

DSAC Module4

Deep Learning (5-1)

2023년 11월 18일~12월 16일

권오준

(ojkwon@deu.ac.kr)



딥러닝 응용

이미지 분류 (Image Classification)

- **이미지 분류(Classification) 문제**
 - 특정 데이터를 식별하고 **정해진 카테고리(Category)**로 정리, **분류**하는 것
- **컴퓨터 비전에서의 분류 문제**
 - 주어진 이미지 및 영상 내의 특정 객체(Object)를 식별하는 것을 의미
 - 식별된 정보를 토대로 라벨(Label) 형태로 표현
- 예를 들어 주어진 고양이 사진에서 사진 내에 있는 객체가 고양이라고 식별하고 판단하는 것
- 이미지 및 영상에서 보이는 객체의 모습은 일반적이지 않을 수 있다.
 - 같은 종의 동물이더라도 각도, 위치, 조도 등 여러 요인들로 인해 다르게 보여 식별하는데 어려울 수 있다.
 - 이러한 요인들에도 불구하고 해당 객체를 식별하는 것이 중요

객체 탐지 (Object Detection)

- 객체 탐지(객체 검출, Object Detection)
 - 컴퓨터 비전(Computer Vision)의 중요 분야 중 하나
 - 이미지 및 영상 속에서 특정 객체의 정보 및 위치 등을 탐지하는 것을 의미
 - 즉, 전체 디지털 이미지 및 영상 내에서 유의미한 특정 객체를 탐지 또는 감지하는 것
 - 이미지 검색(Image Retrieval), 이미지 주석(Image Annotation), 얼굴 인식(Face Detection), 비디오 추적(Video Tracking) 등 다양한 분야의 문제를 해결하기 위한 초석이 되는 기술로써 사용

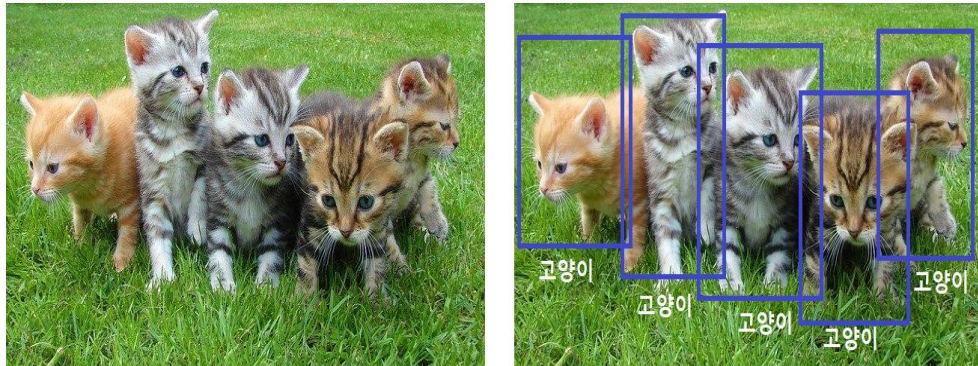
지역화 (Localization)

- 지역화(Localization) 문제
 - 해당 객체가 이미지에서 어디에 있는지 위치 정보를 나타내는 것
 - 바운딩 박스(Bounding Box) 형태로 표현
- 분류와 지역화, 이 두 문제를 해결하는 것이 객체 탐지의 주목표라 할 수 있다.
- 즉, 이미지에서 특정 객체가 어떤 객체인지 분류하고 판단해야 하며 위치까지 파악해야 한다는 것이다.

객체 탐지 (Object Detection)

- 단순 이미지 분류와 객체 탐지
 - 객체 탐지는 어느 위치에(=지역화, Localization), 어떤 종류의 객체가 있는지(=분류, Classification)에 대한 것을 모두 포함

객체 탐지(Object Detection) = 분류(Classification) + 지역화(Localization)



고양이

단순 이미지 분류(Image Classification)와 객체 탐지(Object)의 차이

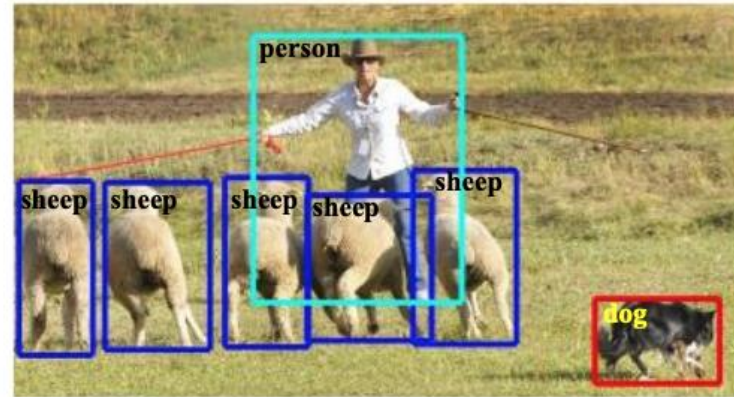
객체 검출

- 이미지 분류(Classification)
 - 이미지에 어떤 객체들이 들어 있는지만 알아내는 것
 - 여기서는 이미지에 사람, 양, 개가 있다는 것을 찾아낸다
- 객체 검출(Detection)
 - 찾아낸 객체의 위치까지 알아내는 것
 - 일반적으로는 객체가 있는 위치를 **박스 형태**로 찾아낸다
- 세그멘테이션(Segmentation)
 - 객체의 위치를 박스형태가 아니라 **비트 단위**로 찾아낸다
 - 즉, 각 비트가 어떤 객체에 속하는지를 분류해내는 것
- 객체 인스턴스 세그멘테이션(Instance Segmentation)
 - 이미지를 비트 단위로 어느 객체에 속하는지를 찾아낼 뿐 아니라 각 객체를 **서로 다른 객체로 구분**하는 기능까지 수행

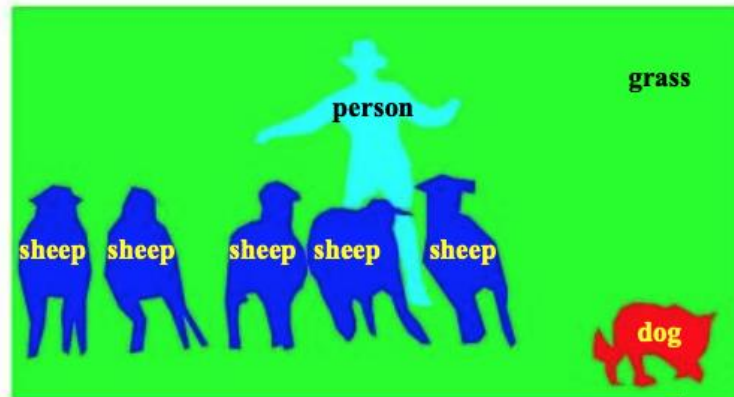
객체 검출



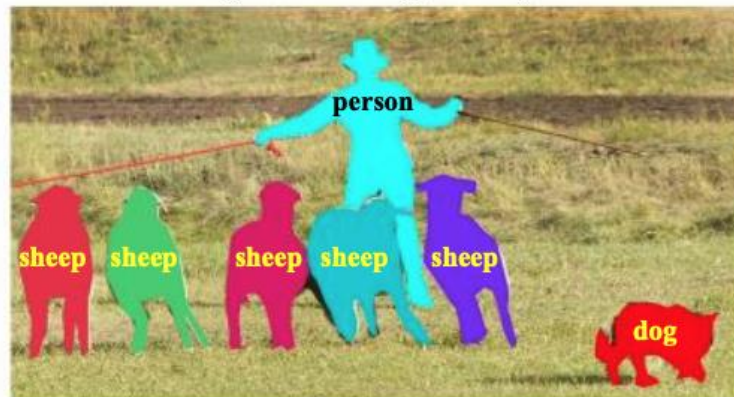
(a) Object Classification



(b) Generic Object Detection
(Bounding Box)

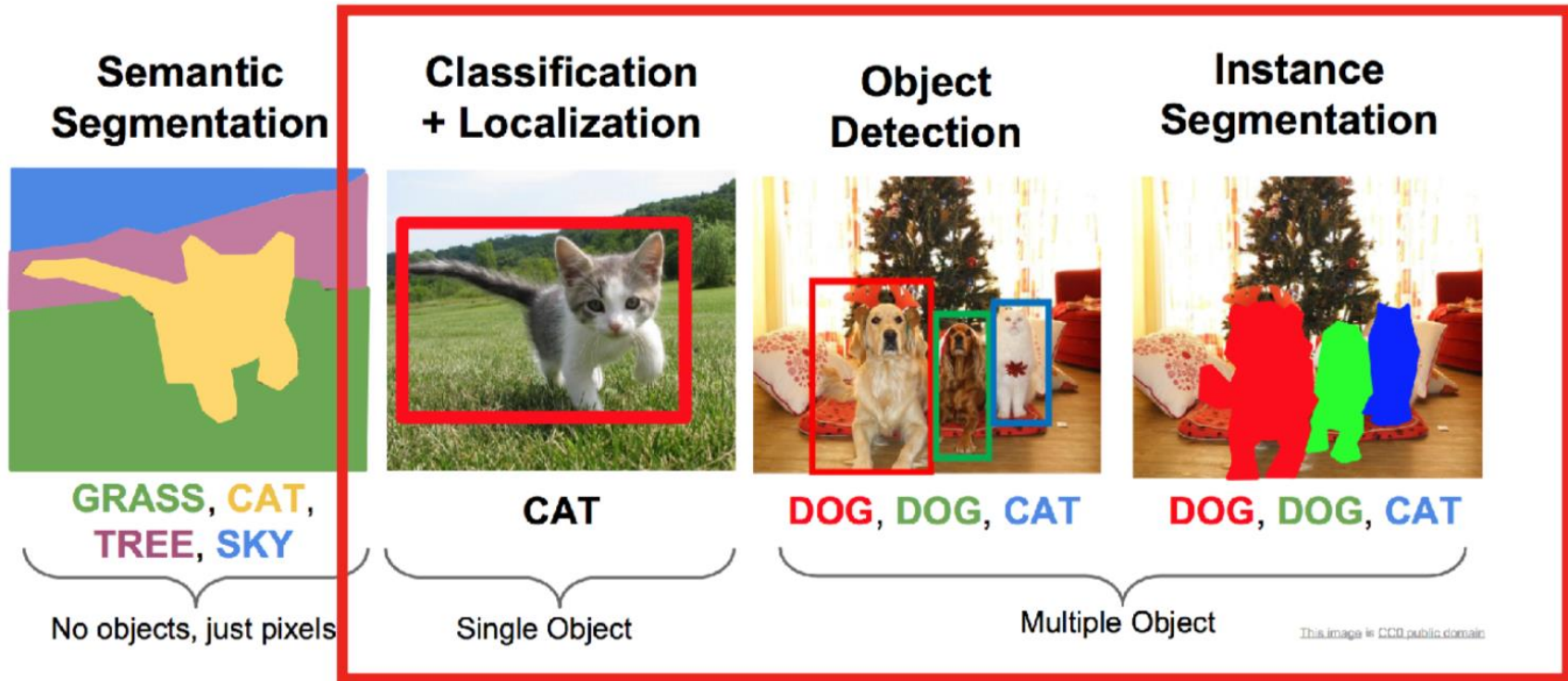


(c) Semantic Segmentation



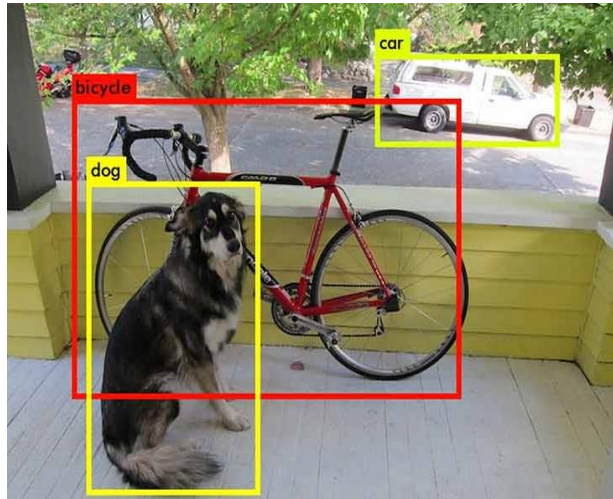
(d) Object Instance Segmentation

이미지 객체 검출

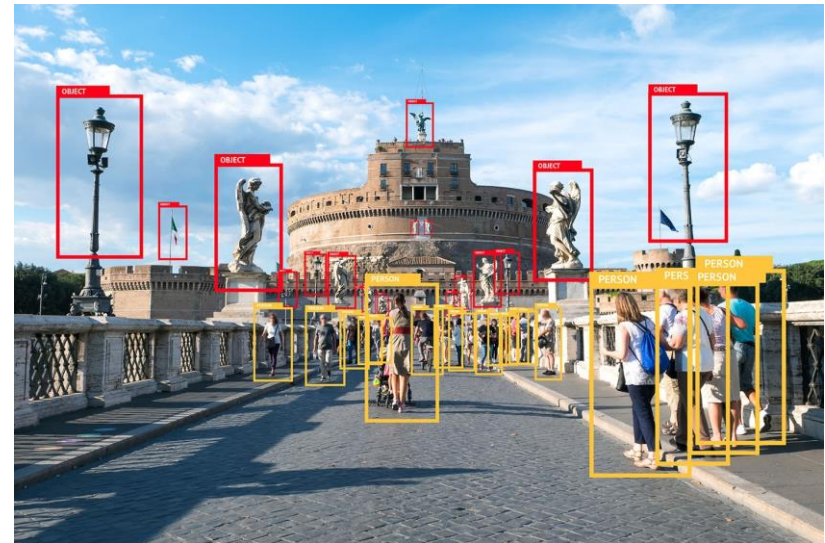
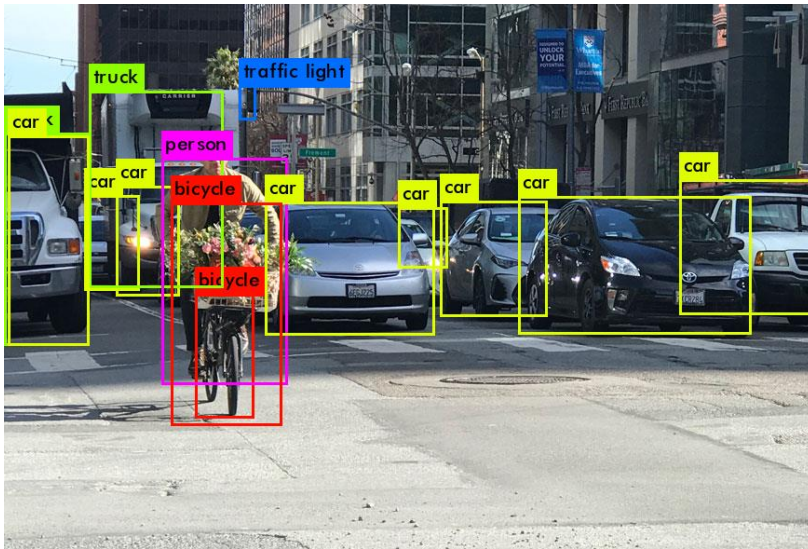


다중 객체 분류 (Multi-Labeled Classification)

- 다중 객체 분류(Multi-Labeled Classification)
 - 하나의 객체만 분류하는 것이 아닌 **복수 개의 객체를 분류**하는 것
- 이미지 및 영상 내에서는 서로 다른 객체가 존재할 수 있다. 그리고 실제 환경에서는 하나의 객체만 존재하는 것이 더 흔하지 않다. 따라서 다중 객체 분류는 객체 탐지에서 중요한 개념



다중 객체 분류 (Multi-Labeled Classification)



객체 위치 검출

- 한편 객체의 위치를 찾는 문제
 - 분류가 아니라 회귀(regression) 문제
 - 즉, 객체의 **경계 박스의 좌표 값**은 회귀 문제로 예측
 - 이 방법은 사람의 자세를 예측하는 **포즈 검출**, **기준점 검출**에서도 사용

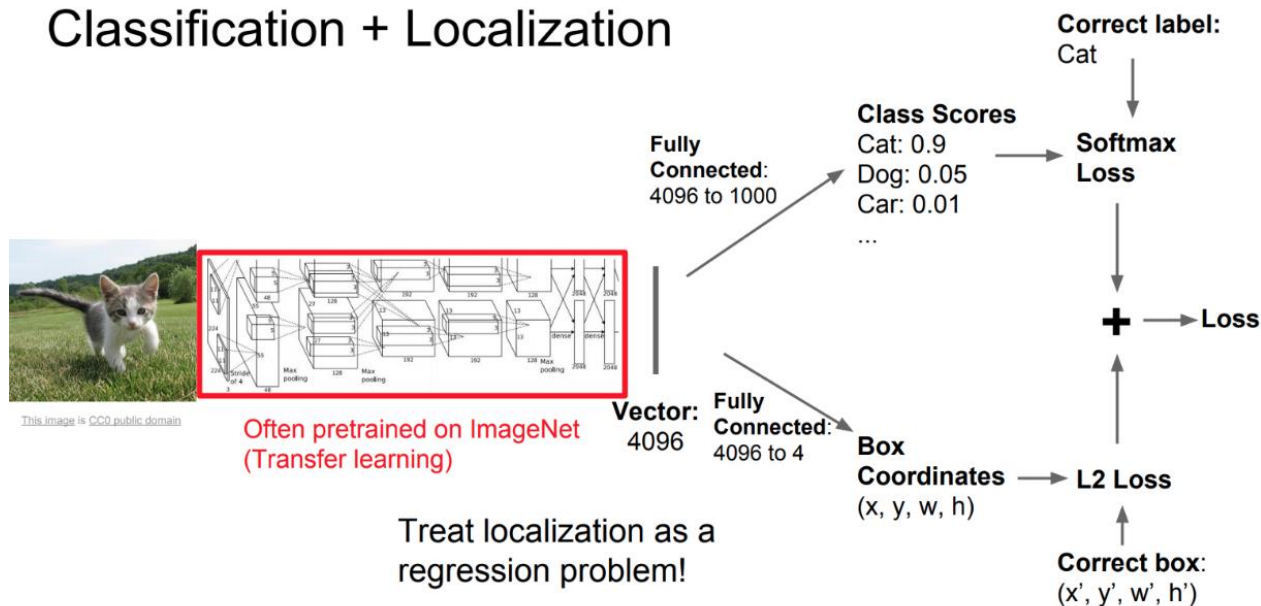


(자세 검출의 예)

손실 함수

- 객체 검출의 2가지 작업
 - 이미지에 어떤 객체가 있는지를 분류(classification)하는 작업
 - 객체가 있는 위치(좌표)를 계산하여 예측하는 기능(localization)
- 손실 함수
 - 객체 검출의 2가지 작업을 동시에 고려하여 학습

Classification + Localization



손실 함수

- 학습을 시키려면
 - 입력 이미지에 대해서 **객체의 종류**와 **위치**를 찾아야 한다
- 레이블 - 2가지 제공해야
 - 이미지 클래스
 - 이미지가 위치한 박스의 좌표
- 출력이 두 가지 이상인 멀티 출력 모델을 만들어야 하며 손실함수는 분류에 관한 손실과 회귀에 관한 손실의 합으로 구성

$$Loss = \alpha * \text{Softmax_Loss} + (1 - \alpha) * \text{L2_Loss}$$

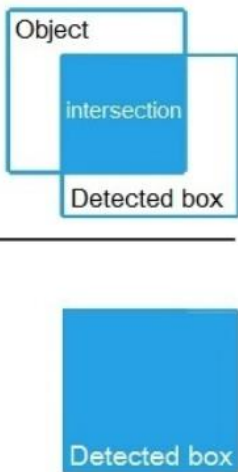
- 두 가지 성분의 손실의 상대적인 비중은 α 값으로 조정

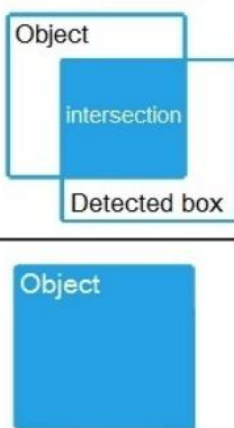
성능 평가

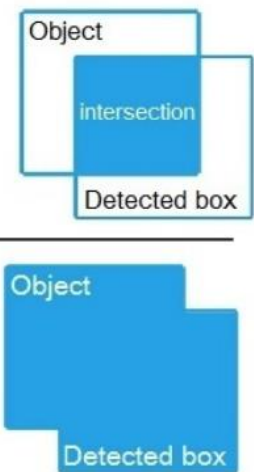
- 객체 검출에서는 단순히 어떤 객체가 있는지를 찾는 분류 문제가 아니므로 정확도 등 만으로 성능을 평가할 수 없고, **예측한 객체의 위치가 실제 객체의 위치와 얼마나 일치하는지**를 평가해야 한다
- 성능 평가척도
 - IoU (Intersection over Union)
 - mAP (mean Average Precision)
 - AR (Average Recall)

객체검출 성능 평가 – 정밀도, 리콜, IoU

- 객체검출 성능 평가: 객체의 위치를 얼마나 실제 위치와 겹치게 잘 찾았는지 평가
 - 정밀도 (precision) : 예측한 면적분에 겹치는 면적
 - 리콜 (recall) : 실제 이미지 면적분에 겹치는 면적
 - IoU : 두 가지 면적의 전체집합 부분에 비하여 겹치는 부분의 면적의 비율


$$\text{Precision} = \frac{\text{Area of Intersection}}{\text{Area of Detected box}}$$


$$\text{Recall} = \frac{\text{Area of Intersection}}{\text{Area of Object}}$$


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

성능 평가 – IoU

- 실제 이미지 예



성능 평가 – mAP

- 객체 검출에서 검출할 객체가 여러 개가 있을 때 이들의 평균 예측 성능을 표현하는데 mAP가 많이 사용
- mAP
 - 검출 작업의 평균 정확도
 - 여러 객체에 대해 각각 AP를 구하고 이의 평균치를 구한 것
 - mAP 계산
- 일반적으로 mAP가 0.5 이상이면 true positive라고 판단

객체 검출 방식 – Sliding window 방식

- 슬라이딩 윈도우 방식 (초기)

- 객체 검출을 위해서 전체 이미지를 작은 크기로 나누고 각 부분을 대상으로 객체를 찾는 작업을 수행
- 시간이 오래 걸리고 더욱이 **윈도우 크기**와 **객체의 크기**가 **다를 때** **찾지 못함**

슬라이딩 윈도우 방식

탐지기의 종류

- 탐지기(Detector)
 - 지역화와 분류의 진행되는 순서에 따라 다음과 같이 두 가지 방식으로 나눌 수 있다.

1-Stage Detector: 지역화(Localization), 분류(Classification) 과정을 동시에 진행

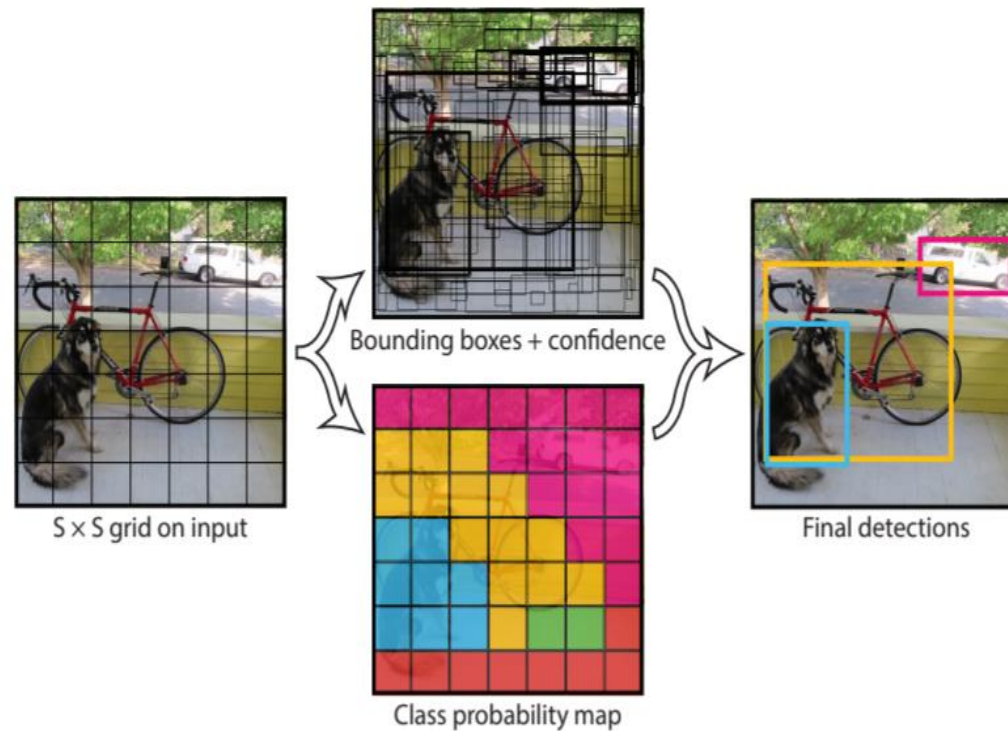
2-Stage Detector: 지역화, 분류 과정을 순차적으로 진행.

1-Stage Detector

- 객체의 위치를 찾는 **지역화**(Localization)와 객체를 식별하고 분류하는 **분류**(Classification) 과정이 **동시에** 이루어지는 탐지 방식
- 장점
 - 두 과정이 동시에 이루어지기 때문에 **속도는 빠르다**
- 단점
 - 평균적으로 비교적 **정확도가 더 낮다**
- 빠른 처리 속도를 요구하는 영상(Video)에 많이 적용되는 경우가 많다
- 이 방식의 대표적인 모델
 - YOLO (You Only Look Once)
 - SSD (Single Shot Detector)
 - RetinaNet

1-Stage Detector

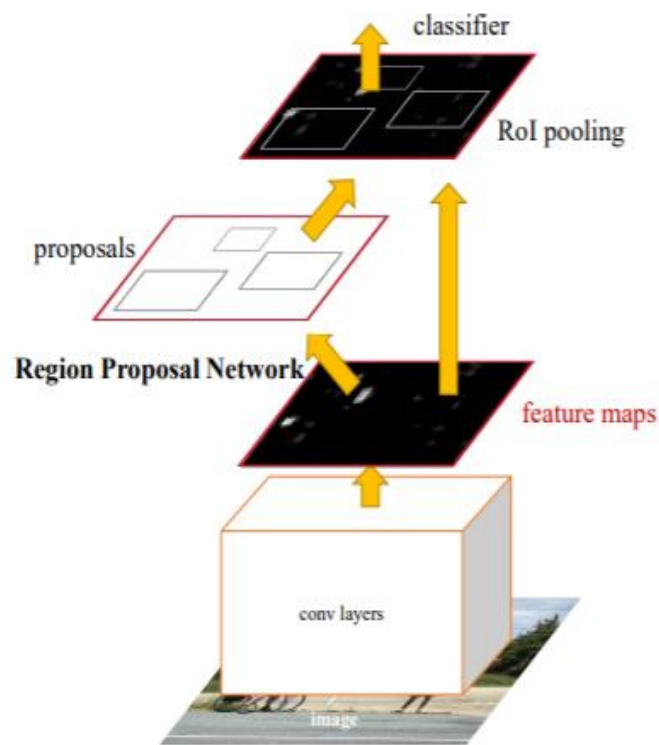
- 두 가지의 과정이 동시에 이루어져서 결과가 나오는 것을 확인할 수 있음



1-Stage 방식의 YOLOv1 모델

2-Stage Detector

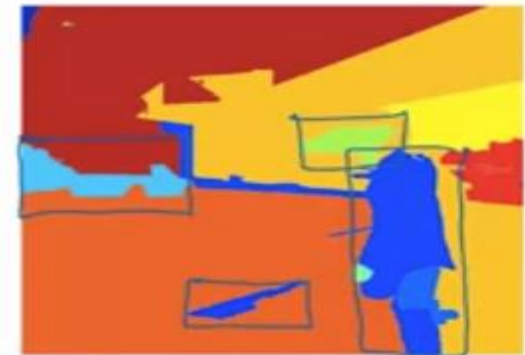
- 순차적으로 **지역화** 과정이 **먼저** 이루어지고 해당 영역에서 객체를 찾는 식으로 진행하는 탐지 방식
- 장점
 - 느리다
- 단점
 - 비교적 **정확도가 높다**
- 이 방식의 대표적인 모델
 - R-CNN (Regions with CNN features)
 - Faster R-CNN
 - DenseNet



2-Stage 방식의 R-CNN 모델

객체 검출 방식 – R (Regional) CNN

- Region Proposal (R-CNN 에 적용)
 - Sliding window 를 전체에 대해 하지 않고 물체가 있을 것 같은 영역에 대해서만 함. (스케일링 등이 필요 없어 빠름): segmentation 이용
 - Region Proposal method, Classification, Box Regression 문제를 **개별적으로 취급**하여 **개별 학습**
 - 2-stage 탐지기



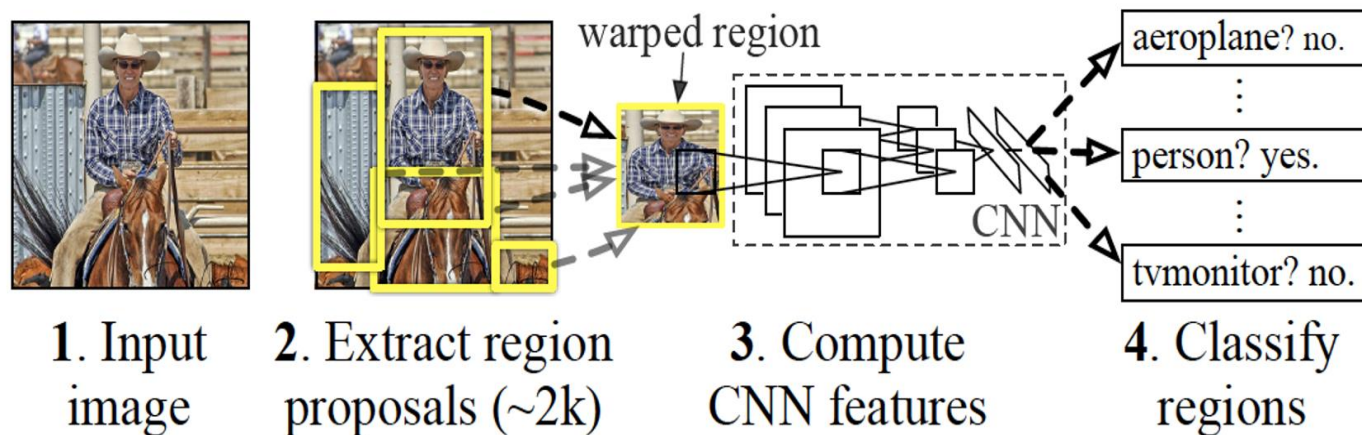
Segmentation algorithm
~2,000

객체 검출 방식 – R (Regional) CNN

- 2014년 CNN 모델을 객체 탐지 분야에 최초로 적용한 딥러닝 모델
- 단순 이미지 분류(Classification) 뿐만 아니라 객체 검출 분야에서도 높은 수준의 성능을 보여줌
- “Regions with Convolutional Neural Networks features”의 약자
- 설정한 영역(Region)을 CNN의 Feature(입력 값)로 활용하여 객체 탐지(Object Detection)를 수행하는 신경망
- 즉, 이미지 및 영상 내에 특정 객체가 있을 만한 영역에서 특징 추출 (Feature Extraction) 성능이 뛰어난 CNN(합성곱신경망)을 적용시켜 객체를 탐지하겠다는 것이다.
- 즉, 이미지 내 특정 영역을 먼저 기준으로 잡고(=Localization), CNN 모델을 활용하여 객체를 분류(=Classification)

R (Regional) CNN 기본 구조

R-CNN: *Regions with CNN features*

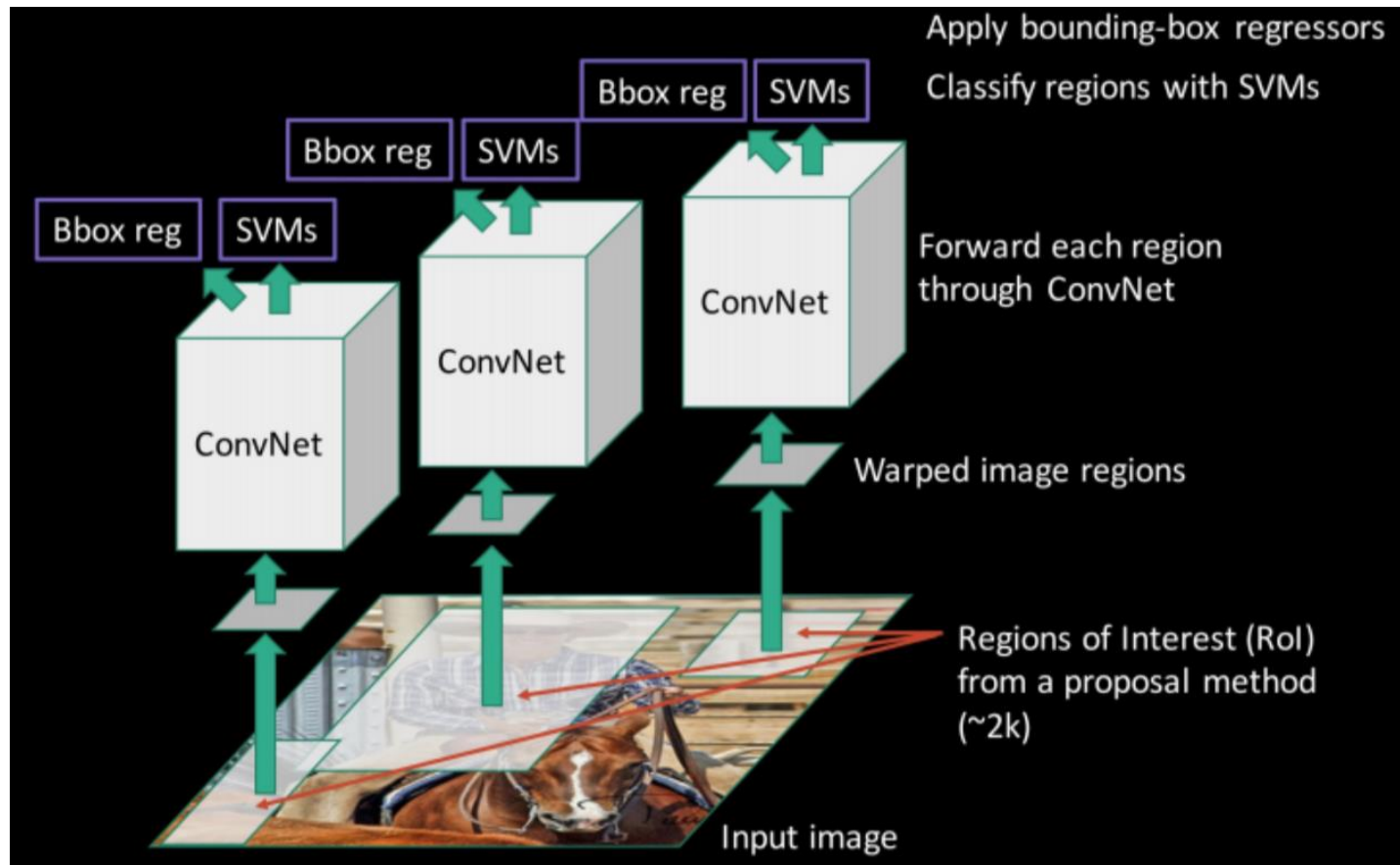


Task: →

1. Region Proposal

2. Region Classification

R (Regional) CNN



R (Regional) CNN 기본 구조

- 2-Stage Detector 탐지 방식을 사용
 - 객체의 위치를 찾는 Region Proposal
 - 객체를 분류하는 Region Classification
- 이 두 단계를 처리하기 위한 총 네 가지 모듈로 나뉘 세분화

R-CNN 모델 구성 모듈	기능 및 설명
Region Proposal	카테고리와 무관하게 우선 객체가 있을 법한 영역(Region)을 찾는 모듈(Selective Search)
(사전 훈련된) CNN	찾은 영역(Region)에서 객체를 탐지하기 위해 특성 맵(Feature Map)을 추출하는 모듈
SVM	추출된 특성 맵을 분류하는 모듈
Bounding Box Regression	회귀(Regression)를 활용해 해당 객체의 바운딩 박스를 표현하는 모듈

R-CNN의 단점

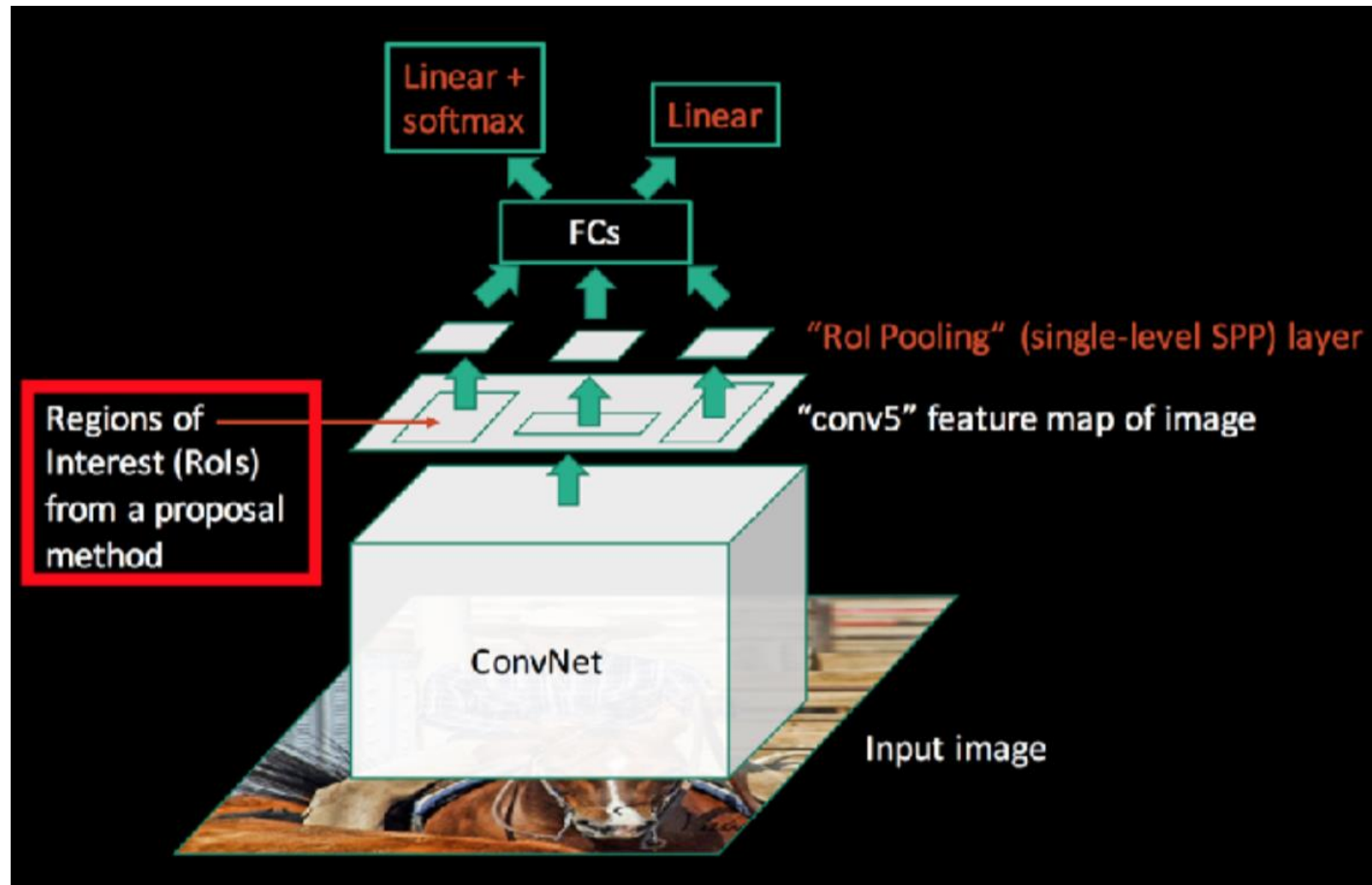
- R-CNN 의 단점 : 느린 속도
 - 이 모델은 선택적 탐색(Selective Search) 알고리즘을 사용해 객체가 있을 법한 영역에 대해 Region Proposal을 생성한다. 생성된 수많은 RoI(Region of Interest)는 다시 사전 훈련된 CNN 모델을 통과하여 각 영역마다 CNN 연산을 수행한다. 생성된 Region Proposal 마다 CNN 연산을 수행하기 때문에 많은 연산량을 요구
 - Region Proposal이 만약 2천개가 생성되었다면, 2천개마다 CNN 연산이 수행됨

Fast R-CNN

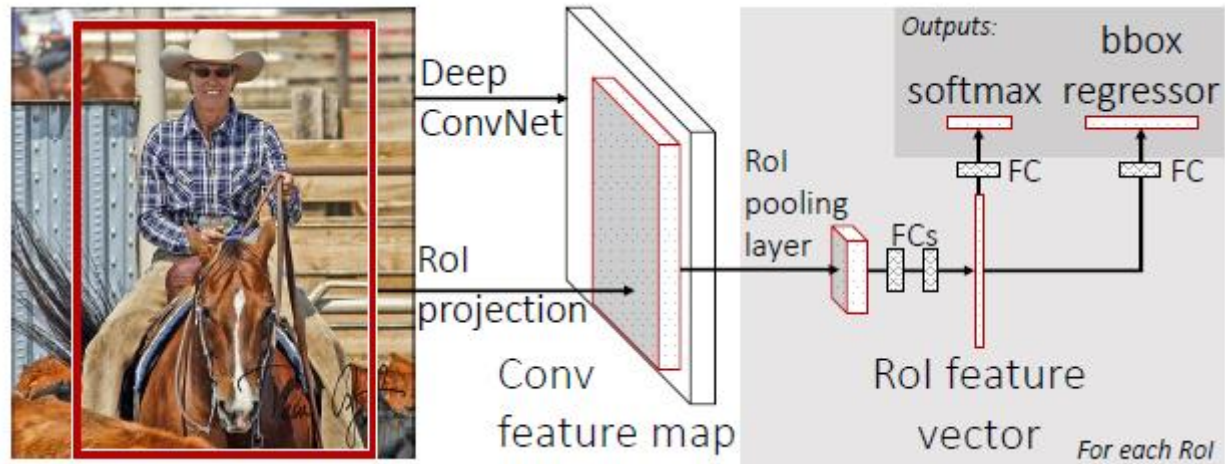
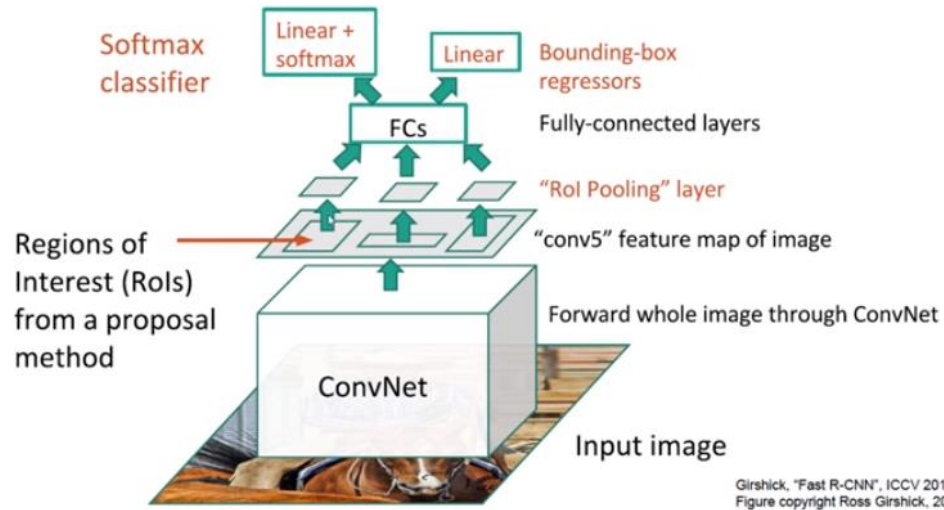
- Fast R-CNN 모델
 - 기존의 R-CNN 모델의 단점을 극복하고자 제안된 객체 탐지 모델
 - R-CNN 모델의 느린 속도 문제를 해결하기 위해
 - Fast R-CNN 모델은 SPP(Spatial Pyramid Pooling)을 적용한 RoI Pooling 과정을 기존의 R-CNN 모델에 적용
 - RoI Pooling 과정은 층(Layer)으로서 모델 구조에 적용됨

Fast R-CNN

Fast R-CNN



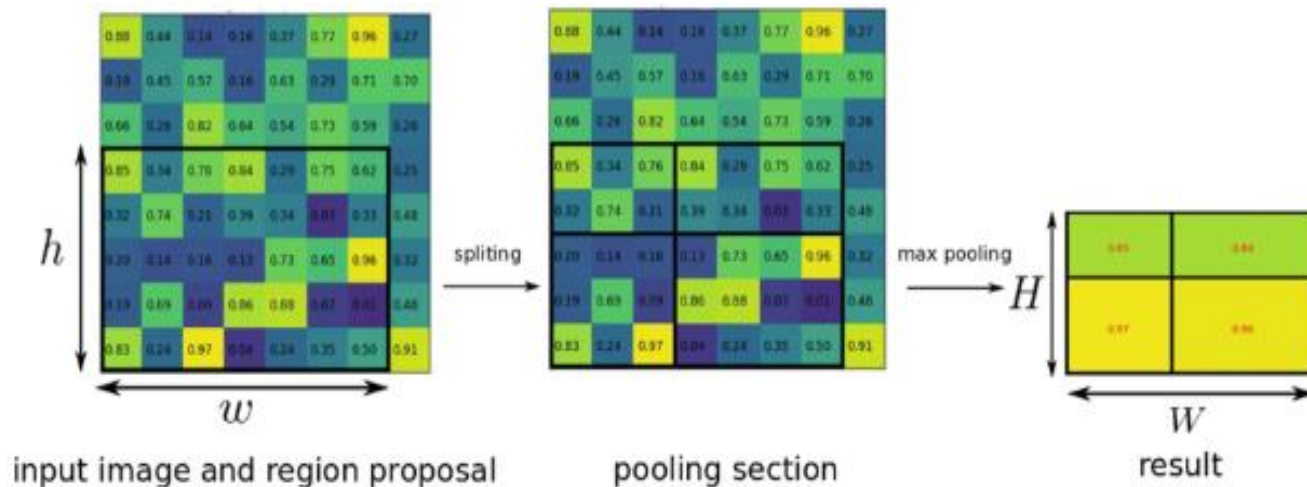
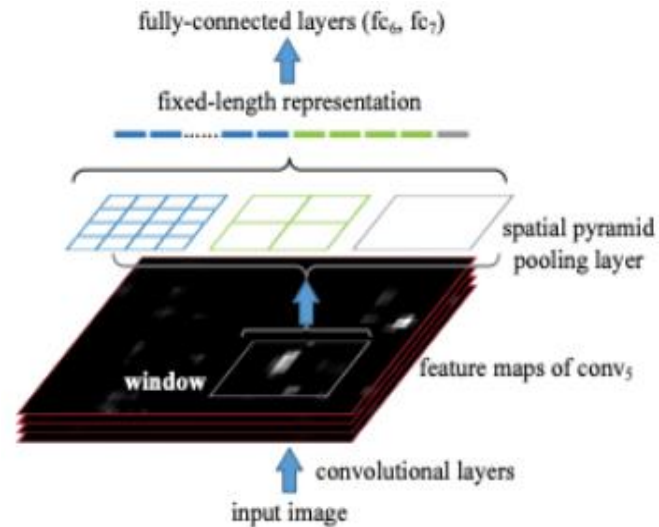
Fast R-CNN



Fast R-CNN

- RoI Pooling 과정
 - Fast R-CNN 모델에서 기존 R-CNN 모델의 느린 처리 속도를 해결하기 위해 고안된 개념
 - 이 과정은 SPP(Spatial Pyramid Pooling) 개념을 적용한 것인데, 기존 전결합망 층(FC Layer, Fully Connected Layer)의 고정적인 입력 크기로 인해 발생하는 문제를 해결하기 위한 것

Fast R-CNN – SPP 개념도와 ROI Pooling 과정



Fast R-CNN

- 합성곱신경망(CNN)은 마지막 층이 전결합망 층으로 구성된다. 이는 CNN 모델의 입력 크기를 고정해야 하는 문제를 초래
- 이를 해결하기 위해 층 자체를 피라미드(Pyramid) 형태로 나누어 특성 맵 (Feature Map)을 나누어 처리하게끔 구성된다. 나누어진 특성 맵은 Max Pooling 연산을 통해 다시 이어붙여 고정된 크기의 전결합망에 입력되는 방식으로 동작
- 미리 설정한 $H \times W$ 크기로 만들어주기 위해서 다음과 같은 크기만큼 경계를 RoI 위에 생성

$$(h/H) * (w/H)$$

Fast R-CNN

- 즉, 특성 맵은 결과적으로 $h \times w$ 크기의 RoI가 $H \times W$ 크기의 고정된 특성 벡터(Feature Vector)로 변환됨
- 이는 수많은 생성된 Region Proposal에 각각 CNN 연산이 필요한 것이 아닌, **단 1번의 CNN 연산만**으로 R-CNN 동작을 가능하게 됨
- 이에 따라 Fast R-CNN 모델은 기존의 R-CNN 모델의 **속도보다 대폭 향상**되어 동작하게 됨

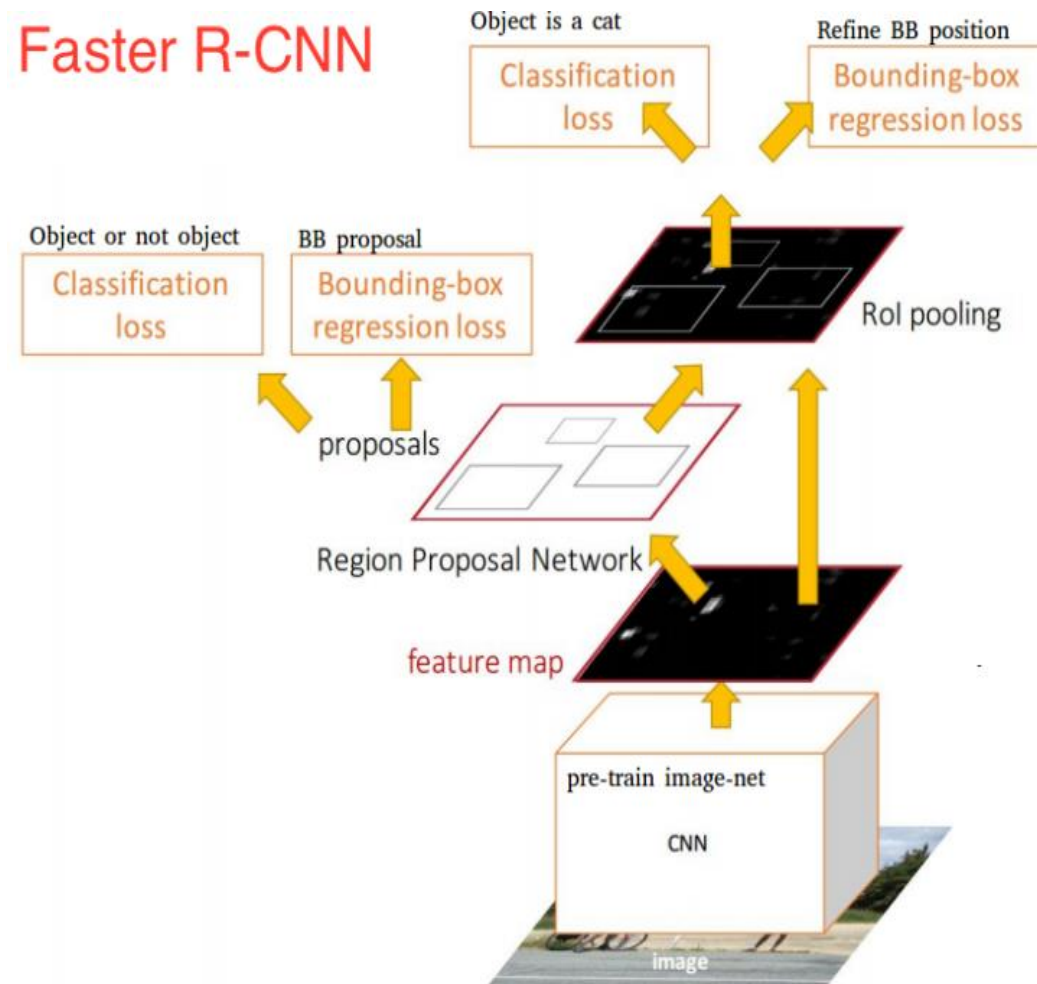
Fast R-CNN

- 기존의 R-CNN 모델의 느린 속도 문제를 어느 정도 해결했으나
- ROI(Region of Interest) 생성에 사용되는 선택적 탐색 알고리즘은 여전히 모델의 외부에서 동작
- 이는 해당 알고리즘이 모델 내부에서 일목 정연하게 같이 동작하는 것이 아닌, 외부에서 처리하여 모델에 알고리즘 결과를 제공하는 형태가 되기 때문에 느린 속도의 또 하나의 원인
- 즉, 많은 처리 속도 향상을 이루어냈으나 실시간 객체 탐지에서 만족할만한 속도는 아님

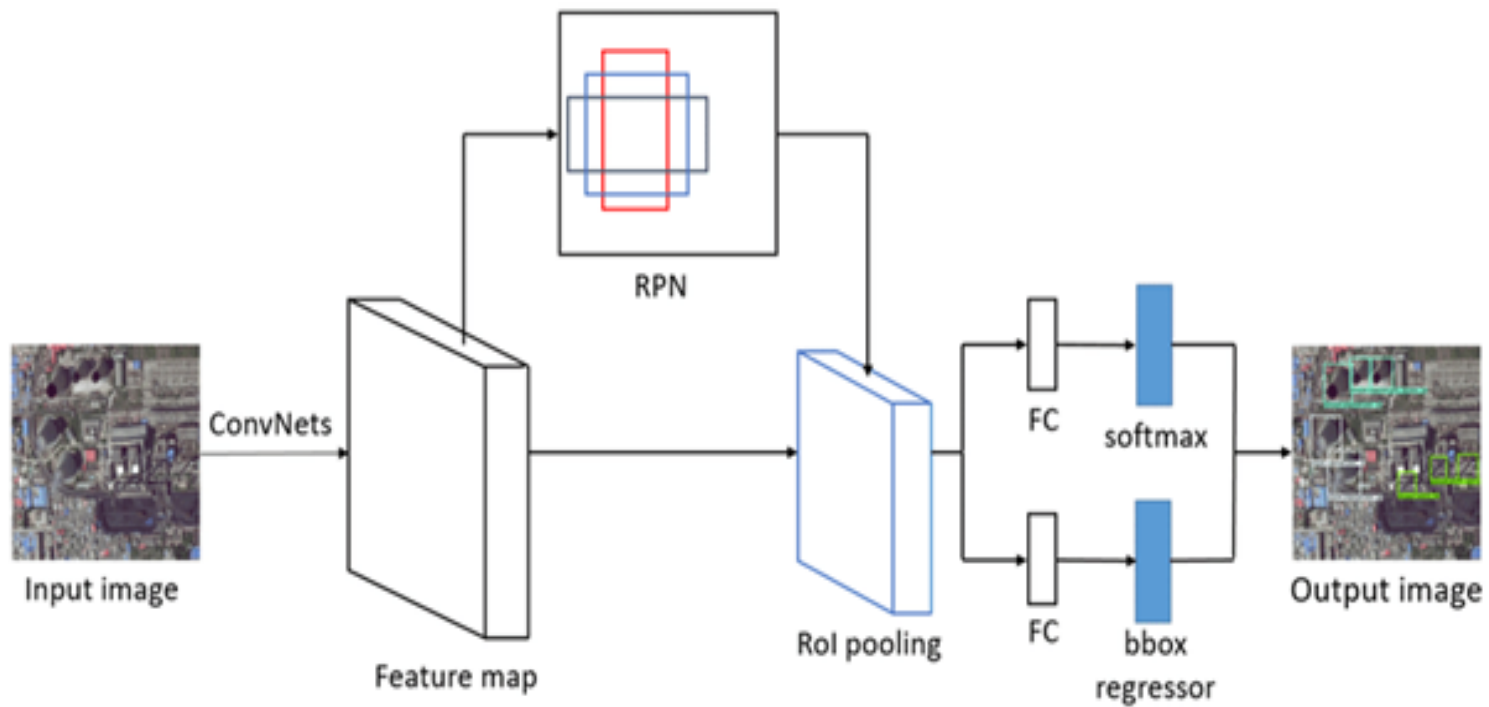
Faster R-CNN

- Fast R-CNN 모델의 구조와 알고리즘을 개선
- 기존의 선택적 탐색(Selective Search) 알고리즘 대신에 RPN(Region Proposal Network) 신경망 모델을 추가하여 이 기능을 대신하는 구조
- 즉, RPN(Region Proposal Network) 망이 Region Proposal을 생성

Faster R-CNN



Faster R-CNN

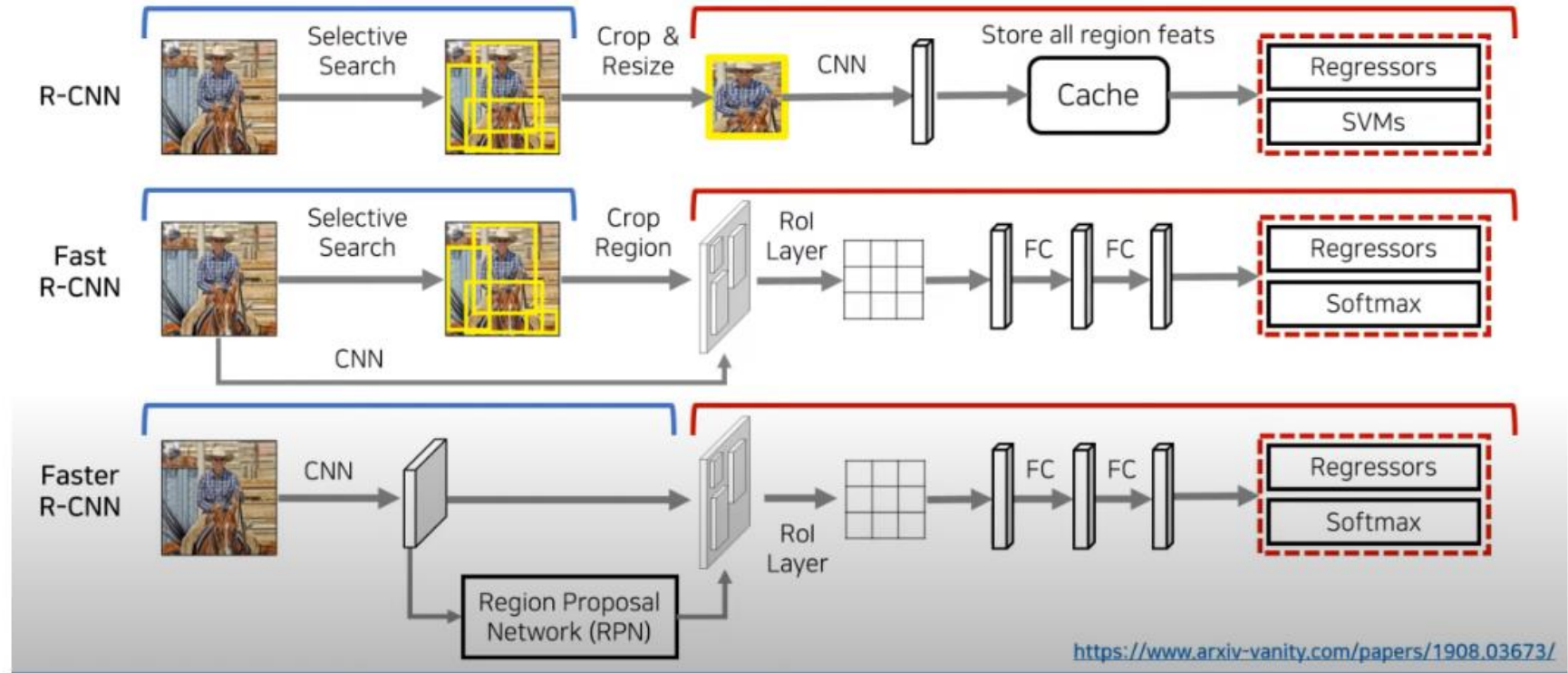


RPN + Fast R-CNN 구조 (Faster R-CNN)

Faster R-CNN

- 기존의 Fast R-CNN 모델 구조에서 **RoI Pooling Layer** 중간에 **RPN 신경망**이 추가되었고, 이 신경망은 기존 알고리즘처럼 모델 외부에서 동작하는 것이 아니라 **모델 내부에서 함께 동작함**
- 기존 Fast R-CNN 모델보다 더 향상된 성능을 보여주었고, **객체 탐지 분야**에서 **좋은 결과**를 보여 줌

객체 검출 방식 – R-CNN 계열 네트워크 구조



R-CNN	Fast R-CNN	Faster R-CNN
	CNN	CNN
ROI (selective search)	ROI (selective search)	ROI (Region Proposal Network)
Warping	ROI Pooling layer	ROI Pooling layer
CNN	FCN	FCN
SVM & BBR	Softmax , BBR	Softmax , BBR

CNN 계열 객체 탐지 모델의 발전

- CNN 계열의 객체 탐지 모델의 발전을 국소적으로 정리

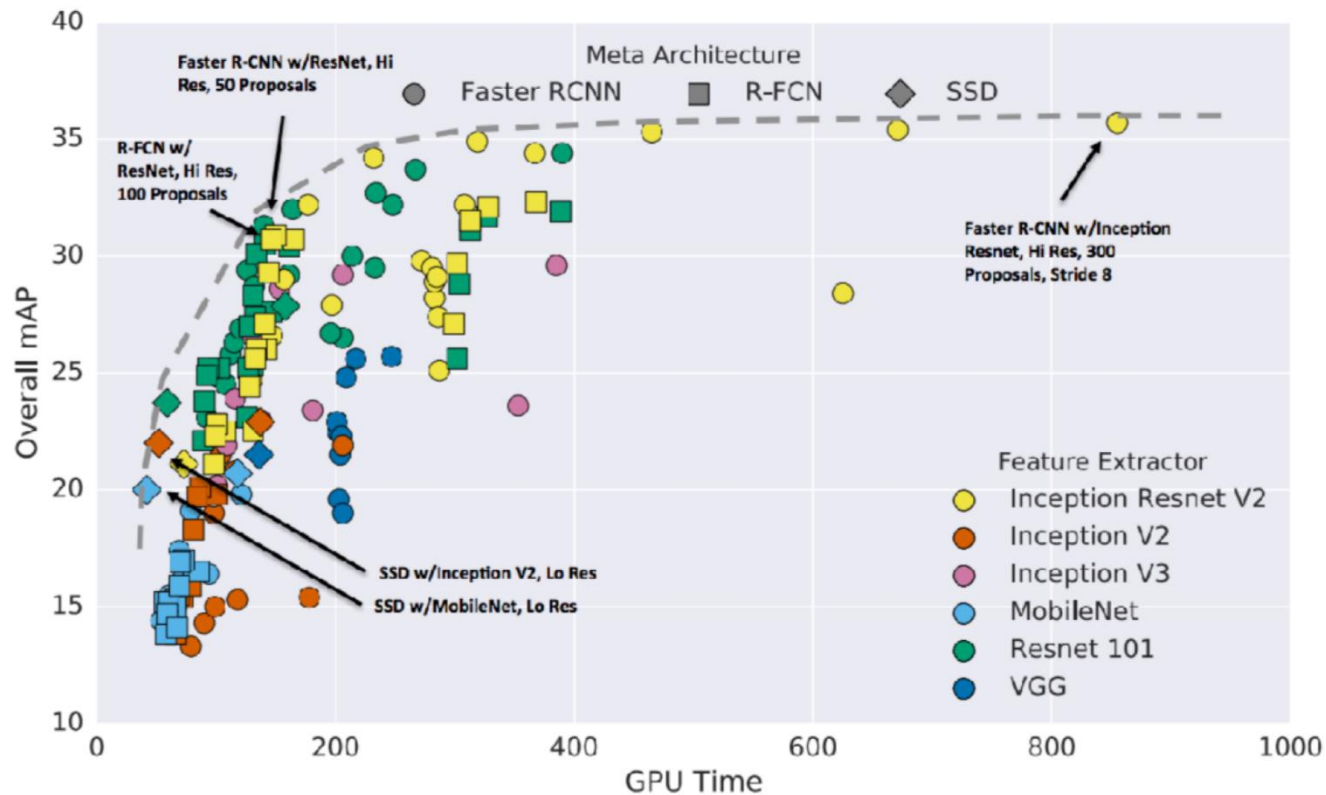
CNN 계열 객체 탐지 모델	동작 과정
R-CNN	1. (선택적 탐색 알고리즘) Region Proposal 추출 2. 각 Region Proposal 마다 CNN 연산 수행 3. 객체 분류와 바운딩 박스 생성
Fast R-CNN	1. (선택적 탐색 알고리즘) Region Proposal 추출 2. (RoI Pooling) 전체 이미지 1번의 CNN 연산 3. 객체 분류와 바운딩 박스 생성
Faster R-CNN	1. (RPN 신경망) Region Proposal 추출 2. (RoI Pooling) 전체 이미지 1번의 CNN 연산 3. 객체 분류와 바운딩 박스 생성

CNN 계열 객체 탐지 모델의 발전

- 합성곱신경망(CNN)은 **이미지의 특징**(Feature)을 추출하는데 **좋은 성능**을 보여 줌.
- 하지만 타 신경망보다 **처리 속도가 느린** 단점을 가지고 있어 **실시간으로 객체를 탐지**하고 처리하는데 **적합하지 않음**
- 객체 탐지 모델은 주로 **처리 속도를 향상하는 방향으로 발전**해왔고, 현재는 실시간으로 객체를 처리할 수 있을만큼 성능 향상을 이루어 냈으며 많은 주제로 모델이 적용되어 개발되고 있음

객체 검출 방식 – 알고리즘 비교

- Resnet을 사용한 Faster R-CNN등이 우수한 성능을 나타냈으나 **최근 Mask-R-CNN과 Yolo**등 성능이 **더 개선된 방식**이 소개됨
 - x 축 : 객체 검출 계산에 필요한 시간
 - y 축 : mAP 즉, 검출 성능



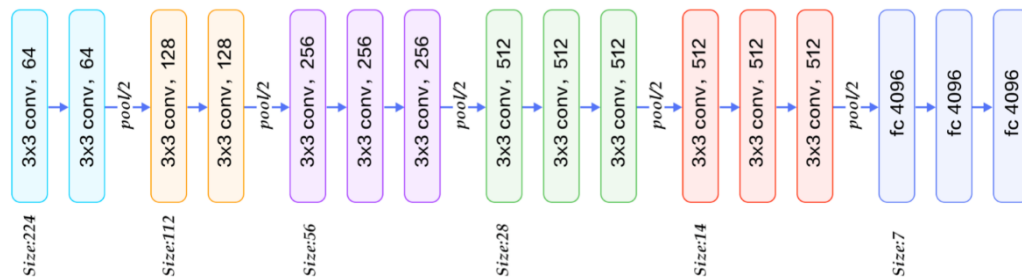
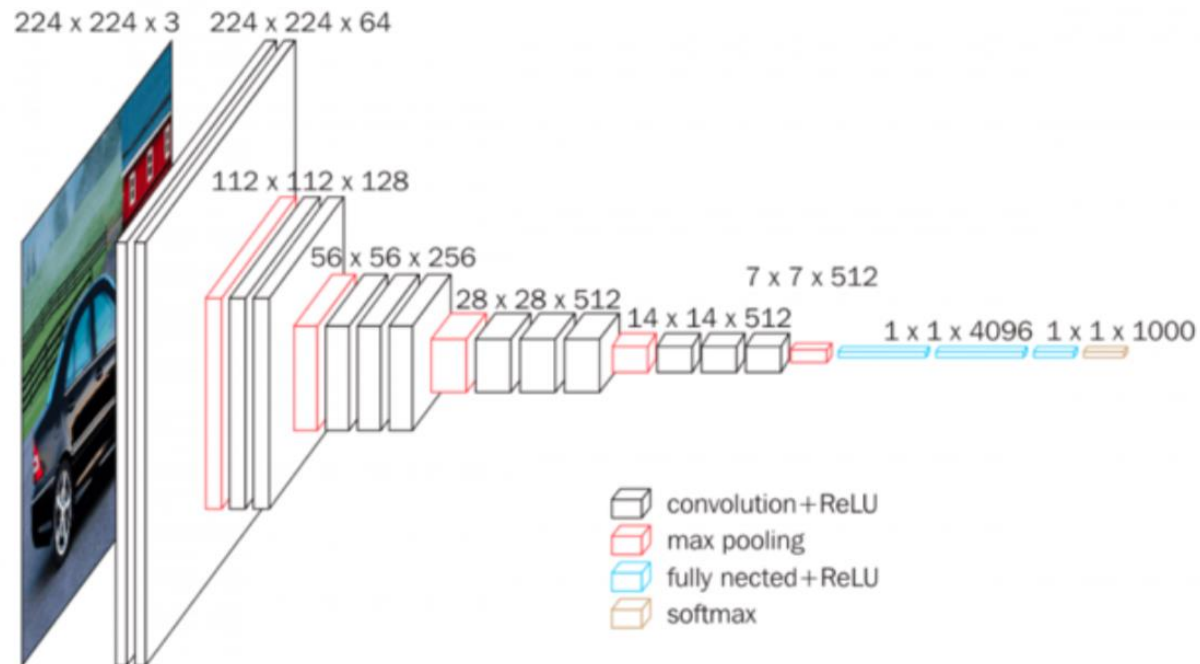
이미지 데이터 셋 (객체 검출) – ImageNet

- 분류, localization, 검출을 위한 데이터를 제공
- 일반적인 사물 1,000 종류의 1,400만 장의 사진
- 경계박스(bounding box) 좌표 값도 제공
- 각 이미지 당 평균 1.1개의 객체를 포함
- 1,000개의 객체 목록

<https://gist.github.com/aaronpolhamus/964a4411c0906315deb9f4a3723aac57>

- ILSVRC(ImageNet Large Scale Visual Recognition Competition)
 - 대표적인 이미지 인식 경진대회
 - 2017년에 인간의 성능을 능가하면서 종료됨
 - 2012년 AlexNet이 큰 성과를 낸 후 딥러닝 모델이 확산
 - 2014년 우승 모델인 VGG16 모델이 전이학습에 널리 사용

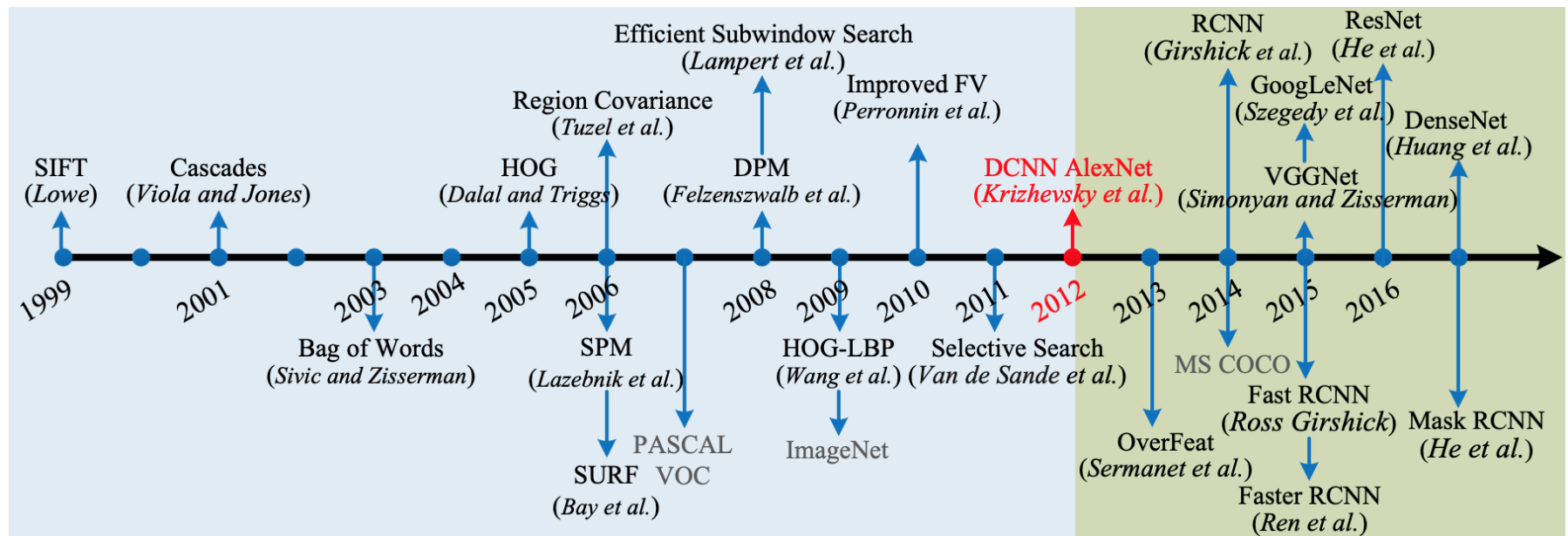
VGG16 모델



이미지 데이터 셋 (객체 검출) – VOC, COCO

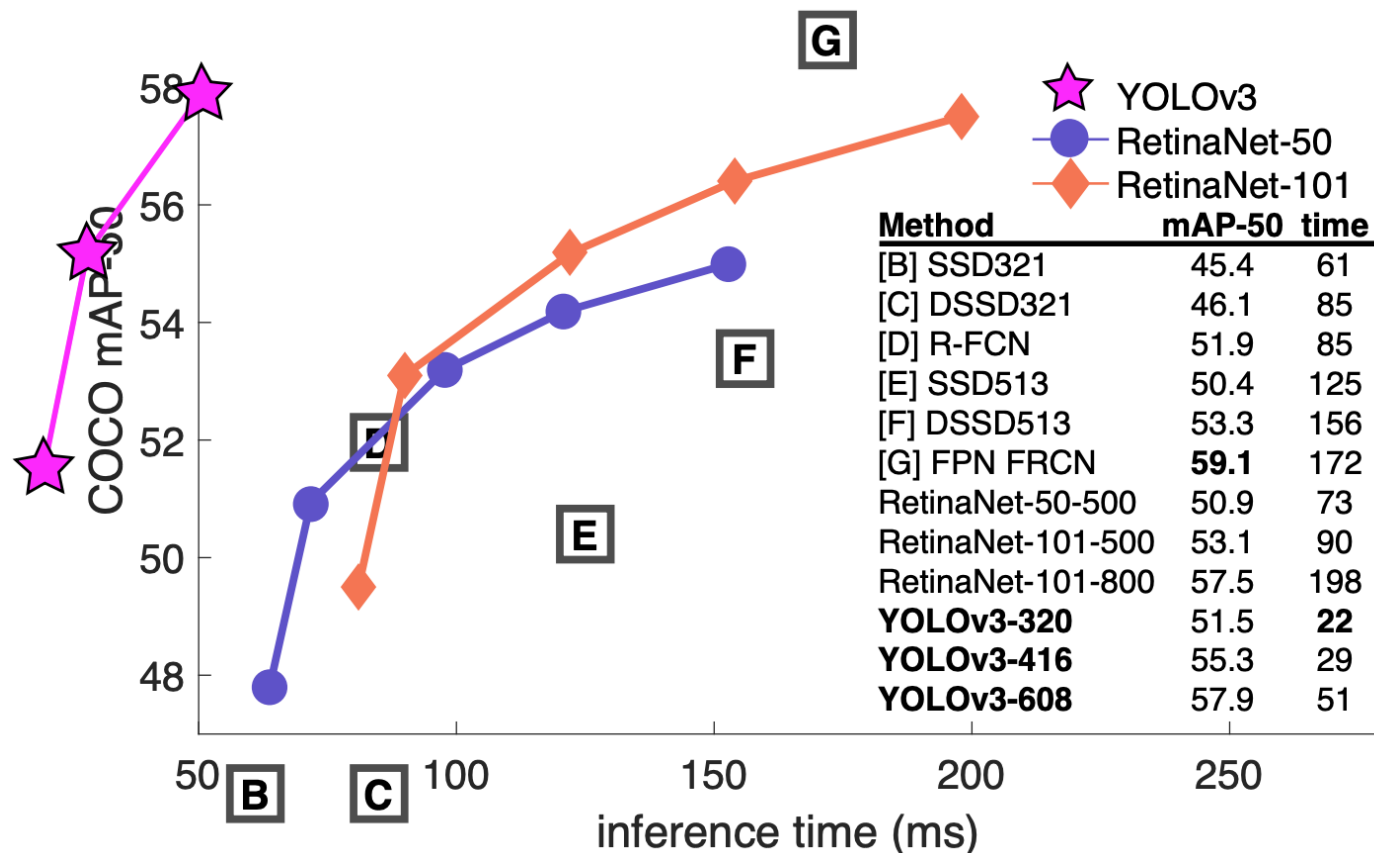
- VoC
 - 20개 클래스, 11,530 이미지, 27,450 annotation을 제공
 - image당 2.4개의 객체를 포함
- COCO
 - MS 사가 주최하는 COCO challenge 대회가 2015년부터 운영 중이며 여기서 제공되는 데이터가 COCO(common object context)
 - 91 카테고리, 20만개의 이미지, 50만개의 annotation, 32만개 영상을 제공
 - image당 평균 7.3 개의 객체를 포함

이미지 데이터 셋 (객체 검출) – 관련 기술



YOLO (You only look once)

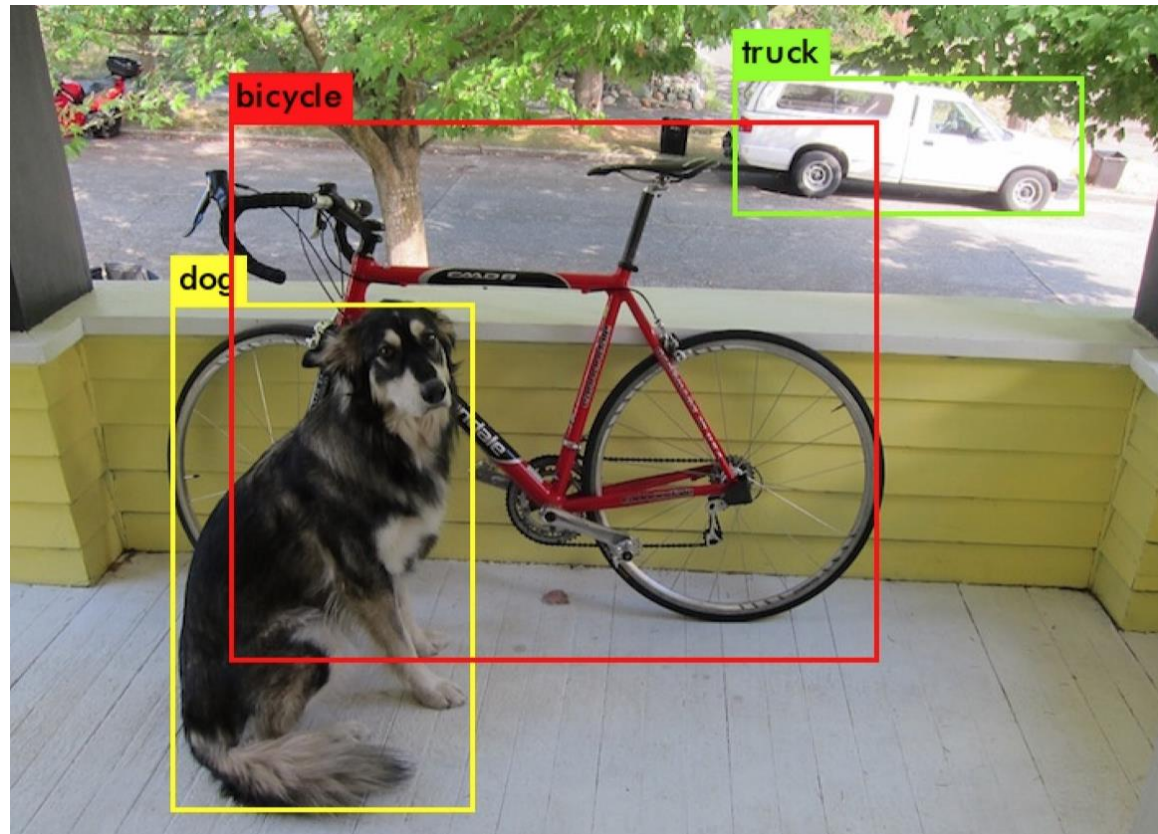
- 1-stage detector
- 현재 객체 검출 방식 중에 **속도가 가장 빠르고 성능도 우수**, 널리 사용



YOLO – 동작 데모

- 객체 인식(classification)과 위치예측(localization)을 동시에 하나의 신경망으로 수행
 - 즉, 객체 예측과 경계박스를 찾는 작업을 동시에 수행하는 Single Shot Detector(SSD)을 수행
 - 기존의 다른 객체 검출 알고리즘들은 별도로 수행
- 입력 이미지 전체를 여러 지역(region)으로 나누고 경계 박스와 각 지역에 대한 객체 존재여부를 확률로 예측
 - 이 경계박스는 예측확률의 가중합으로 계산
- 98개의 검출 결과를 제공 ($7 \times 7 \times 30 = 1470$ 벡터 출력)
- GoogleLeNet 기반으로 동작, C로 구현된 DarkNet을 framework로 사용
 - ❖ 참고: <https://github.com/qjwweee/keras-yolo3>
 - ❖ 동작 데모 : <https://pjreddie.com/darknet/yolo/#demo>

- 동작 데모



YOLO 모델 – 다른 모델과의 비교

- You Only Look Once
- 버전이 높아지면서 발전하고 있는 저명한 객체 인식 및 탐지 모델. 버전
에 따라 모델 명칭 뒤에 숫자가 넘버링
- 이 모델은 당시에 높은 성능과 처리 속도로 좋은 평가를 받고 있던 객
체 인식 및 탐지 모델의 성능을 압도하는 최고의 성능을 보여줌

객체 인식 및 탐지 모델	처리 속도
R-CNN	5초
Fast R-CNN	(영상) 0.5 프레임
Faster R-CNN	(영상) 7.0 프레임
YOLO	(영상) 45.0 프레임 [높은 버전의 YOLO의 경우 155프레임]

YOLOv3 모델 개요

- YOLOv3 모델은 기존의 YOLO, YOLOv2 모델을 개선한 버전의 모델
 - 이전 YOLOv2 보다, 1.5배 더 빠른 성능을 보여줌
- Multi-Scale의 특징(Feature)를 추출하며, **DarkNet-53 모델**을 사용
- 또한 클래스 예측 시에 소프트맥스(Softmax) 대신에 개별 클래스 별로 로지스틱 회귀(Logistic Regression)를 사용
- 기본 동작은 YOLOv2 모델과 비슷하며, 가장 큰 변화점은 **백본(Backbone) 네트워크로 ResNet-101 대신에 DarkNet-53 구조를 채택했다는 것**

YOLOv3 모델 개요

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	$3 \times 3 / 2$	64×64
2x	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	$3 \times 3 / 2$	32×32
8x	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	$3 \times 3 / 2$	16×16
8x	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	$3 \times 3 / 2$	8×8
4x	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLOv3 모델에서 채택한 DarkNet-53 구조

YOLO – Tiny YOLO3

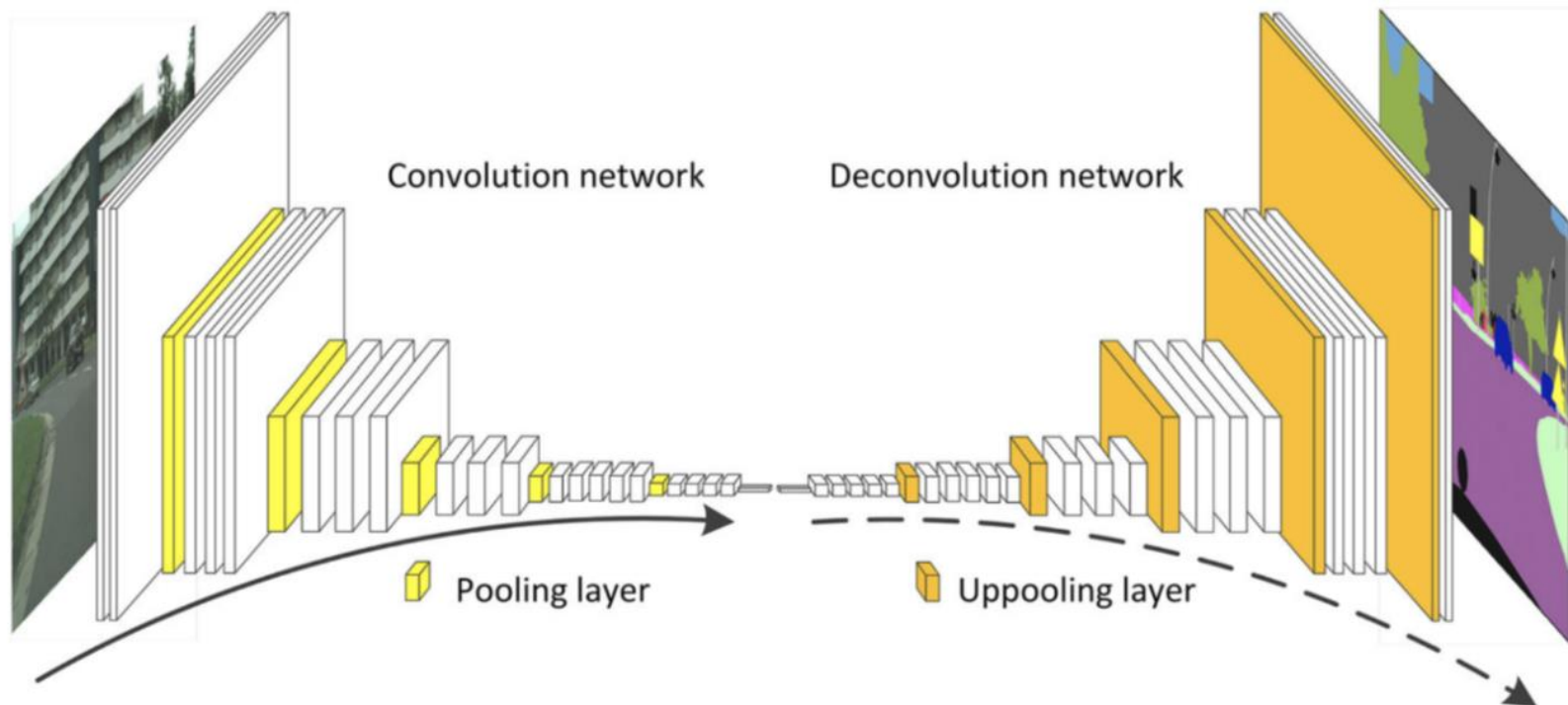
- 정식 YOLO보다 간단하면서 속도가 빠른 tiny YOLO 버전
- 이를 사용하려면 아래와 같이 해당 버전을 설치
 - `!wget https://pjreddie.com/media/files/yolov3-tiny.weights`

객체 분할 (Segmentation)

- 픽셀 단위로 객체의 클래스를 표시하는 작업
- 머신러닝 모델
 - 정보를 계속 줄여나가는 작업을 주로 수행
 - 분류와 회귀는 모두 정보를 줄인 결과 클래스 이름이나 수치를 얻는다.
- 객체 분할
 - 원래 이미지 크기의 이미지를 다시 얻어야 한다
 - 따라서 정보를 축소 한 후에 다시 정보를 늘리는 작업을 수행해야
 - Deconvolution, uppooling 수행

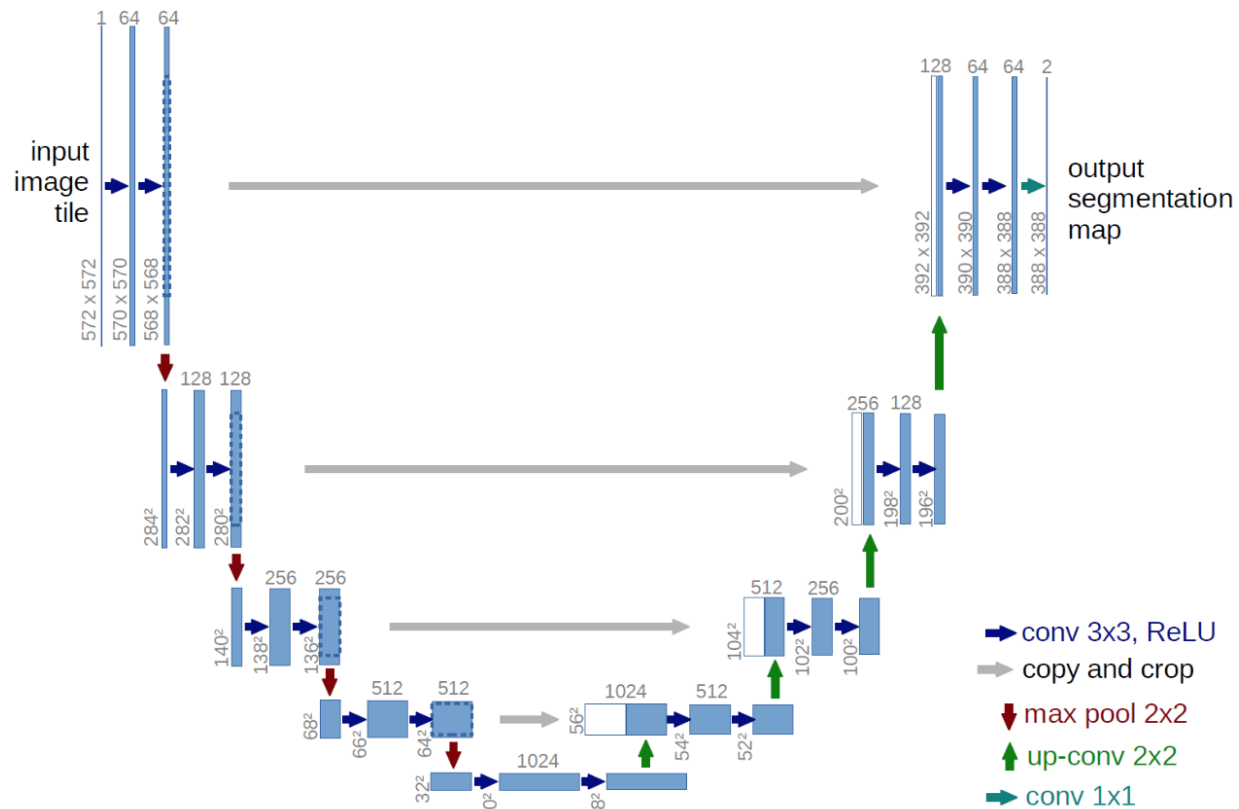
객체 분할 모델 구조

- Encoder-Decoder 구조로 구성



객체 분할 모델 구조 – U-Net

- 객체 분할을 위해서는 U자형 구조를 갖는 U-Net을 주로 사용

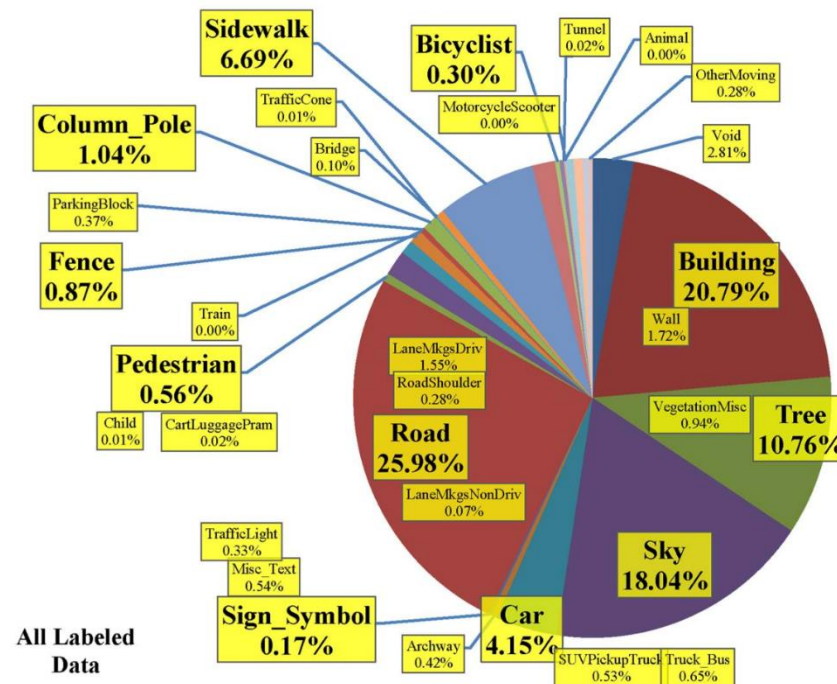


객체 분할 모델 구조 – U-Net

- 실행 결과
 - <https://www.youtube.com/watch?v=ATlcEDSPWXY>
- Mask R CNN과 비교 동영상 비교
 - https://www.youtube.com/watch?v=s8Ui_kV9dhw

객체 분할 – 데이터 셋

- 객체 분할을 위해서 사용되는 데이터 셋 : CAMVID, COCO 등
- CAMVID(Cambridge-driving Labeled Video Database) 데이터
 - <http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/>



객체 분할 – 데이터 셋

- COCO 데이터 셋
 - <http://cocodataset.org/#explore>
- 데이터의 특징
 - Object segmentation
 - Recognition in context
 - Superpixel stuff segmentation
 - 330K images (>200K labeled)
 - 1.5 million object instances
 - 80 object categories
 - 91 stuff categories
 - 5 captions per image
 - 250,000 people with keypoints

Q & A