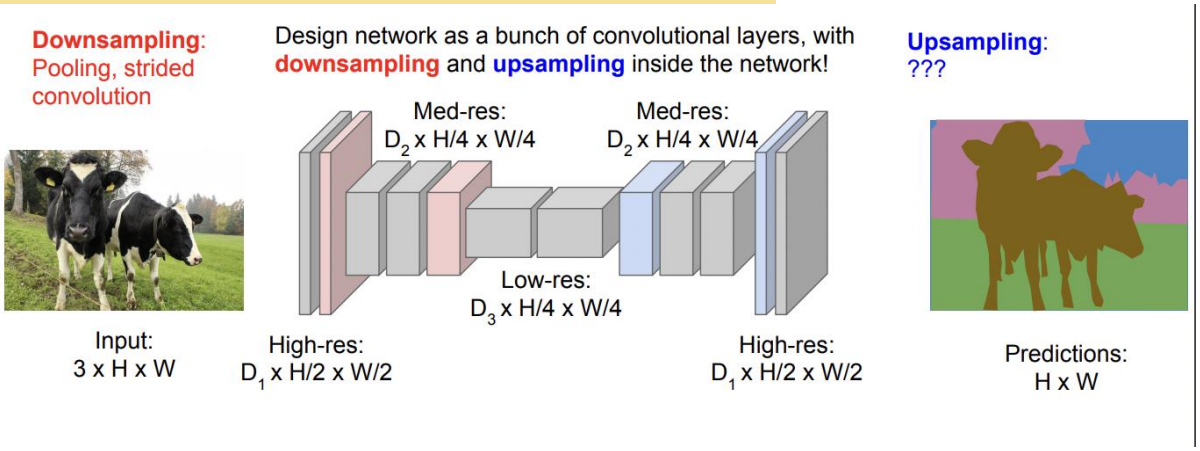
(C: class 개수)

즉 HxW 의 모든 pixel 은 class Ci 에 해당하는 점수를 갖는다.

단점 1) training data 한장을 만드는데 오래걸린다.

단점 2) high resolution 을 끝까지 유지한다 > computational cost 가 부담스럽다.

3) Fully Convolutional with downsampling / upsampling O



원본 resolution(해상도)에서 시작해 downsample 을 몇차례 거치면서 해상도를 낮추다가,

중간부터는 다시 upsampling 으로 해상도를 올리는 방식이 효과가 좋다.

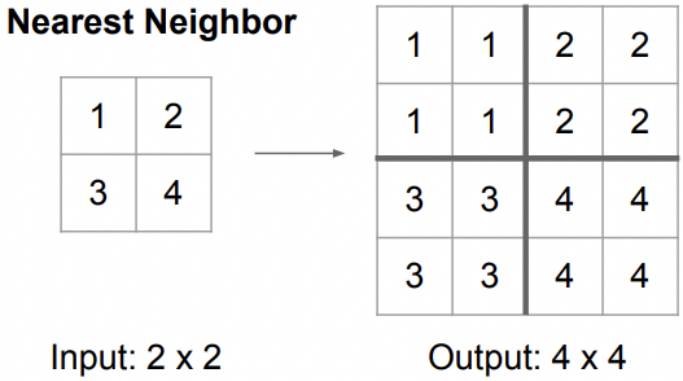
장점 1) 이렇게 되면 위에서 말했던 resolution 을 계속 크게 유지하는데에서 오는 computation cost 문제를 어느정도 해결할수가 있다. 중간 단계에서는 낮은 resolution 을 사용하지만 input 과 output 의 사이즈가 결과적으로 동일하기 때문이다.

Upsampling

Downsampling 에는 max-pool, average-pool 등이 있음.

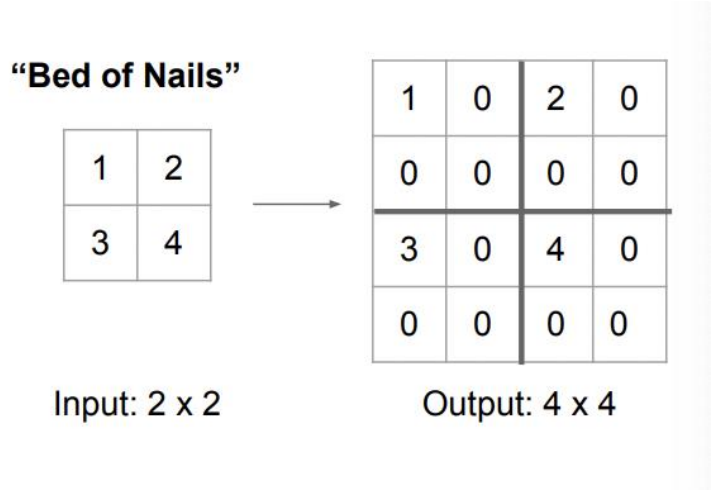
Upsampling 은 어떻게 하는걸까?

2x2 → 4x4 Stride:2



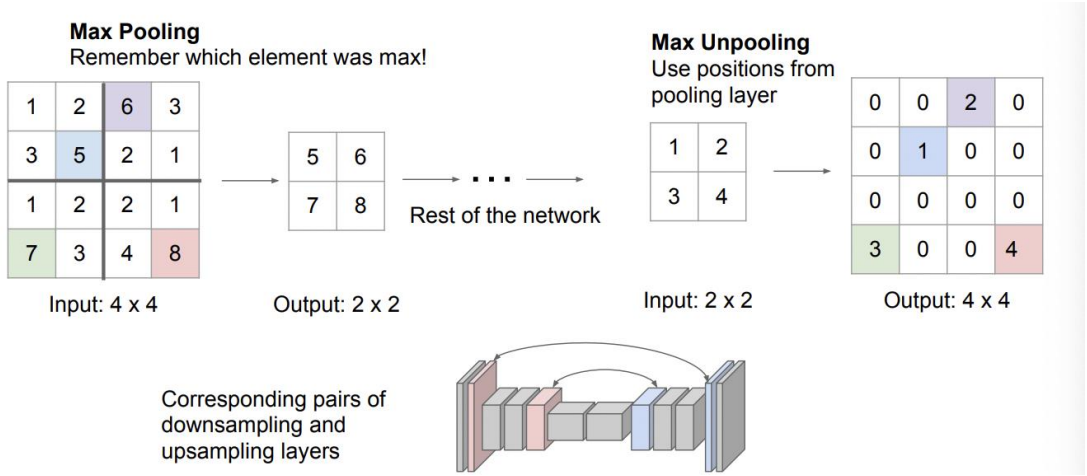
1) Nearest Neighbor

: 이웃한 픽셀에 복사



2) Bed of Nails

: 왼쪽 위 코너에 픽셀값을 복사해 넣고 나머지는 0 으로 채우기



3) Max Unpooling

: 앞선 Pooling layer 에서 선택되었던 자리를 기억해,

Unpooling 단계에서 해당 자리에만 픽셀값을 전달

: (주의) 각 Pooling layer 가 짝이 맞아야 한다.

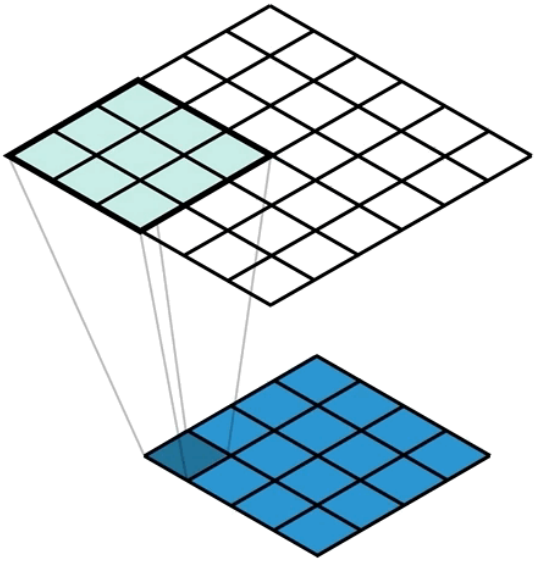
Pooling 은 앞에서부터 1~n 번, Unpooling 은 뒤에서부터 1~n 까지.

Sematic Segmentation 에서 Unpooling 기법이 중요한 이유는, 픽셀이 하나하나 분류되기 때문이다.

클래스의 Boundary 가 나눠지는 경계 픽셀이 분명히 존재할건데, 그 boundary 를 제대로 구별하려면 더 꼼꼼해야 한다.

더군다나 down sampling 과정에서 이미 정보 손실이 일어났기 때문에, 손실을 조금이라도 메꾸기 위해서는 자리라도 기억해 Unpooling 을 조금이라도 더 정확하게 해야 한다.

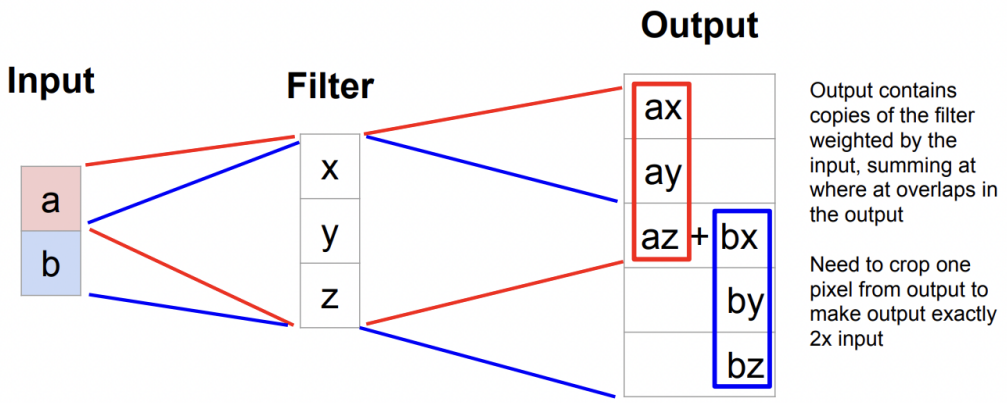
4) Transpose Convolution : Learnable Upsampling



filter 3x3, stride 1, pad 1

Transpose Convolution 은 upsampling 방식의 일종으로 input 이미지의 값을 filter 전체에 곱해서 더하는 방법이다,
위 그림을 예시로 들어 설명하자면, input(파랑색)의 각 pixel 을 3x3 필터에곱한다.
그럼 필터사이즈만큼의 벡터가 나오는데 이를 지정한 stride 만큼 움직이면서 sum up 해서 output 을 만든다.
이때 겹치는 부분은 서로 더해준다.

Transpose Convolution 은 Conv 과정에서 upsampling 을 동시에 진행하기 때문에,
upsampling 에 parameter(weight)가 관여하게 되고, 이는 upsampling 을 learnable 하게 해준다.



1D Vector Transpose Conv

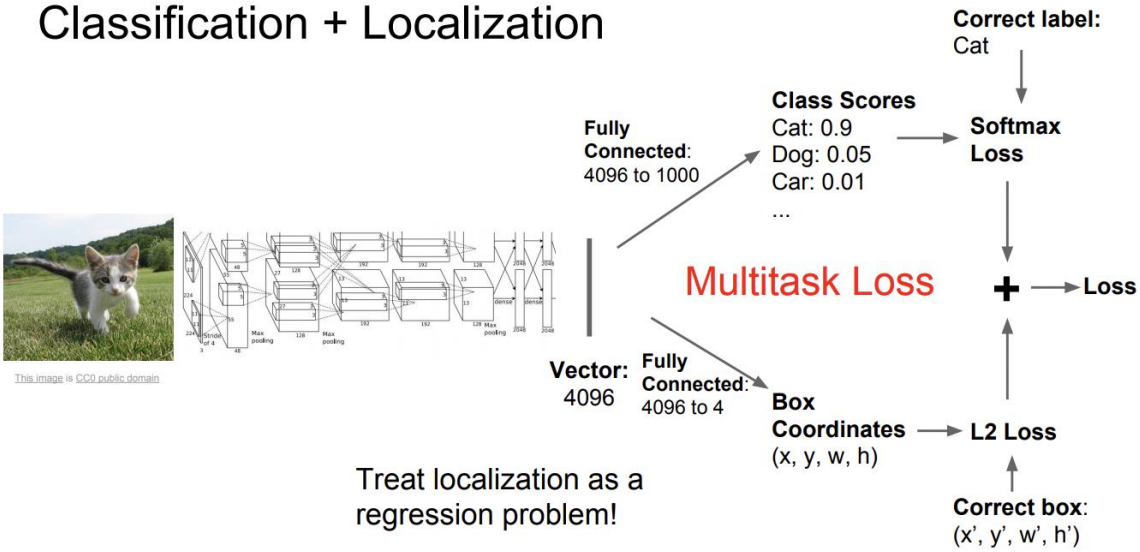
---

2. Classification + Localization

CAT

Classification task 를 수행할때는 왼쪽 이미지를 'Cat'이라고 분류한다.
여기에 Localization task 를 추가하면 'Cat'이 어디에 위치해있는지를 Bounding box 를 그려 파악한다.

한마지로 Localization 은 oject 가 어디에 있는지를 파악한다.



방법은 Classification 과정과 비슷하다.

위의 예시를보면 input 은 우선 AlexNet 을 거쳐 FC layer 를 마지막에 만나게 된다.

이때 Box coordinates 는 4 개의 값이 필요하니, Fully connected Layer 는 4096 to 4 로 vector 를 바꾼다.

그럼 두개의 Loss 가 생기는데 이런 경우를 Multitask Loss 라고 한다.

Multitask Loss

대개의 경우, Loss 에 따라 경사하강을 얼만큼 할지가 정해지는데, 기준이 되는 Loss 가 두개일때는 어떻게 해야될까?

그럴땐 따로 Hyperparameter 를 두어서 각각의 loss 를 반영할 비율을 정해줘야 한다.

하지만 이 hyperparameter 를 정하는게 쉽지 않다.

보통 hyperparameter 를 정할땐 실험적으로 다양한 조합으로 test 를 하면서 Loss 가 어떻게 변하는지 관찰을 한다.

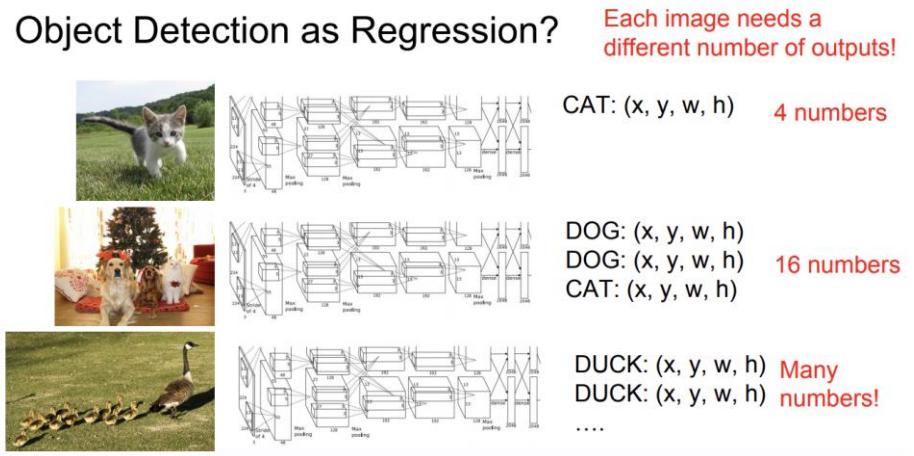
하지만 weighting hyperparemeter 의 경우, loss 의 절대값에 곱해지는 값으로 Loss 의 scale 에 직접적으로 관여하기 때문에 Loss 끼리의 비교가 불가능하다.

이런 경우에는 보통 model 을 평가할수 있는 다른 metric 을 두어서, 그 Metric 을 기준으로 hyperparameter 를 고르는 방법을 택한다.

3. Object Detection

Object Detection 은 한 이미지 내에서 object 를 탐지하고 그게 어떤 Class 에 속하는지까지 정한다.
그냥 Classification 과 마찬가지로 Class 의 종류를 미리 정의해야 한다는 공통점이 있지만,
여태까지 하나의 이미지에 하나의 object 만 있던것과는 달리 Object Detection 은 감별해야할 Object 가 몇개인지 정해져 있지 않다.

1) Regression ❌

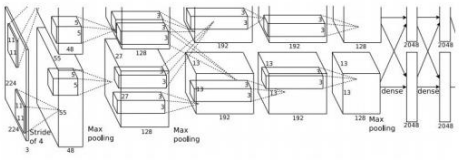


Output 의 개수가 정해져 있지 않기 때문에 문제는 더 어려워 진다.
더군다나 각 물체의 좌표를 Regression 으로 예측하는것도 까다롭다.

2) Sliding Window ❌

Object Detection as Classification: Sliding Window

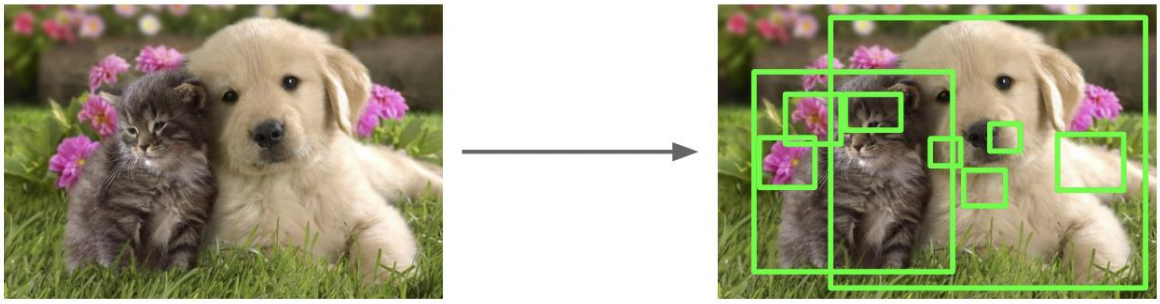
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

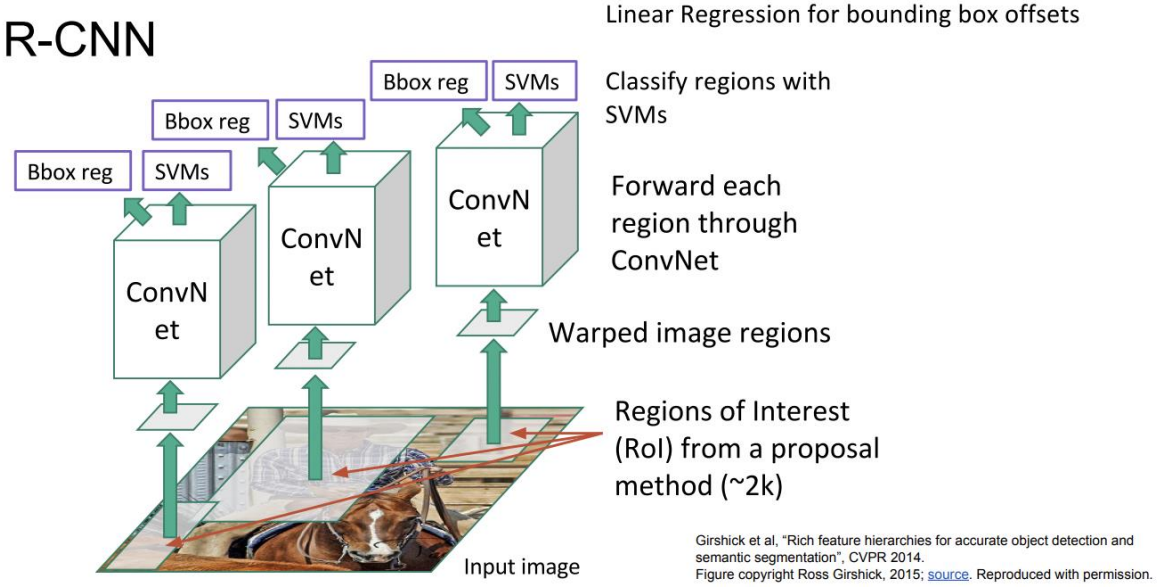
Sliding Window 를 사용해서 각 crop 마다 어떤 Class 에 속하는지 하나하나 분류할수도 있다.
하지만 어떤 좌표에서 어떤 크기로 crop 을 할지 정하는게 매우 어려우며 모든 경우의 수를 다 해보는건 불가능에 가깝기 때문에 Sliding window 방식은 사용하지 않는다.

3) Region Proposal (Not DeepLearning) ❌



Region Proposal 은 DL 은 아니지만 network 에서 많이 활용되는 방법이다.
이미지에서 뭔가 Blobby 한 뭉텅이가 있는 부분을 감지해서 여기에 물체가 있는거같다! 라고 예측이 되는 region 을 1000~2000 개 정도 뱉어낸다. 그럼 높은 확률로 2000 개의 region 중에 물체가 있다고 한다.

4) R - CNN 🔺



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

1. 위에서 본 Region Proposal 방식을 사용해 ROI. Reason of Interest 를 추출한다.

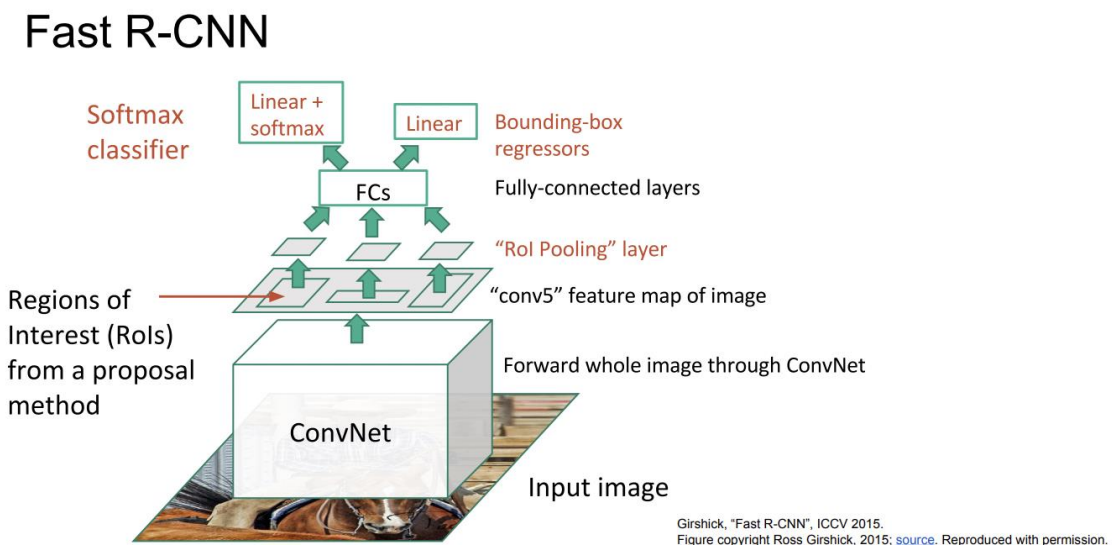
> crop 된 region 의 사이즈는 다 제각각이다. 따라서 region 들을 동일한 사이즈로 warp, 일반화 시켜주는 과정이 필요하다.

2. 각각의 warped 된 region 은 이유 Conv Net 을 거쳐 classifiy 된다.

3. Region 이 detect 를 가능하게 도와줄진 몰라도 해당 region 이 object Boundary 를 정확하게 커버하고 있지 않을 수 있다.
따라서 마지막에 Bounding Box 를 수정해서 완벽하게 하는 과정이 필요하다.

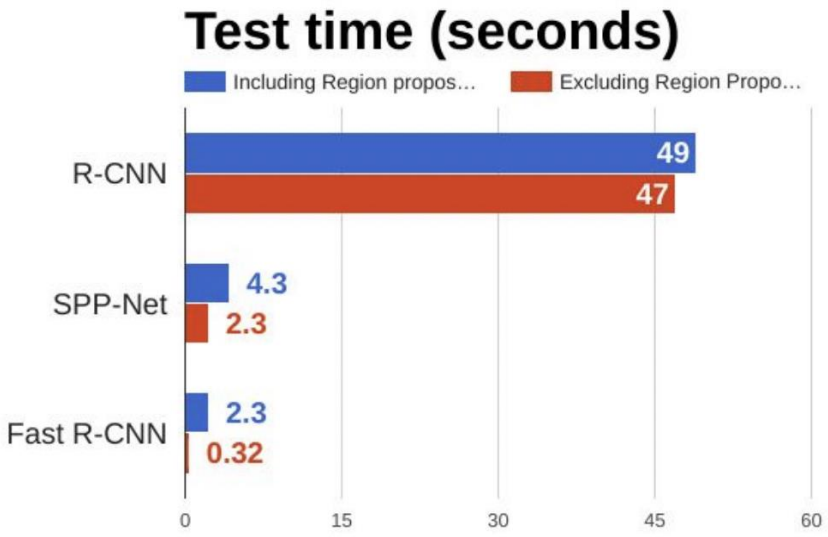
하지만 Training 이 느리고 testing 시에도 하나의 input 에 30 초씩 걸리는 등 단점들이 조금 치명적이다.
그래서 등장한게 **Fast R-CNN**!

5) Fast R - CNN ⭕



얘는 이미지를 통째로 COnv Net 에 넣어 추출한 feature map 에서 ROI 를 추출한다.
나머지 과정은 R-CNN 과 동일하다.

단점: Regin Proposal time 이 오래걸린다.

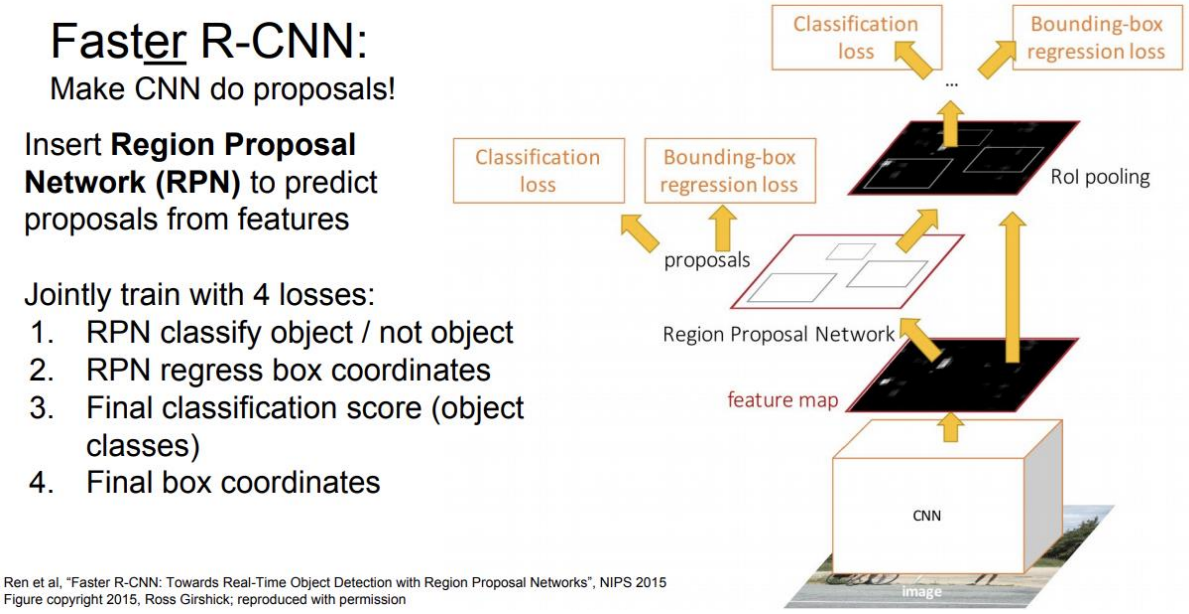


보다시피 Fast R-CNN 이 일반 R-CNN 보다 훨씬 빠른걸 알 수있다, 그림에는 없지만 Training 도 대략 10 배정도 빠르다.

이는 바로 ConvNet 이 shared components 를 효율적으로 처리하기 때문이다.

근데 이렇게 되다보니 Region Proposal(파란색)에 걸리는 시간이 나머지 시간보다 몇배는 오래걸리게 되었다.

6. Faster R - CNN ㅇㅇㅇ



그래서 또한번 등장한게 Faster R-CNN 이다.

Region Proposal 을 Network 로 따로 만들어 전체 Network 내부에서 처리하기 때문에

Fast R-CNN 처럼 Region Proposal 에 걸리는 시간때문에 bottleneck(병목현상)되는 현상이 발생하지 않는다.

---

4. Instance Segmentation

Semantic Segmentation 과 Object Detection 의 hybrid task.

각각의 object 가 어떤 class 에 속하는지 + 각 pixel 이 어떤 class 에 속하는지로 객체의 종류와 위치를 파악한다.