

5

Object detection

강의 소개

영상 내에 존재하는 객체를 인식하는 방법은 픽셀마다의 클래스를 분류하는 segmentation 뿐만 아니라 물체 하나하나마다 bounding box 단위의 예측도 있습니다.

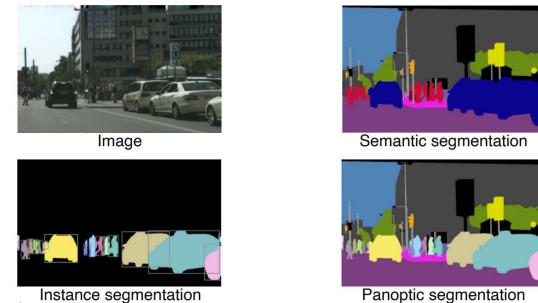
이러한 task를 object detection이라고 하며 자율주행, CCTV 등 다양한 분야에 활용되고 있습니다. Object detection을 위한 모델은 크게 one-stage detector 와 two-stage detector로 구분할 수 있는데 시대의 흐름을 따라 각각의 모델들의 발전사를 소개합니다.

Further Question

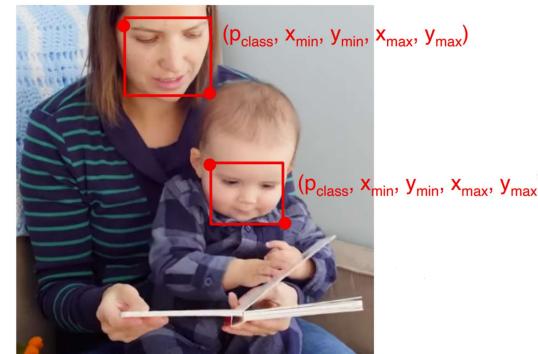
- (1) Focal loss는 object detection에만 사용될 수 있을까요?
- (2) CornerNet/CenterNet은 어떤 형식으로 네트워크가 구성되어 있을까요?

Object detection

What is object detection?



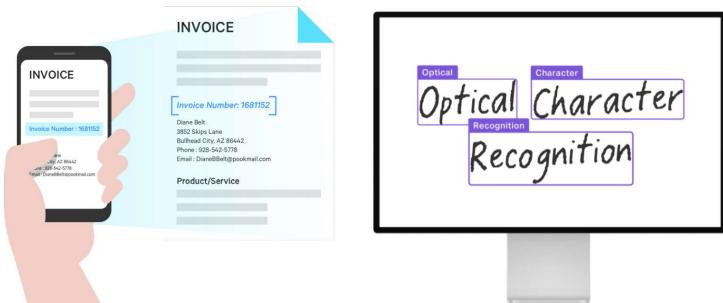
- Semantic segmentation
 - 같은 카테고리인 다른 객체를 분류 X
- Instance segmentation / Panoptic segmentation
 - 같은 카테고리인 다른 객체를 분류 O



- Object detection = Classification + Box localization

What are the applications of object detection?

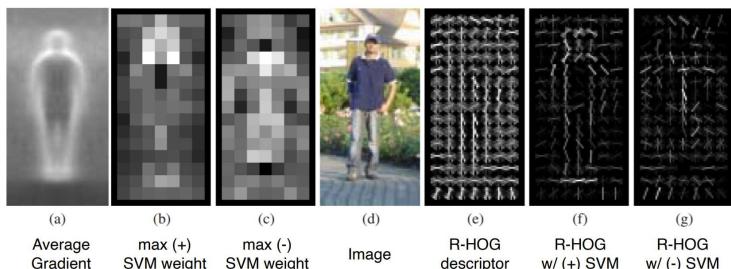
- Autonomous driving
- Optical Character Recognition (OCR)



Two-stage detector (R-CNN family)

Traditional methods - hand-crafted techniques

Gradient-based detector (ex. HOG)



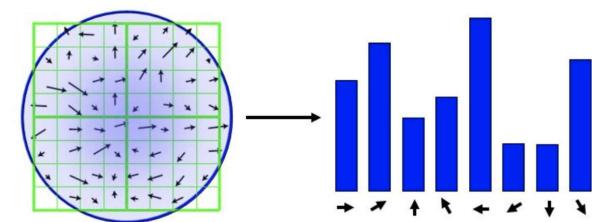
- HOG = Histogram of Oriented Gradients

▼ histogram of oriented gradients (HOG)

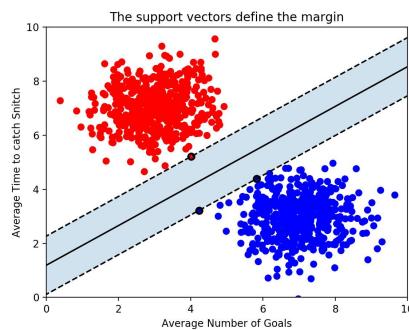
- **Image의 지역적인 Gradient**을 해당 영상의 특징으로 사용하는 방법
- 가장 중요한 특징은 Edge의 양과 방향을 구분하는 특성을 가지고 있다. 또한, Overlap을 이용하여 계산하기 때문에 어느 정도 Shift

에도 적용할 수 있는 능력이 있다. 마치 Haar-like가 영역으로 특징을 구하기 때문에 잡음에 둔감한 것과 같이 HOG의 Overlap의 이유는 어느정도의 변화를 받아 들일 수 있다는 것이다.

- 참고 : [\[Image Processing\] HOG Algorithm \(tistory.com\)](#)
- Histogram of Gradient은 픽셀의 변화량의 각도와 크기를 고려하여 히스토그램 형태의 feature를 추출하는 방법
- 참고 + 읽어보기 : [Histogram of Gradient \(HoG\) \(donghwa-kim.github.io\)](#)



- SVM = Support Vector Machine → 선형 classifier
- ▼ SVM
 - 서포트 벡터 머신(이하 SVM)은 결정 경계(Decision Boundary), 즉 분류를 위한 기준 선을 정의하는 모델이다. 그래서 분류되지 않은 새로운 점이 나타나면 경계의 어느 쪽에 속하는지 확인해서 분류 과정을 수행할 수 있게 된다.



- 결정 경계는 데이터 군으로부터 최대한 멀리 떨어지는 게 좋다. 실제로 서포트 벡터 머신(Support Vector Machine)이라는 이름에서 **Support Vectors**는 결정 경계와 가까이 있는 데이터 포인트들을 의미한다. 이 데이터들이 경계를 정의하는 결정적인 역할을 하는 셈이다.
- 가운데 실선이 하나 그어져있는데, 이게 바로 '결정 경계'가 되겠다. 그리고 그 실선으로부터 검은 태두리가 있는 빨간점 1개, 파란점 2개까지 영역을 두고 점선을 그어놓았다. 점선으로부터 결정 경계까지의 거리가 바로 '마진(margin)'이다.
- 여기서 일단 결론은 하나 얻을 수 있다. **최적의 결정 경계는 마진을 최대화한다.**
- 그리고 위 그림에서는 x축과 y축 2개의 속성을 가진 데이터로 결정 경계를 그었는데, 총 3개의 데이터 포인트(서포트 벡터)가 필요했다. 즉, **n개의 속성을 가진 데이터에는 최소 n+1개의 서포트 벡터가 존재한다는 걸 알 수 있다.**
- 이번엔 SVM 알고리즘의 장점을 하나 알 수 있다.
- 대부분의 머신러닝지도 학습 알고리즘은 학습 데이터 모두를 사용하여 모델을 학습한다. 그런데 **SVM에서는 결정 경계를 정의하는 게 결국 서포트 벡터이기 때문에 데이터 포인트 중에서 서포트 벡터**

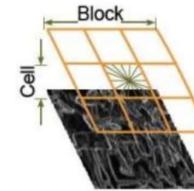
만 잘 골라내면 나머지 쓸 데 없는 수많은 데이터 포인트들을 무시할 수 있다. 그래서 매우 빠르다.

- 참고 : [서포트 벡터 머신\(Support Vector Machine\) 쉽게 이해하기 - 아무튼 위라밸\(hleecaster.com\)](#)

- R-HOG descriptor → 경계선 분포

▼ R(rectangular)-HOG descriptor

- HOG descriptor는 모든 block 영역에서 정규화된 cell 히스토그램 요소의 이어붙인 벡터(concatenated vector)가 된다.



- **R-HOG block**은 정사각형 그리드로, block당 cell의 수, cell당 픽셀 수, 그리고 cell 히스토그램당 채널 수를 파라미터로 나타낸다.

- 논문에 따르면 최적의 파라미터는 9개 히스토그램 채널을 갖는 8*8 cell 의 16*16 블록이다.

- 참고 : [공부하는 티벳여우 \(tistory.com\)](#)

- R-HOG w/ (+) SVM → 사람 O ↔ R-HOG w/ (-) SVM → 사람 X

Selective search

- bounding box proposal

▼ selective search

- Segmentation과 Exhaustive search 두 가지 방법을 결합하여 후보영역을 추천한다.

- **Segmentation** : 이미지 구조를 사용하여, 샘플링 프로세스를 안내

- **Exhaustive Search** : 모든 객체의 위치(Locations)를 찾아내는 것
- detector는
 - 1) generic detector로 candidate objects 영역을 찾기 위해 exhaustive search를 진행하고
(region proposal)
 - 2) 이 candidate object에 대해 인식 알고리즘을 실행한다.
 - → object들이 각기 다른 shape을 가지고 있다면 windows로 scan하여 region proposal 하는 것이 옳은 방법일까? (고정된 window 사이즈는 각기 다른 object의 size나 shape을 포착하기 어렵기 때문이다)
 - 만약 object recognition을 실행하기 전에 아래와 같이 이미지를 올바르게 segment하면 segmented result에 대해서 candidate object로 사용할 수 있지 않을까?
 - http://vision.stanford.edu/teaching/cs231b_spring1415/slides/slides.html#slide_1
 - 위 방법을 활용해서 제안한 것이 **selective search**- Selective search 시 고려사항
 - 1. Capture All Scales**
 - object는 이미지 내에서 어떤 크기(any scale)로도 나타날 수 있다.
 - 또한 일부 object는 다른 object보다 boundary가 명확하지 않는다.

- 따라서 selective search에서는 모든 object의 크기(scales)를 고려해야 한다.

- 이는 hierarchical 알고리즘을 사용하여 자연스럽게 달성 가능

2. Diversification

- 영역은 색상, 질감, 또는 parts가 둘러싸여 있기 때문에 객체를 형성 가능
- 음영 및 빛의 색상과 같은 lighting condition은 regions이 물체를 형성하는 방식에 영향을 미칠 수 있음.
- 따라서, 대부분의 경우 잘 작동하는 single strategy 대신,
- 모든 경우를 처리할 수 있는 diverse set of strategies가 필요.

3. Fast to Compute

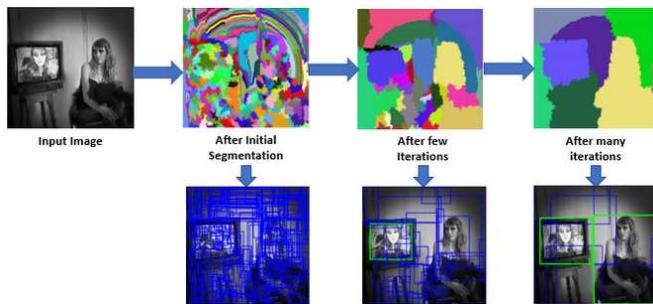
- selective search의 목표는 실제 object recognition framework에서 사용할 수 있는 가능한 object locations 집합을 생성하는 것
- 이 집합의 생성은 computational bottleneck이 되어서는 안된다.
- 따라서, 이 알고리즘은 빠르다

• selective search process

- 1. Input 이미지에 sub-segmentation 진행**
- 2. 반복적으로 작은 영역을 큰 영역으로 결합 (greedy algorithm을 사용)**

- 1) Set of regions에서, 가장 유사한 두 가지를 선택.
- 2) 이 선택한 두 가지를 한 가지, 큰 영역으로 결합한다.
- 3) 위 1, 2 steps를 여러 iteration 동안 반복한다.

- 3. Segmented region proposals를 사용하여 candidate object locations를 생성**



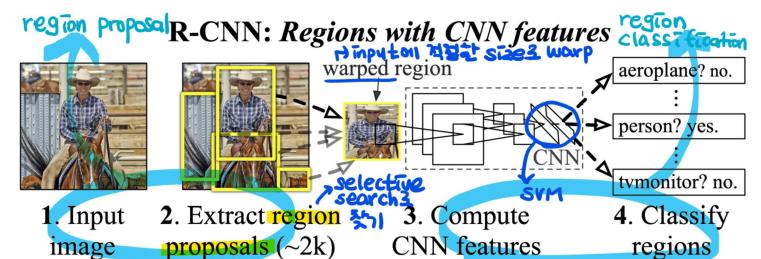
- selective search의 단점
 - region proposal 과정이 실제 object detection CNN과 별도로 이루어지기 때문에,
 - selective search를 사용하면 end-to-end로 학습이 불가능하고, 실시간 적용에도 어려움이 있음
- 참고 : [Selective Search 간단히 정리...](#) (tistory.com)



1. Over-segmentation
 - 같은 색깔끼리 묶기 (잘게 분할)
2. Iteratively merging similar regions
 - 비슷한 영역끼리 (반복적으로) 합치기 (ex. gradient 분포, 색상)
3. Extracting candidate boxes from all remaining segmentations
 - 각 영역을 포함하는 가장 tight한 bounding box

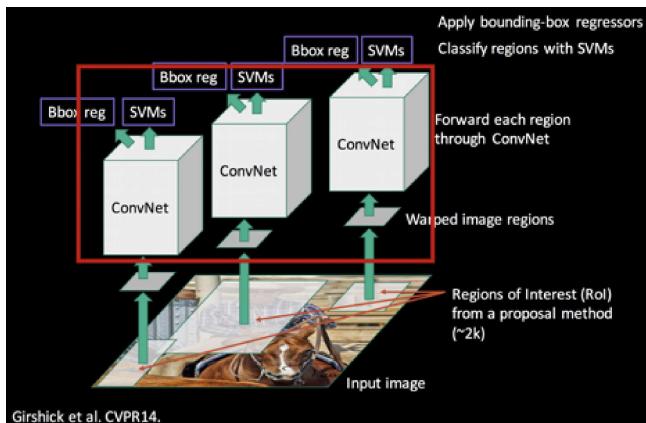
R-CNN

- object detection에 image classification network 사용



- 각 region proposal 하나하나마다 모델에 넣어야 하므로, 시간 多 소요
- ▼ R-CNN
- R-CNN은 'Regions with Convolutional Neuron Networks features'의 약자로, 즉 설정한 Region을 CNN의 feature(입력값)로 활용하여 Object Detection을 수행하는 신경망이라는 의미를 담고 있다.
 - R-CNN의 기본적인 구조는 2-stage Detector라고 한 것처럼 전체 task를 두 단계로 나눌 수 있는데, 우선 물체의 위치를 찾는 Region Proposal, 그리고 물체를 분류하는 Region Classification이다. 이

두 가지 task를 처리하기 위해 수행되는 **R-CNN의 구조**를 총 네 가지 모듈로 나눠서 살펴볼 것이다.

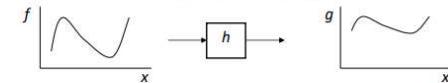


1. 이미지에 있는 데이터와 레이블을 투입한 후 카테고리에 무관하게 물체의 영역을 찾는 **Region Proposal**.
 2. 그리고 proposal 된 영역으로부터 고정된 크기의 Feature Vector를 warping/crop 하여 CNN의 인풋으로 사용한다. 여기서 CNN은 이미 ImageNet을 활용한 사전훈련된 네트워크를 사용한다.
 3. CNN을 통해 나온 feature map을 활용하여 선형 지도학습 모델인 **SVM**(Support Vector Machine)을 통한 분류, 그리고
 4. Regressor를 통한 **bounding box regression**을 진행한다
- 참고 : [R-CNN 을 알아보자 \(velog.io\)](#)

▼ image warping

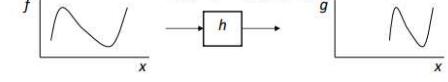
- **image filtering**: change *range* of image

$$g(x) = h(f(x))$$



- **image warping**: change *domain* of image

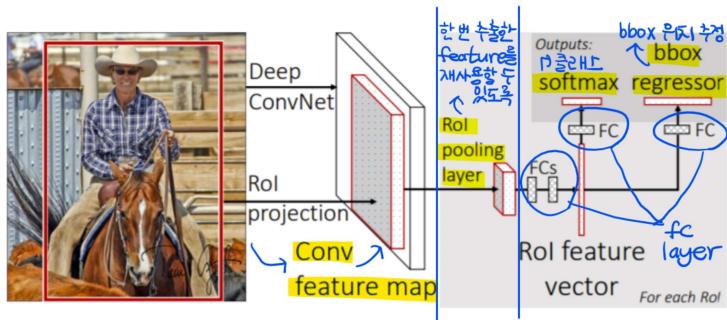
$$g(x) = f(h(x))$$



- 이미지 필터링은 이미지의 범위, 즉 y축 값을 변화시키는 것 (색 변화 등)
- 이미지 와핑은 이미지의 도메인, 즉 x 축 값이 변화시키는 것 (위치 변화 만)
- 정리하자면 이미지 필터링과 이미지 와핑의 차이는 위치 변화 유무입니다.
- 참고 : [컴퓨터비전] 7. Transformations and warping :: 컴퓨터에 빠진 학생의 기록 ([tistory.com](#))
- 읽어보기 : [OpenCV] 이미지 와핑(Warping), 탑뷰(Top View) : 네이버 블로그 ([naver.com](#))

Fast R-CNN

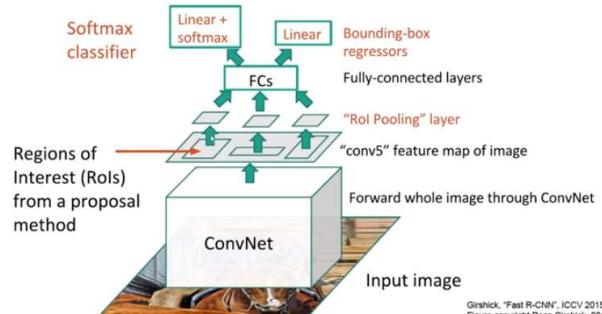
- Recycle a pre-computed feature for multiple object detection → **RoI pooling layer** 도입
- RoI = Region of Interest
 - region proposal이 제시한 object의 위치 후보
 - bounding box가 주어지면 RoI에 해당하는 feature만 추출



1. Conv. feature map from the original image
 - 미리 feature map 추출
 - fully convolutional layer는 input의 크기에 상관없이 feature map 추출 가능
2. RoI feature extraction from the feature map through RoI pooling
 - fixed dimension을 갖도록 resize
 - 일정 크기로 resampling
3. Class and box prediction for each RoI

▼ Fast R-CNN

Fast R-CNN

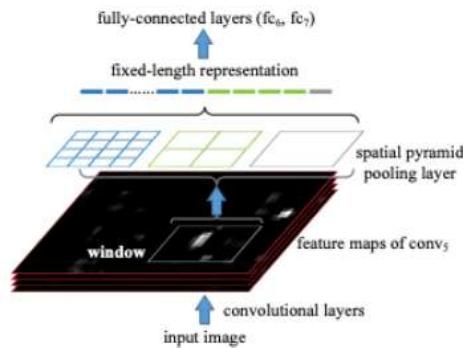


- Fast R-CNN 프로세스
 - 1-1. R-CNN에서와 마찬가지로 Selective Search를 통해 RoI를 찾는다.
 - 1-2. 전체 이미지를 CNN에 통과시켜 feature map을 추출한다.
 2. Selective Search로 찾았었던 RoI를 feature map크기에 맞춰서 projection시킨다.
 3. projection시킨 RoI에 대해 RoI Pooling을 진행하여 고정된 크기의 feature vector를 얻는다.
 4. feature vector는 FC layer를 통과한 뒤, 구 브랜치로 나누게 된다.
 - 5-1. 하나는 softmax를 통하여 selective search로 찾은 box의 위치를 조정한다.
- 참고 : [2. faster R-CNN :: 공부하려고 만든 블로그 \(tistory.com\)](#).

▼ RoI pooling

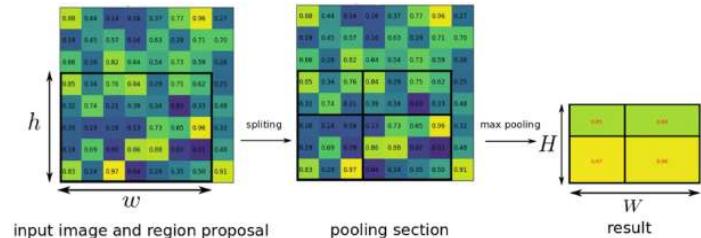
- R-CNN에서 CNN output이 FC layer의 input으로 들어가야했기 때문에 CNN input을 동일 size로 맞춰줘야 했다. 따라서 원래 이미지에서 추출한 RoI를 crop, warp을 통해 동일 size로 조정했다.

- 그러나 실제로 "FC layer의 input이 고정인거지 CNN input은 고정이 아니다" 따라서 CNN에는 입력 이미지 크기, 비율 관계없이 input으로 들어갈 수 있고, FC layer의 input으로 들어갈때만 size를 맞춰주기만 하면된다.
- 여기서 Spatial Pyramid Pooling(SPP)이 제안된다.



- SPP에서는 먼저 이미지를 CNN에 통과시켜 feature map을 추출한다.
- 그리고 미리 정해진 4x4, 2x2, 1x1 영역의 피라미드로 feature map을 나눠준다. 피라미드 한칸을 bin이라 한다.
- bin내에서 max pooling을 적용하여 각 bin마다 하나의 값을 추출하고,
- 최종적으로 피라미드 크기만큼 max값을 추출하여 3개의 피라미드의 결과를 쭉 이어붙여 고정된 크기 vector를 만든다.
- 정리하자면, 4x4, 2x2, 1x1 세 가지 피라미드가 존재하고, max pooling을 적용하여 각 피라미드 크기에 맞게 max값을 뽑아낸다.
- 각 피라미드 별로 뽑아낸 max값들을 쭉 이어붙여 고정된 크기 vector를 만들고 이게 FC layer의 input으로 들어간다.
- 따라서 CNN을 통과한 feature map에서 2천 개의 region proposal을 만들고 region proposal마다, SPPNet에 집어넣어 고정된 크기의 feature vector를 얻어낸다.

- 이 작업을 통해 모든 2천개의 region proposal마다 해야했던 2천번의 CNN연산이 1번으로 줄었다.
- Fast R-CNN에서 적용된 1개의 피라미드 SPP로 고정된 크기의 feature vector를 만드는 과정을 "RoI Pooling"이라 한다.



위 그림의 가장 좌측 그림이 feature map이고 그 안에 $h \times w$ 크기의 검은색 box가 투영된 RoI이다.

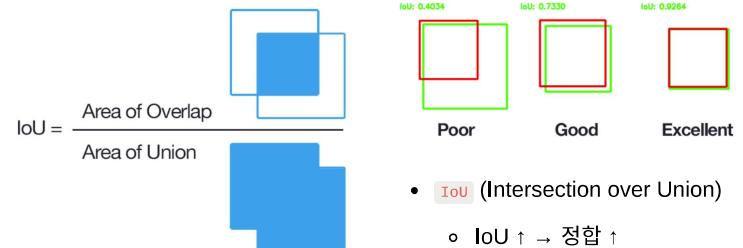
- Fast R-CNN에서 먼저 입력 이미지를 CNN에 통과시켜 feature map을 추출한다.
- 그 후 이전에 미리 Selective search로 만들어놨던 RoI(region proposal)을 feature map에 projection시킨다.
 - (1) 미리 설정한 $H \times W$ 크기로 만들어주기 위해서 $(h/H) * (w/W)$ 크기 만큼 grid를 RoI위에 만든다.
 - (2) RoI를 grid크기로 split시킨 뒤 max pooling을 적용시켜 결국 각 grid 칸마다 하나의 값을 추출한다.
- 위 작업을 통해 feature map에 투영했던 $h \times w$ 크기의 RoI는 $H \times W$ 크기의 고정된 feature vector로 변환된다.
 - "원래 이미지를 CNN에 통과시킨 후 나온 feature map에 이전에 생성한 RoI를 projection시키고, 이 RoI를 FC layer input 크기에 맞게 고정된 크기로 변형할 수가 있다"

- R-CNN에서는 CNN을 통과한 후 각각 서로 다른 모델인 SVM(classification), bounding box regression(localization) 안으로 들어가 forward됐기 때문에 연산이 공유되지 않았다. (* bounding box regression은 CNN을 거치기 전의 region proposal 데이터가 input으로 들어가고, SVM은 CNN을 거친 후의 feature map이 input으로 들어가기에 연산이 겹치지 않는다.)
- 그러나 위 그림을 다시보면 **RoI Pooling**을 추가함으로써 이제 RoI영역을 CNN을 거친후의 feature map에 투영시킬 수 있었다.
- 따라서 **동일 data**가 각자 softmax(classification), bbox regressor(localization)으로 들어가기에 **연산을 공유한다**.
- 이는 이제 모델이 **end-to-end**로 한 번에 학습시킬 수 있다는 뜻이다.
- 참고 : ([노문리뷰](#)) Fast R-CNN 설명 및 정리 :: 프라이데이 ([tistory.com](#))

Faster R-CNN

- End-to-end object detection by neural region proposal**
 - end-to-end → object detection의 모든 파트가 neural network 기반
 - neural region proposal** → region proposal을 neural network로 개선 시킴
 - Fast R-CNN까지 region proposal은 selective search에 의존 → 별도의 알고리즘을 사용하므로, 데이터만으로 성능을 개선하는 데에 한계 O
- ▼ selective search의 한계
 - Fast R-CNN에서도 R-CNN에서와 마찬가지로 **RoI**를 생성하는 **Selective search** 알고리즘은 CNN외부에서 진행되므로 이 부분이 속도의 bottleneck이다.
 - 참고 : ([노문리뷰](#)) Fast R-CNN 설명 및 정리 :: 프라이데이 ([tistory.com](#))

IoU (Intersection over Union)

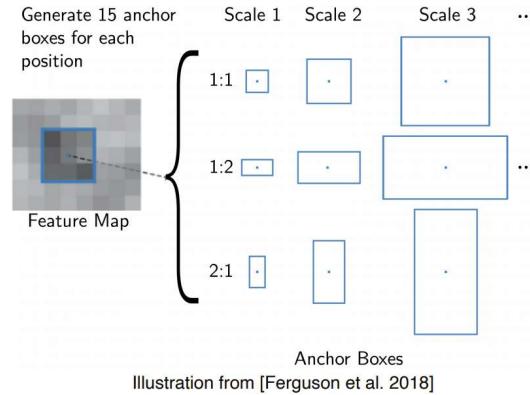


- **IoU** (Intersection over Union)
 - $\text{IoU} \uparrow \rightarrow$ 정합 ↑

Anchor boxes

▼ anchor boxes

- 각 픽셀을 중앙에 두고 크기와 종횡비가 서로 다른 Bounding Boxes를 생성합니다.
- 이러한 Bounding Boxes를 Anchor Boxes라고 하며, 이를 통해 물체를 감지하게 됩니다.
- 참고 : [[Computer Vision](#)] Anchor Boxes(앵커 박스) - 1 ([tistory.com](#))
- region proposal 구현
- A set of **pre-defined bounding boxes**
 - IoU with GT > 0.7 → **positive sample** → bounding box O
 - IoU with GT < 0.3 → negative sample → bounding box X

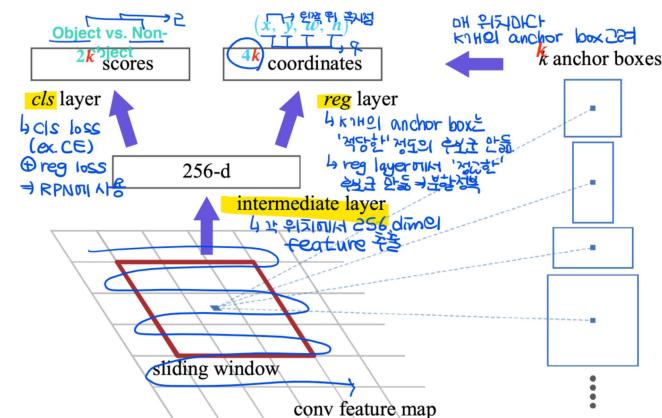


- anchor box의 ratio, scale도 여러 가지

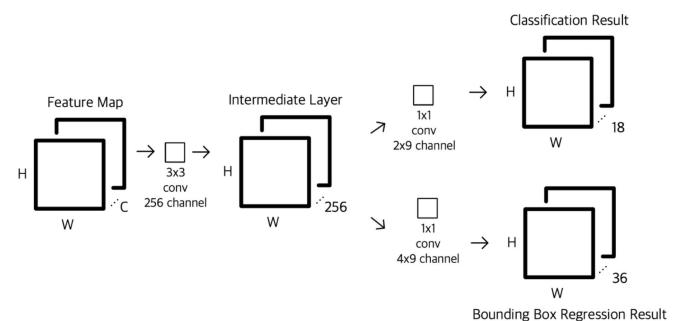
Faster R-CNN

- Time-consuming selective search (third party)을 사용하는 대신 \rightarrow Region Proposal Network (RPN)로 대체

▼ 🔍 Region Proposal Network



- RPN의 input 값은 이전 CNN 모델에서 뽑아낸 feature map이다. Region proposal을 생성하기 위해 feature map 위에 $n \times n$ window를 sliding window 시킨다.
- 참고 : [RPN\(Region Proposal Network\) 정리 \(velog.io\)](#)



- Faster R-CNN 알고리즘

- CNN을 통해 뽑아낸 feature map을 입력으로 받는다. 이 때, feature map의 크기를 $H \times W \times C$ (세로*가로*채널수)로 잡는다.
- feature map에 3×3 convolution을 256 channel (or 512 channel) 만큼 수행한다.
이 때, padding을 1로 설정해주어 $H \times W$ 가 보존되도록 진행하며, intermediate layer 수행 결과 $H \times W \times 256$ (or 512) 크기의 두 번째 feature map을 얻는다.
- 2 번째 feature map을 입력으로하여 classification (cls layer)과 bounding box regression (reg layer) 예측 값을 계산한다.
+) 이 과정은 Fully Connected Layer가 아니라 1×1 컨볼루션을 이용하여 계산하는 Fully Convolution Network의 특징을 갖는다. 이는 입력 이미지의 크기에 상관없이 동작할 수 있도록 하기 위함이며 자세한 내용은 [Fully Convolution Network](#)을 참고 하길 바란다.

4. Classification layer에서는 1×1 컨볼루션을 (2(Object인지 아닌지를 나타냄) \times 9(앵커(Anchor) 개수)) 채널 수 만큼 수행하며, 그 결과로 $H \times W \times 18$ 크기의 feature map을 얻는다. $H \times W$ 상의 하나의 인덱스는 피쳐맵 상의 좌표를 의미하고, 그 아래 18개의 채널은 각각 해당 좌표를 Anchor로 삼아 k개의 앵커 박스들이 object인지 아닌지에 대한 예측 값을 담는다. 즉, 한 번의 1×1 컨볼루션으로 $H \times W$ 개의 Anchor 좌표들에 대한 예측을 모두 수행할 수 있다. 이제 이 값들을 적절히 reshape 해준 다음 Softmax를 적용하여 해당 Anchor가 Object일 확률 값을 얻는다.

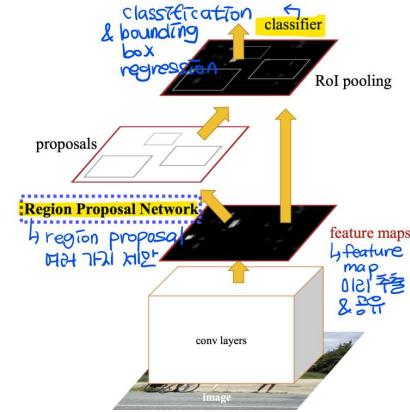
- + 여기서 앵커(Anchor)란, 각 슬라이딩 윈도우에서 bounding box의 후보로 사용되는 상자를 의미하며 이동불변성의 특징을 갖는다.

5. Bounding Box Regression 예측 값을 얻기 위한 1×1 컨볼루션을 (4×9) 채널 수 만큼 수행하며 regression이기 때문에 결과로 얻은 값을 그대로 사용한다.

6. Classification의 값과 Bounding Box Regression의 값을 통해 ROI를 계산한다.

- + 먼저 Classification을 통해서 얻은 물체일 확률 값을 정렬한 다음, 높은 순으로 K개의 앵커를 추려낸다. 그 후, K개의 앵커들에 각각 Bounding box regression을 적용하고 Non-Maximum-Suppression을 적용하여 ROI를 구한다.

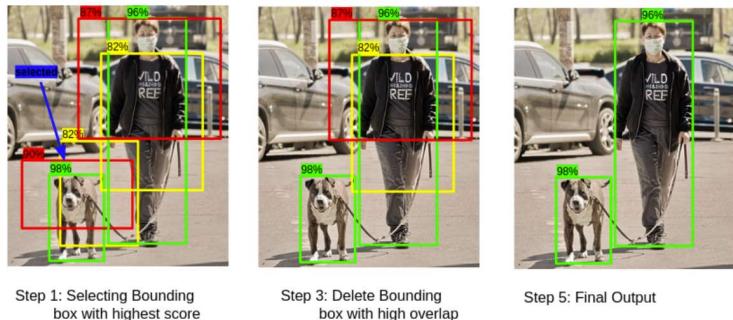
- 참고 : 2. faster R-CNN :: 공부하려고 만든 블로그 ([tistory.com](#))



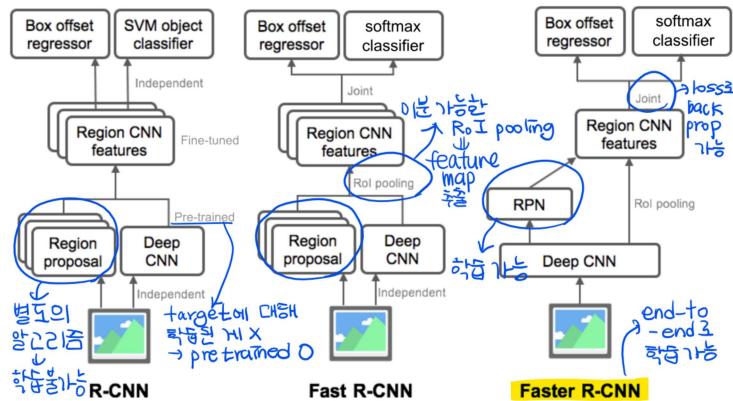
Non-Maximum Suppression (NMS)

▼ NMS

- NMS를 가장 쉽게 설명 하자면 '얼마나 겹쳐 있는지'를 판단하고 일정 크기 이상 겹칠 경우 삭제 하는 방법이다.
- IOU 방법을 이용하여 나온 비율 (0~1까지) 값을 가지고 비교를 하여 겹치는 박스를 제거 하는 방법이다.
- 참고 : [NMS\(Non-maximum Suppression\) 이란? IOU부터 알자](#) ([tistory.com](#))
- 여러 anchor box 中 filtering
 - Step 1: Select the box with the highest objectiveness score
 - Step 2: Compare IoU of this box with other boxes
 - Step 3: Remove the bounding boxes with IoU 50%
 - Step 4: Move to the next highest objectiveness score
 - Step 5: Repeat steps 2-4

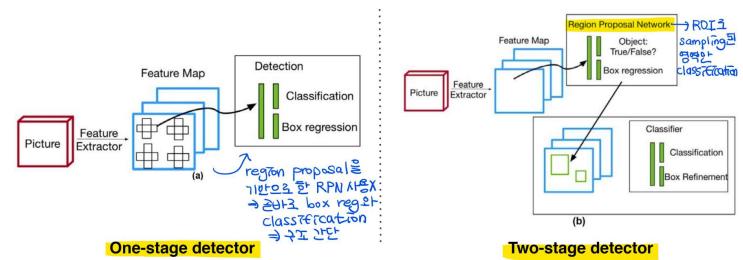


Summary of the R-CNN family



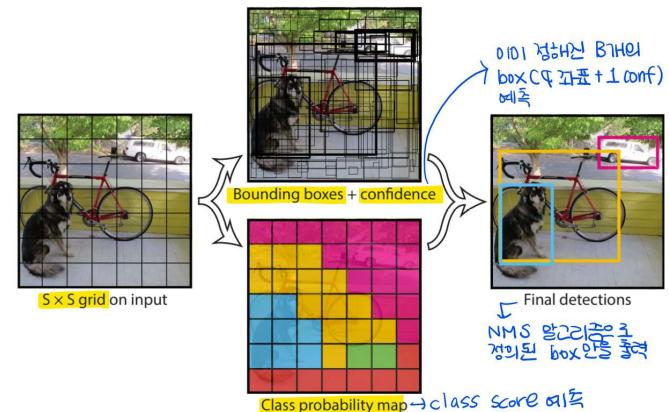
Single-stage detector

One-stage VS. Two-stage



- One-stage
 - No explicit ROI pooling
 - 정확도를 포기하더라도 속도 ↑ ⇒ real time detection 가능

YOLO (You Only Look Once)



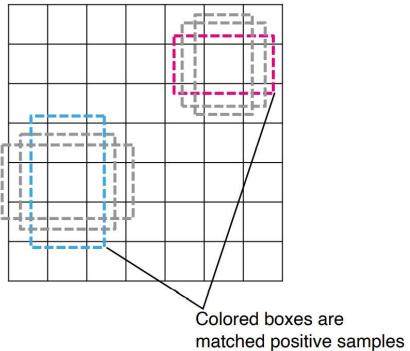
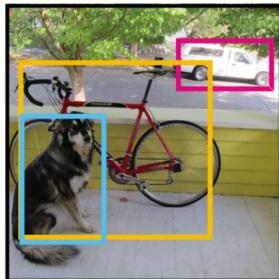
▼ YOLO

- 하나의 이미지 데이터를 입력하면 물체들을 찾아 **bounding box**와 가장 높은 확률의 **box**를 찾는다. (중앙 상단)

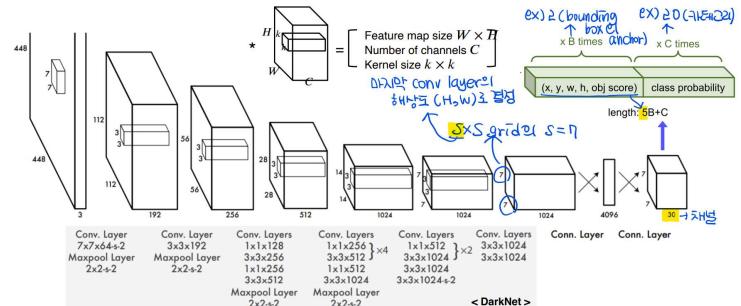
- 각 그리드셀에서 물체의 확률과 IOU값을 비교해 가장 높은 확률을 가진 클래스를 검출해 해당 그리드셀의 클래스를 확정 (중앙 하단)
- 확정된 클래스의 확률이 일정한 값(threshold)이상의 그리드셀을 이용해 bounding box를 그린다. (우측 사진)
- 참고 : [YOLO \(You Only Look Once \)란 무엇인가 — 두리안의 코딩 나무 \(tistory.com\)](#).

▼信心 confidence

- 예측 box와 실제 정답 사이의 IOU
- 만약 cell에 object가 존재하지 않는다면, confidence score는 0이 된다. confidence score는 이 시스템이 물체를 포함한다는 예측을 얼마나 확신하는지, 박스에 대한 예측이 얼마나 정확한지를 의미한다.
- 참고 : [YOLO\(You Only Look Once\)란? \(tistory.com\)](#).



- ground truth와 매치된 영역을 positive bounding box로 설정



Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45

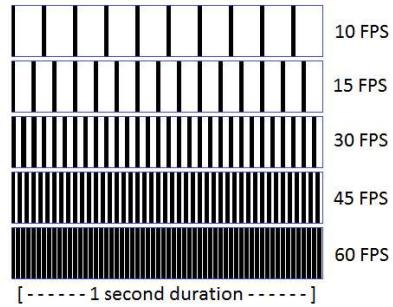
Less Than Real-Time

Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16 [28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

- mAP : mean Average Precision (metrics)
- FPS : Frame Per Second (speed)

▼ FPS

- 1초당 몇 이미지 Frame이 우리에게 보여지는지

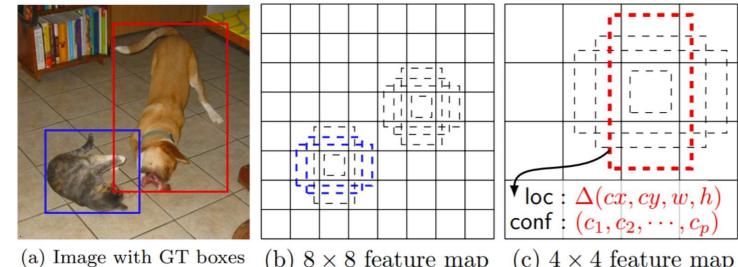


- 위 사진을 보면 10 FPS는 중간에 텀이 많아서 영상이 뚝 뚝 뚝 끊겨 보일 것이다. 하지만 60FPS는 끊길새 없이 계속 바뀌니 영상이 연속적으로 보일 것이다.
 - fps가 높을 수록 빠르다
 - 참고 : [YOLO에서 mAP와 FPS란? \(tistory.com\)](#).

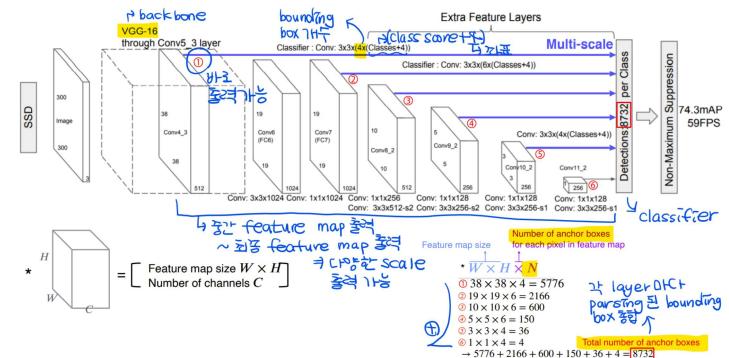
SSD

Single Shot MultiBox Detector

- YOLO → 마지막 layer에서만 prediction 진행 \Rightarrow localization ↓
 - ▼  YOLO 한계
 - 기존의 yolo에서는 CNN을 통과한 마지막 레이어의 피쳐맵만 사용하여 작은 물체에 대한 정보가 사라진다는 비판이 있었습니다
 - 참고 : 갈아먹는 Object Detection [8].yolov2.yolo9000 (tistory.com)



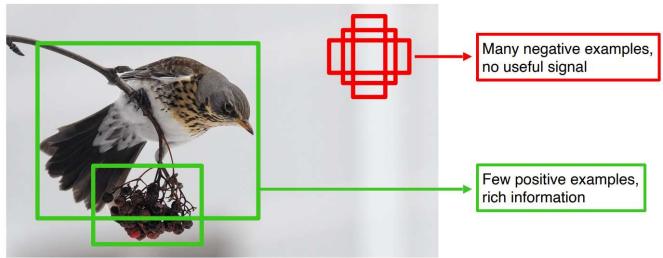
- 각 해상도에 적절한 bounding box scale 구조 만들 \Rightarrow box 크기 다름 (파란색, 빨간색)
 - The use of **multi-scale outputs** attached to multiple feature maps enable effectively modeling a diverse space of possible box shapes



Single-stage detector vs. two-stage detector

Focal loss

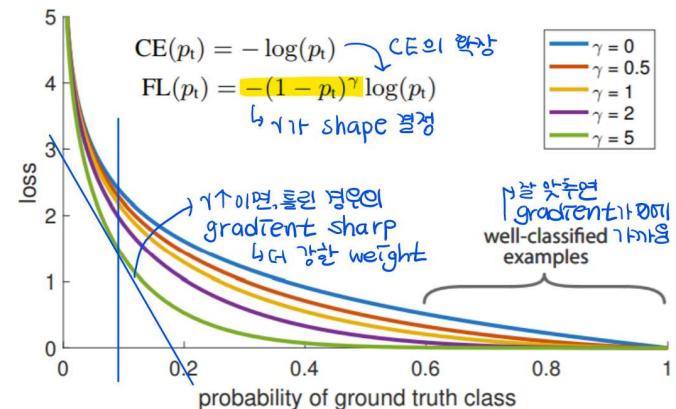
Class imbalance problem



- negative anchor box 개수 >> positive anchor box 개수
 - 배경이 넓고 + 물체는 일부 \Rightarrow positive sample 개수 $\downarrow \leftrightarrow$ negative sample 개수 \uparrow
- Single-stage detectors (ex. YOLO, SSD) \Rightarrow RoI pooling X \Rightarrow 모든 영역의 loss 계산해야 \rightarrow class imbalance에 취약

Focal loss

- improved cross entropy loss
- class imbalance 처리
 - over-weights** hard or misclassified examples
 - down-weights** easy examples
 - 잘 맞추면 loss $\downarrow \leftrightarrow$ 틀리면 loss \uparrow

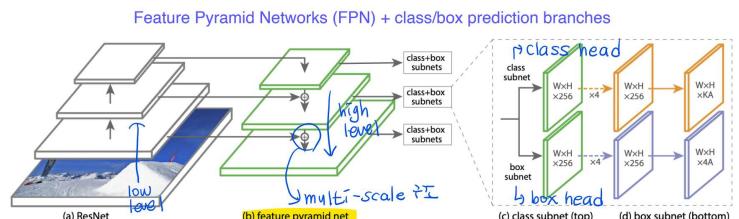


RetinaNet

▼ RetinaNet

- RetinaNet은 기존 detection 알고리즘에서 classifier의 loss로 사용되던 cross entropy loss에서 변형된 focal loss를 제안했습니다. **focal loss**를 이용해 class imbalance 문제를 해결한 논문입니다.
- 참고 : [1-Stage detector 리뷰 \(chacha95.github.io\)](#).

RetinaNet is a one-stage network



- multi-scale 구조 (feature pyramid net) \rightarrow 각 scale 별로 object detection

- class head + box head → 각 위치마다 dense하게 수행

▼ Feature Pyramid Networks (FPN)

- 아키텍처를 크게 bottom-up pathway와 top-down pathway로 나누어서 설명한다.
- **bottom-up** 은 resolution을 줄여가며 feature를 얻는 forward 과정이고, **top-down**은 low resolution이지만 strong semantic을 가지고 있는 feature를 다시 원래 resolution으로 내려주면서 기존의 feature와 결합한다.

• **Bottom-up Pathway**

- Bottom-up Pathway는 그냥 convolution network를 이용한 forward computing 부분인데 여기에서는 ResNet을 사용했다. 한 stage마다 스케일을 2씩 줄여주면서 convolution 연산을 수행한다.

• **Top-down Pathway**

- **bottom-up**에서 올라왔던 feature들은 lower-level의 정보를 가지고 있지만, 일단 subsampling이 많이 되기 전이니까 위치 정보는 좀 더 정확하게 가지고 있다. 그리고 top-down에서 내려오는 feature들은 higher-level의 정보를 가지고 있지만 위치 정보는 많이 잃어버렸을 수 있기 때문에 두 feature를 합쳐서 서로의 장점을 가져온다.

- 생각보다 더하는건 단순했는데, 일단 아까 2배로 줄였으니까 다시 2배 upsampling을 해주고(이 때 그냥 간단하게 nearest neighbor upsampling을 쓴다.), 1x1 convolution으로 채널 수를 맞춰준 다음에 elementwise로 더해준다.
- 최종적으로 마지막 feature map까지 왔으면 3x3 conv를 적용해서 마무리 해준다.
- 참고 : [Feature Pyramid Networks for Object Detection 논문 정리 \(velog.io\)](#).

Detection with Transformer

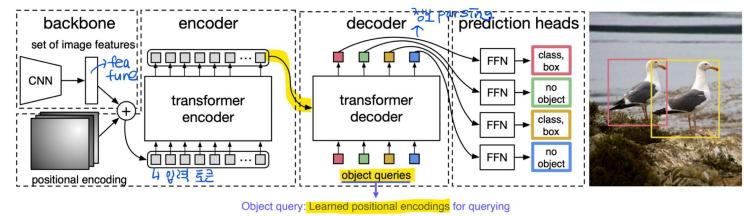
DETR (DEtection TRansformer)

▼ DETR

- DETR은 CNN Backbone + Transformer + FFN (Feed Forward Network)로 구성되어 있습니다.
- 참고 : [\[Review\] End-to-end object detection with Transformers.\(하루노문 P02\). \(tistory.com\)](#)

Transformer

- ViT (Vision Transformer) by Google
- DeiT (Data-efficient image Transformer) by Facebook
- DETR (DEtection TRansformer) by Facebook



- query → 이 위치에 해당하는 object? → query의 응답을 detection 구조로 활용
- Hyperparameter → N object queries = max, N objects exist in a single images
 - 하나의 이미지를 볼 때, 최대 n개의 위치에서 object detection 가능하다고 미리 정함

▼ transformer

- **encoder**

- encoder의 역할은 한마디로 말해서 **image를 이해하는 역할**이라고 볼 수 있다. 이 때 MHSA(multi-head self attention)에서는 image feature vector 간 distance에 상관없이 image feature vector를 분석하는 과정을 거치게 되는데 결과적으로 image 내에서 어떤 **grid** 가 **object**이고 어떤 **grid**가 **background**인지를 학습하게 된다.

- decoder**

- Decoder의 역할은 크게 2가지로 볼 수 있다.
 - (1) object query의 이해
 - (2) object query가 image feature vector들과 무슨 관련이 있는지를 조사
- (1) object query는 decoder의 MHSA에서 서로간의 관계를 조사하면서 object에 대해 이해하는 과정을 거치게 된다. **object query** 만 MHSA에 넣는게 무슨 의미가 있나 싶을 수 도 있지만 object끼리 비교하는 과정을 거쳐야 서로간의 공통점, 차이점을 비교하면서 해당 object에 대한 이해도가 높아질 수 밖에 없다. 예를 들어 코끼리 object와 기린 object를 비교한다고 치면 코끼리는 기린과 마찬가지로 다리가 4개지만 귀가 크고 코가 길며 똥뚱하구나!라는 object의 특성을 이해할 수 있다.
- (2) MHA(cross attention) 과정을 설명하기에 앞서 잠깐 query, key, value의 개념에 대해 짚고 넘어가자.
 - Query : 영향을 받는 entity
 - Key : 영향을 주는 entity
 - Value : Q, K matmul을 통해 계산된 attention score를 적용 할 entity.
 - 즉 cross attention에서 일어나는 일을 말로 풀어서 설명하면 "object query가 image feature로부터 어떤 영향을 받는지, 얼마나 관련이 있는지를 조사"하게 된다고 할 수 있다.

Detecting objects as points

- bounding box로 regression X → 대신, 물체의 중심점 or (왼쪽 위, 오른쪽 아래 점)을 찾기



Reference

- Object detection
 - Kirillov et al., Panoptic Segmentation, CVPR 2019
- Two-stage detector (R-CNN Family)
 - Dalal et al., Histograms of Oriented Gradients for Human Detection, CVPR 2005
 - Uijlings et al., Selective Search for Object Recognition, IJCV 2013
 - Girshick et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014
 - Girshick et al., Fast R-CNN, ICCV 2015
 - Ferguson et al., Detection and Segmentation of Manufacturing Defects with Convolutional Neural Networks and Transfer Learning, Smart and Sustainable Manufacturing Systems 2018
 - Ren et al., Faster R-CNN: Towards Real-Time Object detection with Region Proposal Networks, NeurIPS 2015
- Single-stage detector
 - Ndonhong et al., Wellbore Schematics to Structured Data Using Artificial Intelligence Tools, Offshore Technology

Conference 2019

- Redmon et al., You Only Look Once: Unified, Real-Time Object detection, CVPR 2016
- Liu et al., SSD: Single Shot MultiBox Detector, ECCV 2016

4. Single-stage detector vs. two-stage detector

- Lin et al., Focal loss for Dense Object detection, ICCV 2017

5. Detection with Transformer

- Vaswani et al., Attention Is All You Need, NeurIPS 2017
- Carion et al., End-to-end Object Detection with Transformers, ECCV 2020
- Zhou et al., Objects as Points, arXiv 2019