Cs231n Lecture 11 summary
Detection and Segmentation

1. Semantic Segmentation
2. Instance Segmentation
3. Classification + Localization
4. Object Detection
    - R-CNN
    - Fast R-CNN
    - Faster R-CNN
    - Yolo / SSD

1. Semantic Segmentation

   지금까지 배운 것은 앞에서 image가 들어오면 어떤 deep neural network를 통과해 결과값이 나오는 것이었다. 즉 이미지가 들어오면 카테고리가 출력이 되는 것이다. 하지만 이제 다른 Computer Vision tasks에 대해 배우게 된다.

   먼저 Semantic Segmentation은 입력으로 이미지가 들어오면 출력으로 이미지의 픽셀에 카테고리를 정하게 된다. 일종의 classification 분류 문제라고 할 수 있다. 다른 점은 이미지 전체에 카테고리가 매겨지는 것이 아니라 모든 픽셀에 카테고리가 매겨진다. 그래서 이미지에 같은 개체가 여러 개여도 하나로 인식되는 단점이 있다. 이 단점을 해결한 것이 instance segmentation이다.

2. Instance segmentation

   위의 예제에서 Semantic segmentation을 위해 sliding window 방법을 생각해 볼 수 있다. 입력 이미지를 작은 단위로 쪼개고 이 단위 부분을 classification을 진행한다. 하지만 이 방법은 비용이 크기 때문에 좋은 방법이 아니다. 그리고 만약 인접해 있는 영역이라면 특징을 공유할 수 있지만 그렇게 하지 못하기 때문에 좋지 않다. 그래서 개선한 방법이 fully convolutional network이다. 여기에는 FC layer가 없고 conv layer로만 구성되어 있다. 여기에 zero padding을 사용하면 이미지의 공간 정보가 손실되지 않기 때문에 결과로 C * H * W가 나온다. C는 카테고리의 개수이다. 이 출력 tensor는 입력 이미지의 모든 픽셀 값에 대해 score를 매긴 값이다. 이렇게 하려면 모든 픽셀의 classification loss를 계산하고 평균값을 취한다.

   이런 방식으로 train data를 일일히 만들려면 굉장히 힘드므로 여기에서는 모든 픽셀의 카테고리를 알고 있다는 전제하에 진행한다. 하지만 이 방식에도 문제가 있는데, 이미지의 apatial size를 계속 유지시켜 주기 때문에 계산량이 너무 커진다. 그래서 나온 방법이 downsampling과 upsampling이다. 중간에 downsampling을 하고 upsampling을 진행해 입력 이미지의 해상도와 같게 만드는 것이다.

   Upsampling의 첫번째 방법은 Unpooling이다. Nearest neighbor 방법, bed of nails 방법이 있는데 전자는 숫자를 동일하게 해서 키우는 것이고 후자는 1번째 위치에 input의 값을 넣고 나머지는 0으로 채우는 방법이다.

   두 번째 방법은 max unpooling이다. Max pooling을 진행할 때 그 위치를 기억해 값을 넣고 나머지를 0으로 채우는 방법이다. 이 방법을 쓰는 이유는 max pooling을 하게 되면 특징맵의 공간 정보를 잃게 되는데 공간 정보를 균형있게 유지할 수 있기 때문에 좋은 방법이다.

   세 번째 방법은 transpose convolution이다. 앞의 방법은 고정된 함수이지만 이 방식에서는 학습을 진행한다.

3. Classification + Localization

이는 이미지가 어떤 카테고리에 속하는지 분류하는 것 뿐만 아니라 실제 객체가 어디에 위치하는지 box를 쳐 표시하는 것이다. Localization에서는 object detection과는 다르게 객체가 오직 하나뿐이라고 가정한다. 구조는 Image classification과 비슷하지만 1개의 FC layer가 추가로 존재한다. 여기서 box corrdinate를 진행한다. 이것은 W, H, x, y의 값을 가지고 있다.

따라서 하나는 score를 반환하고 다른 하나는 bbox의 좌표를 반환한다. 이 네트워크를 학습시키려면 2개의 loss가 존재한다. 이 2개 loss를 합친 loss를 multi-task loss라고 부른다. 이 gradient를 구하려면 네트워크 가중치들의 각각의 미분 값을 계산해야 한다. 따라서 두 loss의 가중치의 합이 최종 loss이다.

이 방법을 적용할 수 있는 문제 중 가장 대표적인 것이 human pose estimation이다. 입력으로 이미지가 들어가고 출력으로는 14개의 관절위치 좌표 값이 나온다.

4. Object Detection

Object detection에서도 고정된 카테고리가 있다. 컴퓨터가 여러 개의 객체를 인식해야 하기 때문에 쉽지 않은 문제이다. 처음 많은 사람들이 사용한 방법은 sliding window 방법이다. 옆으로 진행하면서 어떤 카테고리인지 물으면서 가는 방식이다. 하지만 문제가 발생하는데, 객체가 몇 개인지 그리고 어디에 존재하는지, 크기가 얼마나 되는지 알 수 없다. 그래서 region proposal 방식을 사용한다. 이 방식은 객체가 있을 법한 후보를 찾아낸 뒤 CNN의 입력으로 해서 추출한다.

이 방법이 R-CNN이다. 이미지가 주어지면 region proposal을 얻기 위해 RPN을 수행한다. Region proposal은 다른 말로 ROI라고도 하는데, 따라서 총 2000개의 ROI를 얻어낸다. 하지만 사이즈가 제각각 이므로 고정된 크기로 바꿔준다. 그리고 이것을 CNN에 통과시킨다. Supervised 학습이기 때문에 이미지 내 모든 객체의 bbox가 있어야 train이 가능하다. 그러나 R-CNN은 계산비용이 많고 느리다는 단점이 있다.

그래서 제안된 것이 fast R-CNN이다. 각 ROI마다 CNN을 수행하지 않고 전체 이미지에서 수행하기 때문에 고해상도 feature map을 얻을 수 있다. 하지만 region proposal을 계산하는 데 걸리는 시간이 길기 때문에 병목 현상이 발생한다.

이를 해결하기 위해 faster R-CNN이 등장한다. 이미지가 입력으로 들어오면 전체 CNN을 통해 feature map을 뽑아낸 뒤 계산한다. 따라서 4개의 loss가 한번에 계산된다.

다음은 YOLO와 SSD이다. 이 네트워크들은 Feed forward를 한 방향으로 수행하는 네트워크이다. 각 task를 따로 계산하지 않고 하나의 Regression 문제로 풀어보자는 것이다. 출력은 3d를 가지는 Tensor가 된다.