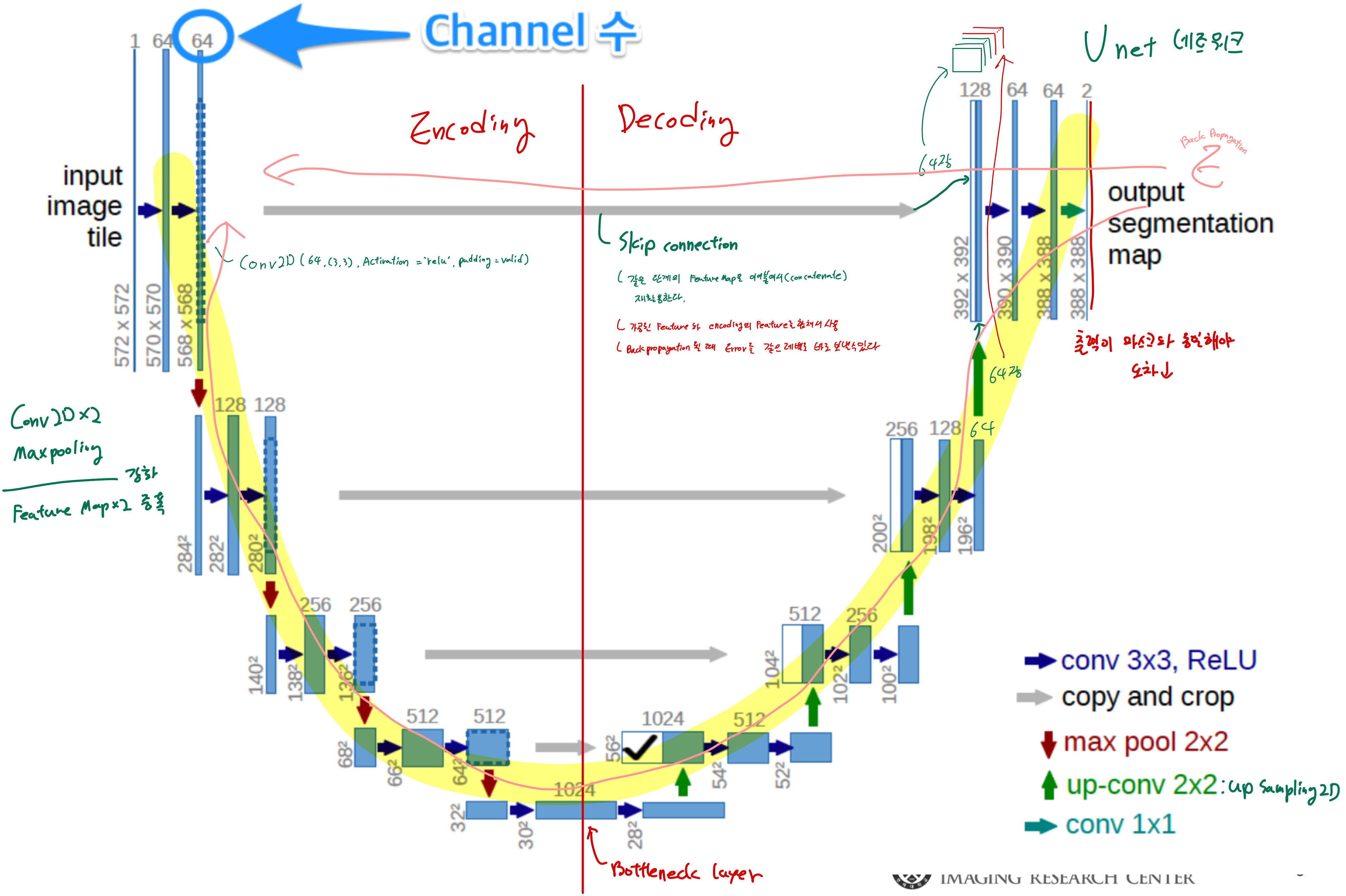
Contracting Path

Expanding Path



Channel 수

Unet 네트워크

Encoding

Decoding

Conv2D (64, (3, 3), Activation = 'relu', padding = valid)

Conv2D ×2
Maxpooling
강화
Feature Map ×2 증폭

Skip connection

(같은 단계의 Feature Map을 이어붙여서 (concatenate) 재활용한다.

(가공된 Feature 와 encoding의 Feature를 합쳐서 사용

(Backpropagation 될 때 Error를 같은 레벨로 바로 보낼수있다

출력이 마스크와 동일해야 오차↓

Bottleneck layer

➡ conv 3x3, ReLU
➡ copy and crop
⬇ max pool 2x2
⬆ up-conv 2x2 : up Sampling2D
➡ conv 1x1

14일차 segmentation → pixel classification

- Size는 2^n으로 표현

- batch = np.random.choice(files, size=batch_size)

  ↳ img 1 & mask 1
  경로 2 & " 2
  3 & " 3
  ⋮ ⋮
  n mask n

- if batch_size == 4 : (4,256,256,3) Batch network에 입력

- batch_x = [img N-dim1 , " 2 , …]

  batch_y = [img N-dim1 , " 2 , …]

  ↳ numpy를 List로 감싸기

- preprocess the binary mask

  mask [mask >=2] =0

  mask [mask !=0] =1

- 분리할 것 두가지[배경, 물체] → ex) 배경, 강아지, 고양이

  $\underbrace{\text{배경}}_{0}, \underbrace{\text{물체}}_{1}$ → ex) $\underbrace{\text{배경}}_{0}, \underbrace{\text{강아지}}_{1}, \underbrace{\text{고양이}}_{2}$

  ↳ pixel classification

  $\begin{bmatrix}1\\0\\0\end{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix}$

4D-Tensor $(\underbrace{4}_{\text{샘플}}, \underbrace{224, 224}_{\text{2차원 좌표}}, \underbrace{2}_{\text{pixel당 경우의 수}})$

Input

n(224, 224, 3)

pixel당 가질 수 있는 경우의 수

↳ channel

target

n(224 , 224 , 3)

↳ 배경, 고양이 강아지

- Binary mask

0	1	1
0	1	1
1	1	1

⇒ onehot encoding

Logit vector

0 : [1, 0]

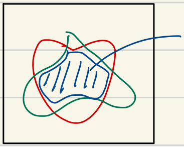
1 : [0, 1]

1	0	0
1	0	0
0	0	0

0	1	1
0	1	1
1	1	1

IoU metric

- 분류 acc : 잘 분류 했는지



Segmentation에서 정확도를 측정하는 방법

A : 예측라벨

B : 정답

$$IOU = \frac{A \cap B}{A \cup B}$$

- def mean_iou(y_true, y_pred):

```
    yt0 = y_true[:,:,:,0]  # 정답 마스크
                                        Sigmoid, 경계선 부분의 값을 0.4~0.5로 처리한다
    yp0 = K.cast(y_pred[:,:,:,0] > 0.5, 'float32')  # True:1, False:0
    inter = tf.count_nonzero(tf.logical_and(tf.equal(yt0, 1), tf.equal(yp0, 1)))
    union =
```