# 파이썬 뿌수기

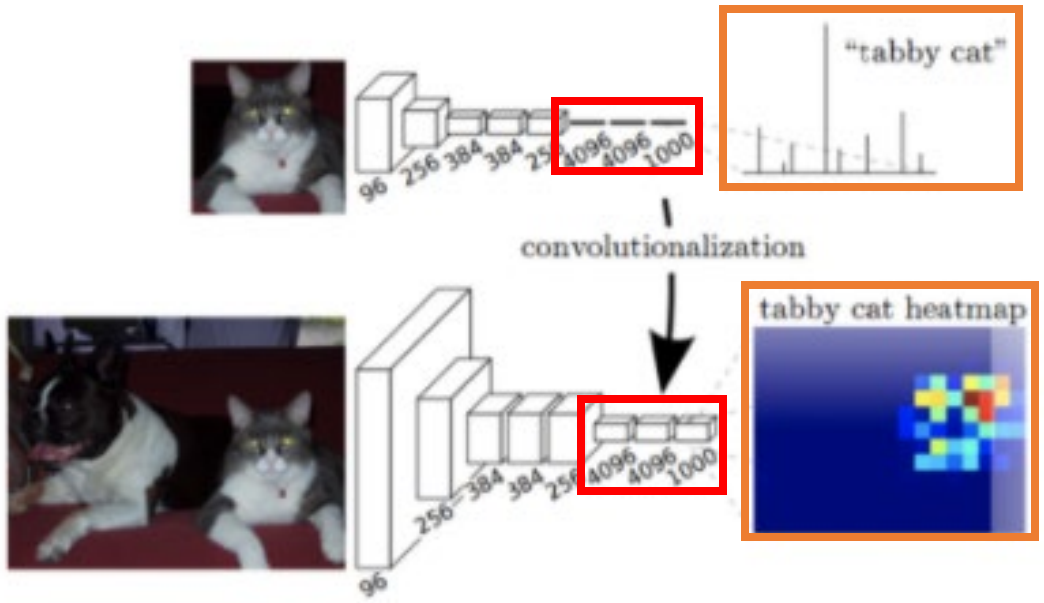
DeconvNet, SegNet

2020.10.28
김현우

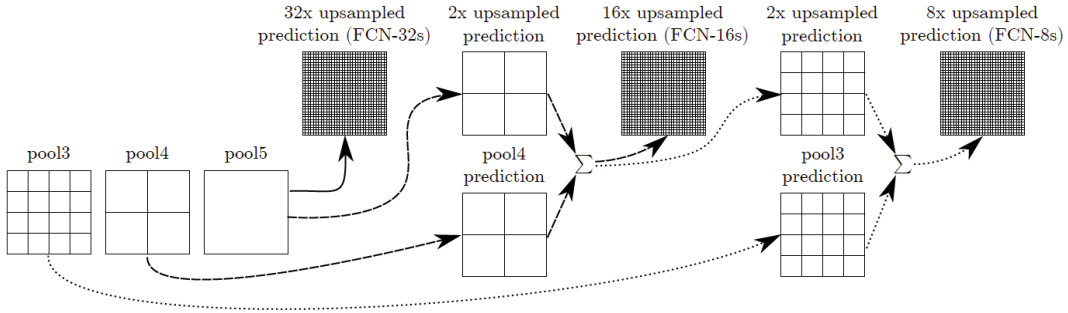
# 1 Fully Convolutional Network

## Fully Convolutional Network 요약

1. Fully Connected Layer를 Fully Convolutional Layer로 변경




2. Residual Connection 기법을 이용



# 2 Deconvolution Network

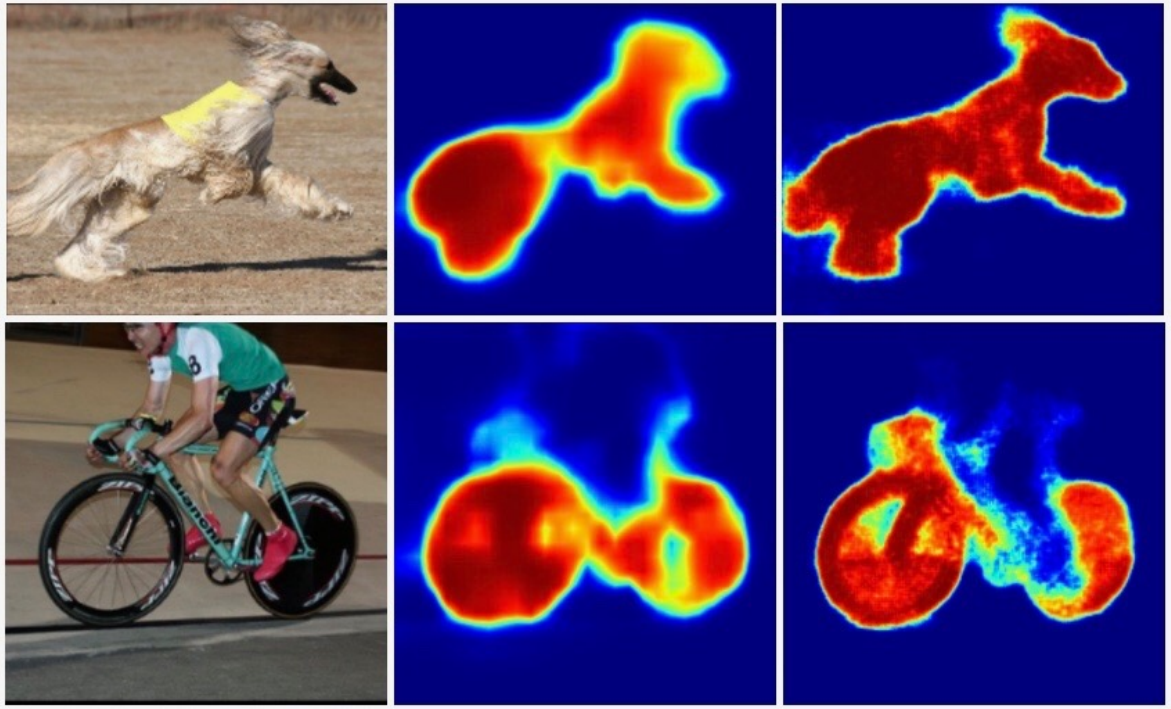
## FCN의 단점 (전반적인 Segmentation 문제)

### 1. Object의 크기에 약함


(a) Inconsistent labels due to large object size


(b) Missing labels due to small object size

### 2. Detail한 부분을 맞추지 못함



(a) Input image (b) FCN-8s (c) Ours

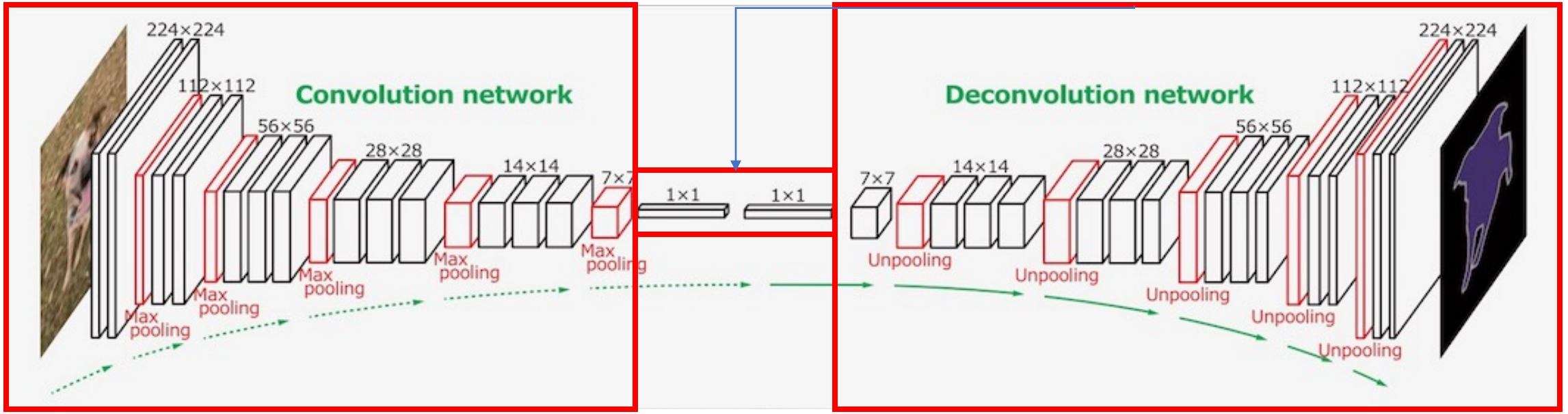
# 2 Deconvolution Network

## 문제를 해결하는 방법

Layer를 더 깊게 쌓자 !!!

**1x1 Convolution (Fully Connected Layer)**

- Fully Connected Layer라는 표현을 사용했지만, Convolution을 이용했음




VGG 네트워크

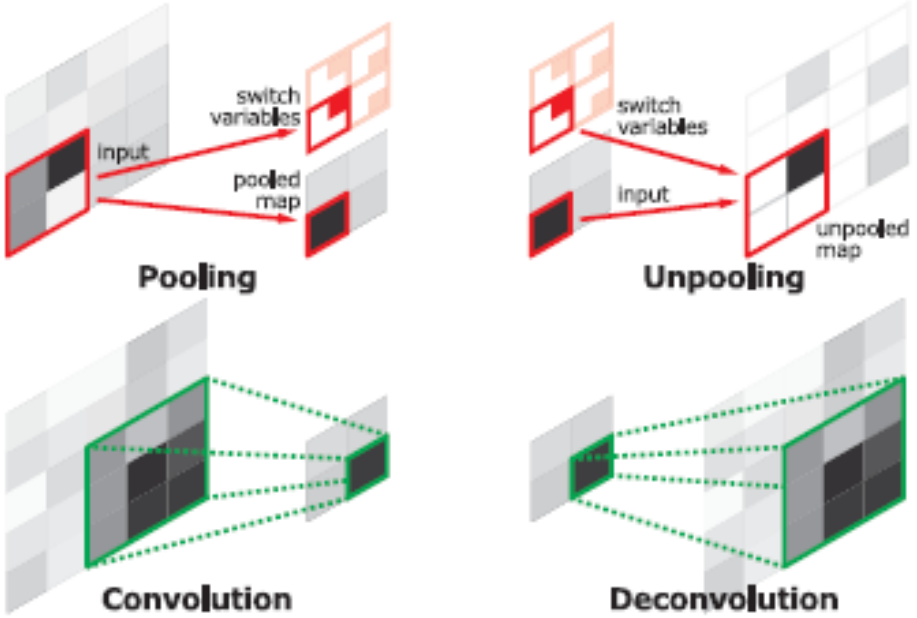
VGG 네트워크 대칭

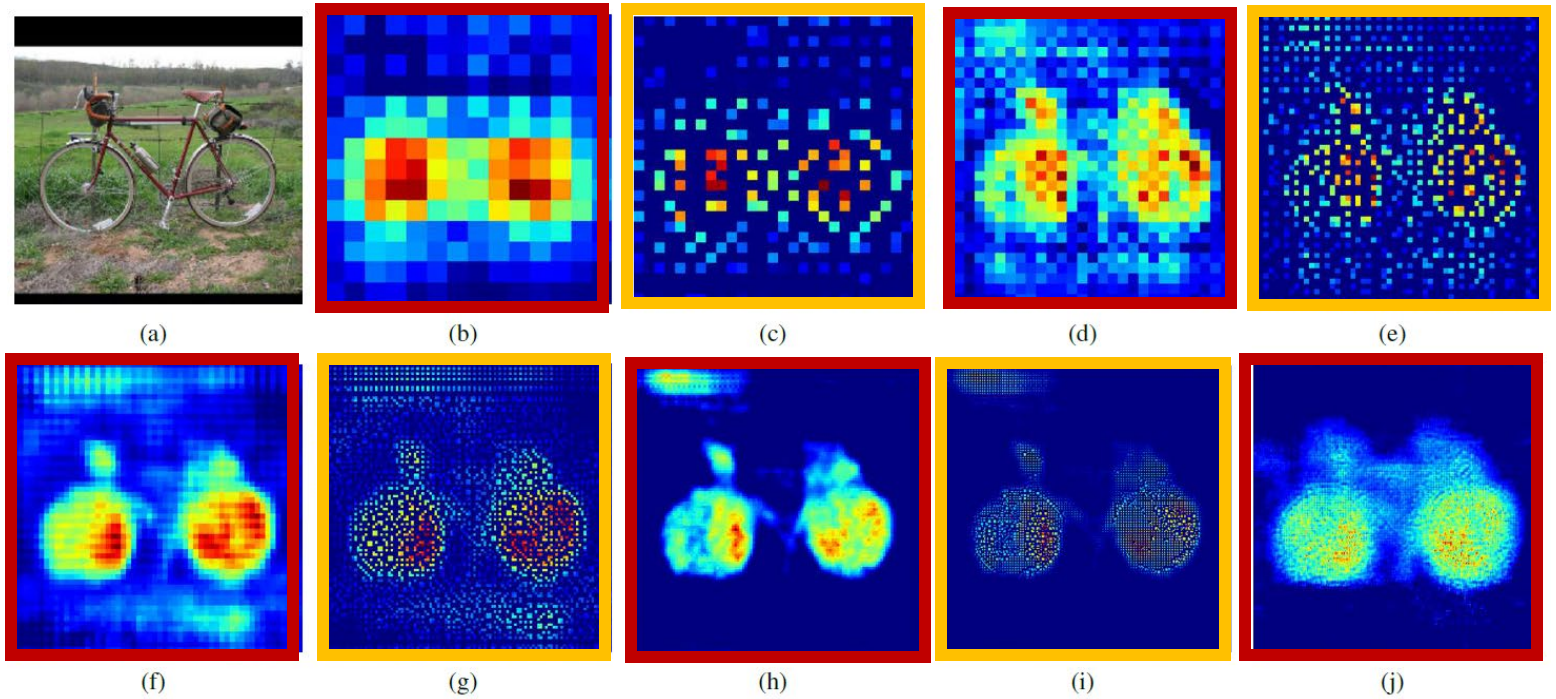
# 2 Deconvolution Network

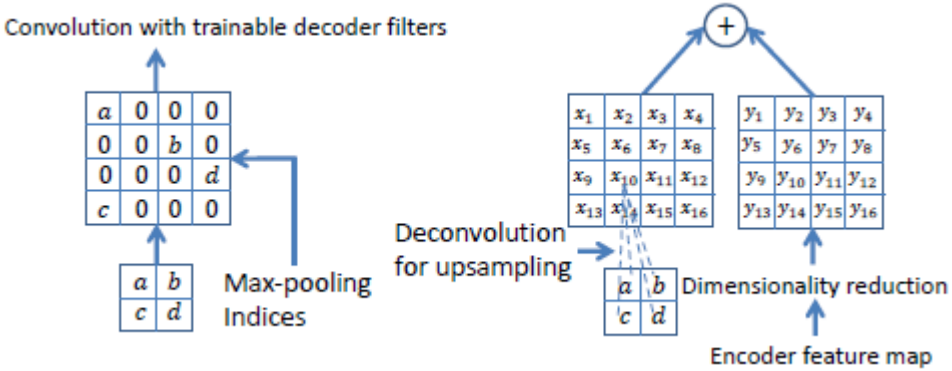
## Pooling 방법

- Unpooling : 이미지의 디테일한 경계를 학습
- Deconvolution : 이미지의 전반적인 부분을 학습

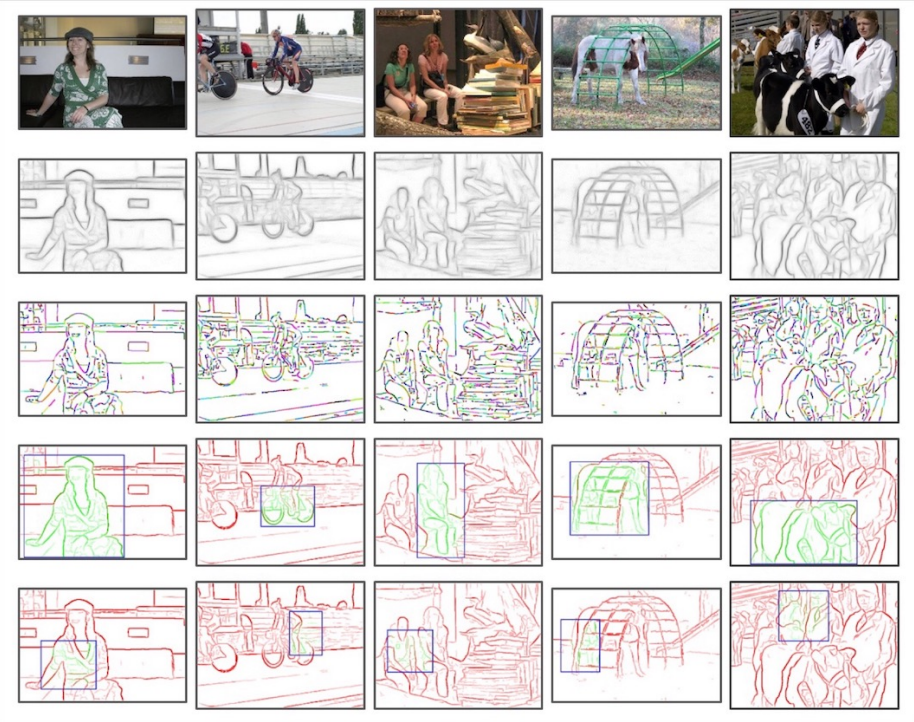
-> 이미지의 Detail한 부분을 해결



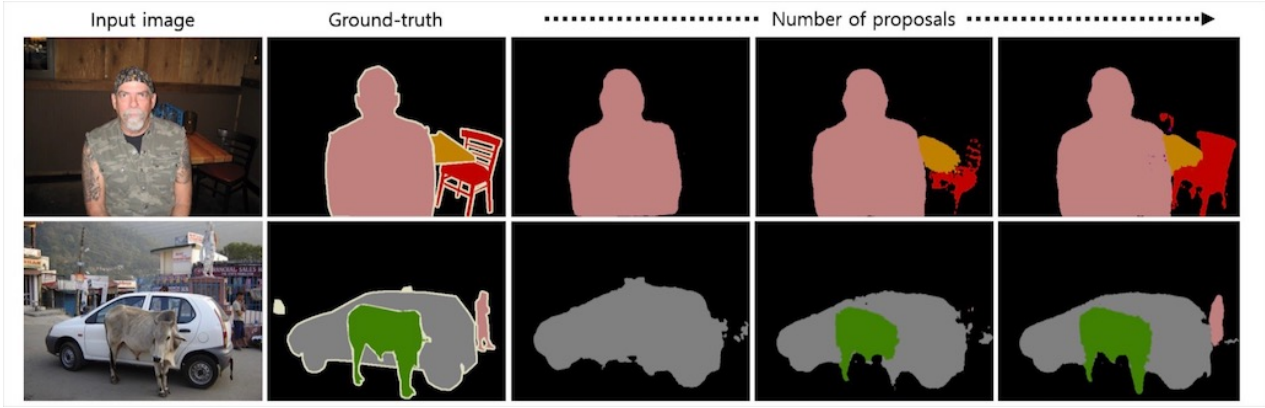



# 2 Deconvolution Network



## Two-Stage Training

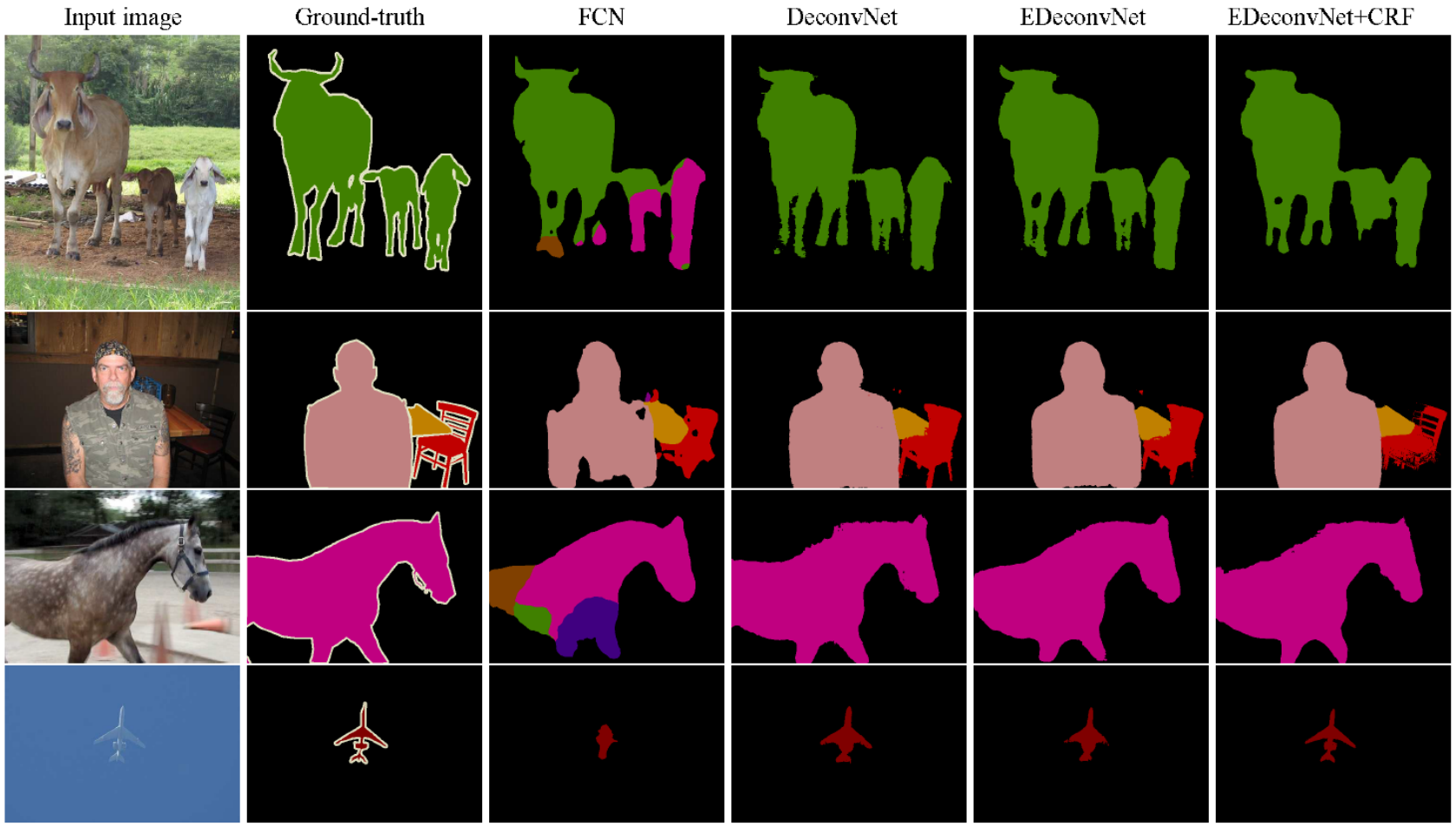
1. First Stage Training (Edge-box Object Proposal)
   - 객체가 있을 영역을 다양한 크기의 상자로 확인
   - 학습시에는 우선 실제 정답이 상자의 가운데에 들어가도록 이미지를 Crop
2. Second Stage Training
   - 1차 학습 후, IoU가 일정 값 이상인 이미지로 2차 학습 진행
   - 이렇게 학습한 edge-box를 추론 시에도 적용(edge-box의 수가 늘수록 성능 상승)




Learning Details
- Batch Normalization
- VGGNet weigh로 ConvNet을 initialized
- DeconvNet은 zero-mean Gaussians으로 initialized
- 64 Batch Size

# 2 Deconvolution Network

## Results



# 3 SegNet

## Abstract

1. Encoder – Decoder으로 구성된 Pixel Wise Segmentation 방법
2. Upsampling 방법을 이용한 보간 방법
3. 메모리와 계산시간의 효율적인 방법
   - VGG16의 네트워크를 사용하고, Fully Connected Layer을 제거
   - Upsampling시에 Maxpooling에서 사용한 index를 가져옴 (Unpooling)

# 3 SegNet

## Purpose

1. 자율주행과 관련된 구조들을 pixel-wise segmentation 하기 위해 설계된 모델

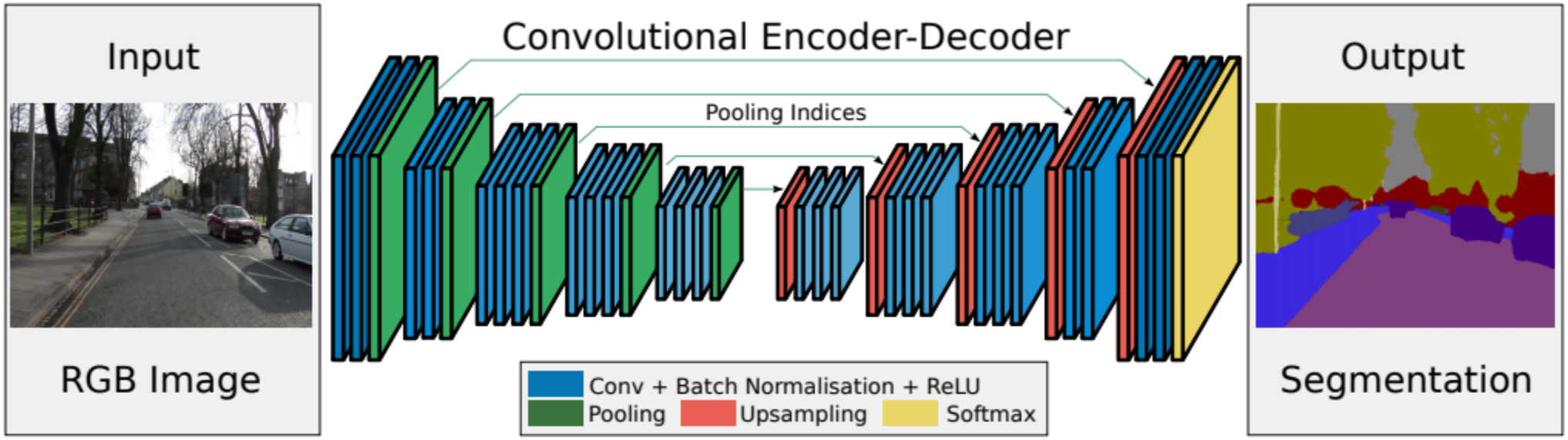
2. Object간의 관계가 있다는게 중요한 포인트

   - (차, 도로), (나무, 잔디), (차, 차), (보행자, 횡단보도), 등등

# 3 SegNet

## Network

1. Encoder – Decode 형태의 네트워크 구성

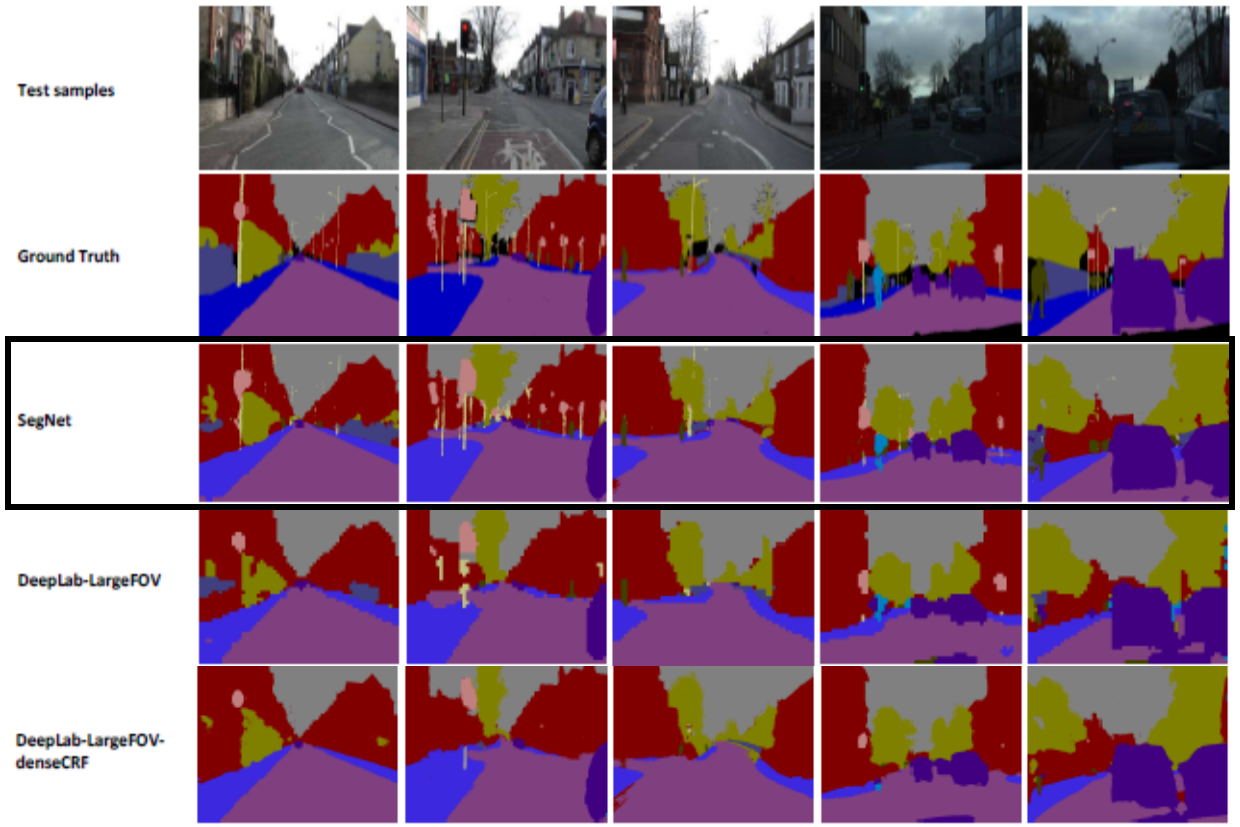


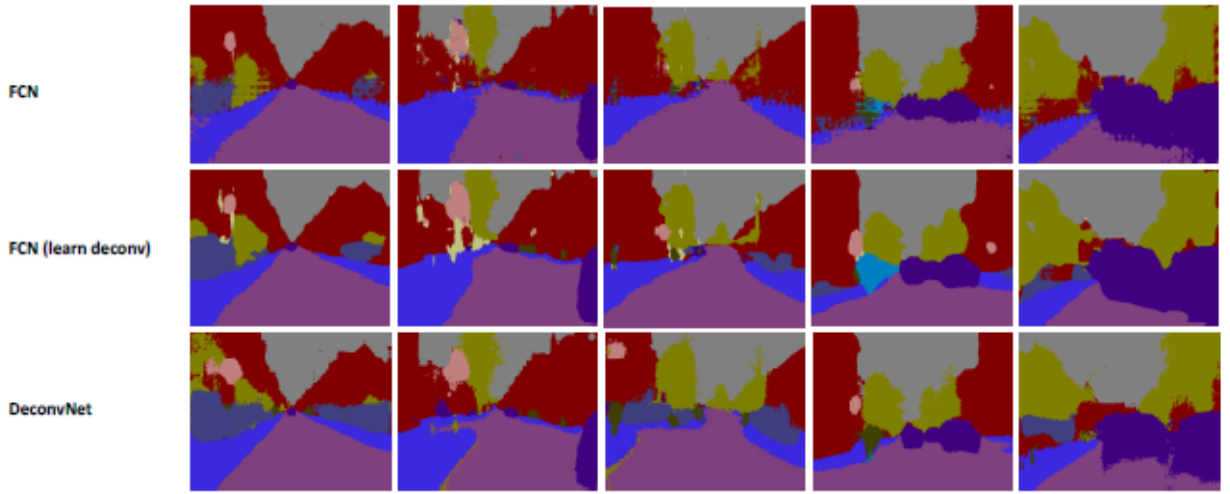

2. Upsampling시에 Pooling Indices 사용 -> 파라미터 수가 줄어서 속도 향상 및 이전의 정보 활용

3. Fully Connected Layer 제거 -> 속도 향상

# 3 SegNet

## Results

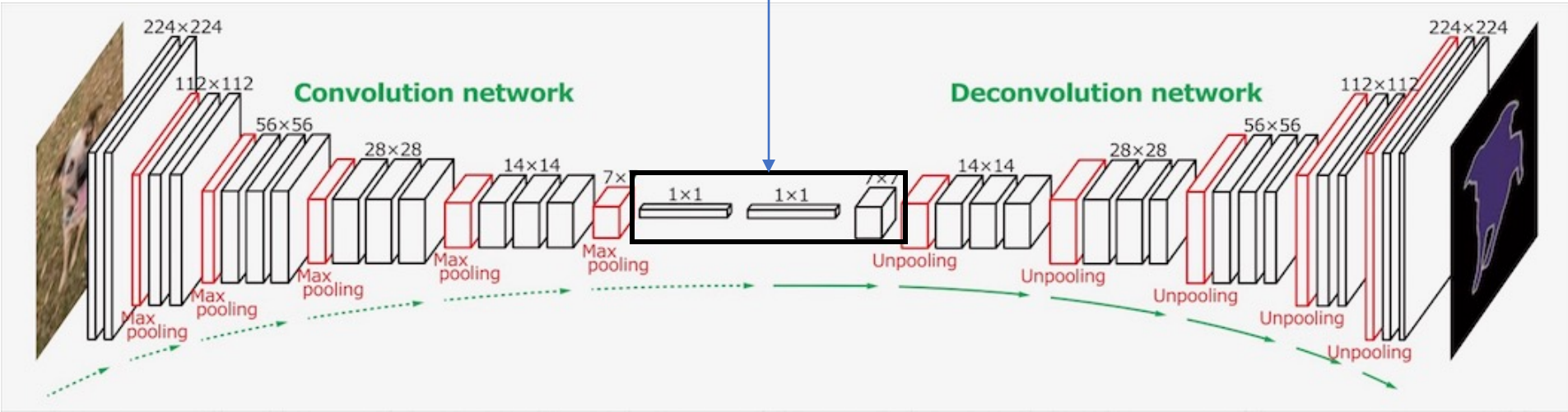


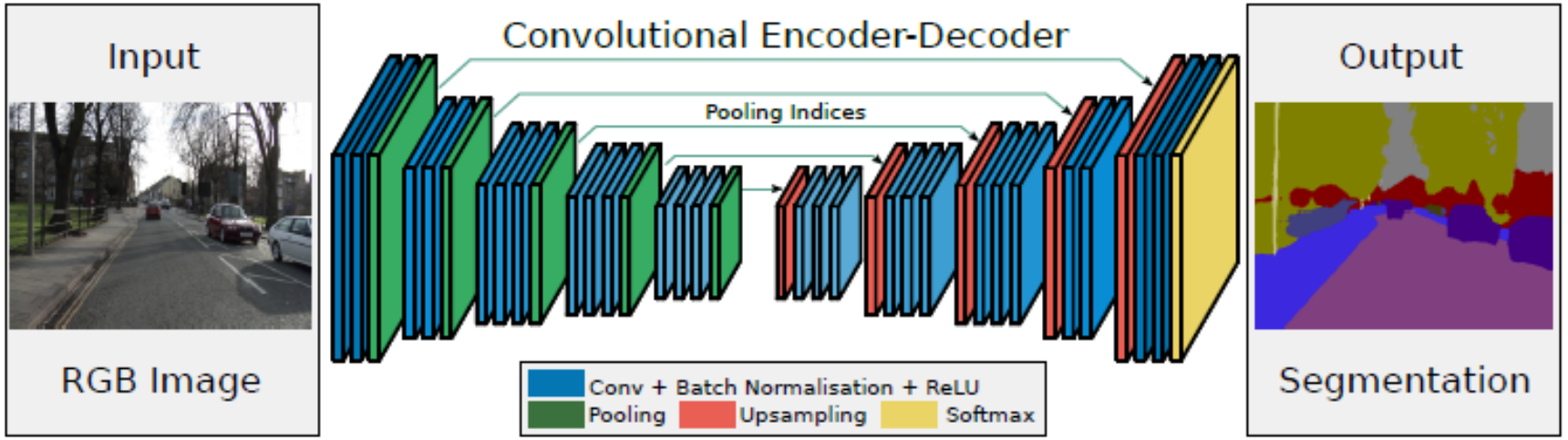



# 4 Appendix

## DeconvNet vs SegNet

Fully Connected Layer의 존재
-> Memory와 속도의 차이가 발생

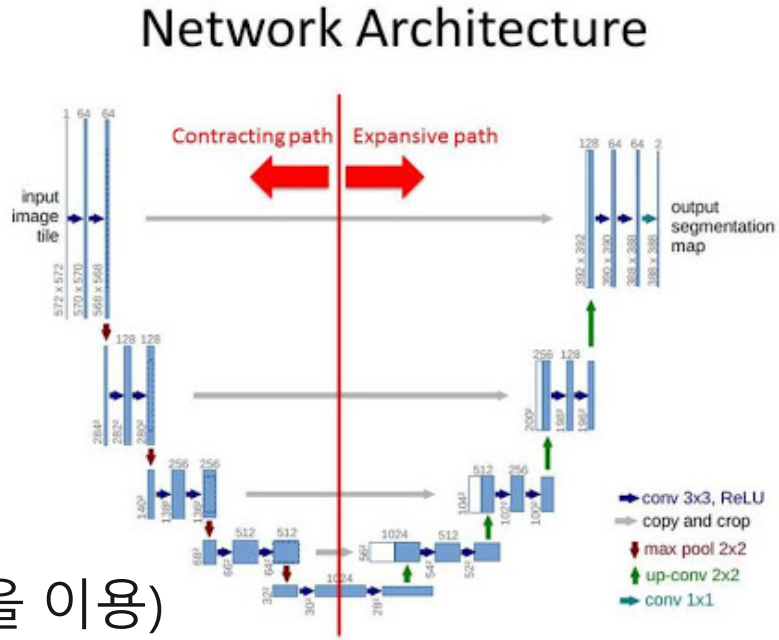


# 4 Appendix

## SegNet vs UNet

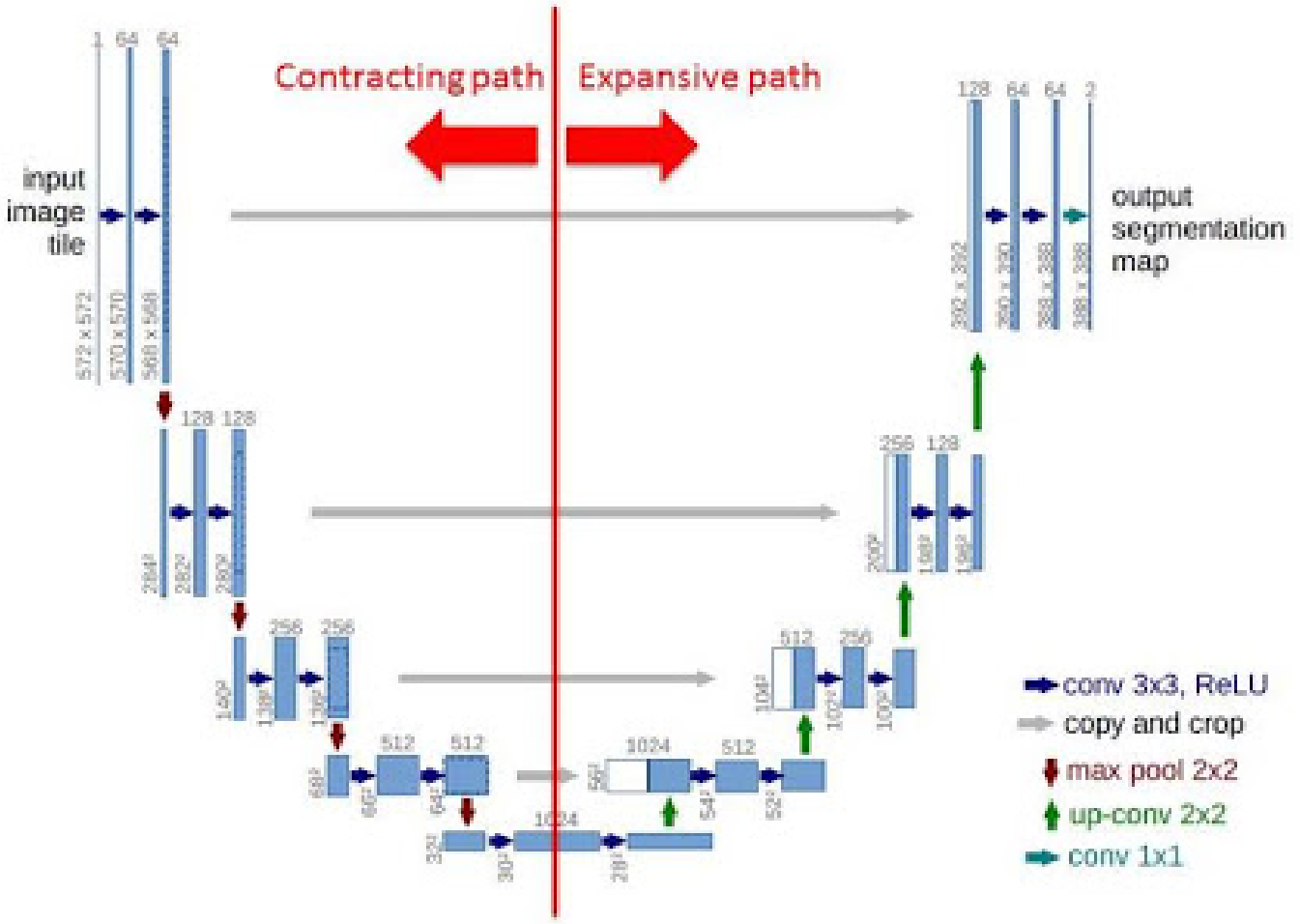
1. UNet은 Bio 이미지에서 주로 사용되는 방법




2. UNet은 Unpooling 방법을 사용하지 않음 (Convolution을 이용)

3. 기본적으로 네트워크가 더 많은 메모리를 사용

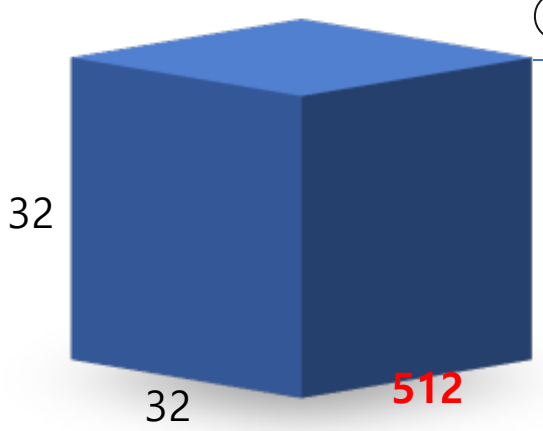
# Network Architecture



# 4 Appendix

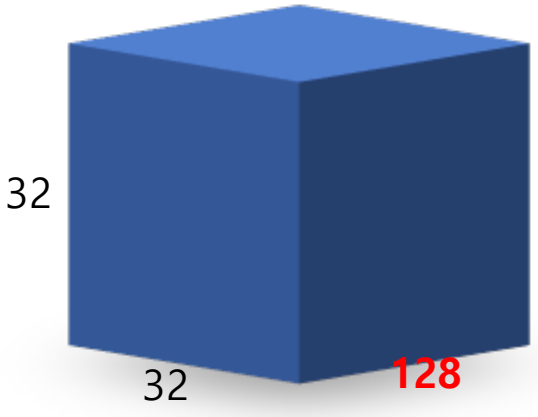
## 1x1 Convolution

① Decreasing dimensionality




nn.Conv2d(512, 128, 1, 1, stride=1, zero_padding=1)

- You can use a 1x1 convolutional layer to reduce $n_C$ but not $n_H$, $n_W$.
- You can use a pooling layer to reduce $n_H$, $n_W$, and $n_C$.



# 4 Appendix

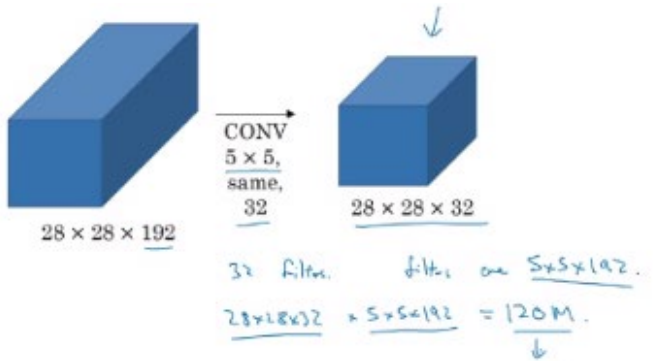
## 1x1 Convolution

② Reduce Number of Parameter -> More Faster

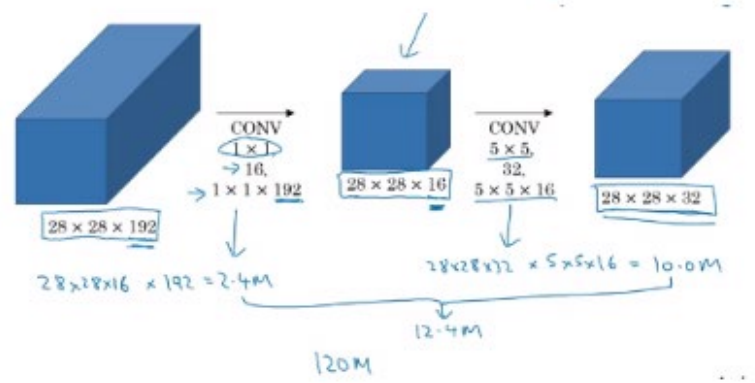
```
input (256 depth) -> 1x1 convolution (64 depth) -> 4x4 convolution (256 depth)

input (256 depth) -> 4x4 convolution (256 depth)
```

i.




ii.



# 4 Appendix

## 1x1 Convolution

③ Remember the position information