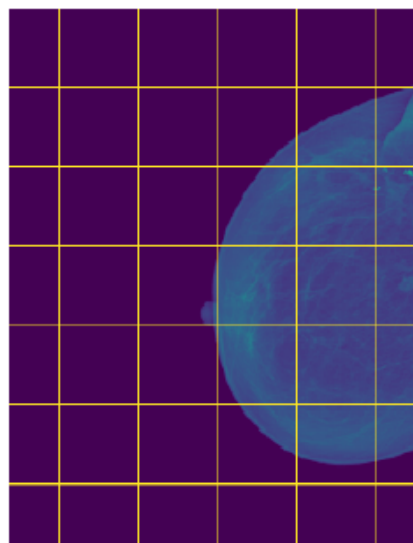


# U-Net

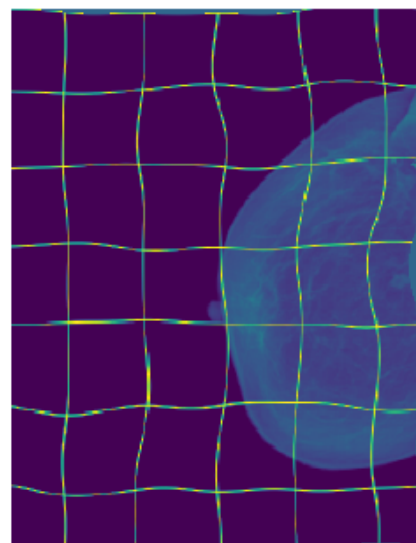
: Convolutional Networks for Biomedical Image Segmentation



## Augmentation - Elastic Deformation

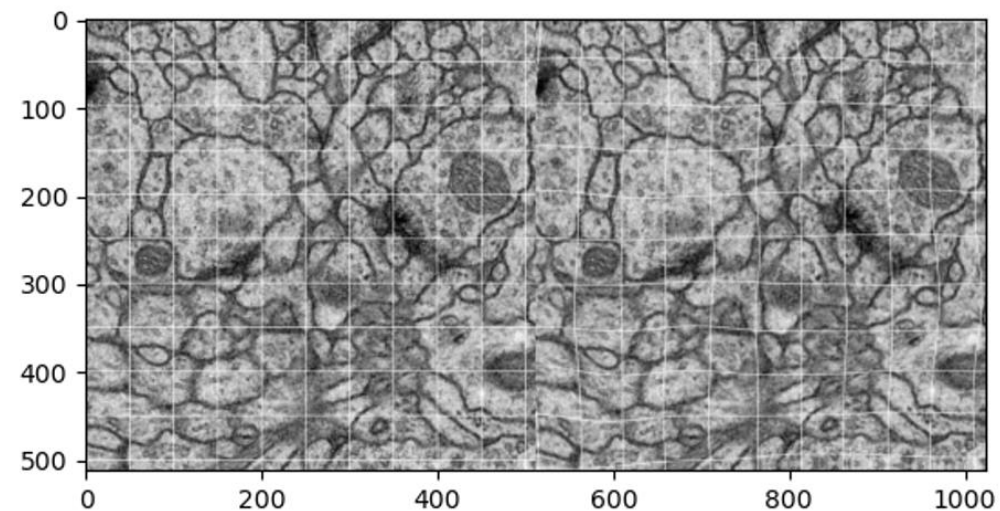


(a) Original



(b) Deformed

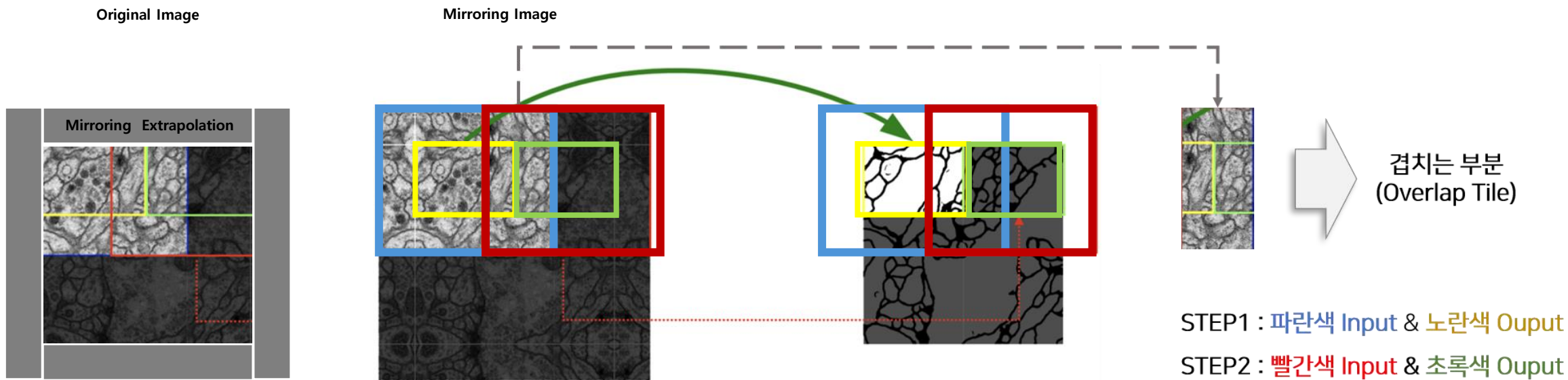
각 Pixel이 랜덤하게 다른 방향으로 뒤틀리도록 변형하는 방식



의학 데이터의 경우 세포들의 움직임에 따라 사진이 다르기 때문에 **Elastic Deformation** 사용하기 적합

→ U-Net은 의학 데이터 분석하기 위해 만들어진 모델이기에 **Elastic Deformation** Augmentation 적합

## Unet – Overlap Tile Strategy



- 우선 이미지가 큰 경우 이미지를 자른 후 각 이미지에 해당하는 Segmentation 진행
- Original Image Mirroring Extrapolation
- Mirroring 적용한 Image에서 파란색 영역을 Input으로 넣으면 노란색 영역이 Output으로 추출
- 동일하게 초록색 영역을 Segmentation 하기 위해서는 빨간색 영역을 모델의 Input으로 사용

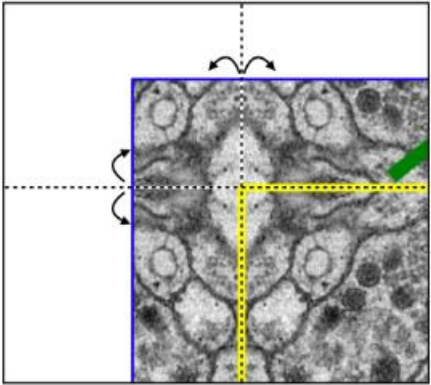


→ Padding을 하지 않으니 mirroring으로 만들어진 부분이 conv 연산을 거치면서 (missing data 발생) feature map에 덜 반영되는 건 괜찮지만 Original Image가 반영되지 않으면 안됨.

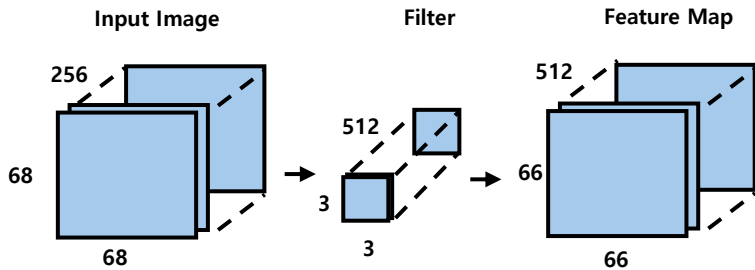
겹치는 부분이 있어 결국 모든 Original Image는 반영됨.

# Unet - Contracting Path

## Mirroring Extrapolation & Overlap-tile strategy 같이 적용

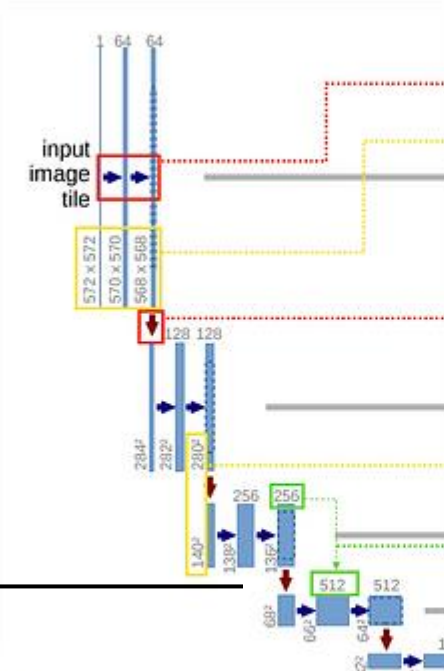


- Missing data 해결
- 파란색 박스의 빈 공간을 노란색 영역이 거울에 반사된 형태로 채우는 방식
- 테두리를 좌우대칭을 하는 것이 zero padding보다 실제 삭제되지 않은 모습에 더 가까울 것이다.



Down, Up Sampling 진행 시 채널의 수가 2배로 증가하고 감소하는 이유  
-> Conv 연산에서 filter의 채널 수에 따라 feature map의 채널 수 변경

- ➡ conv 3x3, ReLU
- ➡ copy and crop
- ⬇ max pool 2x2
- ⬆ up-conv 2x2
- ➡ conv 1x1



## Contracting Path

각 Contracting Step 마다  
3x3 convolution을 두 차례씩 반복  
(단, 패딩이 없음으로 Feature Map이 조금씩 줄어듦)

Convolution에는 ReLU 연산이 포함됨

각 Contracting Step 마다  
2x2 max-pooling (stride: 2) 연산을 수행함  
이 때, Feature map의 크기가 절반으로 줄어듦

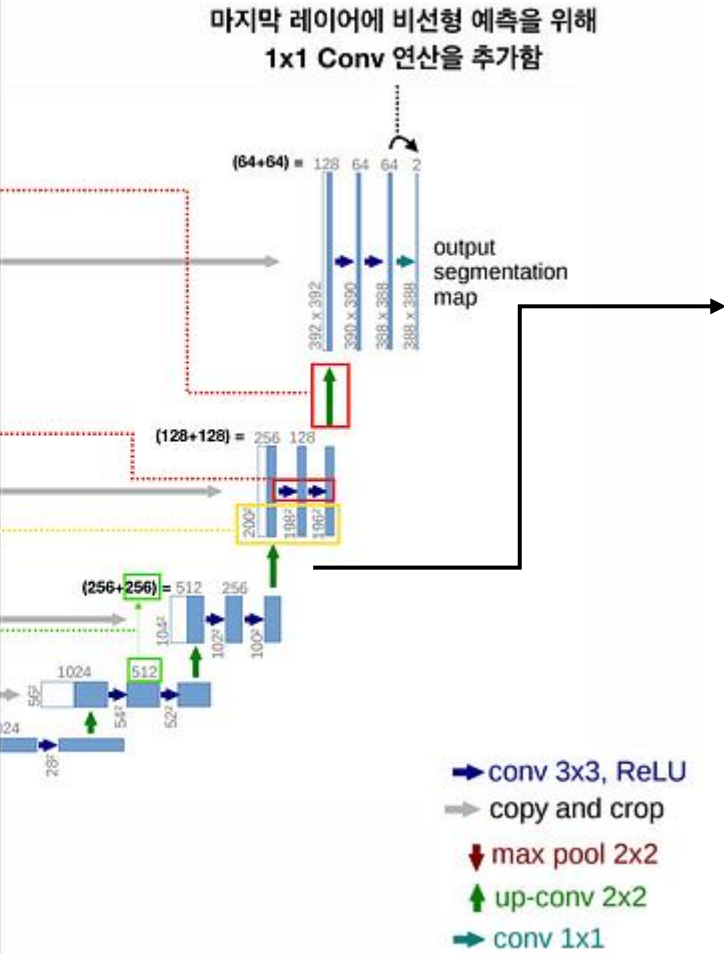
Down-sampling 마다 채널의 수가 2배로 늘어남

# Expanding Path

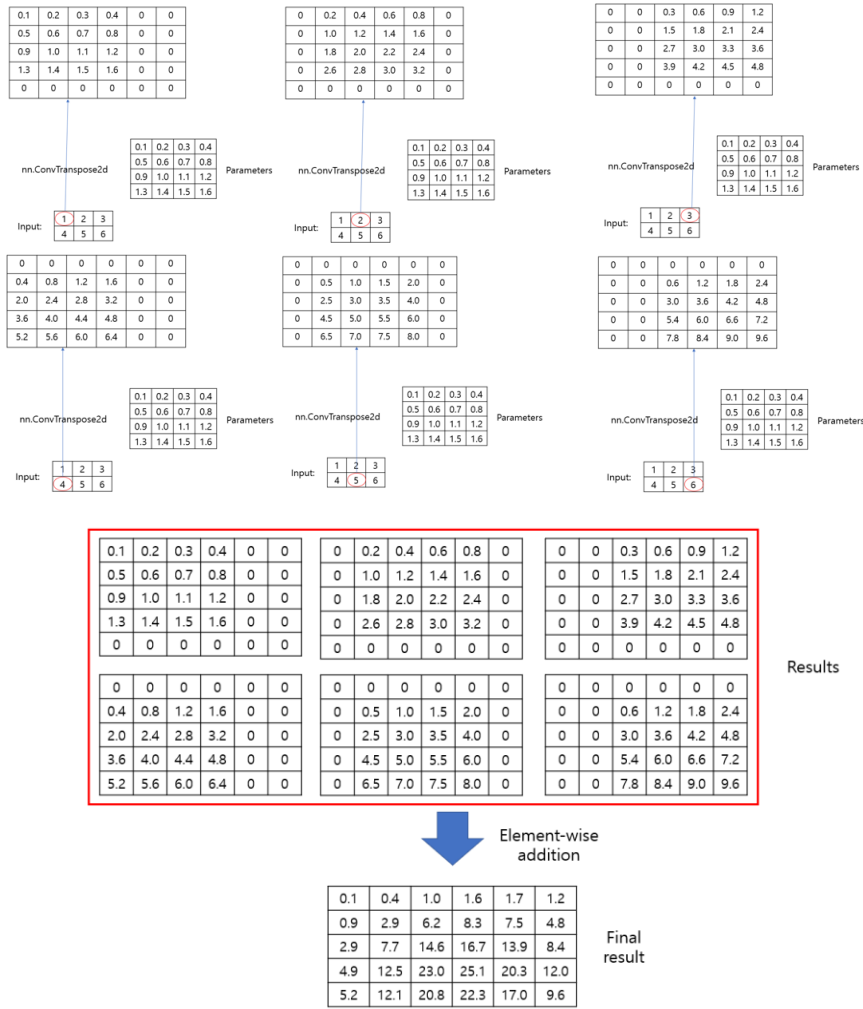
각 Expanding Step마다  
2x2 Up-convolution을 수행함  
이 때, Feature map의 크기가 두 배로 늘어남

각 Expanding Step마다  
3x3 convolution을 두 차례씩 반복 (ReLU 포함)  
(단, 패딩이 없으므로 Feature Map이 조금씩 줄어듬)

Up-sampling마다 채널의 수가 절반으로 줄어듬

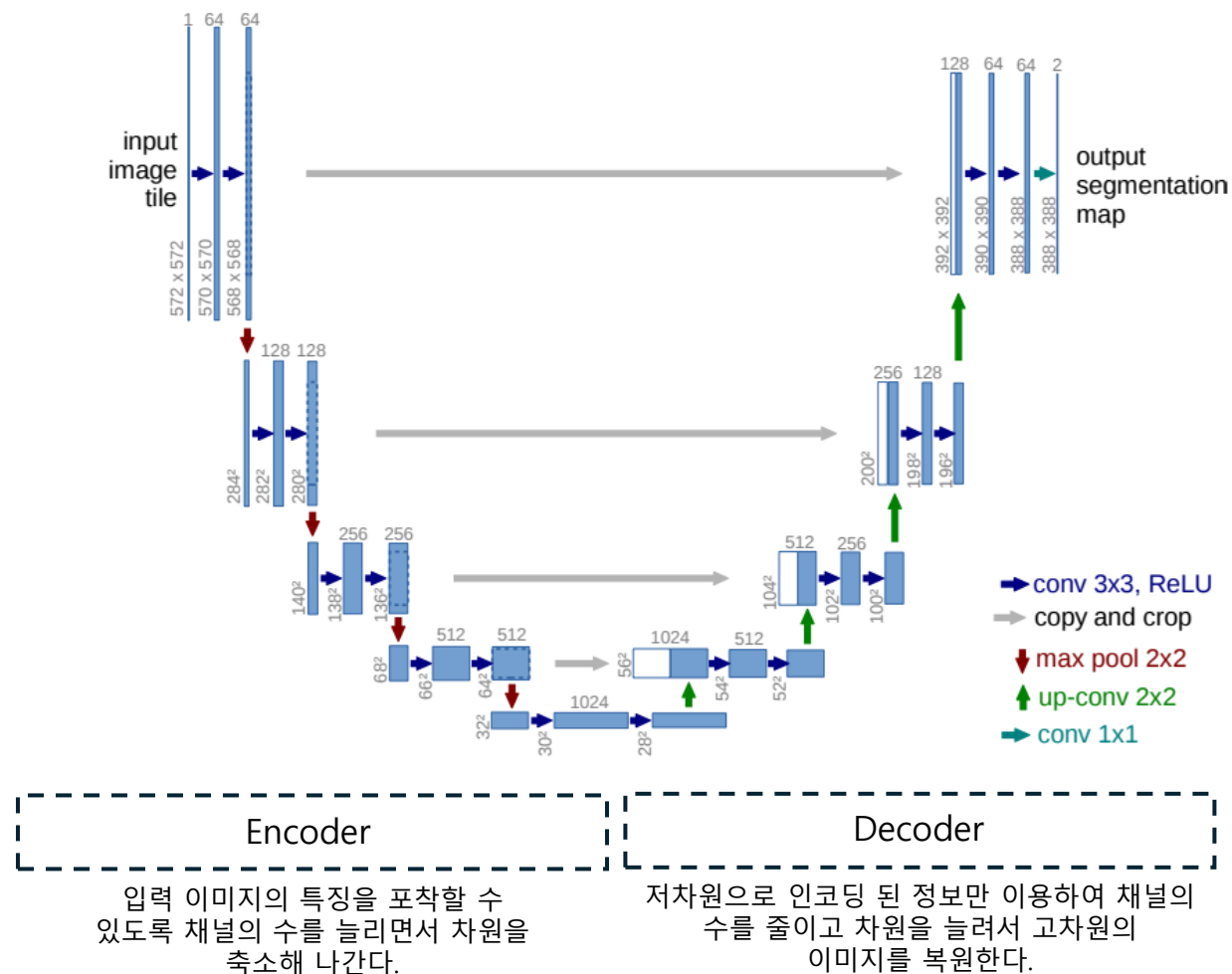


## Transpose Convolution



- 위 그림과 같은 방법으로 up sampling 진행함.
- 근데 U-Net은 2x2 Transpose convolution이고 stride = 2 이기에
- 겹치지 않으니 최종적으로 더하는 부분은 없다.

## Unet - Skip Connection



인코딩 단계에서 차원 축소를 거치면서 이미지 객체에 대한 자세한 위치 정보를 잃고, 디코딩 단계에서도 저차원의 정보만을 이용하기 때문에 위치 정보 손실을 회복하지 못하게 된다.



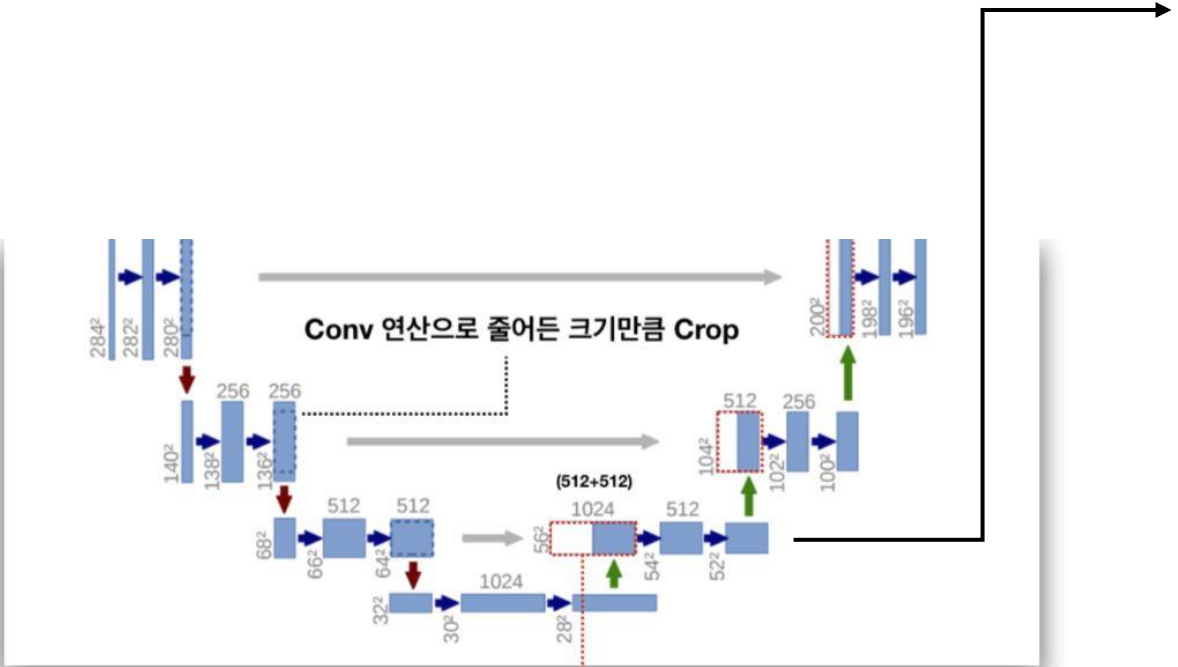
저차원 뿐만 아니라 **고차원 정보도 이용하여** 이미지의 특징을 추출함과 동시에 **정확한 위치 파악도** 가능하게 하자. -> Skip connection = Concatenation



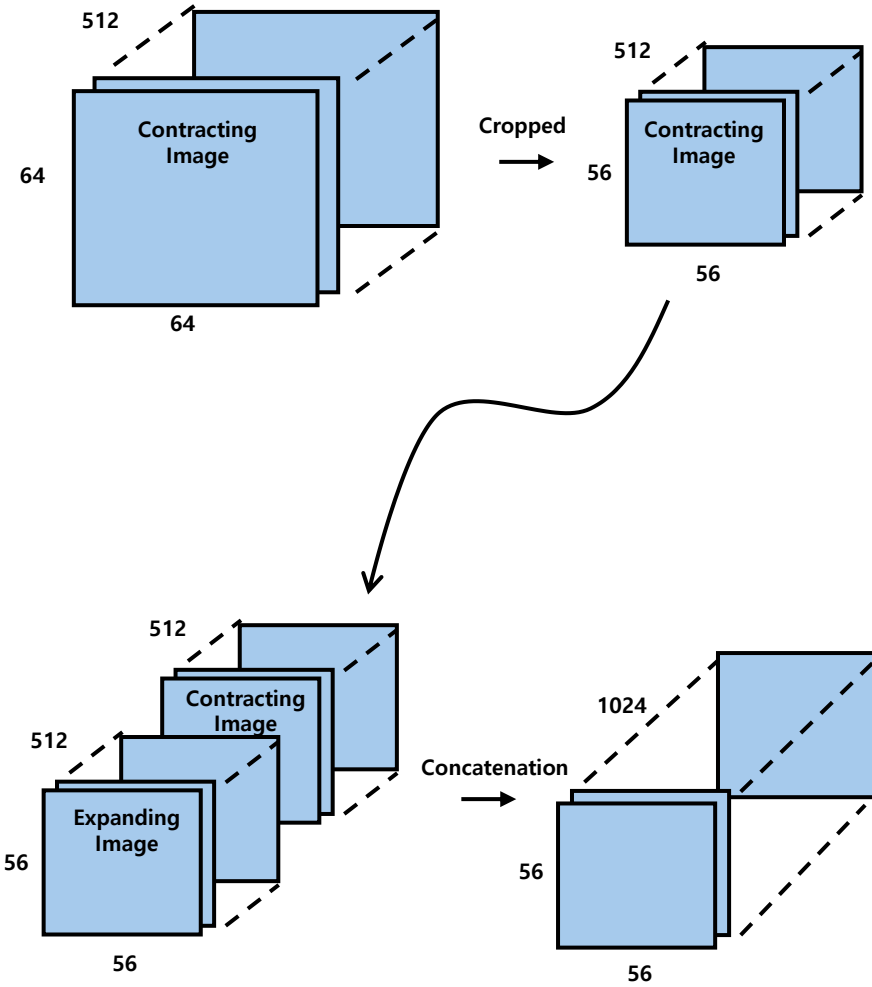
인코딩 단계의 각 레이어에서 얻은 특징을 디코딩 단계의 각 레이어에 합치는 과정



Unet – Concatenation



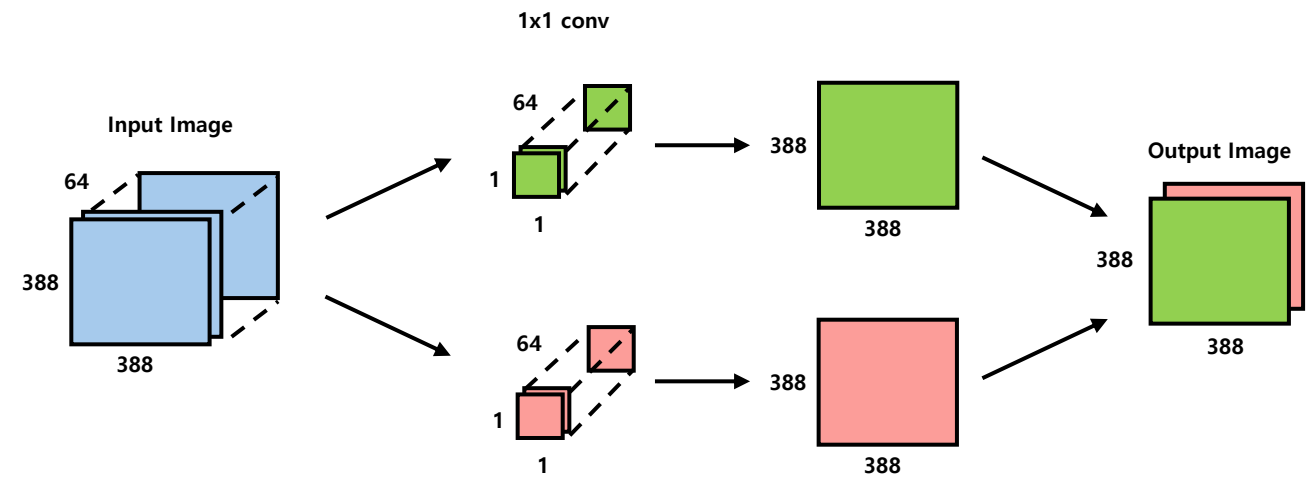
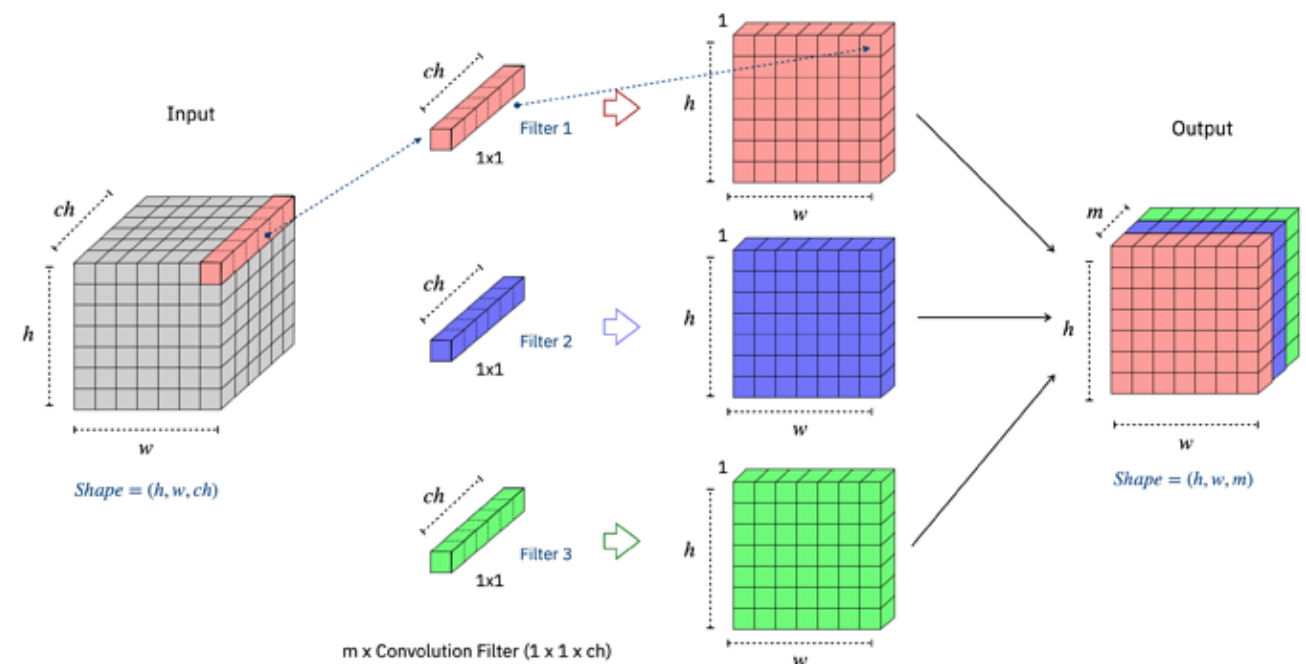
Concatenation



각 Expanding Step 마다  
Up-conv 된 특징맵은 Contracting path의 Cropped된 특징맵과  
Concatenation 함

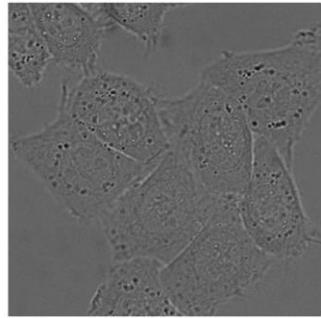
- conv 3x3, ReLU
- copy and crop
- ↓ max pool 2x2
- ↑ up-conv 2x2
- conv 1x1

Unet – 1x1 conv

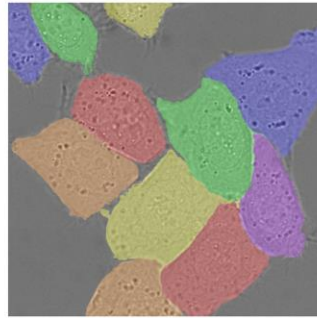




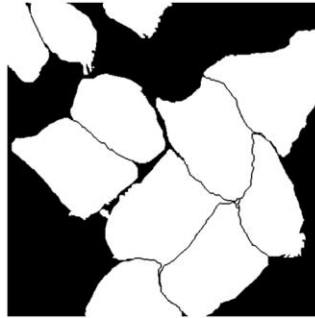
## Unet – Loss Function



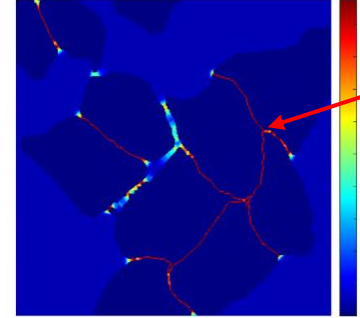
원본 이미지



이미지 분할 목표



생성된 분할 이미지



Weight Map 시각화

W(x)는 객체의 경계 부분에서 큰 값을 갖는 것을 확인

$$\begin{aligned} Loss &= \sum_x w(x) \log(p_{l(x)}(x)) \\ p_k(x) &= \exp(a_k(x)) / (\sum_i^K \exp(a_i(x))) \\ w(x) &= w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}\right) \end{aligned}$$

$a_k(x)$  : 픽셀 x가 Class k일 값(픽셀 별 모델의 Output)

$p_k(x)$  : 픽셀 x가 Class k일 확률(0~1)

$l(x)$  : 픽셀 x의 실제 Label

$w_0$  : 논문의 Weight hyper-parameter, 논문에서 10으로 설정

$\sigma$  : 논문의 Weight hyper-parameter, 논문에서 5로 설정

$d_1(x)$  : 픽셀 x의 위치로부터 가장 가까운 경계와 거리

$d_2(x)$  : 픽셀 x의 위치로부터 두번째로 가까운 경계와 거리

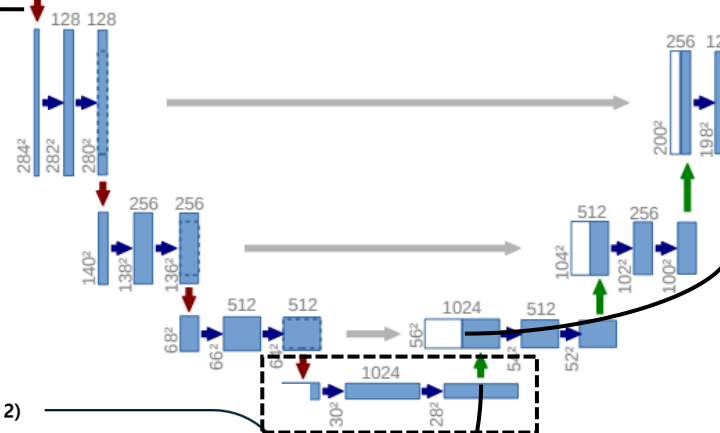
각 픽셀이 경계와 얼마나 가까운지에 따른 Weight-Map을 만들고 학습할 때 경계에 가까운 픽셀의 Loss를 Weight-Map에 비례하게 증가 시킴으로써 **경계를 잘 학습**하도록 설계하였습니다.

W(x)는 **픽셀 x와 경계의 거리가 가까우면** 큰 값을 갖게 되므로 **해당 픽셀의 Loss 비중이 커지게** 됩니다. 즉, 학습 시 경계에 해당하는 픽셀을 잘 학습하게 됩니다.

3x3 Convolution (Stride = 1, Padding = None)



2x2 MaxPooling (Stride = 2)



Up-Sampling (Transposed Convolution)

U-Net : 2x2 Transposed Convolution (stride = 2)

Input Image (4x4)

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

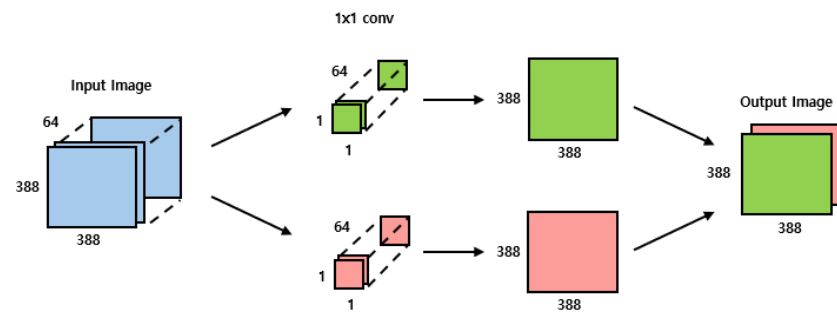
Kernel (2x2)

x	y
z	w

=

a-x	a-y	b-x	b-y	c-x	c-y	d-x	d-y
a-z	a-w	b-z	b-w	c-z	c-w	d-z	d-w
e-x	e-y	f-x	f-y	g-x	g-y	h-x	h-y
e-z	e-w	f-z	f-w	g-z	g-w	h-z	h-w
i-x	i-y	j-x	j-y	k-x	k-y	l-x	l-y
i-z	i-w	j-z	j-w	k-z	k-w	l-z	l-w
m-x	m-y	n-x	n-y	o-x	o-y	p-x	p-y
m-z	m-w	n-z	n-w	o-z	o-w	p-z	p-w

Conv 1x1



Skip Connection (Concatenation)

- conv 3x3, ReLU
- copy and crop
- ↓ max pool 2x2
- ↑ up-conv 2x2
- conv 1x1

