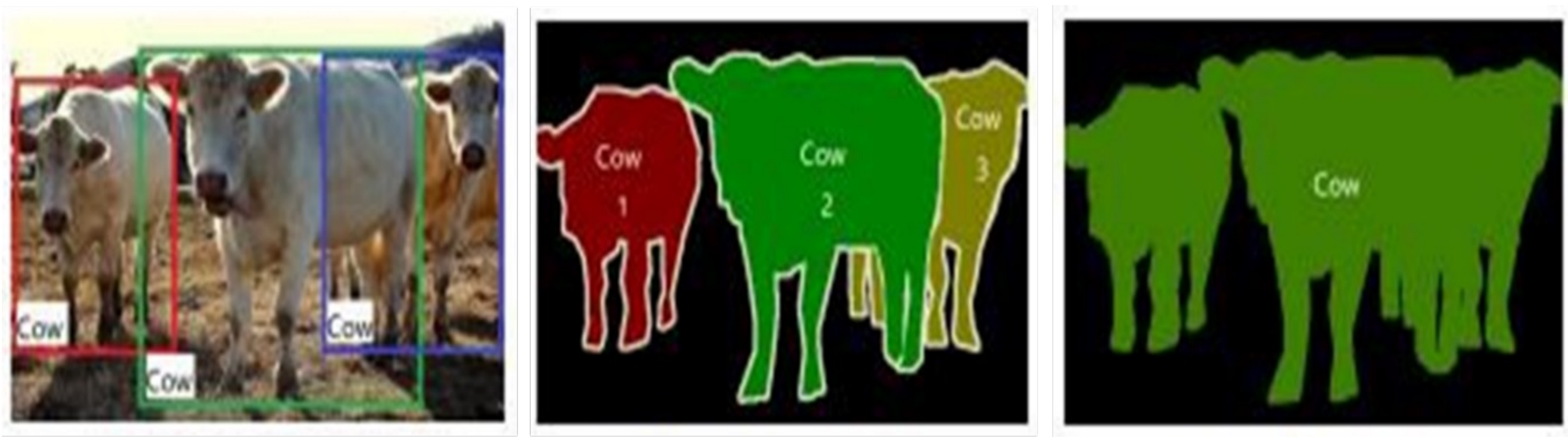
Image segmentation

Sang Yup Lee

Image segmentation

- What is it?
  - 이미지를 픽셀 단위로 구분하여 분할하는 것
  - Image segmentation with deep learning is about using a model to assign a class to each pixel in an image, thus segmenting the image into different zones (such as “background” and “foreground,” or “road,” “car,” and “sidewalk”).
- 주요 종류
  - Semantic segmentation
    - 픽셀을 class 단위로 구분
  - Instance segmentation
    - 동일 클래스의 다른 instance 도 구분 (다음 슬라이드 참고)
  - 둘을 합쳐서 Panoptic segmentation 이라고도 함

Detection vs. Segmentation



Object detection Instance segmentation Semantic segmentation

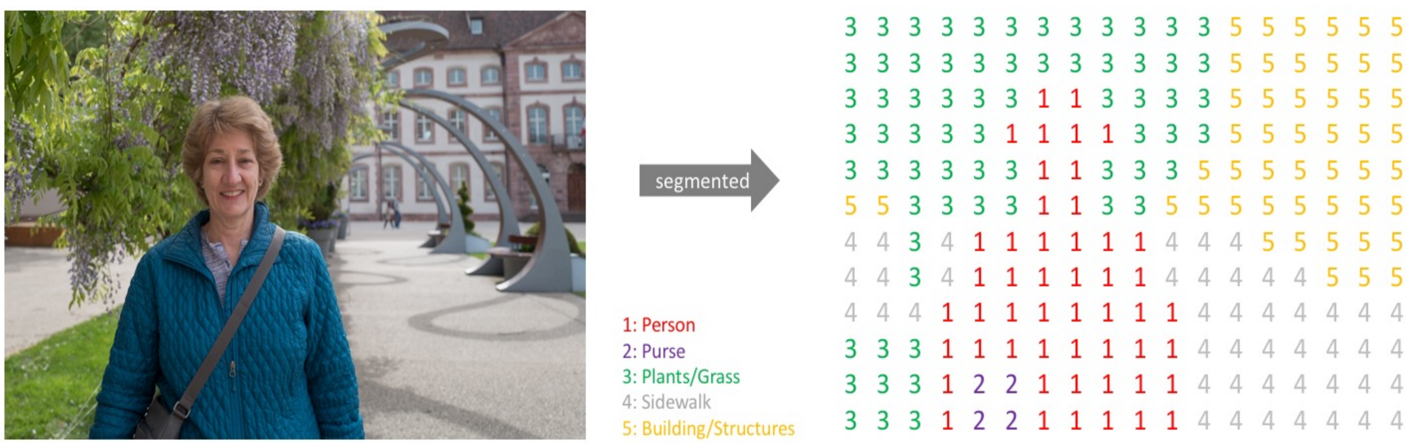
Semantic Segmentation

Semantic segmentation

- What is it?
  - Semantic segmentation is the task of assigning a class to every pixel in a given image
    - a.k.a., dense prediction
  - Note that we're not separating instances of the same class
- 주요 모형
  - CNN-based
    - FCN, U-Net, Deeplab series, InternImage
  - Transformer-based
    - TransUNet, SegFormer
  - Multi-modal approach
    - ONE-PEACE

Semantic segmentation

- The goal is to take either a RGB color image (height×width×3) or a grayscale image (height×width×1) and output a segmentation map where each pixel contains a class label represented as an integer (height×width×1).

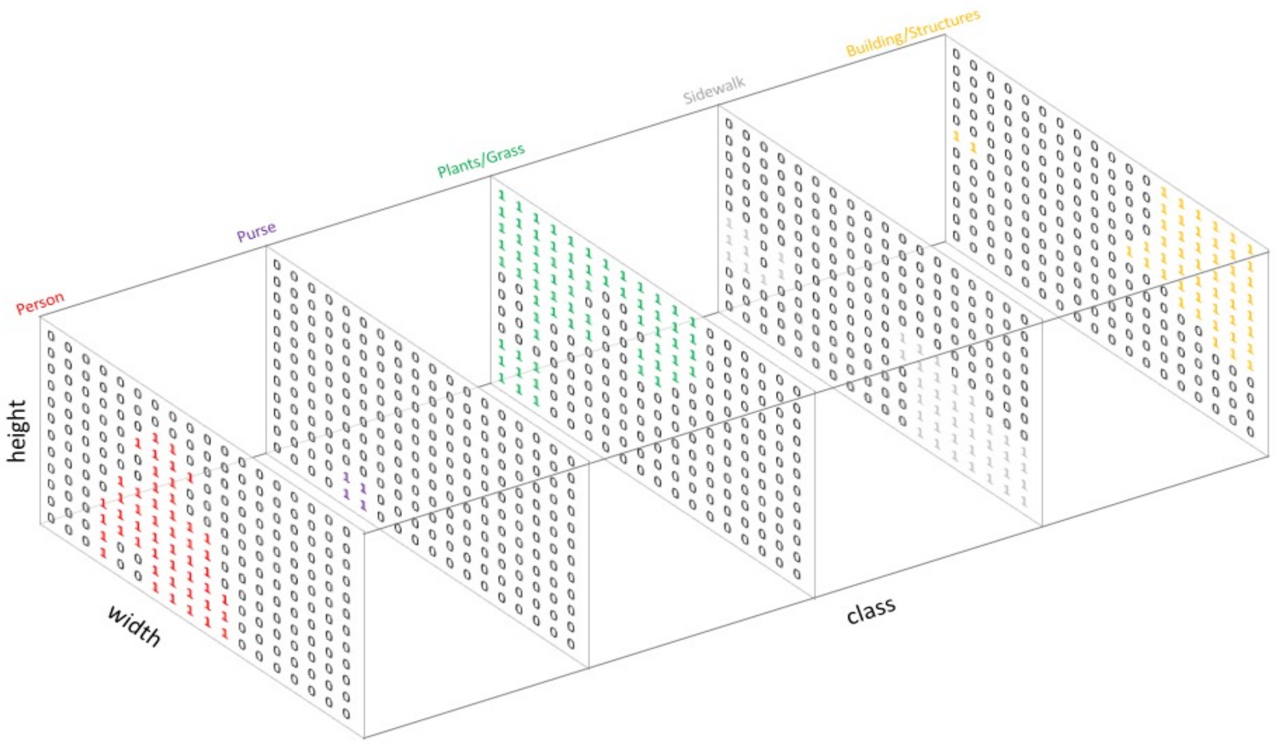


Input

Semantic Labels

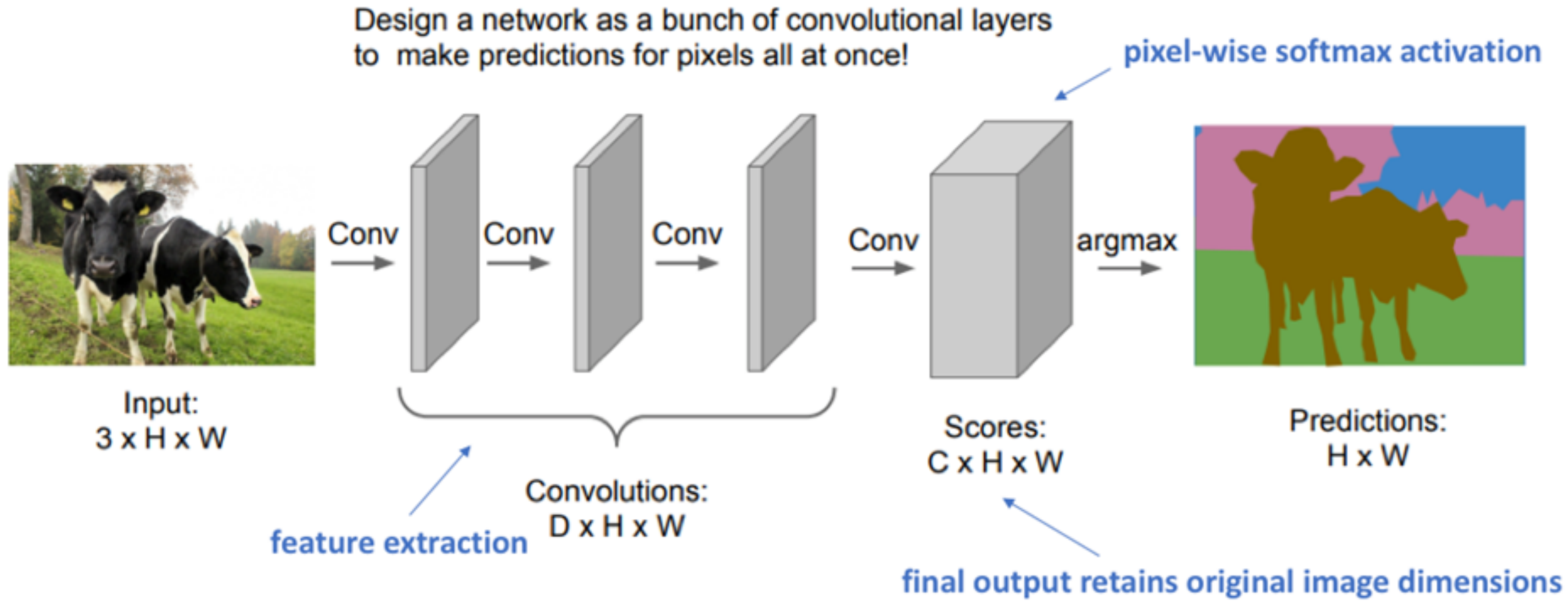
Semantic segmentation

- 정답 데이터
  - 정답 정보를 one-hot encoding 형태로 표현 가능
    - 하나의 클래스마다 하나의 채널 존재



Sematic segmentation

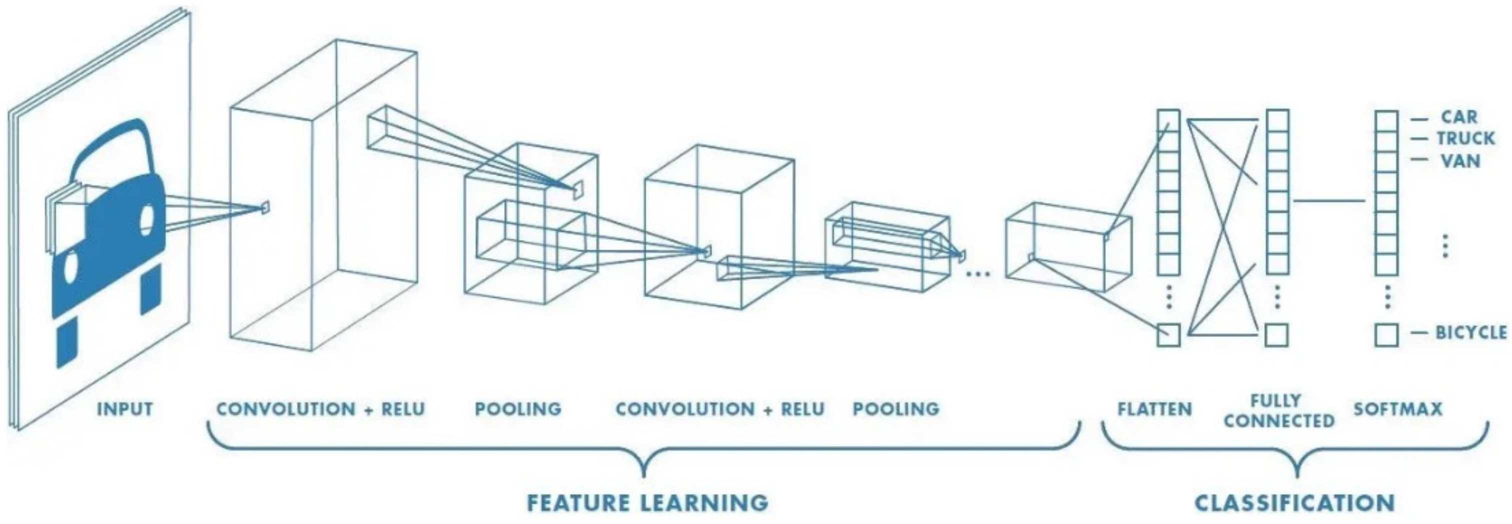
- Simple approach



  - 하지만 문제 존재

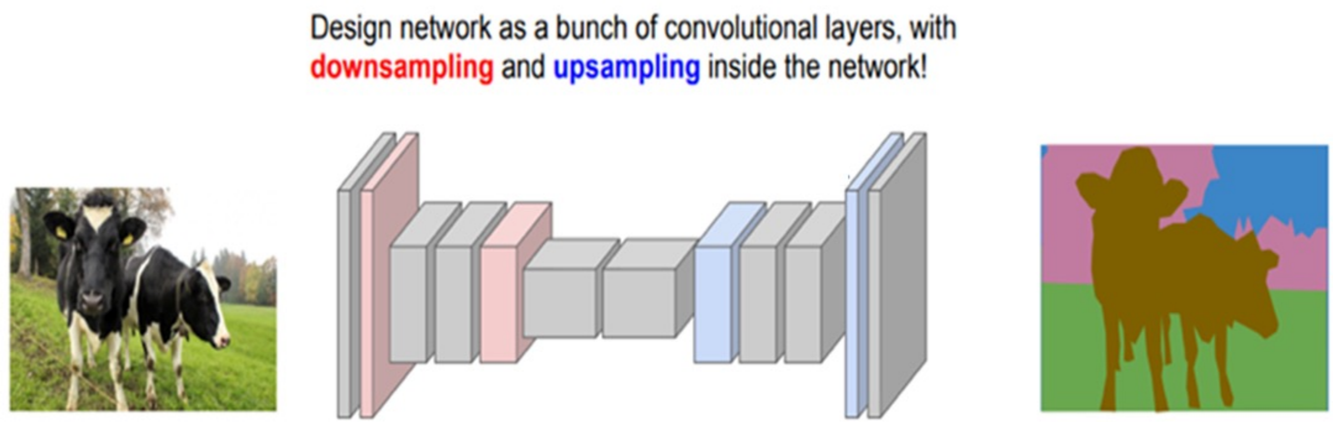
Semantic segmentation

- CNN 구조



Sematic segmentation

- Semantic segmentation model 구조
  - Encoder-decoder 구조
    - Encoder: downsampling => 클래스간 차이를 학습 (혹은 정보 추출)
    - Decoder: upsampling the feature representations into a full-resolution segmentation map

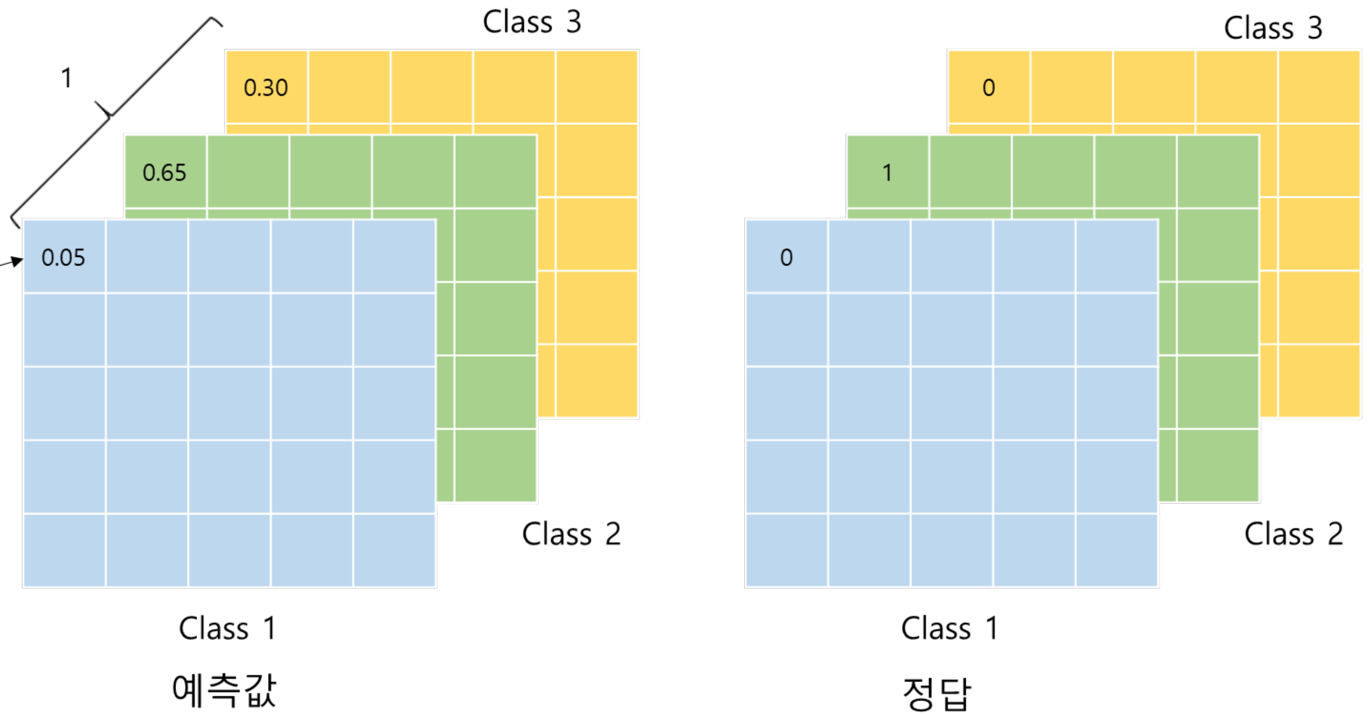


Semantic segmentation

- Loss function
  - Cross entropy

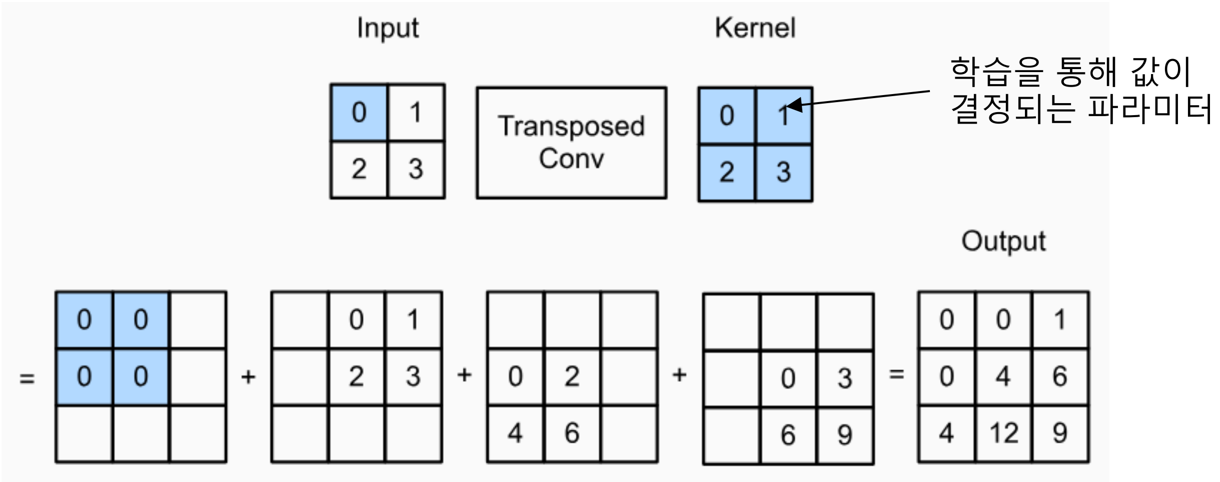
- 이미지의 첫번째 셀이 Class1에 속할 확률
- 이는 소프트맥스 함수를 이용해서 계산

- 각 채널은 각 픽셀이 특정 클래스에 속할 확률값을 지님



Semantic segmentation

- Upsampling (feature map의 크기 확대) 방법들
  - Transposed convolution (a.k.a., deconvolution)

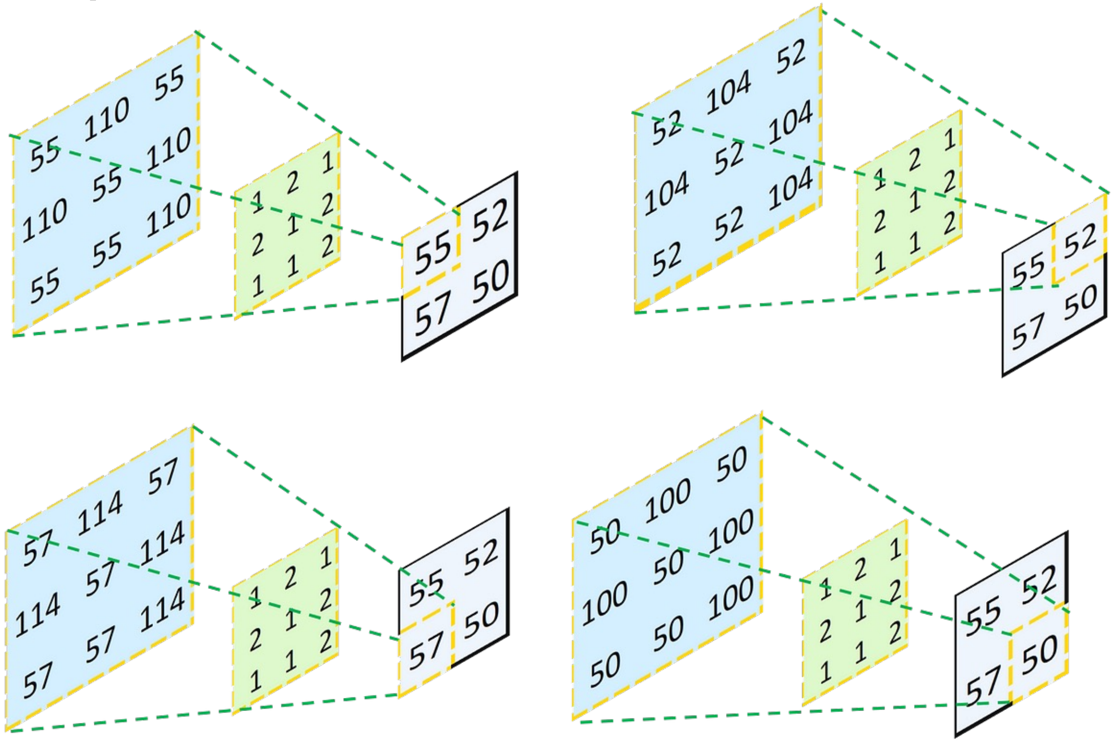


- 목표 크기: 3x3
- Stride = 1

  - 예제 코드
    - Tranposed_convolution.ipynb

Semantic segmentation

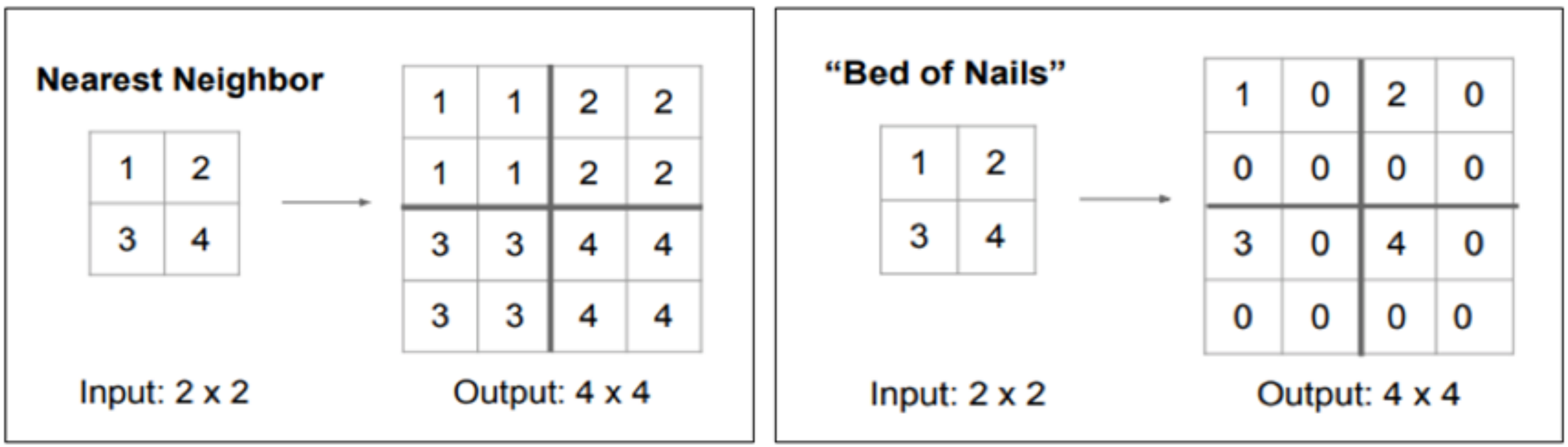
- Transposed convolution



<Source: https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>

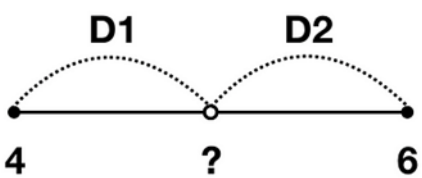
Semantic segmentation

- Upsampling 방법들
  - Upsampling

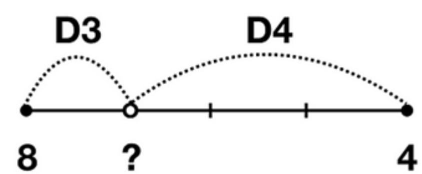


Semantic segmentation

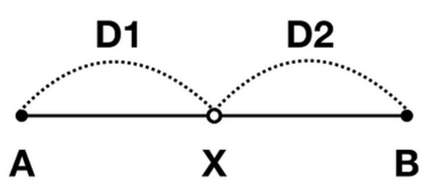
- Interpolation
  - Linear interpolation
    - 직선 위의 한 점의 값을 다른 두 값을 이용해서 계산



D1=D2



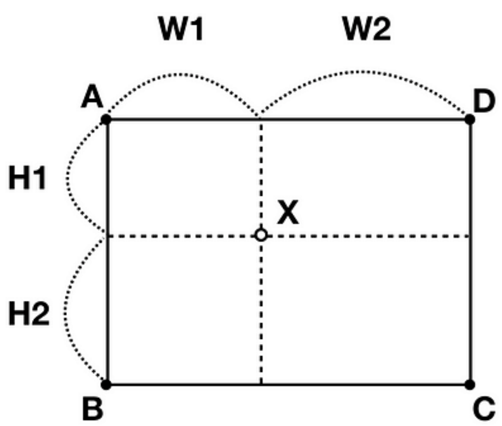
3xD3=D4



$$A\frac{D2}{D1+D2}+B\frac{D1}{D1+D2}$$

주요 개념

- Interpolation
  - Bilinear interpolation
    - 2차원에서의 interpolation



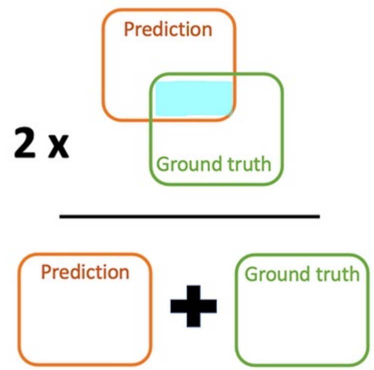
5	X		1
3			0

<이미지의 경우>

$$X = \left(A\frac{H2}{H1+H2} + B\frac{H1}{H1+H2}\right)\frac{W2}{W1+W2} + \left(D\frac{H2}{H1+H2} + C\frac{H1}{H1+H2}\right)\frac{W1}{W1+W2}$$

Semantic segmentation

- Python code
  - Semantic_seg_basic_example.ipynb
- Performance metrics
  - Pixel accuracy
    - 전체의 픽셀중에서 정답 클래스가 제대로 예측된 픽셀의 비중
    - 클래스 불균형에 취약
  - mean IoU
    - 각 클래스에 대한 IoU의 평균
  - Dice coefficient

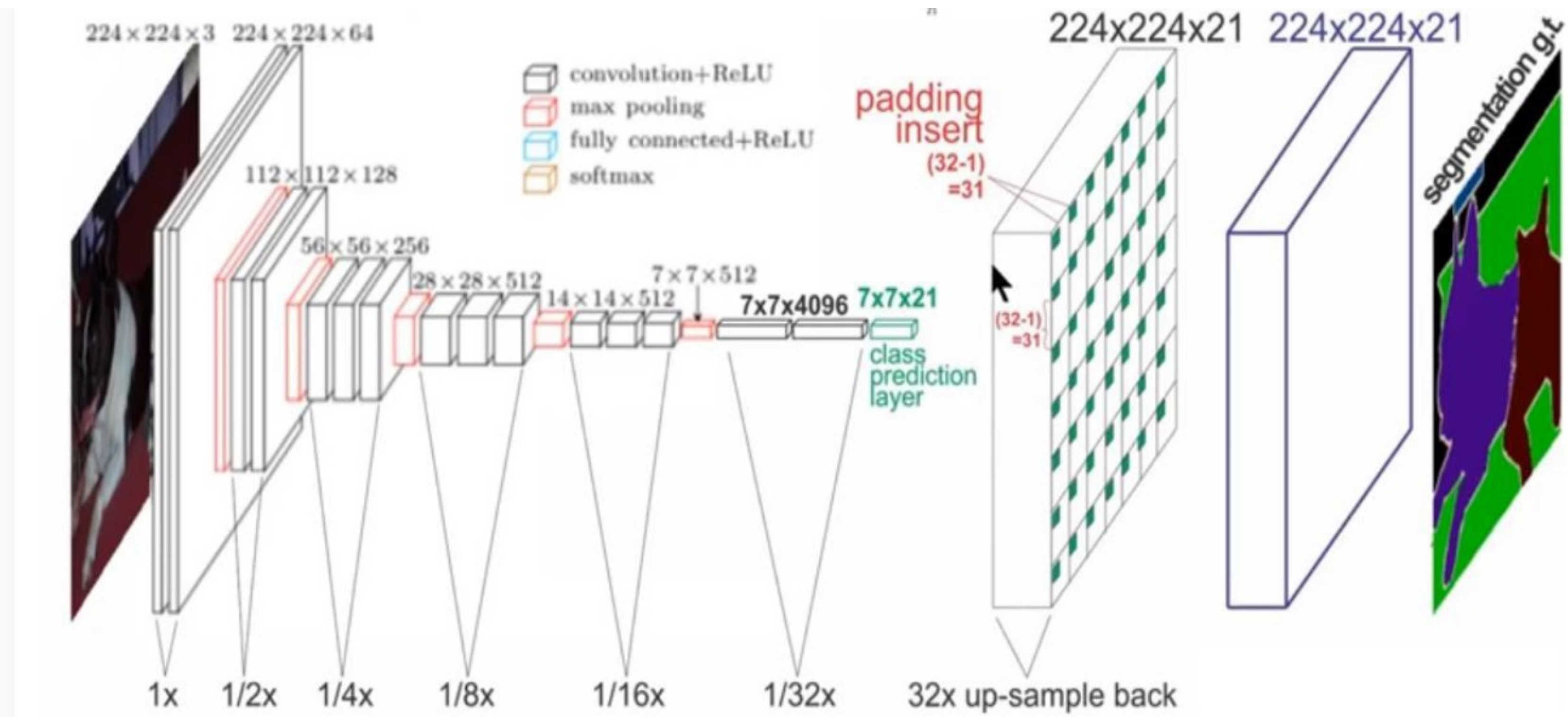


FCN (Fully Convolutional Networks)

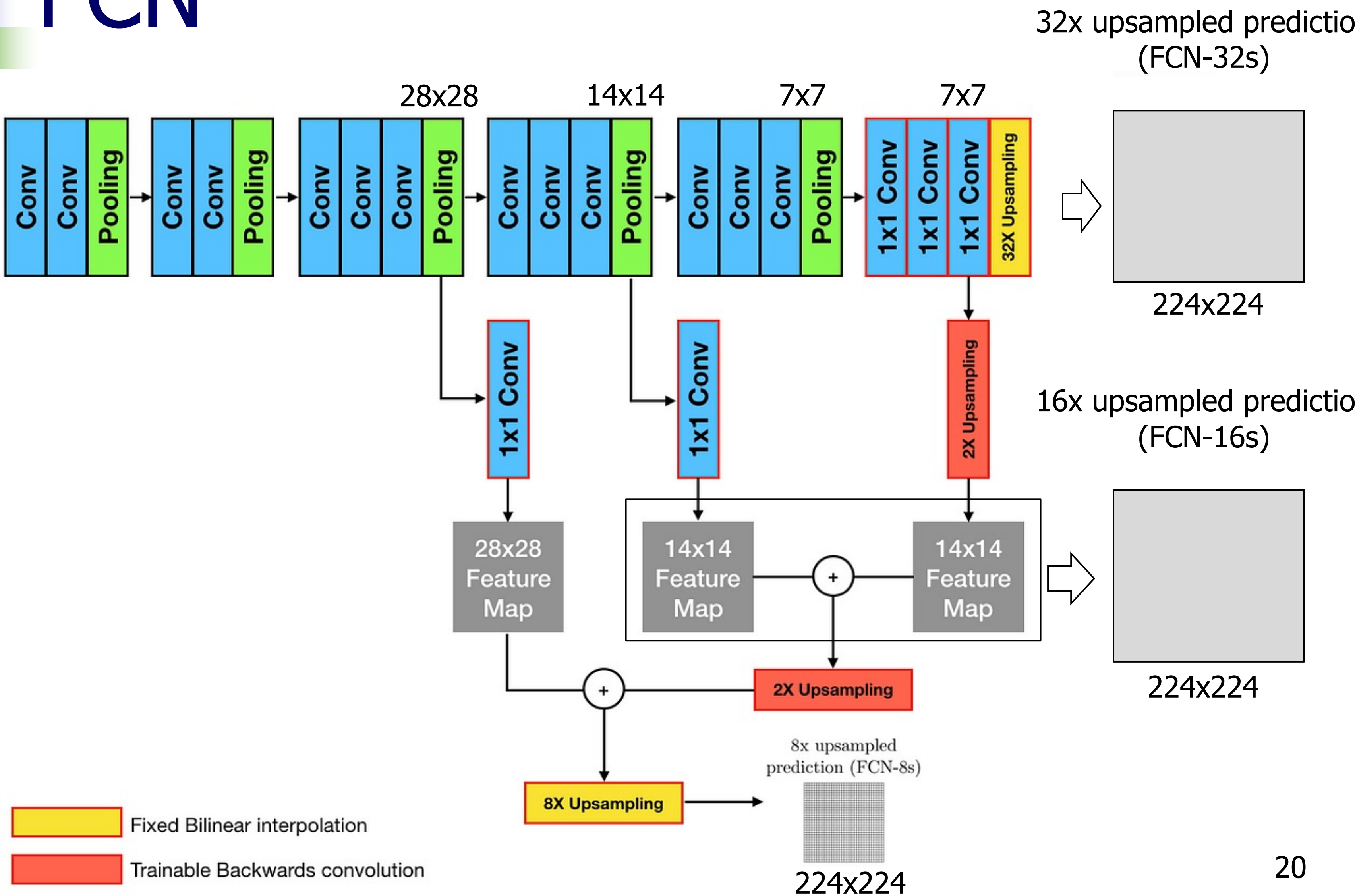
FCN

PASCAL VOC 데이터 20 + 1 (백그라운드) 클래스
원래 이미지 형태로 확대 => 그리고 픽셀별 예측
AlexNet 이용

- 모형의 구조
  - FCL를 convolutional layer로 대체



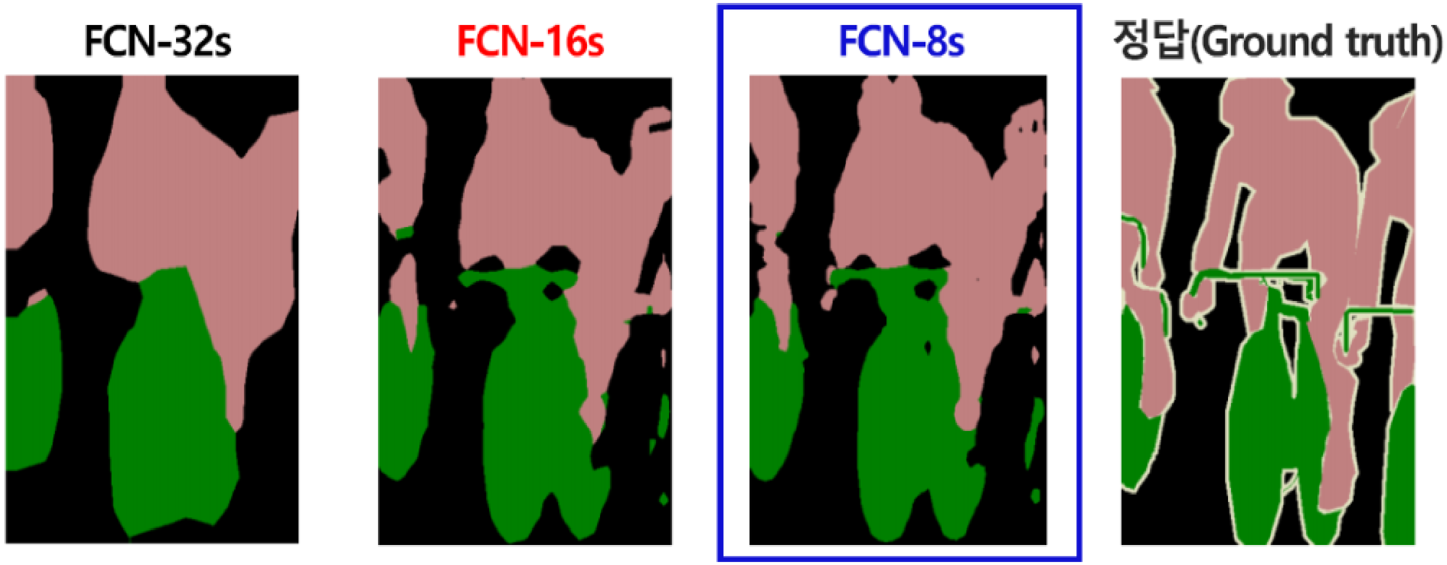
FCN



32x upsampled predictio
(FCN-32s)
28x28
14x14
7x7
7x7
Conv
Conv
Pooling
Conv
Conv
Pooling
Conv
Conv
Conv
Pooling
Conv
Conv
Conv
Pooling
Conv
Conv
Conv
Pooling
1x1 Conv
1x1 Conv
1x1 Conv
32X Upsampling
224x224
1x1 Conv
1x1 Conv
2X Upsampling
16x upsampled predictio
(FCN-16s)
28x28 Feature Map
14x14 Feature Map
+
14x14 Feature Map
224x224
+
2X Upsampling
8x upsampled prediction (FCN-8s)
8X Upsampling
224x224
Fixed Bilinear interpolation
Trainable Backwards convolution

FCN

- 모형의 성능

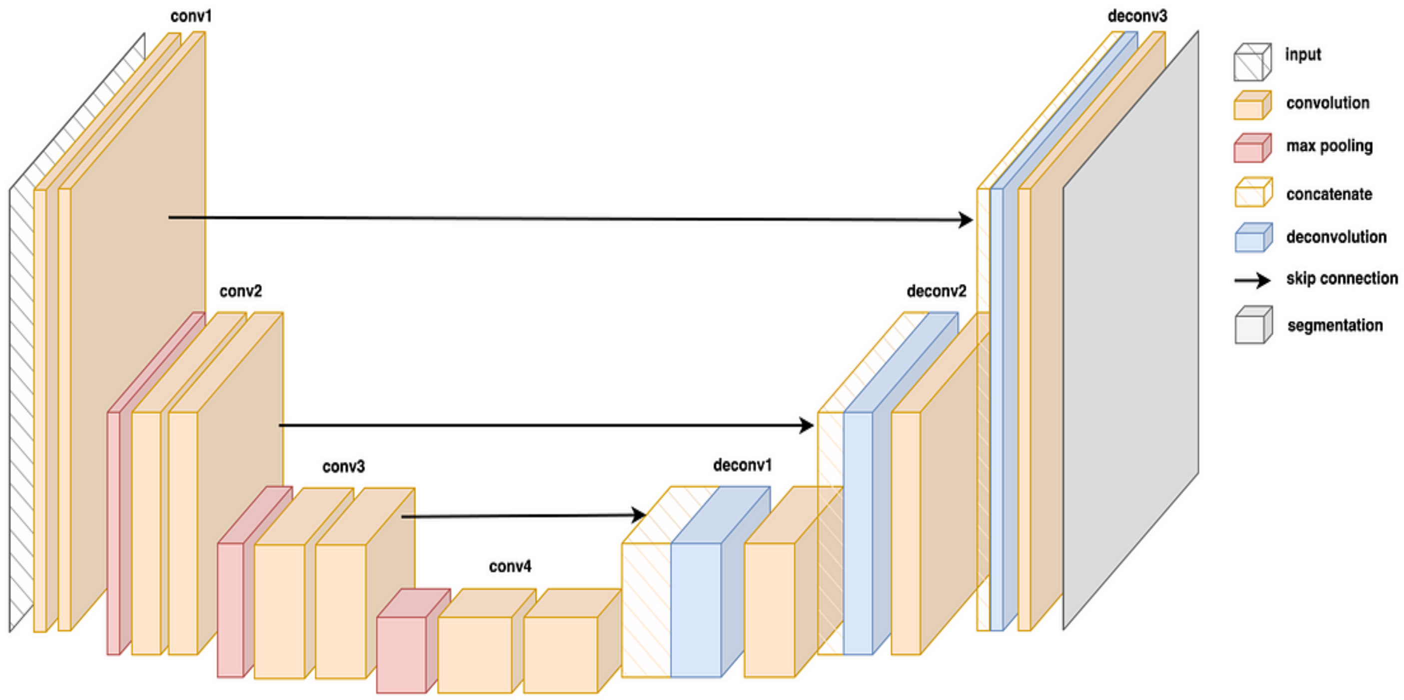


	FCN-32s	FCN-16s	FCN-8s
IoU (Intersection over Union)	59.4(%)	62.4(%)	62.7(%)

U-Net

U-Net

- 구조


conv1
conv2
conv3
conv4
deconv1
deconv2
deconv3
input
convolution
max pooling
concatenate
deconvolution
skip connection
segmentation

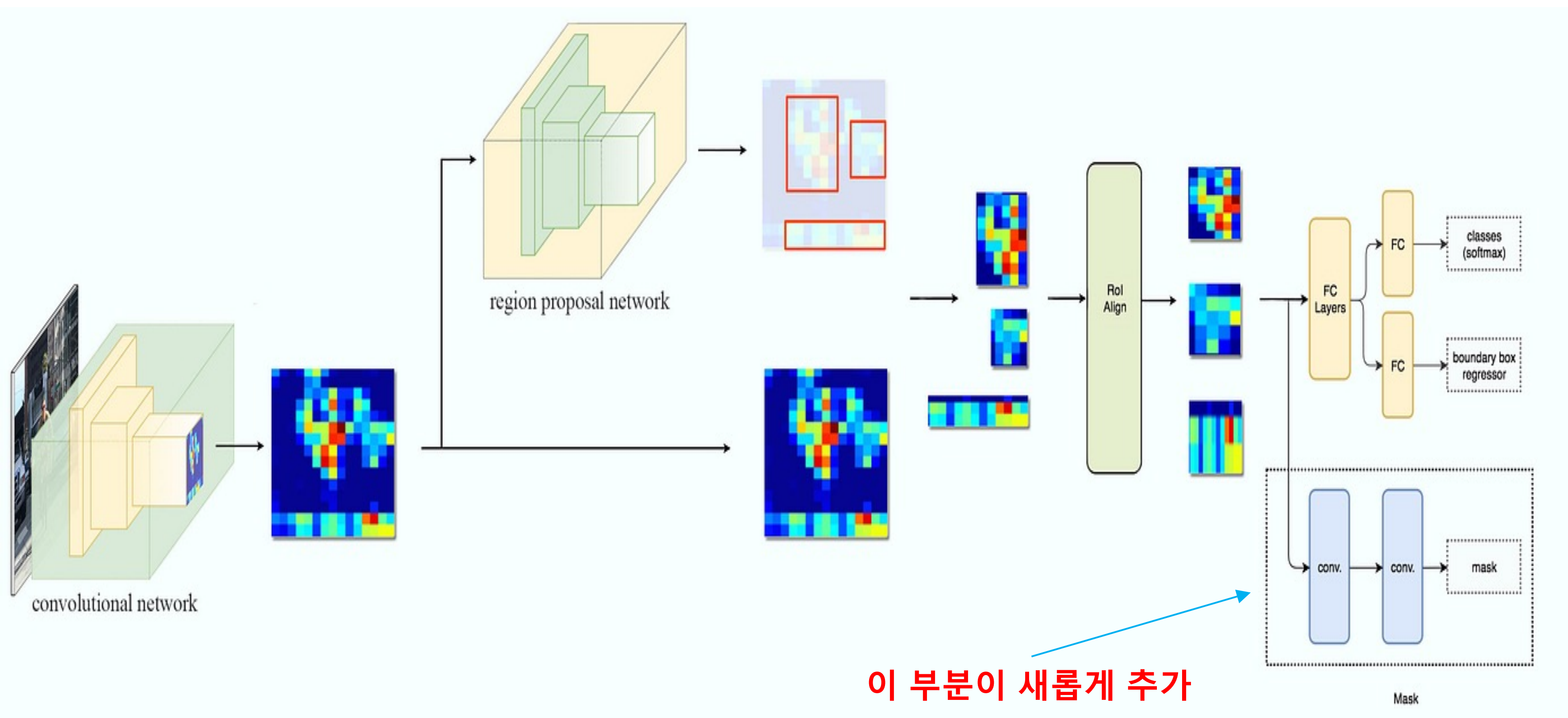
U-Net

- Python code
  - UNet_Keras.ipynb

Instance Segmentation

Mask R-CNN

- 모형의 구조
  - Faster RCNN + FCN



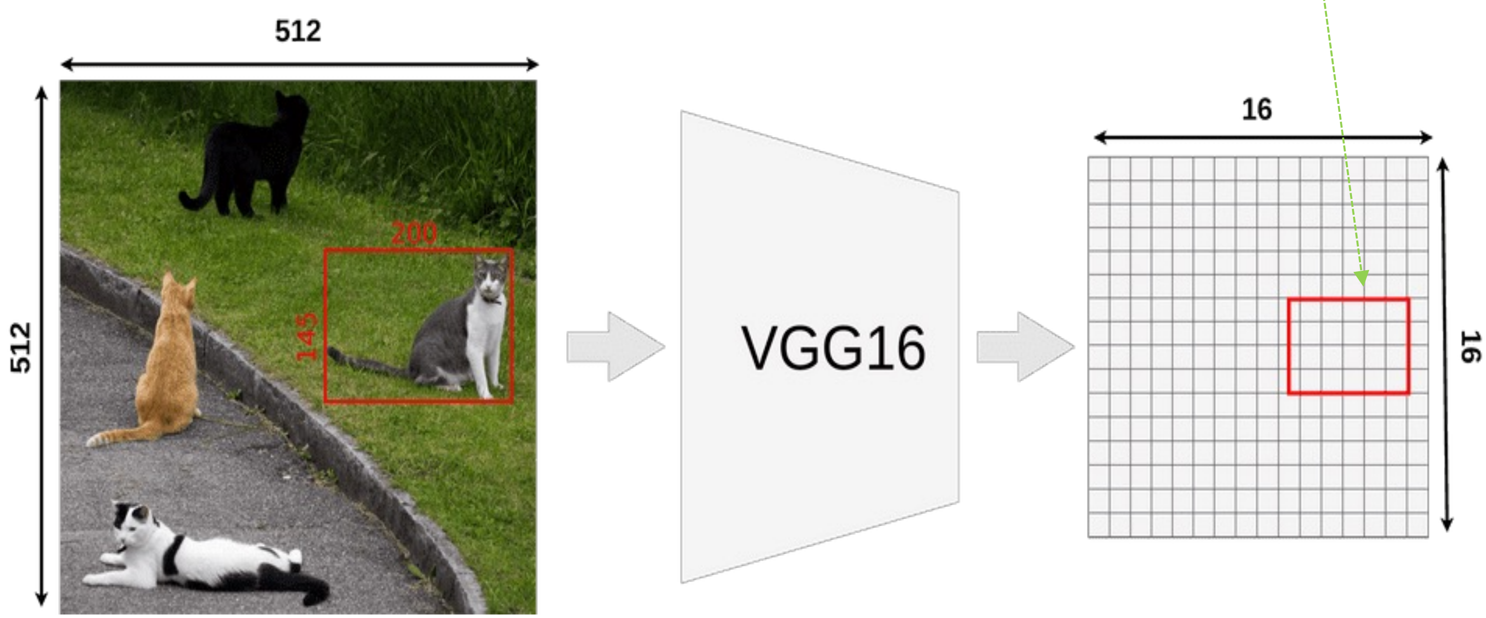
Mask R-CNN

- 비용함수
  - $L = L_{cls} + L_{bbox} + L_{mask}$

Mask R-CNN

- RoI Align

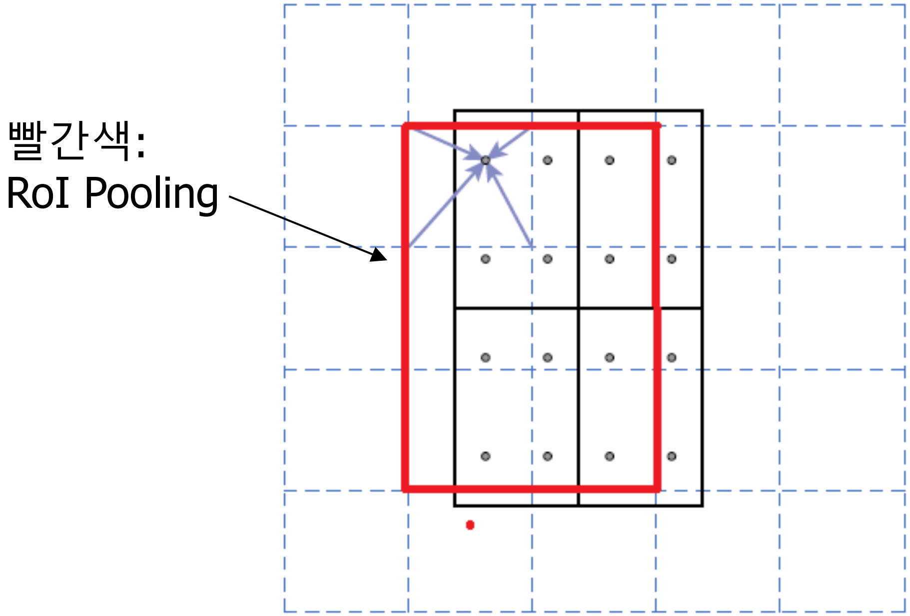
RoI: RPN을 이용해서 출력



- Feature map에 RoI를 매핑한 이후 고정된 크기의 feature map (혹은 벡터)를 추출해야 한다.
- 사용되는 방법: RoI Pooling, RoI Align

Mask R-CNN

- RoI Align
  - RoI pooling의 경우, 정보 손실 발생



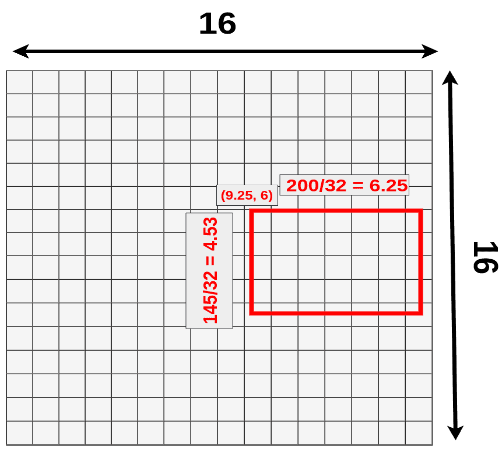
RoIPool의 경우 feature map에 맞추기 위해 반올림 (예를 들어, 아래 그림의 빨간 색과 같이 수행)

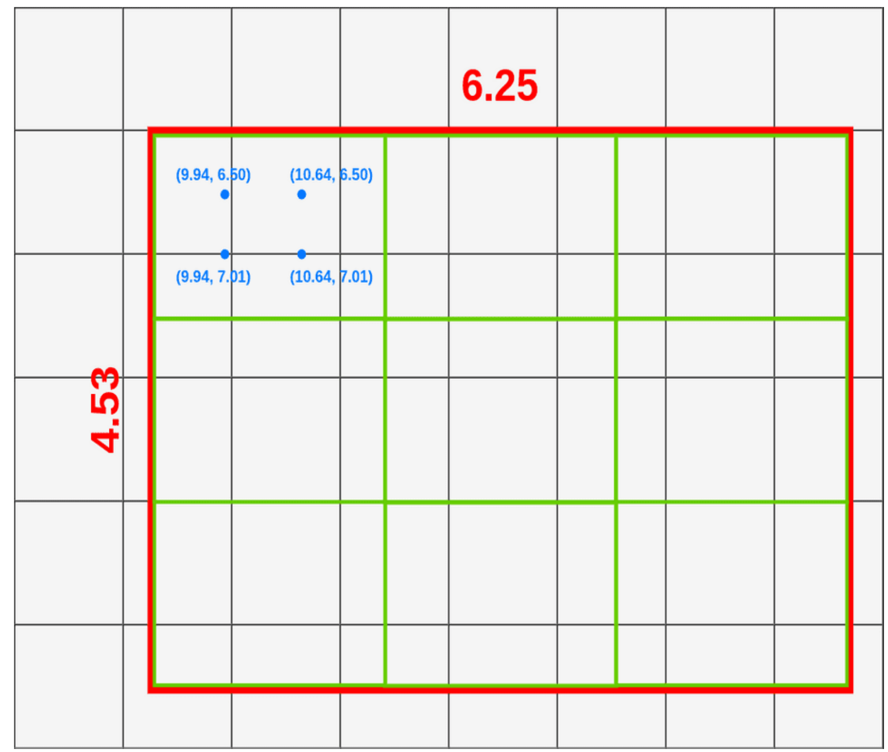
그리고 그 다음 특정한 크기의 feature map (위와는 다른 feature map임)을 추출하기 위해 RoI를 동일한 비중으로 분할하지 못함

Mask R-CNN

- RoI Align

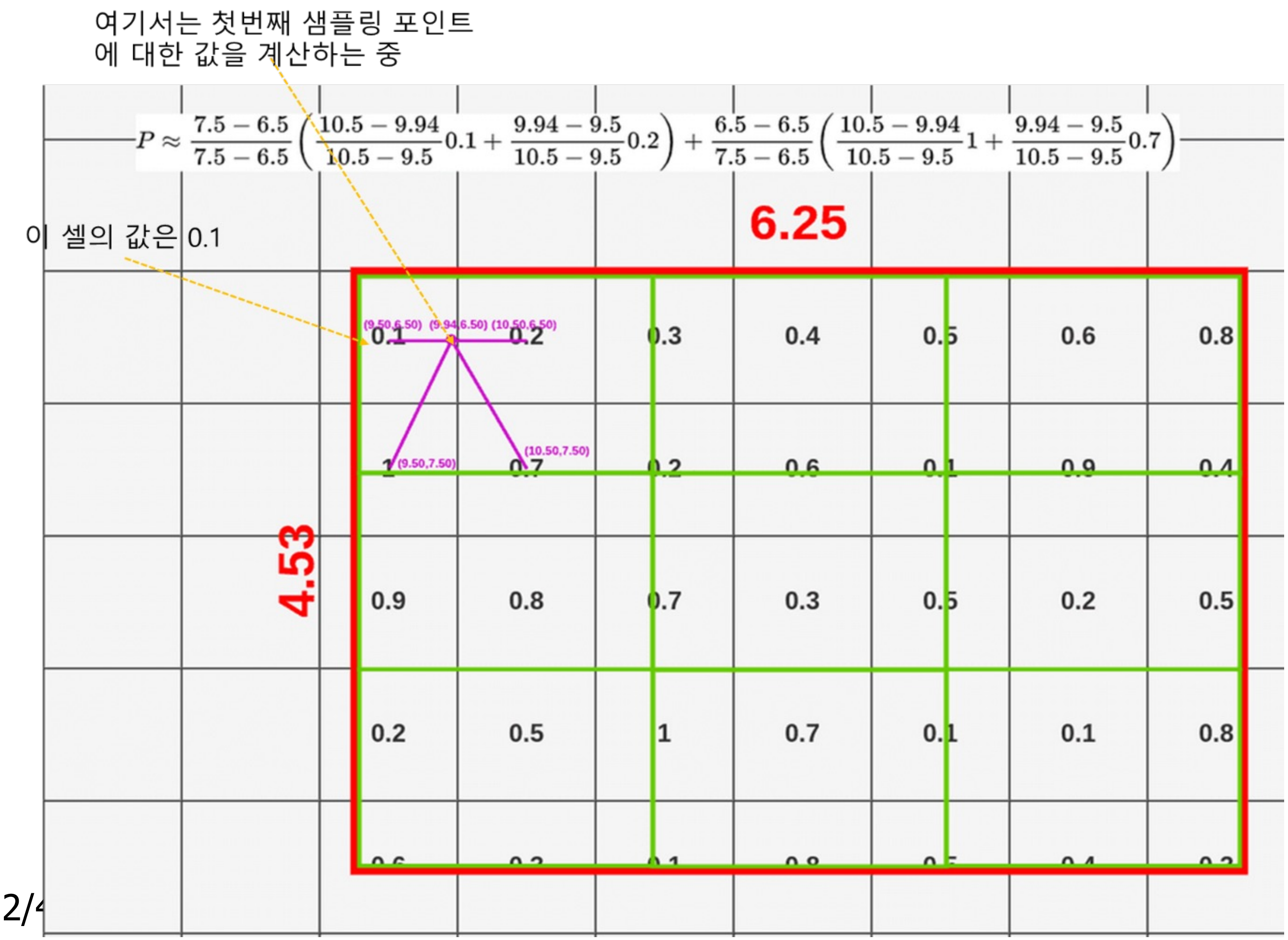
- 4개의 포인트 지정
- 각 포인트 값을 bilinear interpolation 방법을 사용해서 계산 ⇒ 인접한 4개의 셀의 값을 이용해서 bilinear 보간법 사용





3x3의 결과를
얻고자 하는 경우

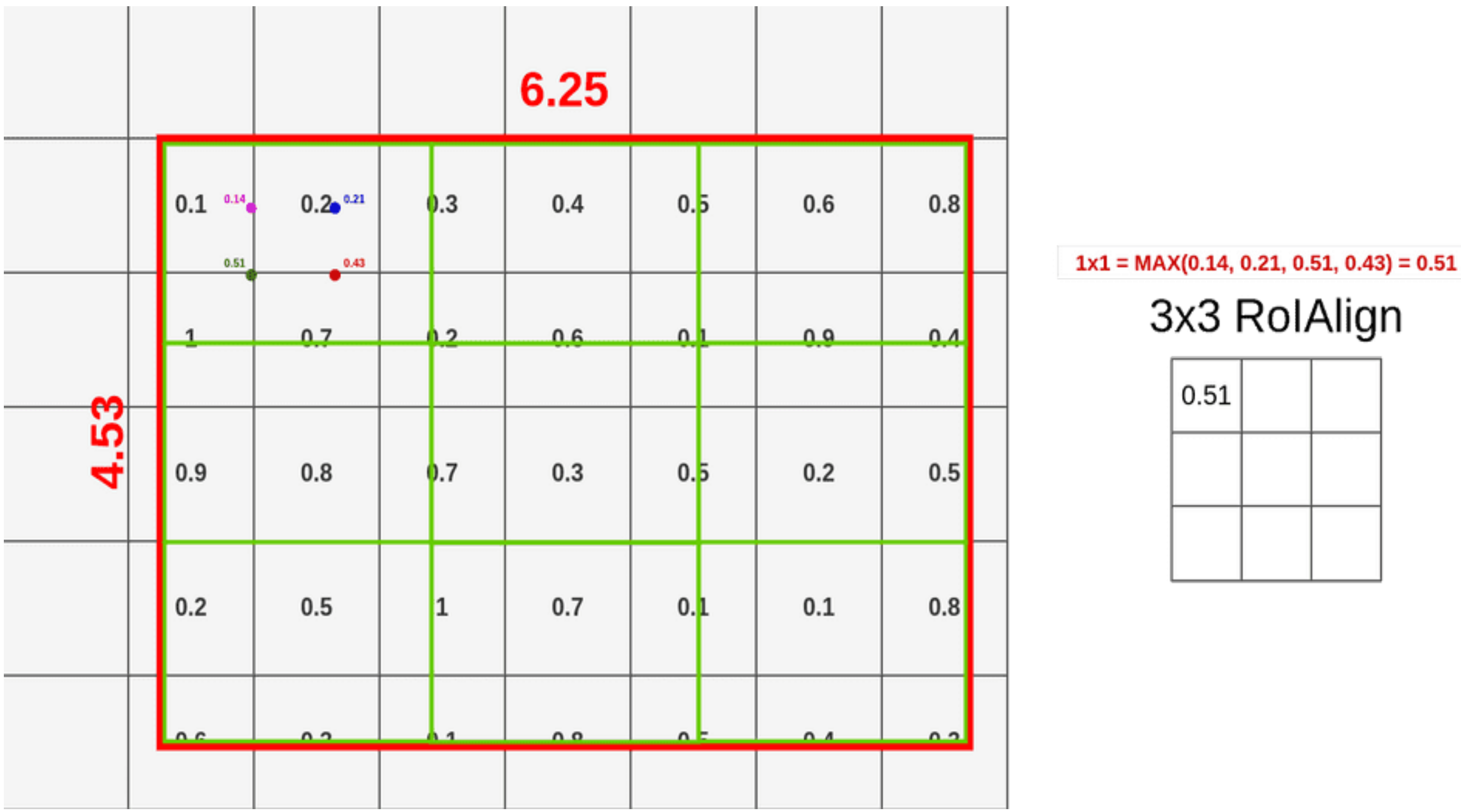
Mask R-CNN



여기서는 첫번째 샘플링 포인트
에 대한 값을 계산하는 중
$P \approx \frac{7.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}0.1+\frac{9.94-9.5}{10.5-9.5}0.2\right)+\frac{6.5-6.5}{7.5-6.5}\left(\frac{10.5-9.94}{10.5-9.5}1+\frac{9.94-9.5}{10.5-9.5}0.7\right)$
이 셀의 값은 0.1
6.25
4.53
(9.50,6.50) (9.94,6.50) (10.50,6.50)
(9.50,7.50)
(10.50,7.50)
0.1 0.2 0.3 0.4 0.5 0.6 0.8
1 0.7 0.2 0.6 0.1 0.9 0.4
0.9 0.8 0.7 0.3 0.5 0.2 0.5
0.2 0.5 1 0.7 0.1 0.1 0.8

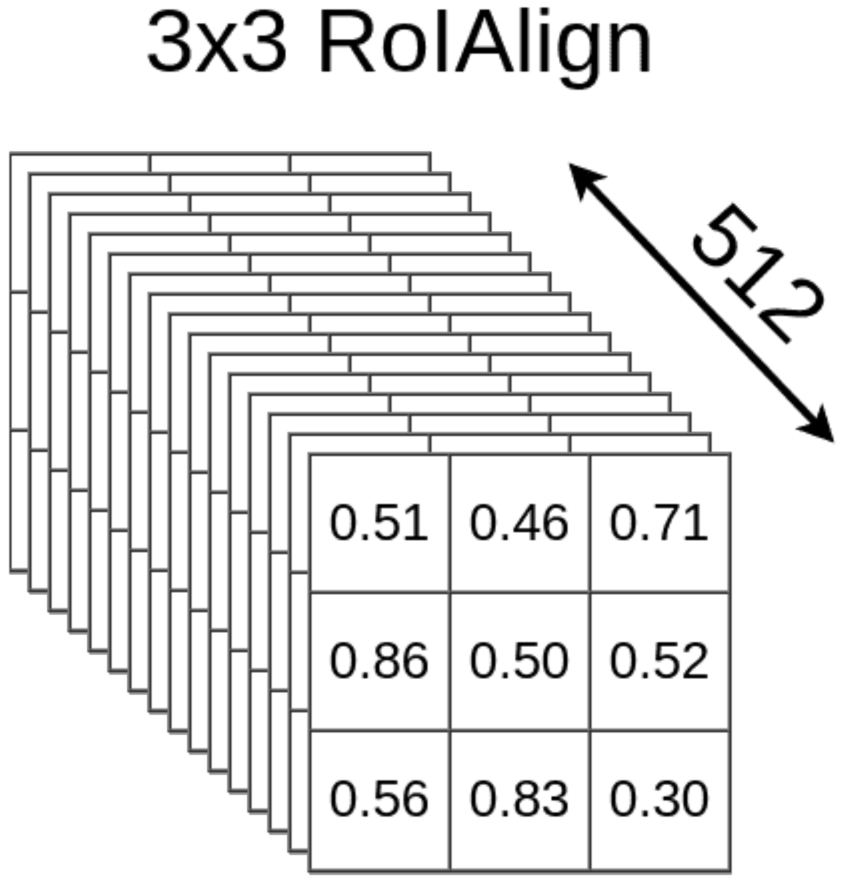
Mask R-CNN

- RoI Align



Mask R-CNN

- RoI Align
  - 이를 모든 레이어에 대해 수행



Q & A