

# CNN을 활용한 이미지 딥러닝 실습

## – Object Detection과 Segmentation –

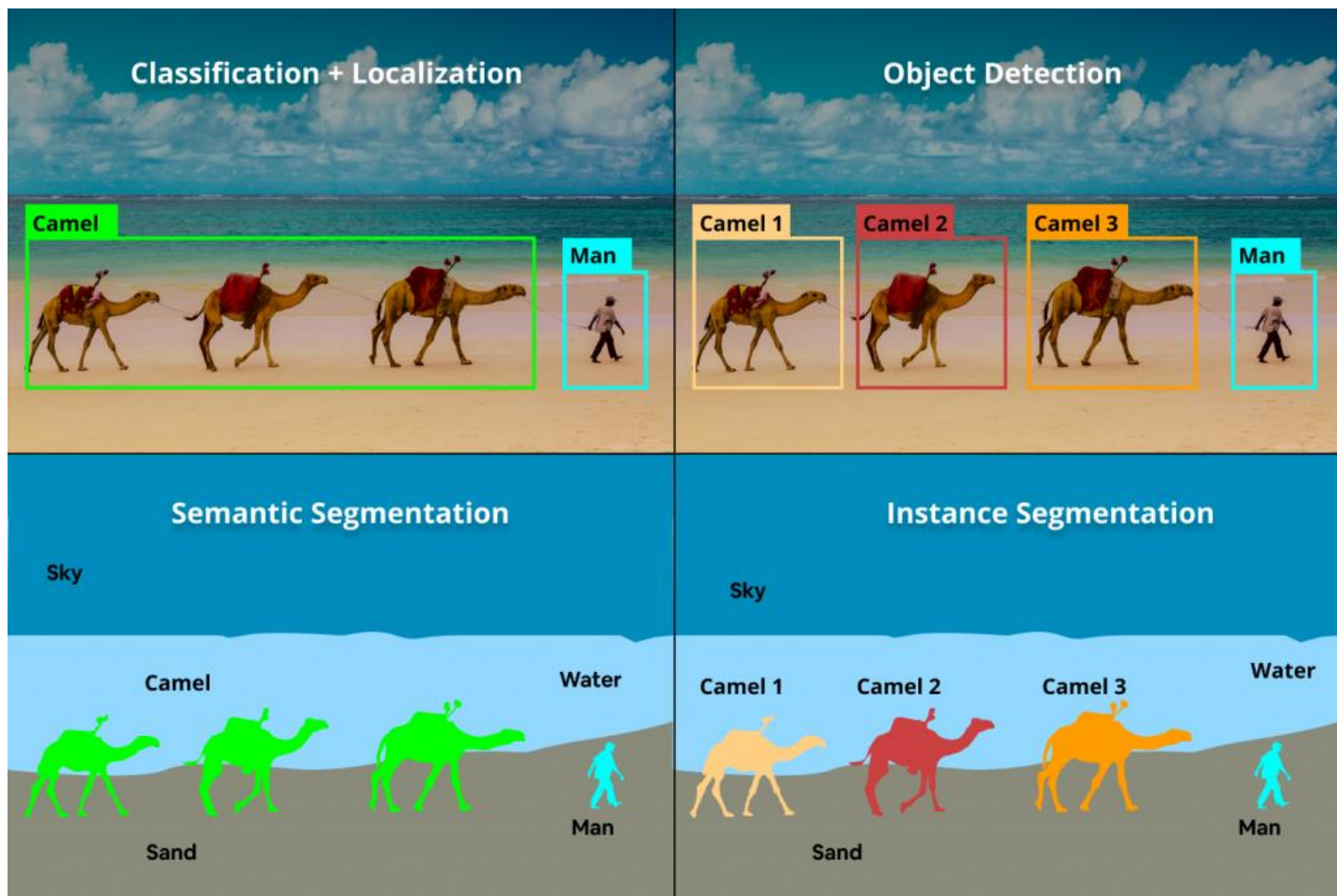
---



## Contents

- I. CHAPTER 1.
- II. CHAPTER 2.

# Object Detection과 Segmentation



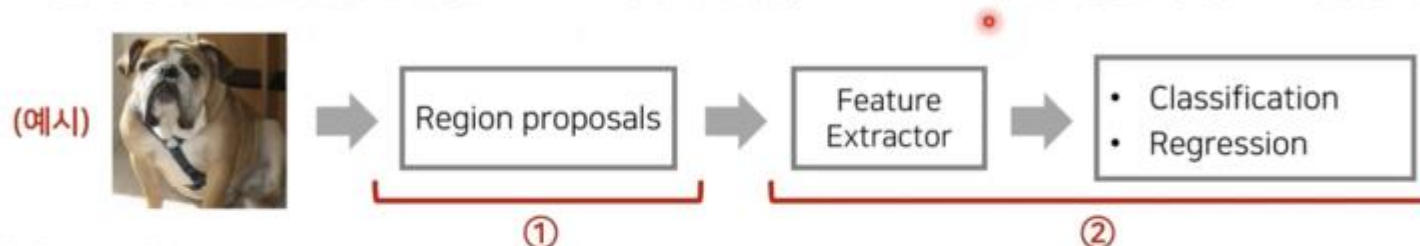
# CHAPTER 1

## 대표 논문 이론

# Object Detection

- 2-Stage Detector

- 물체의 ① 위치를 찾는 문제(localization)와 ② 분류(classification) 문제를 순차적으로 해결합니다.

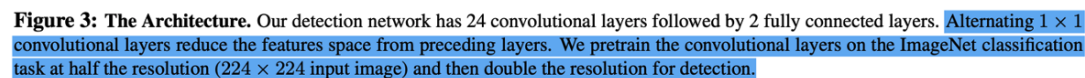


- 1-Stage Detector

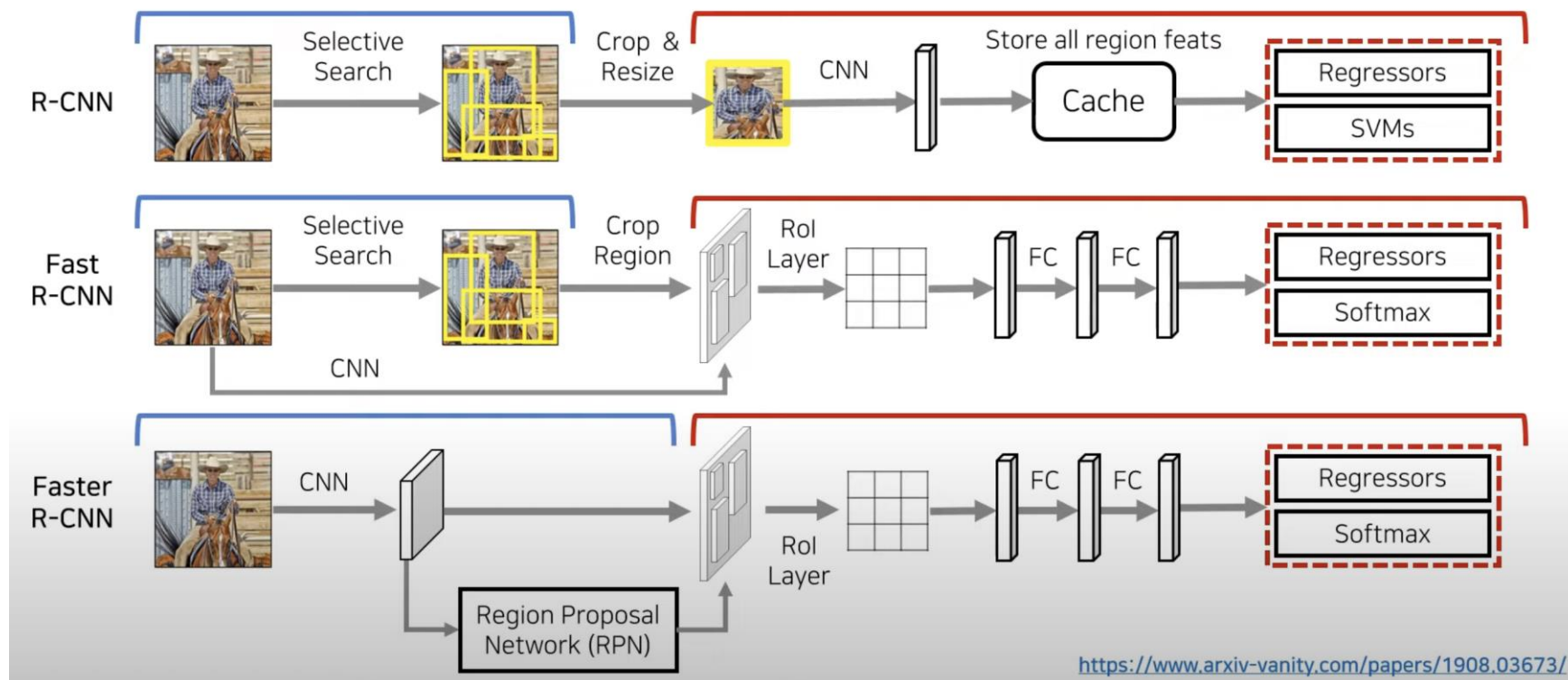
- 물체의 위치를 찾는 문제(localization)와 분류(classification) 문제를 한 번에 해결합니다.



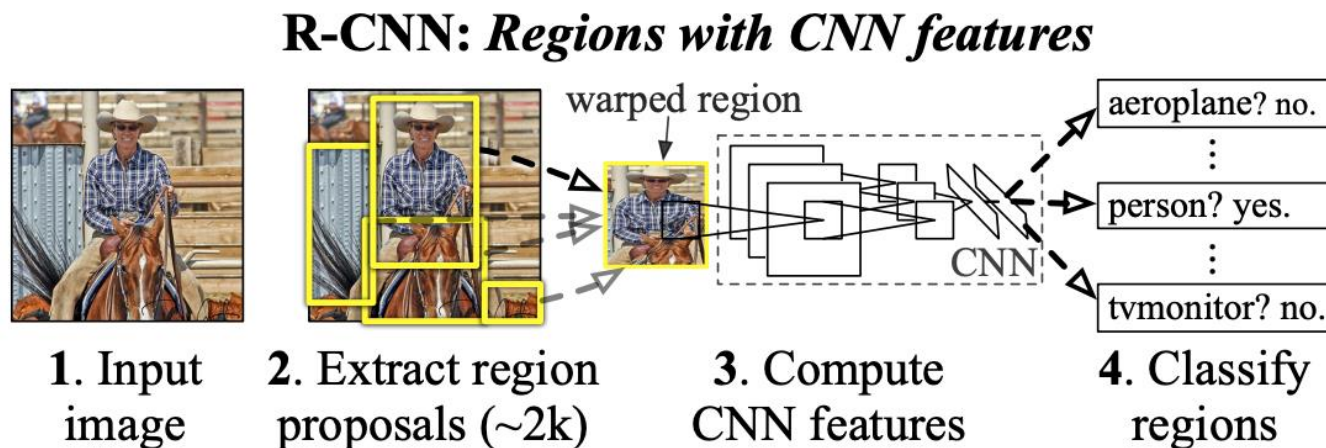
### R-CNN: *Regions with CNN features*



# Object Detection: R-CNN



# Object Detection: R-CNN



## 1. Region Proposal

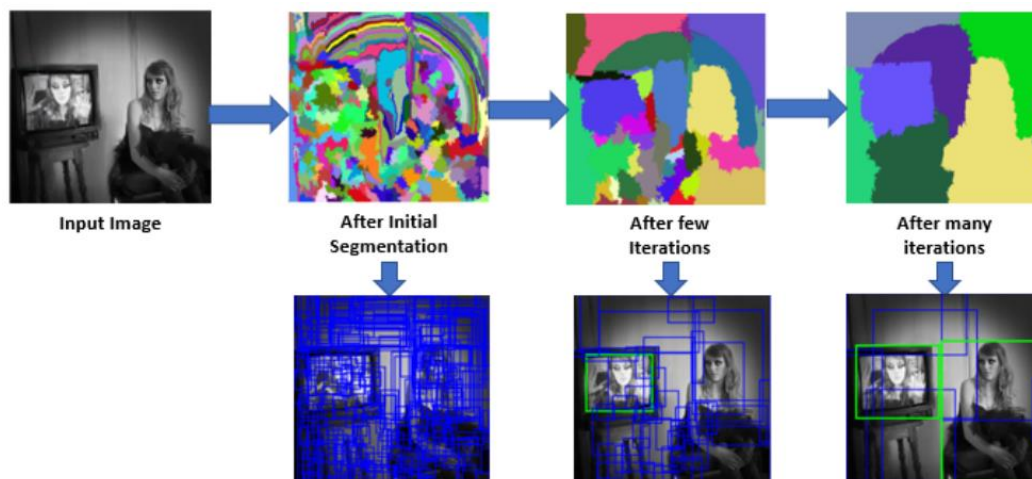
- Selective Search 사용
- Category-independent region proposal 생성
- Candidate region 을 지정함
- 약 2000개의 region proposal 추출



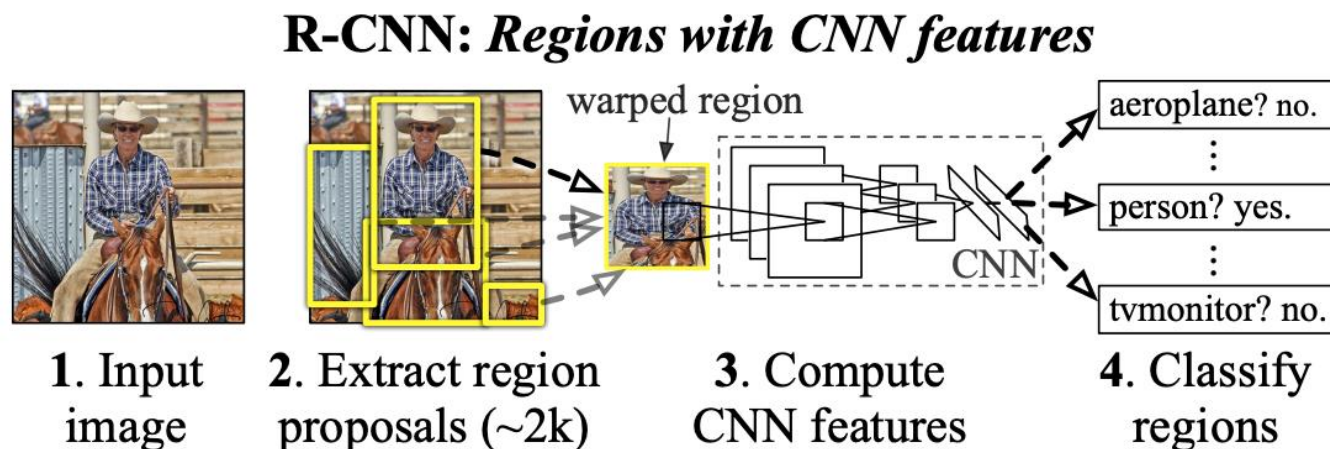
# Object Detection: R-CNN

## Selective Search

- segmentation과 유사도를 이용해 물체가 있을 법한 공간을 찾아내는 방법
  - 1. 입력 영상에 대해 segmentation을 실시해서 이를 기반으로 후보 영역을 찾기 위한 후보군을 설정
  - 2. 초기에는 많은 후보군이 생성됨
  - 3. 유사도를 기반으로 segmentation 처리된 컬러맵을 통합해나감 (가장 큰 유사도를 가진 두 영역을 선택한 후 큰 영역으로 결합하는 방식을 여러 iteration 동안 반복)
  - 4. 이를 기반으로 물체의 위치를 Localization 함
- 비교적 높은 정확도를 얻을 수 있지만, end-to-end로 학습이 불가능하고, 실시간 적용에 어려움이 있음.



# Object Detection: R-CNN

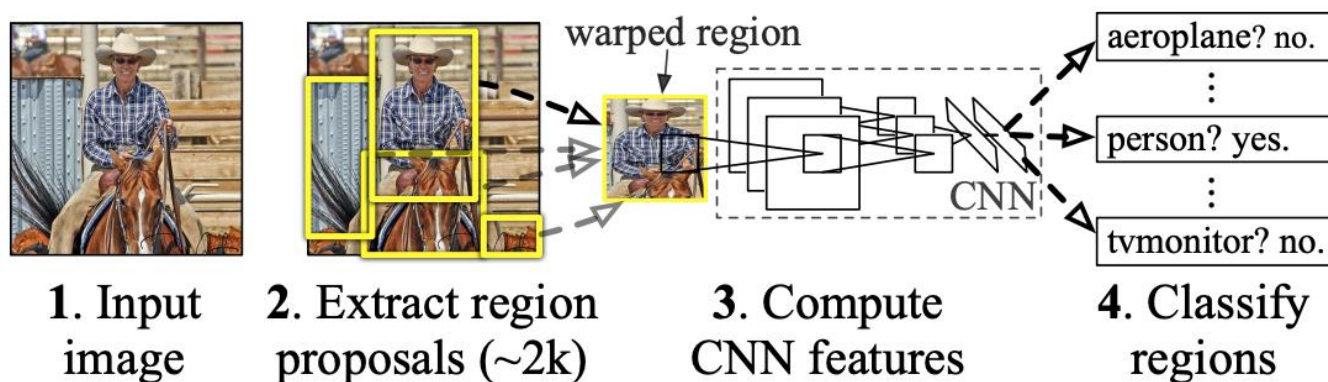


## 2. Feature Extraction (CNN)

- Selective Search로 얻은 후보군 region들을 CNN에 적용할 수 있도록 227 x 227 사이즈로 wrap 함 (찌그러뜨림) → 본래 사이즈나 비율을 고려하지 않고 모든 픽셀을 작은 bounding box 안에 wrap
- CNN을 통해 Feature vector (4096 dim)을 추출함 (AlexNet conv.)
- 227 x 227 RGB 이미지를 5개의 convolution layer와 2개의 fully-connected layers를 통해 forward propagation 해서 feature를 계산

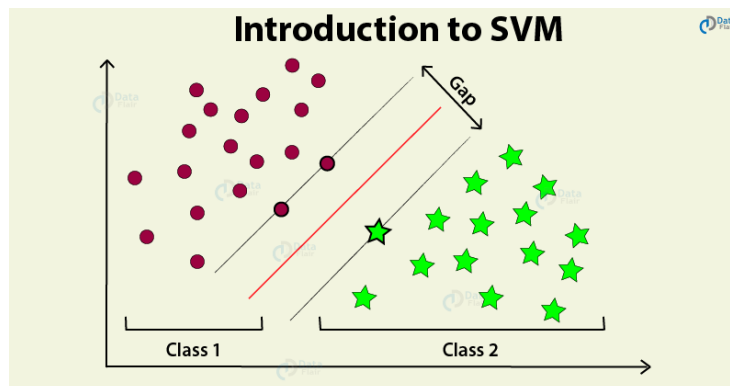
# Object Detection: R-CNN

## R-CNN: *Regions with CNN features*



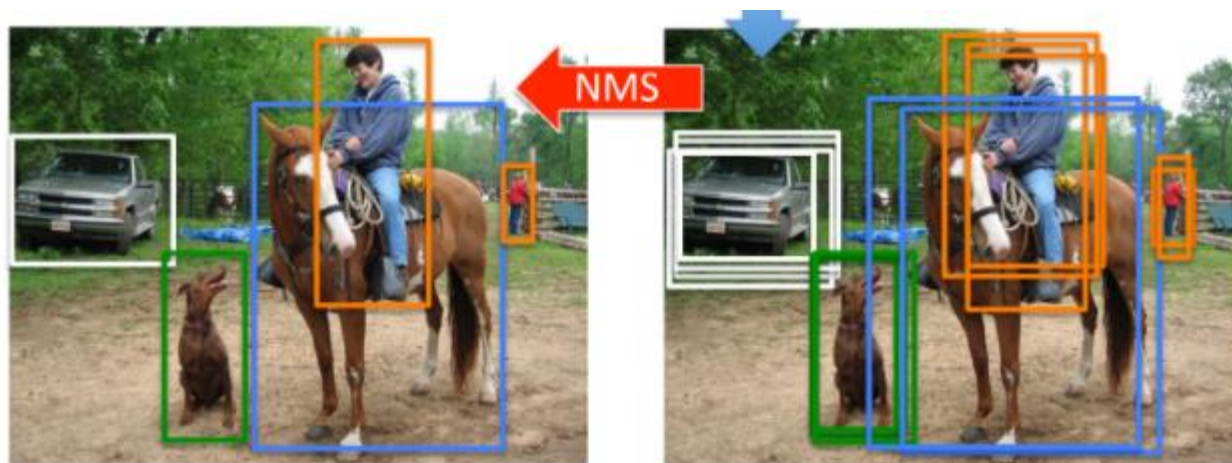
## 3. Classification (SVM)

- Support Vector Machine을 사용해서 클래스 별 score를 계산 후 region proposal을 분류함




## 4. NMS (Non-Maximum Suppression)

- Predict된 클래스별로 적용해서 중복을 제거함
- 하나의 Object를 가리키는 여러 개의 bounding box를 하나로 줄이기 위해서 사용됨
- Confidence가 높은 순서대로 내림차순
- IoU를 이용해서 box별로 값을 계산한 후 일정 기준(threshold) 이상이라면 같은 object를 가리키고 있다고 판단해 confidence가 낮은 box를 제거함



# Object Detection: IoU

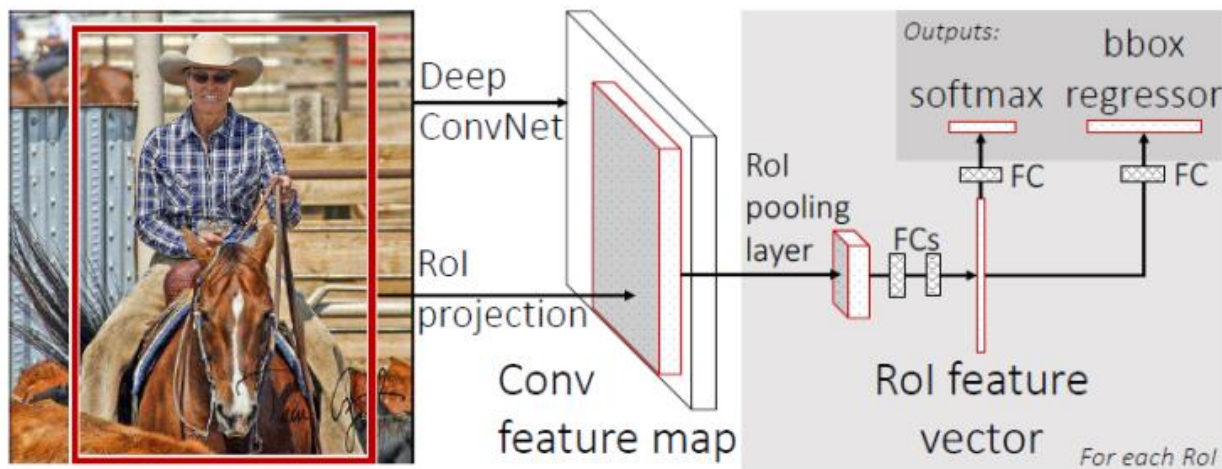
- Intersection of Union
- 두 장의 사진에서 서로 겹치는 부분의 비율
- 두 Region proposal 사이의 유사도를 비교 할때 사용됨
- 혹은 모델 학습에서 prediction과 target 이미지의 유사도를 계산할 때도 사용됨

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


## 5. Box Regression

- 예측된 box의 Localization 성능을 올리기 위해 수행
- Region proposal box 좌표  $P=(P_x, P_y, P_w, P_h)$  (박스의 중심좌표와 width, height)
- Ground truth box 좌표  $G=(G_x, G_y, G_w, G_h)$
- Region proposal box가 ground-truth box로 변형되기 위한 mapping을 학습함.

# Object Detection: Fast R-CNN

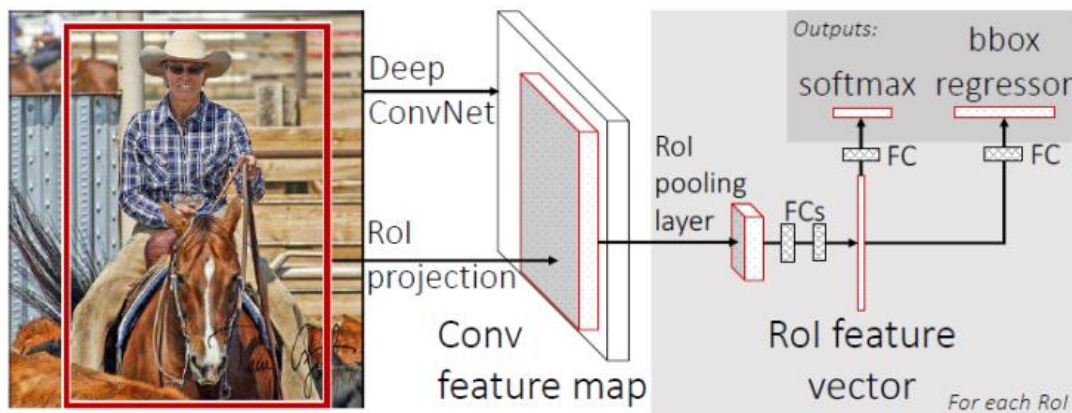


## 1. 이전 연구의 문제점

- Rough Localization 들에 대해 더 정밀한 정제가 필요하다.
- R-CNN은 CNN-fc\_layer, SVM classification, Box regression 세 단계로 구성되어 느리고 낮은 유연성을 가진다.
- 2000개의 proposal을 메모리와 디스크에 write 하는 것은 시간, 공간적으로 expensive
- Proposal을 생성하고 정제하는 과정을 모든 test 이미지에도 적용하므로 object detection 결과 도출 자체가 느림.



# Object Detection: Fast R-CNN

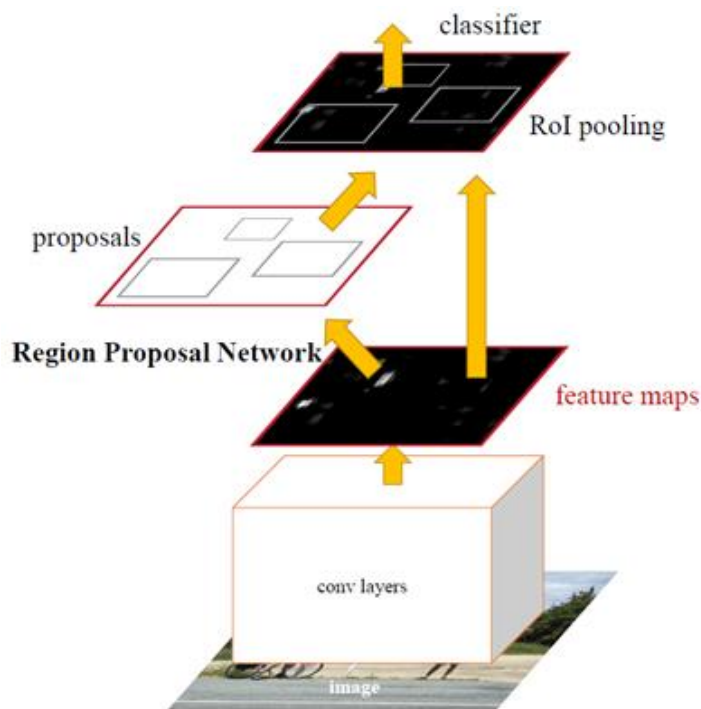


## 2. Solution

- Proposal의 분류와 정제를 single-stage training algorithm으로 통합함
- 1. Selective search로 region proposal을 뽑아냄 (wrap X)
- 2. 원본 이미지 한 장을 CNN에 넣어 feature map을 추출
- 3. Region proposal과 feature map을 projection 해서 object proposal을 생성
- 4. RoI Pooling layer에 넣어 max pooling을 통해 고정길이 feature vector 생성
- 5. FC Layer로 들어간 후 output을 두 갈래로 나눔
  - 1) softmax 적용 후 K개의 클래스들에 대한 확률 값을 구함 → Classification
  - 2) 네 개의 실수 값을 클래스 별로 생성. 이 값으로 클래스에 따라 Box Position을 인코딩 함.



# Object Detection: Faster R-CNN



## 1. 이전 연구의 문제점

- Region proposal 방식의 오랜 시간 소요

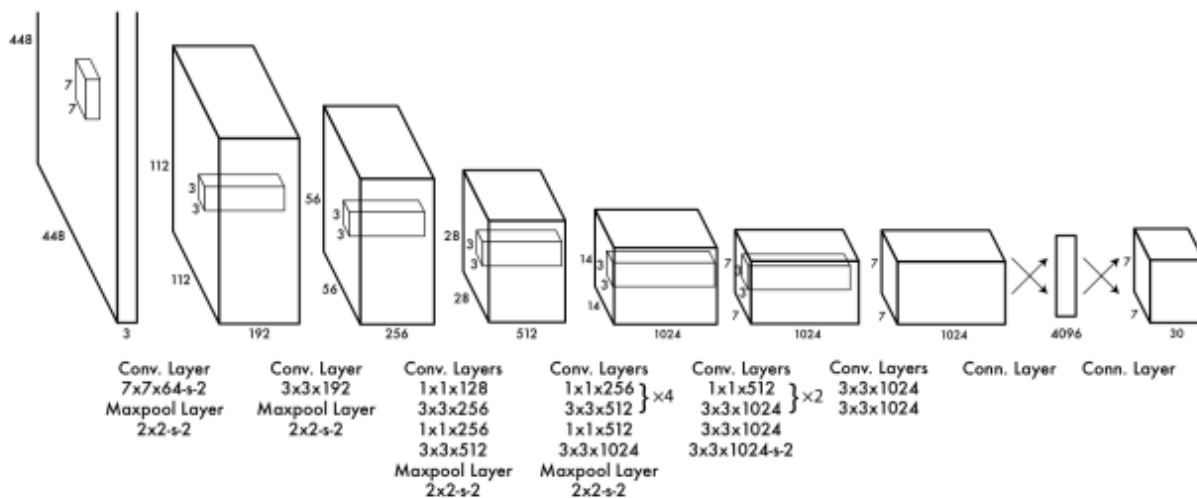
## 2. Solution

- RPN (Region Proposal Network) 제안
- Region Proposal을 CPU가 아닌 GPU를 활용해 신경망 구조 안에서 해결함 → 실제 detection에서 큰 속도 개선

1. RPN은 ImageNet을 사용하여 학습된 모델로부터 초기화 되어 Region proposal task를 위해 end to end로 학습 (pretrain)
2. 1에 학습된 RPN에서 Region Proposal만 가져온 후 Fast R-CNN 모델 학습
3. 다른 파라미터는 고정하고 RPN에만 연결된 층을 학습 (finetune)
4. 공유하는 CNN과 RPN은 고정하고 Fast R-CNN만 학습

# Object Detection: Yolo

One-stage: Yolo

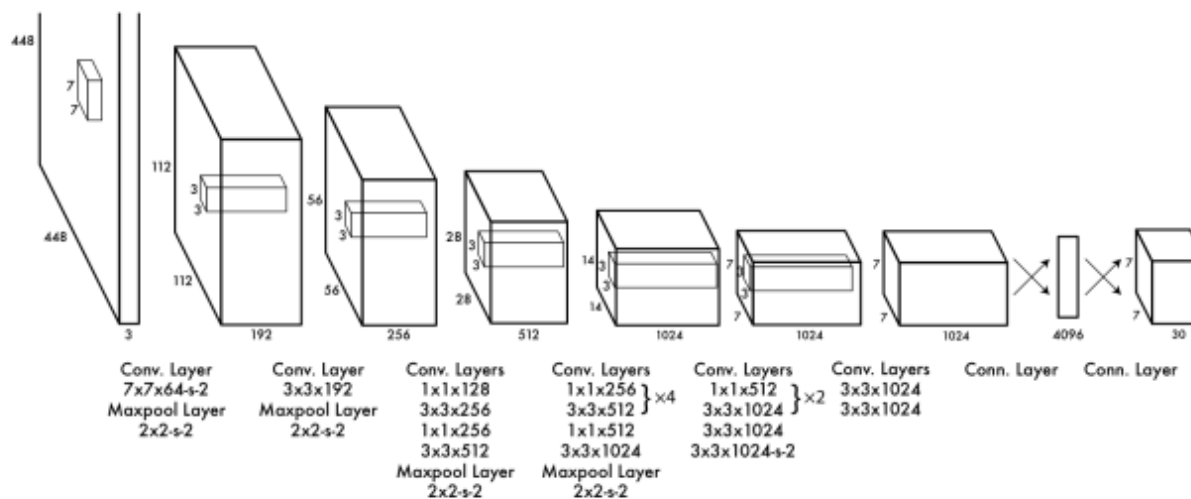


**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

- Object Detection 문제를 Bounding box와 classification probabilities에 대한 Regression 문제로 생각
  - Single Neural Network가 Bounding Box와 class probabilities를 full input image에서 바로 계산
- ➔ End to End

# Object Detection: Yolo

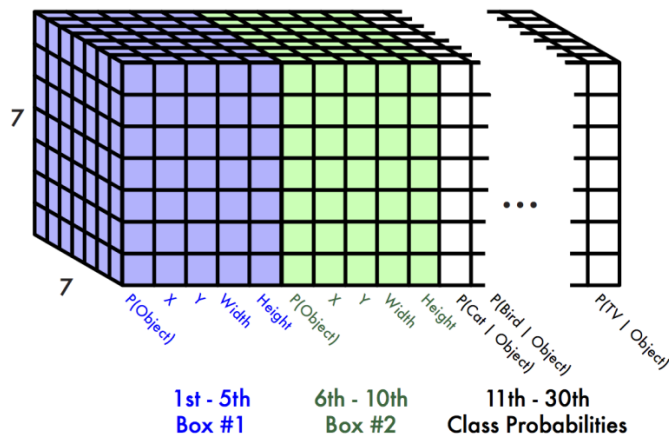
One-stage: Yolo



**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

1. Resize Input Image 448 x 448
2. Run CNN
3. Non-max suppression (NMS)

# Object Detection: Yolo



loss function:

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)
 \end{aligned}$$

# Segmentation



Input

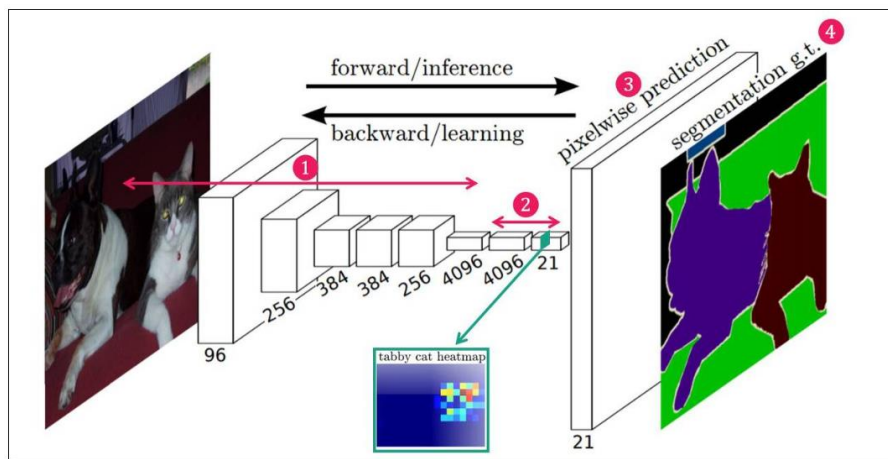


- 1: Person
- 2: Purse
- 3: Plants/Grass
- 4: Sidewalk
- 5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4	4

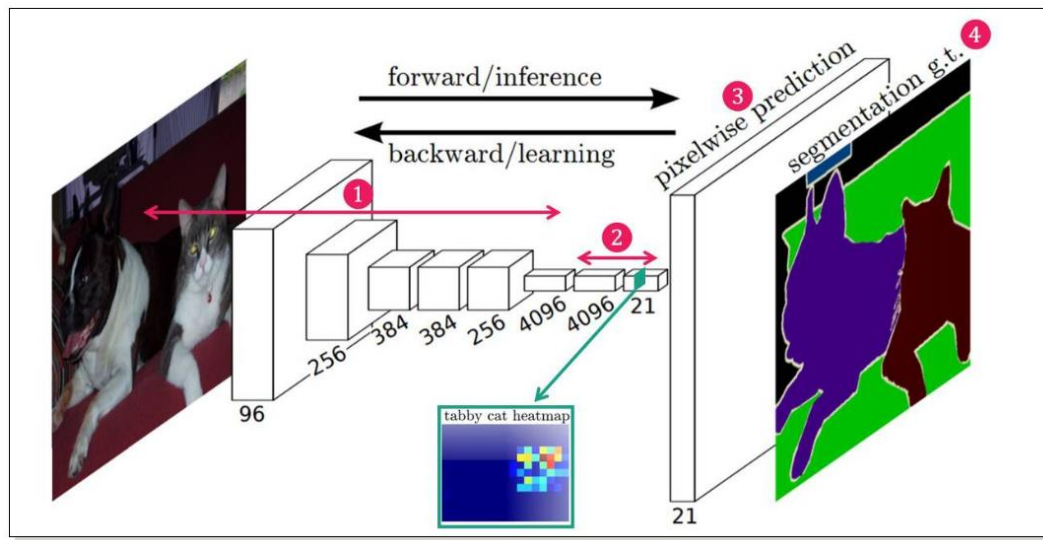
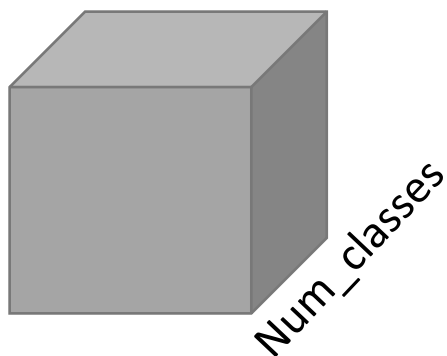
Semantic Labels

# Segmentation 대표 모델: FCN



- VGG, AlexNet과 같은 기존 CNN classification 모델들을 segmentation에서 수정 없이 재사용하는 것은 불가능
- 왜냐하면 Fully connected layer에 들어가면서 위치정보가 소실되기 때문 (Segmentation은 픽셀들 사이 위치 정보가 매우 중요)
- FCN에서는 위치정보의 소실을 방지하기 위해, 그리고 어떤 크기의 입력 이미지도 허용하기 위해(segmentation에서는 고해상도 이미지를 주로 사용함) fully-connected 층을 1x1 convolution 층으로 교체
- 결과적으로 네트워크 전체가 convolution 층으로 이루어짐

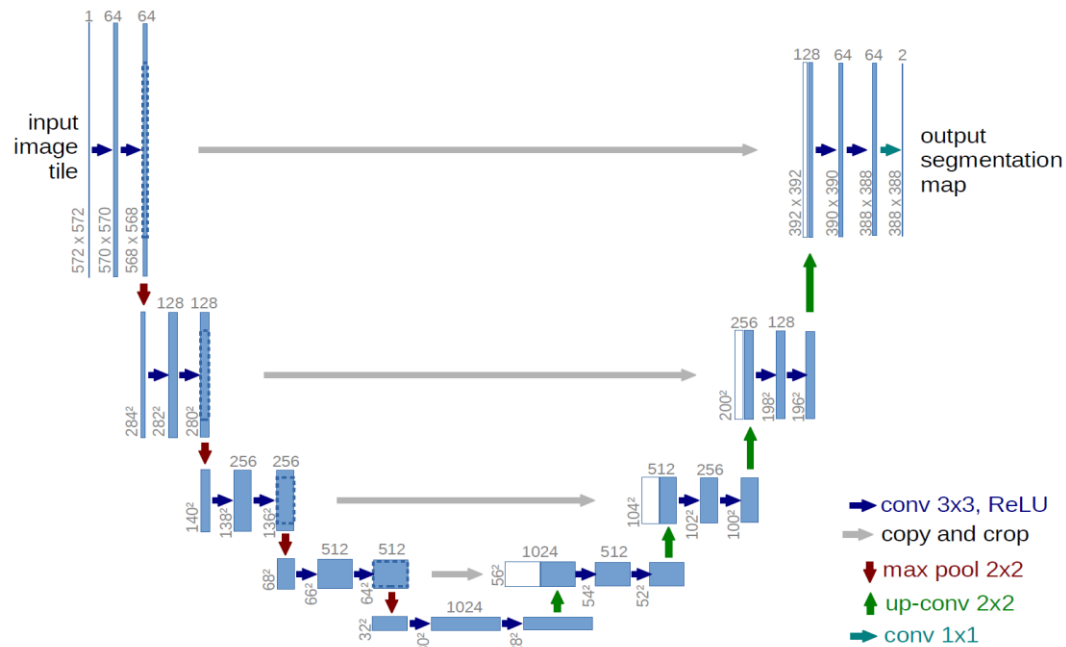
# Segmentation 대표 모델: FCN



- Convolution 연산을 거쳐 나온 output feature map의 channel(depth)는 클래스 개수와 동일
- 즉, 한 장의 feature가 클래스 하나를 대표함
- Down sampling(cnn) 과정에서는 raw 이미지에서 high level feature를 추출
- (H/32, W/32)의 크기를 가지는데, 이는 feature map의 한 픽셀이 원래 이미지의 (32, 32) 만큼을 대표함을 의미
- 픽셀 당 softmax 연산, 해당 픽셀이 n번 클래스에 속할 확률이 높다면, n-1 번째 feature에서 그 픽셀의 값이 가장 높은 확률을 가지도록 학습

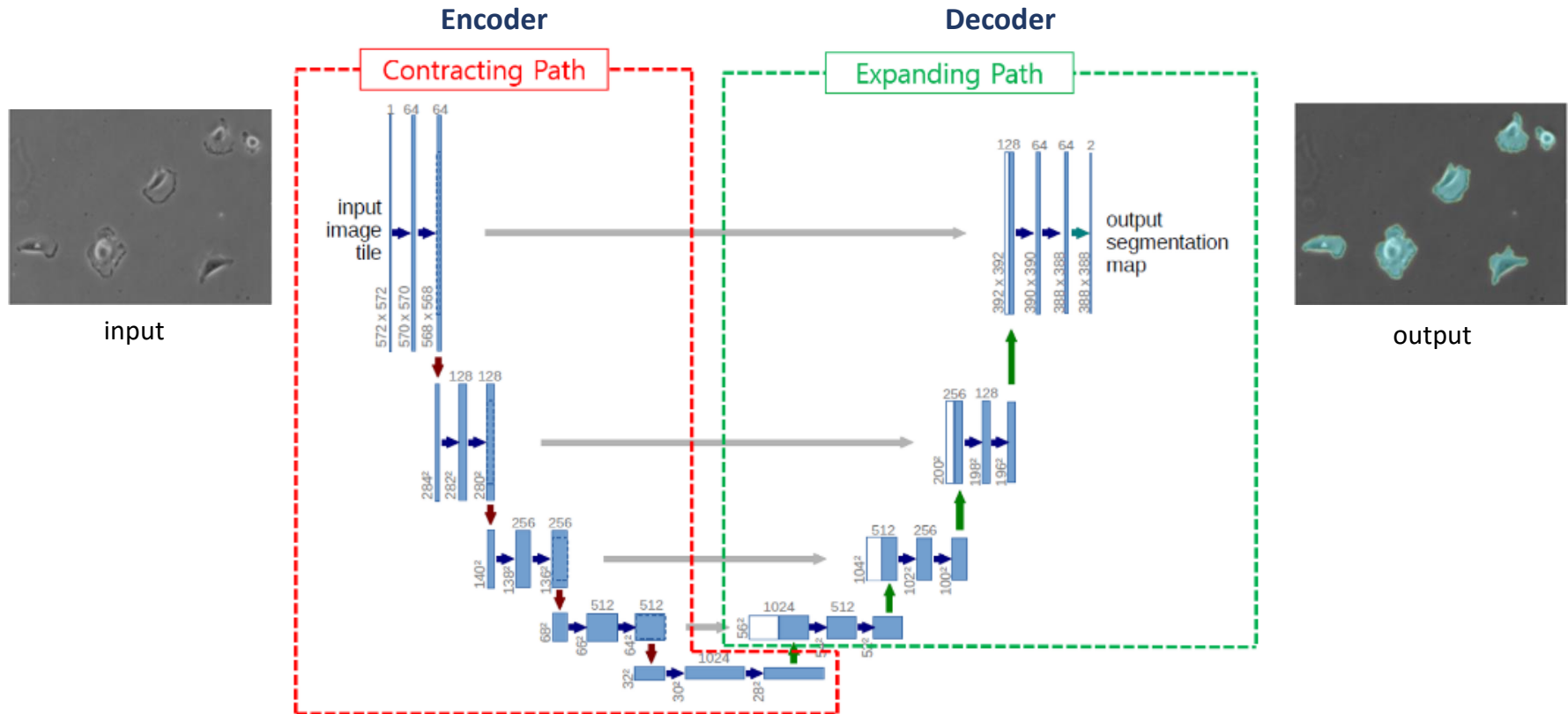
# Segmentation 대표 모델: UNet

## U-Net: Convolutional Networks for Biomedical Image Segmentation





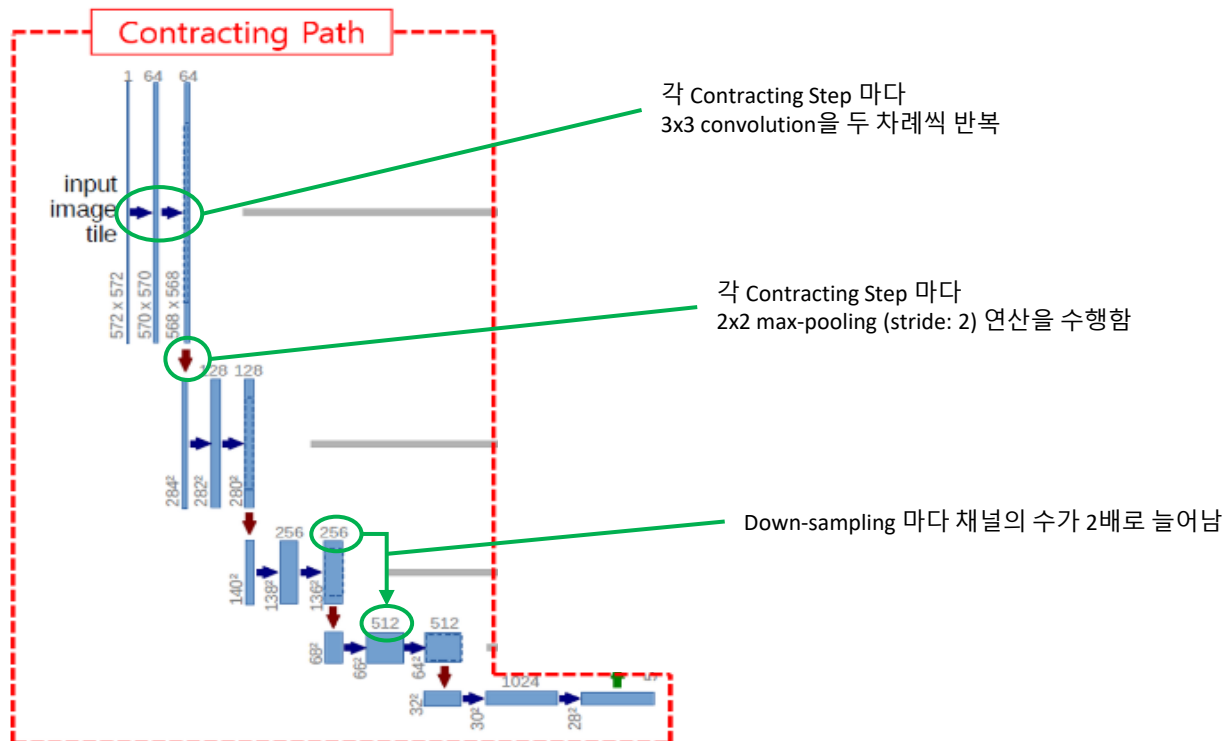
# Segmentation 대표 모델: UNet



# Segmentation 대표 모델: UNet

## Contracting Path

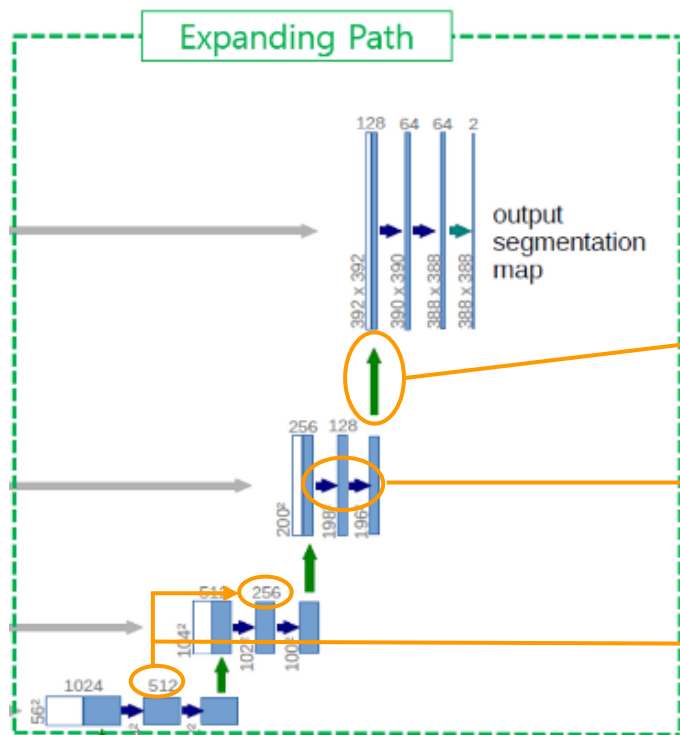
입력 이미지의 Context 포착을 목적으로 구성



# Expanding Path

세밀한 Localization을 위한 구성.

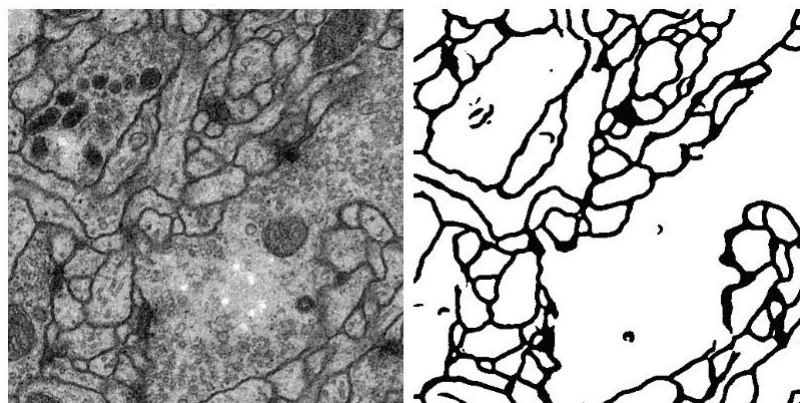
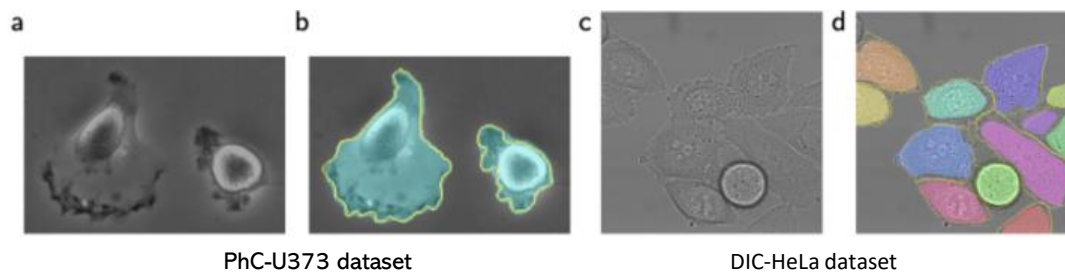
높은 차원의 채널의 갖는 Up-sampling 앞은 레이어의 특징맵을 결합



각 Expanding Step 마다 2x2 Up-convolution을 수행할 때, Feature map의 크기가 두 배로 늘어남

각 Expanding Step 마다 3x3 convolution을 두 차례씩 반복 (ReLU 포함)

Up-sampling 마다 채널의 수가 절반으로 줄어듬



# CHAPTER 2

## 스크래치를 통한 Object Detection

### 모델 학습 실습

수고하셨습니다.