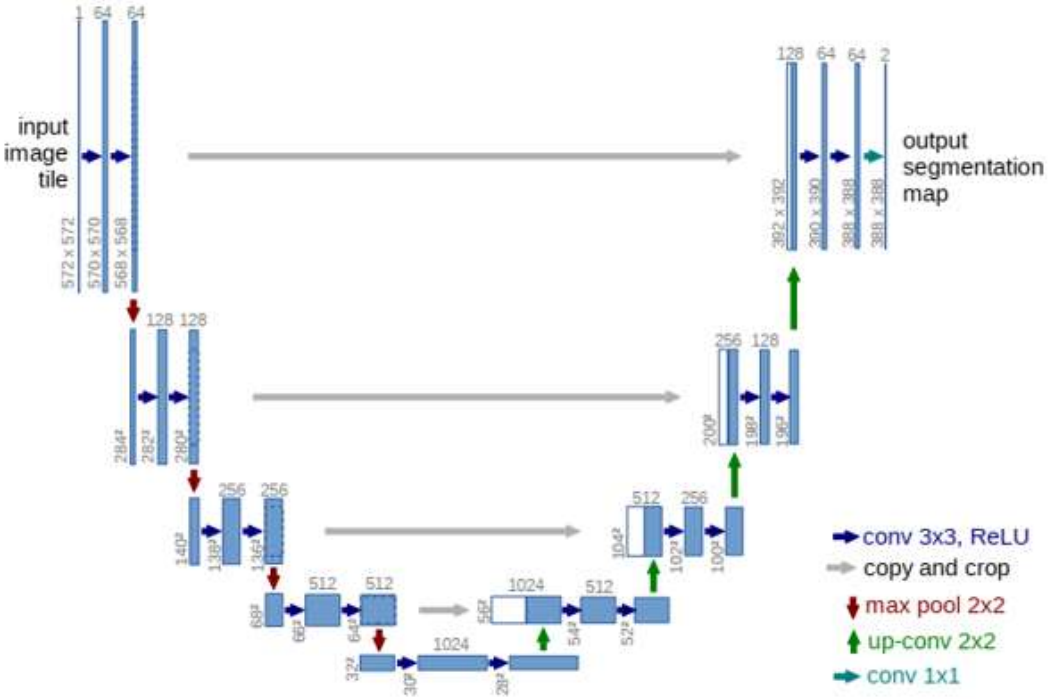
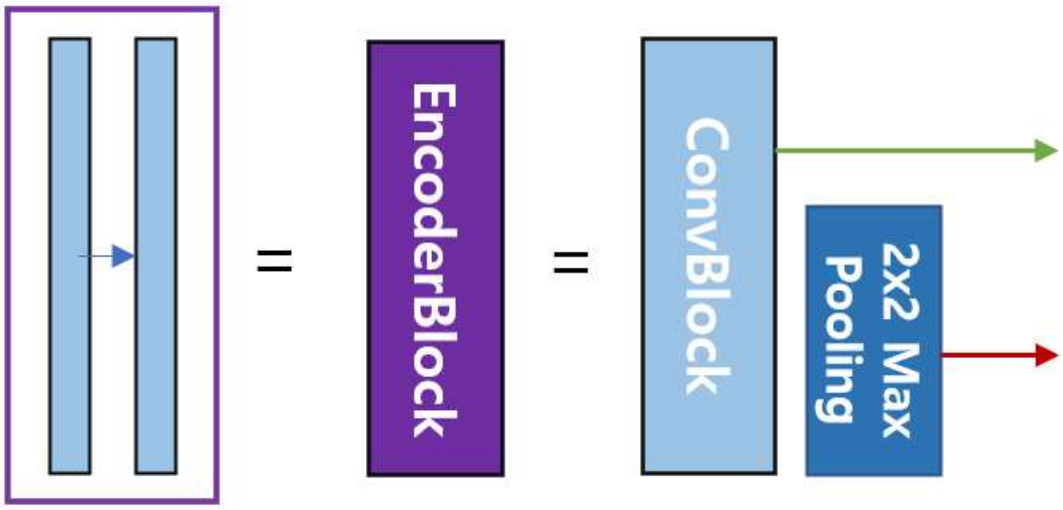
UNET

- 이미지 세그멘테이션를 하기위한 모델
- UNET의 구조를 위해 388*388크기의 Segmentation Map을 얻기 위해서는 572*572 크기의 Input Image가 필요.
  - 이미지 크기가 줄어들어 나오는 이유는 UNET에서 padding없이 Convolution 연산을 반복적으로 수행했기 때문 ## UNET의 구조



- Conv Block = n # of 3*3 Conv+ *Batch Normalization+ Relu+ n# of 3*3 Conv+ Batch Normalization + ReLu
- 3*3 Conv를 두차례 반복때 패딩이 없기 때문에 Feature Map이 줄어듦 ### Contracting Path (축소 경로) = Encoder
- 넓은 범위의 이미지 픽셀을 보며 의미정보(Context Information) 추출 #### Encoder Block
- 

- Encoder Block = (ConvBlock + 2*2 MaxPooling) + (ConvBlock + skip connection)



- ConvBlock에서 나오는 출력이 2개. 하나는 U-Net의 디코더로 복사하기 위한 연결선. 다른 하나는 2*2 MaxPooling으로 다운 샘플링하여 다음 단계의 인코더로 보내는 것.
- 2*2 MaxPooling: Feature map의 크기가 절반으로 줄어듦.
- Down-sampling마다 채널의 수가 2배로 늘어남.

Skip Connection (위치정보 전달)

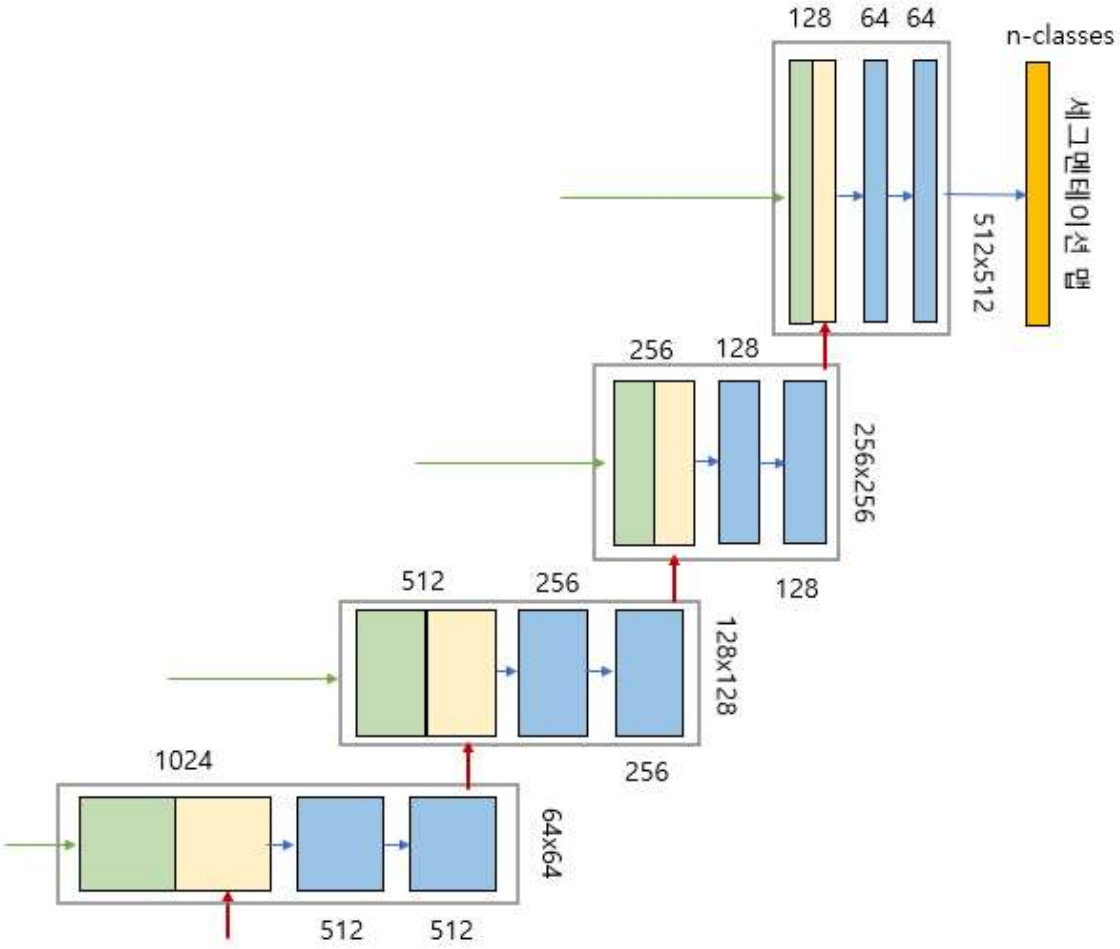
- Encoder 단계에서 대응되는 Decoder layer로 바로 연결하는 것.
- Encoder layer에서 나온 특징맵을 대응하는 Decoder layer에 있는 Cropped된 특징맵과 Concatenation함.

Bridge (Bottle Neck - 전환구간)

- 수축경로에서 확장 경로로 전환되는 구간
- 1개의 ConvBlock레이어

Expanding Path (확장 경로) = Decoder

- 의미정보를 픽셀 위치정보와 결합 (Localization)해 각 픽셀마다 어떤 객체에 속하는지 구분.



DecoderBlock

- DecoderBlock = n# of 2*2 ConvTranspose + Concatenate + ConvBlock
  - 2*2 ConvTranspose할 때 Feature map의 크기가 두배로 늘어남
  - 3*3 Conv를 두번씩 반복하면 패딩이 없음으로 Feature Map이 조금씩 줄어듬
  - Up-sampling마다 채널의 수가 절반으로 줄어듬.

전체적인 구조

- 4개의 EncoderBlock + Bridge + 4개의 DecoderBlock + FCN(Fully Conv Network)

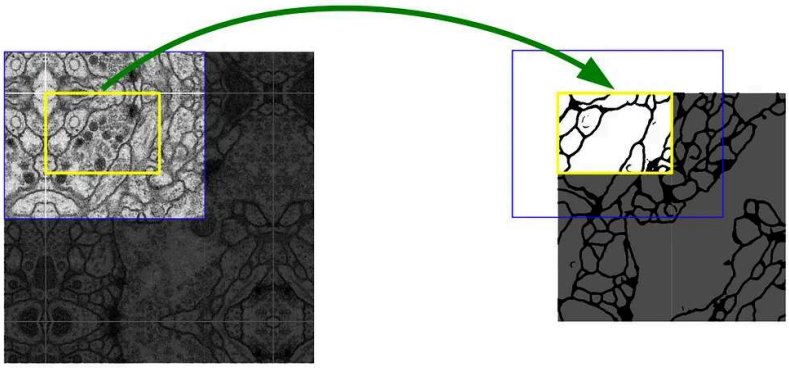
FCN(Fully Convolutional Network)

- FC layer (fully connected layer)는 flatten되기 때문에 이미지의 위치 정보가 사라짐 또한 입력 이미지의 크기가 고정된다. 하지만 FCN은 이 문제들을 다 해결해준다
- Convolutionalization
  - 기존 CNN의 완전 연결 계층들을 합성곱 계층으로 바꿈.
  - 출력 특징맵은 원본 이미지의 위치정보를 가지고 있게됨.
  - 출력 특징맵이 너무 크다는 문제점 (coarse)
  - 따라서 Coarse map을 원본 이미지 크기에 가까운 Dense map으로 변환해주기
- Deconvolution
  - Coarse map에서 Ense map을 얻는 방법은 다양하나 풀링이나 스트라이드를 사용하게되면 tpalf
  - Bilinear Interpolation

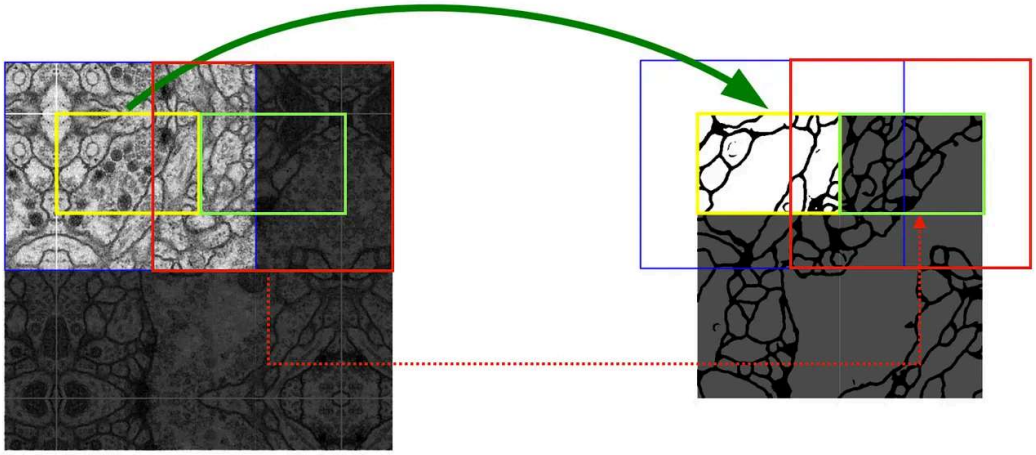
    - Backward Convolution
- Skip Architecture

Overlap-tile strategy & Mirroring Extrapolation

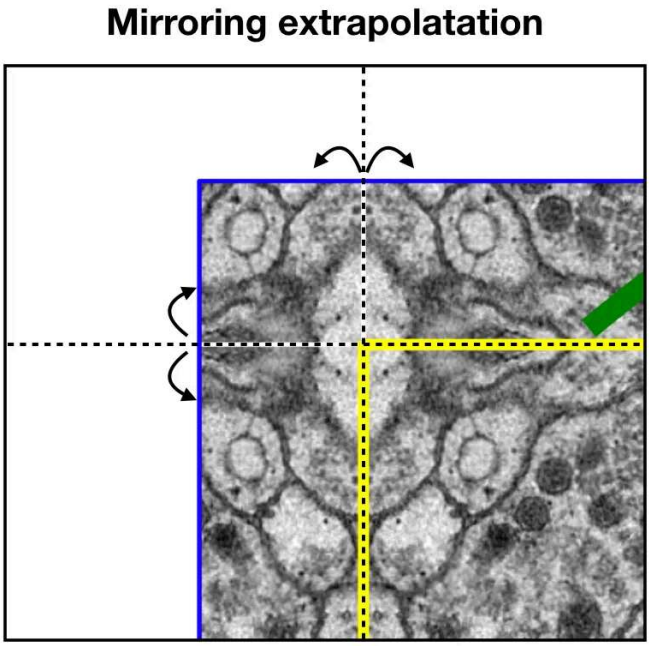
- 고용량의 이미지를 효율적으로 처리하기 위한 방안.
- 용량이 큰 이미지의 경우 이미지 전체를 사용하는 대신 overlap-tile전략을 사용



-
- 파란 영역의 이미지를 입력하면 노란 영역의 Segmentation결과를 얻음

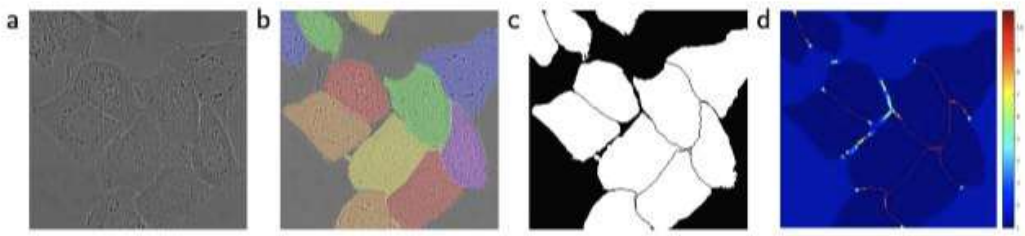


-
- 다음 타일에 대한 Segmentation을 얻기 위해서는 이전 입력의 일부분이 포함되어야함 따라서 overlap이 들어가게 되는 것.
- 이미지 경계 부분 픽셀에 대한 세그멘테이션을 위해 즉, missing data 문제를 해결하기 위해 사용한 것이 mirroring extrapolation.



손실함수 재정의 - Weight Map

- 작은 경계를 잘 분리할 수 있도록 설계
- 각 픽셀이 경계와 얼마나 가까운지에 따른 Weight-Map을 만들고 학습할 때 경계에 가까운 픽셀의 Loss를 Weight-Map에 비례하게 증가시킴으로써 경계를 잘 학습하도록 설계.



(a)는 원본 이미지, (b)는 분할 정답을 씌어놓은 것이고 다른 색은 서로 다른 인스턴스(instance)를 의미함. (c)는 생성된 분할 마스크입니다. 흰색은 새포, 검정색은 배경을 의미함. (d)는 경계 픽셀을 학습하기 위해서 픽셀 단위로 강조된 손실함수의 가중치를 의미함.

Transposed Convolution

Convolution

In [3]:

```
#필요한 라이브러리를 로드합니다.
import os
import math
import numpy as np
import tensorflow as tf

from PIL import Image
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.transform import resize
from glob import glob
```

```
from tensorflow.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
```

In [4]:

```
from albumentations import  HorizontalFlip, RandomSizedCrop, Compose, OneOf, Resize

def build_augmentation(is_train=True):
    if is_train:    # 훈련용 데이터일 경우
        return Compose([
                    HorizontalFlip(p=0.5),    # 50%의 확률로 좌우대칭
                    RandomSizedCrop(         # 50%의 확률로 RandomSizedCrop
                        min_max_height=(300, 370),
                        w2h_ratio=370/1242,
                        height=224,
                        width=224,
                        p=0.5
                        ),
                    Resize(              # 입력이미지를 224X224로 resize
                        width=224,
                        height=224
                        )
                    ])
    return Compose([      # 테스트용 데이터일 경우에는 224X224로 resize만 수행합니다.
                Resize(
                    width=224,
                    height=224
                    )
                ])
```

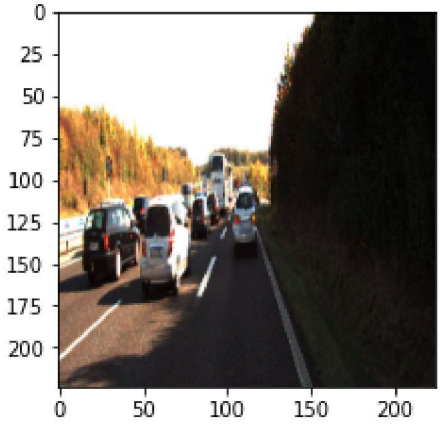
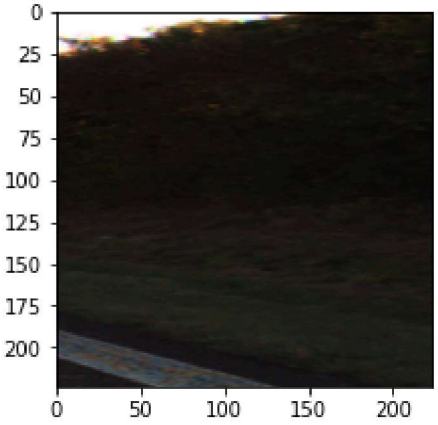
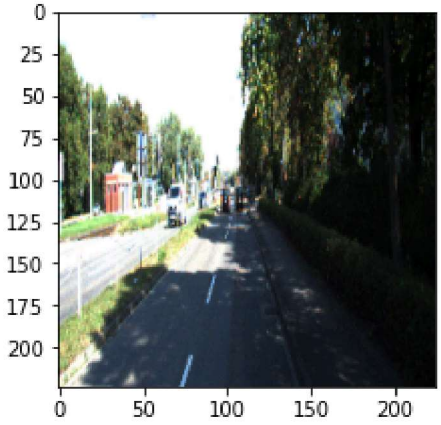
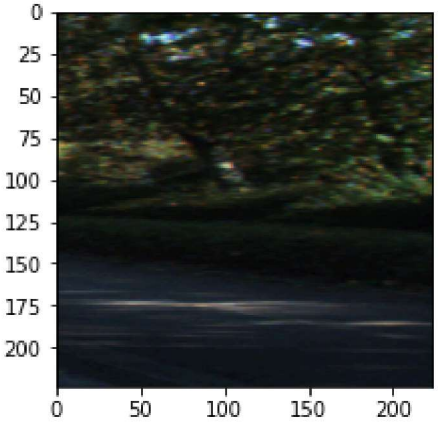
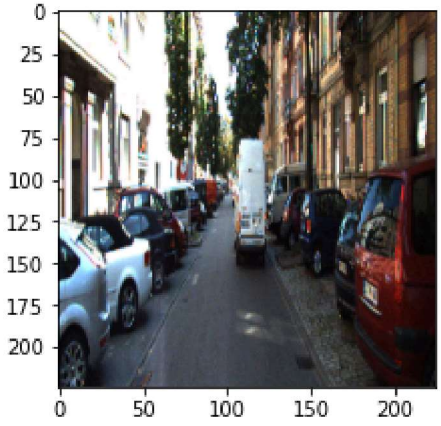
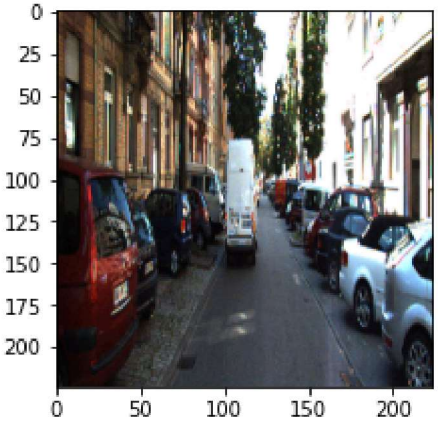
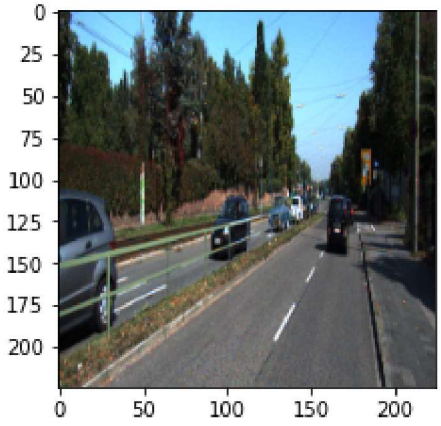
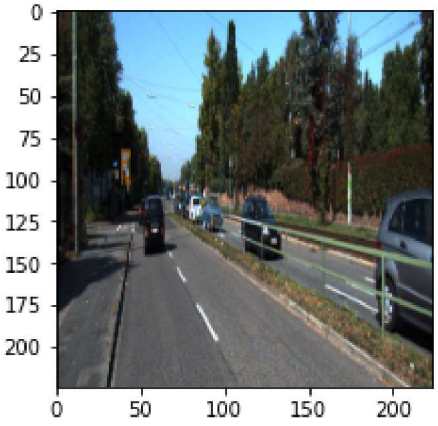
In [5]:

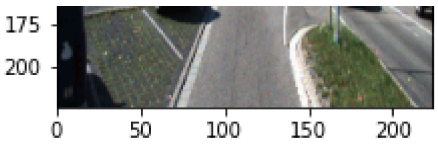
```
dir_path = os.getenv('HOME')+'/aiffel/GoingDeeper/semantic_segmentation/data/trainin

augmentation_train = build_augmentation()
augmentation_test = build_augmentation(is_train=False)
input_images = glob(os.path.join(dir_path, "image_2", "*.png"))

# 훈련 데이터셋에서 5개만 가져와 augmentation을 적용해 봅시다.
plt.figure(figsize=(12, 20))
for i in range(5):
    image = imread(input_images[i])
    image_data = {"image":image}
    resized = augmentation_test(**image_data)
    processed = augmentation_train(**image_data)
    plt.subplot(5, 2, 2*i+1)
    plt.imshow(resized["image"])  # 왼쪽이 원본이미지
    plt.subplot(5, 2, 2*i+2)
    plt.imshow(processed["image"])  # 오른쪽이 augment된 이미지


plt.show()
```



In [6]:

```
'''
KittiGenerator는 tf.keras.utils.Sequence를 상속받습니다.
우리가 KittiDataset을 원하는 방식으로 preprocess하기 위해서 Sequnce를 커스텀해 사용합니다
'''
class KittiGenerator(tf.keras.utils.Sequence):

    def __init__(self,
                 dir_path,
                 batch_size=16,
                 img_size=(224, 224, 3),
                 output_size=(224, 224),
                 is_train=True,
                 augmentation=None):
        '''
        dir_path: dataset의 directory path입니다.
        batch_size: batch_size입니다.
        img_size: preprocess에 사용할 입력이미지의 크기입니다.
        output_size: ground_truth를 만들어주기 위한 크기입니다.
        is_train: 이 Generator가 학습용인지 테스트용인지 구분합니다.
        augmentation: 적용하길 원하는 augmentation 함수를 인자로 받습니다.
        '''
        self.dir_path = dir_path
        self.batch_size = batch_size
        self.is_train = is_train
        self.dir_path = dir_path
        self.augmentation = augmentation
        self.img_size = img_size
        self.output_size = output_size

        # load_dataset()을 통해서 kitti dataset의 directory path에서 라벨과 이미지를 획
        self.data = self.load_dataset()

    def load_dataset(self):
        # kitti dataset에서 필요한 정보(이미지 경로 및 라벨)를 directory에서 확인하고 로
        # 이때 is_train에 따라 test set을 분리해서 load하도록 해야합니다.
        input_images = glob(os.path.join(self.dir_path, "image_2", "*.png"))
        label_images = glob(os.path.join(self.dir_path, "semantic", "*.png"))
        input_images.sort()
        label_images.sort()
        assert len(input_images) == len(label_images)
        data = [ _ for _ in zip(input_images, label_images)]

        if self.is_train:
            return data[:-30]
        return data[-30:]

    def __len__(self):
    # Generator의 length로서 전체 dataset을 batch_size로 나누고 소숫점 첫째자리에서 올림
        return math.ceil(len(self.data) / self.batch_size)

    def __getitem__(self, index):
        # 입력과 출력을 만듭니다.
        # 입력은 resize및 augmentation이 적용된 input image이고
        # 출력은 semantic label입니다.
        batch_data = self.data[
                                index*self.batch_size:
                                (index + 1)*self.batch_size
```

```
                                    ]
        inputs = np.zeros([self.batch_size, *self.img_size])
        outputs = np.zeros([self.batch_size, *self.output_size])

        for i, data in enumerate(batch_data):
            input_img_path, output_path = data
            _input = imread(input_img_path)
            _output = imread(output_path)
            _output = (_output==7).astype(np.uint8)*1
            data = {
              "image": _input,
              "mask": _output,
              }
            augmented = self.augmentation(**data)
            inputs[i] = augmented["image"]/255
            outputs[i] = augmented["mask"]
        return inputs, outputs

    def on_epoch_end(self):
        # 한 epoch가 끝나면 실행되는 함수입니다. 학습중인 경우에 순서를 random shuffle하
        self.indexes = np.arange(len(self.data))
        if self.is_train == True :
            np.random.shuffle(self.indexes)
            return self.indexes
```

In [7]:

```
augmentation = build_augmentation()
test_preproc = build_augmentation(is_train=False)

train_generator = KittiGenerator(
    dir_path,
    augmentation=augmentation,
)

test_generator = KittiGenerator(
    dir_path,
    augmentation=test_preproc,
    is_train=False
)
```

모델 구조 만들기

U-Net

In [14]:

```
def build_convblock(x, filters):
    x = tf.keras.layers.Conv2D(filters = filters, kernel_size =(3,3), padding = 'sam
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation('relu')(x)

    x = tf.keras.layers.Conv2D(filters = filters, kernel_size =(3,3), padding = 'sam
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation('relu')(x)

    return x
def build_encoderblock(x, filters):
    skip = build_convblock(x, filters)
    x = tf.keras.layers.MaxPool2D((2,2))(x)
    return skip, x

def build_decoderblock(x, skip, filters):
    x = tf.keras.layers.Conv2DTranspose(filters = filters, kernel_size = (2,2), stri
```

```
    x = tf.keras.layers.Concatenate()([x,skip])
    x = build_convblock(x, filters)
    return x
```

In [38]:

```
def build_unet(input_shape=(224, 224, 3), num_class = 1):
    # TODO: input_shape에 따라 U-Net을 만들어주세요
    inputs = tf.keras.layers.Input(shape = input_shape)

    #encoder: contracting path - downsample
    skip1, x = build_encoderblock(inputs, 64)
    skip2, x = build_encoderblock(x, 128)
    skip3, x = build_encoderblock(x, 256)
    skip4, x = build_encoderblock(x, 512)

    # bottleneck
    x = build_convblock(x, 1024)

    # decoder: expanding path - upsample
    x = build_decoderblock(x, skip4, 512)
    x = build_decoderblock(x, skip3, 256)
    x = build_decoderblock(x, skip2, 128)
    x = build_decoderblock(x, skip1, 64)

    # outputs using FCN
    if num_class == 1:
        act = 'sigmoid'
    else:
        act = 'softmax'

    outputs = tf.keras.layers.Conv2D(num_class, kernel_size = (1,1), padding = "same
    model = tf.keras.Model(inputs, outputs, name = "U-Net")

    return model

unet = build_unet()
unet.summary()
```

```
Model: "U-Net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================
input_16 (InputLayer)        [(None, 224, 224, 3) 0
_________________________________________________________________
max_pooling2d_60 (MaxPooling2D) (None, 112, 112, 3)  0           input_16[0][0]
_________________________________________________________________
max_pooling2d_61 (MaxPooling2D) (None, 56, 56, 3)    0           max_pooling2d_60[0]
[0]
_________________________________________________________________
max_pooling2d_62 (MaxPooling2D) (None, 28, 28, 3)    0           max_pooling2d_61[0]
[0]
_________________________________________________________________
max_pooling2d_63 (MaxPooling2D) (None, 14, 14, 3)    0           max_pooling2d_62[0]
[0]
_________________________________________________________________
conv2d_307 (Conv2D)          (None, 14, 14, 1024) 28672       max_pooling2d_63[0]
[0]
```

batch_normalization_300 (BatchN	(None, 14, 14, 1024)	4096	conv2d_307[0][0]
conv2d_305 (Conv2D) [0]	(None, 28, 28, 512)	14336	max_pooling2d_62[0]
activation_300 (Activation) 300[0][0]	(None, 14, 14, 1024)	0	batch_normalization_
batch_normalization_298 (BatchN	(None, 28, 28, 512)	2048	conv2d_305[0][0]
conv2d_308 (Conv2D)	(None, 14, 14, 1024)	9438208	activation_300[0][0]
activation_298 (Activation) 298[0][0]	(None, 28, 28, 512)	0	batch_normalization_
batch_normalization_301 (BatchN	(None, 14, 14, 1024)	4096	conv2d_308[0][0]
conv2d_306 (Conv2D)	(None, 28, 28, 512)	2359808	activation_298[0][0]
activation_301 (Activation) 301[0][0]	(None, 14, 14, 1024)	0	batch_normalization_
batch_normalization_299 (BatchN	(None, 28, 28, 512)	2048	conv2d_306[0][0]
conv2d_transpose_28 (Conv2DTran	(None, 28, 28, 512)	2097664	activation_301[0][0]
activation_299 (Activation) 299[0][0]	(None, 28, 28, 512)	0	batch_normalization_
concatenate_76 (Concatenate) [0][0]	(None, 28, 28, 1024)	0	conv2d_transpose_28 activation_299[0][0]
conv2d_309 (Conv2D)	(None, 28, 28, 512)	4719104	concatenate_76[0][0]
batch_normalization_302 (BatchN	(None, 28, 28, 512)	2048	conv2d_309[0][0]
conv2d_303 (Conv2D) [0]	(None, 56, 56, 256)	7168	max_pooling2d_61[0]
activation_302 (Activation) 302[0][0]	(None, 28, 28, 512)	0	batch_normalization_
batch_normalization_296 (BatchN	(None, 56, 56, 256)	1024	conv2d_303[0][0]

```
conv2d_310 (Conv2D)             (None, 28, 28, 512)  2359808     activation_302[0][0]
_____________
activation_296 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
296[0][0]
_____________
batch_normalization_303 (BatchN (None, 28, 28, 512)  2048        conv2d_310[0][0]
_____________
conv2d_304 (Conv2D)             (None, 56, 56, 256)  590080      activation_296[0][0]
_____________
activation_303 (Activation)     (None, 28, 28, 512)  0           batch_normalization_
303[0][0]
_____________
batch_normalization_297 (BatchN (None, 56, 56, 256)  1024        conv2d_304[0][0]
_____________
conv2d_transpose_29 (Conv2DTran (None, 56, 56, 256)  524544      activation_303[0][0]
_____________
activation_297 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
297[0][0]
_____________
concatenate_77 (Concatenate)    (None, 56, 56, 512)  0           conv2d_transpose_29
[0][0]
                                                                 activation_297[0][0]
_____________
conv2d_311 (Conv2D)             (None, 56, 56, 256)  1179904     concatenate_77[0][0]
_____________
batch_normalization_304 (BatchN (None, 56, 56, 256)  1024        conv2d_311[0][0]
_____________
conv2d_301 (Conv2D)             (None, 112, 112, 128 3584        max_pooling2d_60[0]
[0]
_____________
activation_304 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
304[0][0]
_____________
batch_normalization_294 (BatchN (None, 112, 112, 128 512         conv2d_301[0][0]
_____________
conv2d_312 (Conv2D)             (None, 56, 56, 256)  590080      activation_304[0][0]
_____________
activation_294 (Activation)     (None, 112, 112, 128 0           batch_normalization_
294[0][0]
_____________
batch_normalization_305 (BatchN (None, 56, 56, 256)  1024        conv2d_312[0][0]
_____________
conv2d_302 (Conv2D)             (None, 112, 112, 128 147584      activation_294[0][0]
_____________
activation_305 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
```

```
305[0][0]
==================================================================================================
batch_normalization_295 (BatchN (None, 112, 112, 128 512         conv2d_302[0][0]
__________________________________________________________________________________________________
conv2d_transpose_30 (Conv2DTran (None, 112, 112, 128 131200      activation_305[0][0]
__________________________________________________________________________________________________
activation_295 (Activation)     (None, 112, 112, 128 0           batch_normalization_
295[0][0]
__________________________________________________________________________________________________
concatenate_78 (Concatenate)    (None, 112, 112, 256 0           conv2d_transpose_30
[0][0]
                                                                 activation_295[0][0]
__________________________________________________________________________________________________
conv2d_313 (Conv2D)             (None, 112, 112, 128 295040      concatenate_78[0][0]
__________________________________________________________________________________________________
batch_normalization_306 (BatchN (None, 112, 112, 128 512         conv2d_313[0][0]
__________________________________________________________________________________________________
conv2d_299 (Conv2D)             (None, 224, 224, 64) 1792        input_16[0][0]
__________________________________________________________________________________________________
activation_306 (Activation)     (None, 112, 112, 128 0           batch_normalization_
306[0][0]
__________________________________________________________________________________________________
batch_normalization_292 (BatchN (None, 224, 224, 64) 256         conv2d_299[0][0]
__________________________________________________________________________________________________
conv2d_314 (Conv2D)             (None, 112, 112, 128 147584      activation_306[0][0]
__________________________________________________________________________________________________
activation_292 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
292[0][0]
__________________________________________________________________________________________________
batch_normalization_307 (BatchN (None, 112, 112, 128 512         conv2d_314[0][0]
__________________________________________________________________________________________________
conv2d_300 (Conv2D)             (None, 224, 224, 64) 36928       activation_292[0][0]
__________________________________________________________________________________________________
activation_307 (Activation)     (None, 112, 112, 128 0           batch_normalization_
307[0][0]
__________________________________________________________________________________________________
batch_normalization_293 (BatchN (None, 224, 224, 64) 256         conv2d_300[0][0]
__________________________________________________________________________________________________
conv2d_transpose_31 (Conv2DTran (None, 224, 224, 64) 32832       activation_307[0][0]
__________________________________________________________________________________________________
activation_293 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
293[0][0]
__________________________________________________________________________________________________
concatenate_79 (Concatenate)    (None, 224, 224, 128 0           conv2d_transpose_31
[0][0]
                                                                 activation_293[0][0]
```

```
conv2d_315 (Conv2D)             (None, 224, 224, 64) 73792       concatenate_79[0][0]
__________________________________________________________________________________________
batch_normalization_308 (BatchN (None, 224, 224, 64) 256         conv2d_315[0][0]
__________________________________________________________________________________________
activation_308 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
308[0][0]
__________________________________________________________________________________________
conv2d_316 (Conv2D)             (None, 224, 224, 64) 36928       activation_308[0][0]
__________________________________________________________________________________________
batch_normalization_309 (BatchN (None, 224, 224, 64) 256         conv2d_316[0][0]
__________________________________________________________________________________________
activation_309 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
309[0][0]
__________________________________________________________________________________________
conv2d_317 (Conv2D)             (None, 224, 224, 1)  65          activation_309[0][0]
==========================================================================================
=============
Total params: 24,840,257
Trainable params: 24,828,481
Non-trainable params: 11,776
__________________________________________________________________________________________
```

UNET++

In [39]:

```python
def build_unet_plus(input_shape=(224, 224, 3), num_class = 1):
    # TODO: input_shape에 따라 U-Net을 만들어주세요
    inputs = tf.keras.layers.Input(shape = input_shape)

    #encoder: contracting path - downsample
    skip1, x = build_encoderblock(inputs, 64)
    skip2, x = build_encoderblock(x, 128)
    skip3, x = build_encoderblock(x, 256)
    skip4, x = build_encoderblock(x, 512)

    # bottleneck
    x = build_convblock(x, 1024)

    # prepareation to skip
    up4 = tf.keras.layers.UpSampling2D(size = (2,2))(skip4)
    up3 = tf.keras.layers.UpSampling2D(size = (2,2))(skip3)
    up2 = tf.keras.layers.UpSampling2D(size = (2,2))(skip2)

    skip31 = build_convblock(Concatenate()([up4, skip3]), 256)
    up31 = tf.keras.layers.UpSampling2D(size = (2,2))(skip31)

    skip21 = build_convblock(Concatenate()([up3, skip2]), 128)
    up21 = tf.keras.layers.UpSampling2D(size = (2,2))(skip21)

    skip22 = build_convblock(Concatenate()([up31, skip21]), 128)
    up22 = tf.keras.layers.UpSampling2D(size = (2,2))(skip22)

    skip11 = build_convblock(Concatenate()([up2, skip1]), 64)
    skip12 = build_convblock(Concatenate()([up21, skip11]), 64)
    skip13 = build_convblock(Concatenate()([up22, skip12]), 64)
```

```
    # decoder: expanding path - upsample
    x = build_decoderblock(x, skip4, 512)
    x = build_decoderblock(x, skip31, 256)
    x = build_decoderblock(x, skip22, 128)
    x = build_decoderblock(x, skip13, 64)

    # outputs using FCN
    if num_class == 1:
        act = 'sigmoid'
    else:
        act = 'softmax'

    outputs = tf.keras.layers.Conv2D(num_class, kernel_size = (1,1), padding = "same
    model = tf.keras.Model(inputs, outputs, name = "U_Net_Plus")

    return model

unet_plus = build_unet_plus()
unet_plus.summary()
```

```
Model: "U_Net_Plus"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_17 (InputLayer)           [(None, 224, 224, 3) 0
__________________________________________________________________________________
max_pooling2d_64 (MaxPooling2D) (None, 112, 112, 3)  0           input_17[0][0]
__________________________________________________________________________________
max_pooling2d_65 (MaxPooling2D) (None, 56, 56, 3)    0           max_pooling2d_64[0]
[0]
__________________________________________________________________________________
max_pooling2d_66 (MaxPooling2D) (None, 28, 28, 3)    0           max_pooling2d_65[0]
[0]
__________________________________________________________________________________
max_pooling2d_67 (MaxPooling2D) (None, 14, 14, 3)    0           max_pooling2d_66[0]
[0]
__________________________________________________________________________________
conv2d_326 (Conv2D)             (None, 14, 14, 1024) 28672       max_pooling2d_67[0]
[0]
__________________________________________________________________________________
batch_normalization_318 (BatchN (None, 14, 14, 1024) 4096        conv2d_326[0][0]
__________________________________________________________________________________
conv2d_324 (Conv2D)             (None, 28, 28, 512)  14336       max_pooling2d_66[0]
[0]
__________________________________________________________________________________
activation_318 (Activation)     (None, 14, 14, 1024) 0           batch_normalization_
318[0][0]
__________________________________________________________________________________
batch_normalization_316 (BatchN (None, 28, 28, 512)  2048        conv2d_324[0][0]
__________________________________________________________________________________
conv2d_322 (Conv2D)             (None, 56, 56, 256)  7168        max_pooling2d_65[0]
```

```
[0]
__________________________________________________________________________________________________
conv2d_327 (Conv2D)             (None, 14, 14, 1024) 9438208     activation_318[0][0]
__________________________________________________________________________________________________
activation_316 (Activation)     (None, 28, 28, 512)  0           batch_normalization_
316[0][0]
__________________________________________________________________________________________________
batch_normalization_314 (BatchN (None, 56, 56, 256)  1024        conv2d_322[0][0]
__________________________________________________________________________________________________
conv2d_320 (Conv2D)             (None, 112, 112, 128 3584        max_pooling2d_64[0]
[0]
__________________________________________________________________________________________________
batch_normalization_319 (BatchN (None, 14, 14, 1024) 4096        conv2d_327[0][0]
__________________________________________________________________________________________________
conv2d_325 (Conv2D)             (None, 28, 28, 512)  2359808     activation_316[0][0]
__________________________________________________________________________________________________
activation_314 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
314[0][0]
__________________________________________________________________________________________________
batch_normalization_312 (BatchN (None, 112, 112, 128 512         conv2d_320[0][0]
__________________________________________________________________________________________________
conv2d_318 (Conv2D)             (None, 224, 224, 64) 1792        input_17[0][0]
__________________________________________________________________________________________________
activation_319 (Activation)     (None, 14, 14, 1024) 0           batch_normalization_
319[0][0]
__________________________________________________________________________________________________
batch_normalization_317 (BatchN (None, 28, 28, 512)  2048        conv2d_325[0][0]
__________________________________________________________________________________________________
conv2d_323 (Conv2D)             (None, 56, 56, 256)  590080      activation_314[0][0]
__________________________________________________________________________________________________
activation_312 (Activation)     (None, 112, 112, 128 0           batch_normalization_
312[0][0]
__________________________________________________________________________________________________
batch_normalization_310 (BatchN (None, 224, 224, 64) 256         conv2d_318[0][0]
__________________________________________________________________________________________________
conv2d_transpose_32 (Conv2DTran (None, 28, 28, 512)  2097664     activation_319[0][0]
__________________________________________________________________________________________________
activation_317 (Activation)     (None, 28, 28, 512)  0           batch_normalization_
317[0][0]
__________________________________________________________________________________________________
batch_normalization_315 (BatchN (None, 56, 56, 256)  1024        conv2d_323[0][0]
__________________________________________________________________________________________________
conv2d_321 (Conv2D)             (None, 112, 112, 128 147584      activation_312[0][0]
__________________________________________________________________________________________________
activation_310 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
```

```
310[0][0]
__________________________________________________________________________________________________
__________________
concatenate_86 (Concatenate)    (None, 28, 28, 1024) 0           conv2d_transpose_32
[0][0]
                                                                 activation_317[0][0]
__________________________________________________________________________________________________
__________________
up_sampling2d_52 (UpSampling2D) (None, 56, 56, 512)  0           activation_317[0][0]
__________________________________________________________________________________________________
__________________
activation_315 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
315[0][0]
__________________________________________________________________________________________________
__________________
batch_normalization_313 (BatchN (None, 112, 112, 128 512         conv2d_321[0][0]
__________________________________________________________________________________________________
__________________
conv2d_319 (Conv2D)             (None, 224, 224, 64) 36928       activation_310[0][0]
__________________________________________________________________________________________________
__________________
conv2d_340 (Conv2D)             (None, 28, 28, 512)  4719104     concatenate_86[0][0]
__________________________________________________________________________________________________
__________________
concatenate_80 (Concatenate)    (None, 56, 56, 768)  0           up_sampling2d_52[0]
[0]
                                                                 activation_315[0][0]
__________________________________________________________________________________________________
__________________
up_sampling2d_53 (UpSampling2D) (None, 112, 112, 256 0           activation_315[0][0]
__________________________________________________________________________________________________
__________________
activation_313 (Activation)     (None, 112, 112, 128 0           batch_normalization_
313[0][0]
__________________________________________________________________________________________________
__________________
batch_normalization_311 (BatchN (None, 224, 224, 64) 256         conv2d_319[0][0]
__________________________________________________________________________________________________
__________________
batch_normalization_332 (BatchN (None, 28, 28, 512)  2048        conv2d_340[0][0]
__________________________________________________________________________________________________
__________________
conv2d_328 (Conv2D)             (None, 56, 56, 256)  1769728     concatenate_80[0][0]
__________________________________________________________________________________________________
__________________
concatenate_81 (Concatenate)    (None, 112, 112, 384 0           up_sampling2d_53[0]
[0]
                                                                 activation_313[0][0]
__________________________________________________________________________________________________
__________________
up_sampling2d_54 (UpSampling2D) (None, 224, 224, 128 0           activation_313[0][0]
__________________________________________________________________________________________________
__________________
activation_311 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
311[0][0]
__________________________________________________________________________________________________
__________________
activation_332 (Activation)     (None, 28, 28, 512)  0           batch_normalization_
332[0][0]
__________________________________________________________________________________________________
__________________
batch_normalization_320 (BatchN (None, 56, 56, 256)  1024        conv2d_328[0][0]
__________________________________________________________________________________________________
__________________
```

```
conv2d_330 (Conv2D)             (None, 112, 112, 128 442496      concatenate_81[0][0]
__________________________________________________________________________________________
concatenate_83 (Concatenate)    (None, 224, 224, 192 0           up_sampling2d_54[0]
[0]                                                              activation_311[0][0]
__________________________________________________________________________________________
conv2d_341 (Conv2D)             (None, 28, 28, 512)  2359808     activation_332[0][0]
__________________________________________________________________________________________
activation_320 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
320[0][0]
__________________________________________________________________________________________
batch_normalization_322 (BatchN (None, 112, 112, 128 512         conv2d_330[0][0]
__________________________________________________________________________________________
conv2d_334 (Conv2D)             (None, 224, 224, 64) 110656      concatenate_83[0][0]
__________________________________________________________________________________________
batch_normalization_333 (BatchN (None, 28, 28, 512)  2048        conv2d_341[0][0]
__________________________________________________________________________________________
conv2d_329 (Conv2D)             (None, 56, 56, 256)  590080      activation_320[0][0]
__________________________________________________________________________________________
activation_322 (Activation)     (None, 112, 112, 128 0           batch_normalization_
322[0][0]
__________________________________________________________________________________________
batch_normalization_326 (BatchN (None, 224, 224, 64) 256         conv2d_334[0][0]
__________________________________________________________________________________________
activation_333 (Activation)     (None, 28, 28, 512)  0           batch_normalization_
333[0][0]
__________________________________________________________________________________________
batch_normalization_321 (BatchN (None, 56, 56, 256)  1024        conv2d_329[0][0]
__________________________________________________________________________________________
conv2d_331 (Conv2D)             (None, 112, 112, 128 147584      activation_322[0][0]
__________________________________________________________________________________________
activation_326 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
326[0][0]
__________________________________________________________________________________________
conv2d_transpose_33 (Conv2DTran (None, 56, 56, 256)  524544      activation_333[0][0]
__________________________________________________________________________________________
activation_321 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
321[0][0]
__________________________________________________________________________________________
batch_normalization_323 (BatchN (None, 112, 112, 128 512         conv2d_331[0][0]
__________________________________________________________________________________________
conv2d_335 (Conv2D)             (None, 224, 224, 64) 36928       activation_326[0][0]
__________________________________________________________________________________________
concatenate_87 (Concatenate)    (None, 56, 56, 512)  0           conv2d_transpose_33
[0][0]                                                           activation_321[0][0]
```

```
____________
up_sampling2d_55 (UpSampling2D) (None, 112, 112, 256 0           activation_321[0][0]
____________
activation_323 (Activation)     (None, 112, 112, 128 0           batch_normalization_
323[0][0]
____________
batch_normalization_327 (BatchN (None, 224, 224, 64) 256         conv2d_335[0][0]
____________
conv2d_342 (Conv2D)             (None, 56, 56, 256)  1179904     concatenate_87[0][0]
____________
concatenate_82 (Concatenate)    (None, 112, 112, 384 0           up_sampling2d_55[0]
[0]
                                                                 activation_323[0][0]
____________
up_sampling2d_56 (UpSampling2D) (None, 224, 224, 128 0           activation_323[0][0]
____________
activation_327 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
327[0][0]
____________
batch_normalization_334 (BatchN (None, 56, 56, 256)  1024        conv2d_342[0][0]
____________
conv2d_332 (Conv2D)             (None, 112, 112, 128 442496      concatenate_82[0][0]
____________
concatenate_84 (Concatenate)    (None, 224, 224, 192 0           up_sampling2d_56[0]
[0]
                                                                 activation_327[0][0]
____________
activation_334 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
334[0][0]
____________
batch_normalization_324 (BatchN (None, 112, 112, 128 512         conv2d_332[0][0]
____________
conv2d_336 (Conv2D)             (None, 224, 224, 64) 110656      concatenate_84[0][0]
____________
conv2d_343 (Conv2D)             (None, 56, 56, 256)  590080      activation_334[0][0]
____________
activation_324 (Activation)     (None, 112, 112, 128 0           batch_normalization_
324[0][0]
____________
batch_normalization_328 (BatchN (None, 224, 224, 64) 256         conv2d_336[0][0]
____________
batch_normalization_335 (BatchN (None, 56, 56, 256)  1024        conv2d_343[0][0]
____________
conv2d_333 (Conv2D)             (None, 112, 112, 128 147584      activation_324[0][0]
____________
```

```
activation_328 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
328[0][0]
_____________
activation_335 (Activation)     (None, 56, 56, 256)  0           batch_normalization_
335[0][0]
_____________
batch_normalization_325 (BatchN (None, 112, 112, 128 512         conv2d_333[0][0]
_____________
conv2d_337 (Conv2D)             (None, 224, 224, 64) 36928       activation_328[0][0]
_____________
conv2d_transpose_34 (Conv2DTran (None, 112, 112, 128 131200      activation_335[0][0]
_____________
activation_325 (Activation)     (None, 112, 112, 128 0           batch_normalization_
325[0][0]
_____________
batch_normalization_329 (BatchN (None, 224, 224, 64) 256         conv2d_337[0][0]
_____________
concatenate_88 (Concatenate)    (None, 112, 112, 256 0           conv2d_transpose_34
[0][0]
                                                                 activation_325[0][0]
_____________
up_sampling2d_57 (UpSampling2D) (None, 224, 224, 128 0           activation_325[0][0]
_____________
activation_329 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
329[0][0]
_____________
conv2d_344 (Conv2D)             (None, 112, 112, 128 295040      concatenate_88[0][0]
_____________
concatenate_85 (Concatenate)    (None, 224, 224, 192 0           up_sampling2d_57[0]
[0]
                                                                 activation_329[0][0]
_____________
batch_normalization_336 (BatchN (None, 112, 112, 128 512         conv2d_344[0][0]
_____________
conv2d_338 (Conv2D)             (None, 224, 224, 64) 110656      concatenate_85[0][0]
_____________
activation_336 (Activation)     (None, 112, 112, 128 0           batch_normalization_
336[0][0]
_____________
batch_normalization_330 (BatchN (None, 224, 224, 64) 256         conv2d_338[0][0]
_____________
conv2d_345 (Conv2D)             (None, 112, 112, 128 147584      activation_336[0][0]
_____________
activation_330 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
330[0][0]
_____________
```

```
batch_normalization_337 (BatchN (None, 112, 112, 128 512         conv2d_345[0][0]
_____________
conv2d_339 (Conv2D)             (None, 224, 224, 64) 36928       activation_330[0][0]
_____________
activation_337 (Activation)     (None, 112, 112, 128 0           batch_normalization_
337[0][0]
_____________
batch_normalization_331 (BatchN (None, 224, 224, 64) 256         conv2d_339[0][0]
_____________
conv2d_transpose_35 (Conv2DTran (None, 224, 224, 64) 32832       activation_337[0][0]
_____________
activation_331 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
331[0][0]
_____________
concatenate_89 (Concatenate)    (None, 224, 224, 128 0           conv2d_transpose_35
[0][0]
                                                                 activation_331[0][0]
_____________
conv2d_346 (Conv2D)             (None, 224, 224, 64) 73792       concatenate_89[0][0]
_____________
batch_normalization_338 (BatchN (None, 224, 224, 64) 256         conv2d_346[0][0]
_____________
activation_338 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
338[0][0]
_____________
conv2d_347 (Conv2D)             (None, 224, 224, 64) 36928       activation_338[0][0]
_____________
batch_normalization_339 (BatchN (None, 224, 224, 64) 256         conv2d_347[0][0]
_____________
activation_339 (Activation)     (None, 224, 224, 64) 0           batch_normalization_
339[0][0]
_____________
conv2d_348 (Conv2D)             (None, 224, 224, 1)  65          activation_339[0][0]
==========================================================================================
=============
Total params: 28,828,609
Trainable params: 28,814,017
Non-trainable params: 14,592
_____________
_____________
```

모델 학습하기

In [47]:

```
unet = build_unet()
unet.compile(optimizer = Adam(1e-4), loss = 'binary_crossentropy')
history = unet.fit(
      train_generator,
      validation_data=test_generator,
      steps_per_epoch=len(train_generator),
      epochs=100,
```

```
)

unet.save(model_path)  #학습한 모델을 저장해 주세요.
```

```
Epoch 1/100
11/11 [==============================] - 11s 795ms/step - loss: 0.4153 - val_loss: 0.
6933
Epoch 2/100
11/11 [==============================] - 8s 714ms/step - loss: 0.3161 - val_loss: 0.7
107
Epoch 3/100
11/11 [==============================] - 8s 709ms/step - loss: 0.2821 - val_loss: 0.7
539
Epoch 4/100
11/11 [==============================] - 8s 711ms/step - loss: 0.2482 - val_loss: 0.7
843
Epoch 5/100
11/11 [==============================] - 8s 713ms/step - loss: 0.2520 - val_loss: 0.7
859
Epoch 6/100
11/11 [==============================] - 8s 711ms/step - loss: 0.2377 - val_loss: 0.7
813
Epoch 7/100
11/11 [==============================] - 8s 713ms/step - loss: 0.2141 - val_loss: 0.7
210
Epoch 8/100
11/11 [==============================] - 8s 714ms/step - loss: 0.2032 - val_loss: 0.8
031
Epoch 9/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1984 - val_loss: 0.8
805
Epoch 10/100
11/11 [==============================] - 8s 712ms/step - loss: 0.2041 - val_loss: 0.7
629
Epoch 11/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1859 - val_loss: 0.8
016
Epoch 12/100
11/11 [==============================] - 8s 718ms/step - loss: 0.1839 - val_loss: 0.8
096
Epoch 13/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1747 - val_loss: 0.8
086
Epoch 14/100
11/11 [==============================] - 8s 716ms/step - loss: 0.1899 - val_loss: 0.6
737
Epoch 15/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1820 - val_loss: 0.6
063
Epoch 16/100
11/11 [==============================] - 8s 716ms/step - loss: 0.1798 - val_loss: 0.5
657
Epoch 17/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1707 - val_loss: 0.5
182
Epoch 18/100
11/11 [==============================] - 8s 714ms/step - loss: 0.1757 - val_loss: 0.5
058
Epoch 19/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1744 - val_loss: 0.5
291
Epoch 20/100
11/11 [==============================] - 8s 718ms/step - loss: 0.1792 - val_loss: 0.4
406
```

```
Epoch 21/100
11/11 [==============================] - 8s 708ms/step - loss: 0.1594 - val_loss: 0.4
223
Epoch 22/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1695 - val_loss: 0.4
294
Epoch 23/100
11/11 [==============================] - 8s 707ms/step - loss: 0.1733 - val_loss: 0.4
176
Epoch 24/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1576 - val_loss: 0.3
877
Epoch 25/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1562 - val_loss: 0.3
673
Epoch 26/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1645 - val_loss: 0.3
925
Epoch 27/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1665 - val_loss: 0.4
360
Epoch 28/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1562 - val_loss: 0.4
001
Epoch 29/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1496 - val_loss: 0.3
991
Epoch 30/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1384 - val_loss: 0.3
735
Epoch 31/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1388 - val_loss: 0.2
983
Epoch 32/100
11/11 [==============================] - 8s 714ms/step - loss: 0.1468 - val_loss: 0.2
884
Epoch 33/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1507 - val_loss: 0.3
140
Epoch 34/100
11/11 [==============================] - 8s 716ms/step - loss: 0.1518 - val_loss: 0.3
338
Epoch 35/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1460 - val_loss: 0.3
028
Epoch 36/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1511 - val_loss: 0.2
775
Epoch 37/100
11/11 [==============================] - 8s 708ms/step - loss: 0.1395 - val_loss: 0.2
496
Epoch 38/100
11/11 [==============================] - 8s 714ms/step - loss: 0.1453 - val_loss: 0.2
566
Epoch 39/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1522 - val_loss: 0.2
628
Epoch 40/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1476 - val_loss: 0.3
410
Epoch 41/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1550 - val_loss: 0.3
189
Epoch 42/100
```

```
11/11 [==============================] - 8s 712ms/step - loss: 0.1325 - val_loss: 0.2
907
Epoch 43/100
11/11 [==============================] - 8s 708ms/step - loss: 0.1370 - val_loss: 0.3
211
Epoch 44/100
11/11 [==============================] - 8s 716ms/step - loss: 0.1462 - val_loss: 0.3
072
Epoch 45/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1518 - val_loss: 0.2
402
Epoch 46/100
11/11 [==============================] - 8s 718ms/step - loss: 0.1206 - val_loss: 0.2
134
Epoch 47/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1262 - val_loss: 0.2
315
Epoch 48/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1248 - val_loss: 0.2
148
Epoch 49/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1336 - val_loss: 0.2
871
Epoch 50/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1393 - val_loss: 0.3
218
Epoch 51/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1364 - val_loss: 0.2
291
Epoch 52/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1349 - val_loss: 0.2
083
Epoch 53/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1467 - val_loss: 0.2
502
Epoch 54/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1316 - val_loss: 0.2
309
Epoch 55/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1329 - val_loss: 0.2
289
Epoch 56/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1341 - val_loss: 0.3
448
Epoch 57/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1286 - val_loss: 0.1
713
Epoch 58/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1274 - val_loss: 0.1
853
Epoch 59/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1238 - val_loss: 0.1
793
Epoch 60/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1228 - val_loss: 0.1
772
Epoch 61/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1234 - val_loss: 0.2
157
Epoch 62/100
11/11 [==============================] - 8s 714ms/step - loss: 0.1201 - val_loss: 0.1
958
Epoch 63/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1073 - val_loss: 0.1
```

```
956
Epoch 64/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1172 - val_loss: 0.2
069
Epoch 65/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1260 - val_loss: 0.2
190
Epoch 66/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1199 - val_loss: 0.1
931
Epoch 67/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1418 - val_loss: 0.1
978
Epoch 68/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1289 - val_loss: 0.2
407
Epoch 69/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1234 - val_loss: 0.2
356
Epoch 70/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1098 - val_loss: 0.1
854
Epoch 71/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1243 - val_loss: 0.1
794
Epoch 72/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1248 - val_loss: 0.1
878
Epoch 73/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1239 - val_loss: 0.1
647
Epoch 74/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1369 - val_loss: 0.2
465
Epoch 75/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1101 - val_loss: 0.2
197
Epoch 76/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1068 - val_loss: 0.2
300
Epoch 77/100
11/11 [==============================] - 8s 708ms/step - loss: 0.1102 - val_loss: 0.1
740
Epoch 78/100
11/11 [==============================] - 8s 716ms/step - loss: 0.1106 - val_loss: 0.2
306
Epoch 79/100
11/11 [==============================] - 9s 736ms/step - loss: 0.1244 - val_loss: 0.2
353
Epoch 80/100
11/11 [==============================] - 9s 736ms/step - loss: 0.1262 - val_loss: 0.2
106
Epoch 81/100
11/11 [==============================] - 8s 721ms/step - loss: 0.1238 - val_loss: 0.2
073
Epoch 82/100
11/11 [==============================] - 8s 717ms/step - loss: 0.1084 - val_loss: 0.2
060
Epoch 83/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1105 - val_loss: 0.1
873
Epoch 84/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1021 - val_loss: 0.1
949
```

```
Epoch 85/100
11/11 [==============================] - 8s 715ms/step - loss: 0.1062 - val_loss: 0.2
584
Epoch 86/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1142 - val_loss: 0.2
611
Epoch 87/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1087 - val_loss: 0.2
271
Epoch 88/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1180 - val_loss: 0.2
525
Epoch 89/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1176 - val_loss: 0.2
220
Epoch 90/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1104 - val_loss: 0.2
470
Epoch 91/100
11/11 [==============================] - 8s 713ms/step - loss: 0.1173 - val_loss: 0.1
811
Epoch 92/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1064 - val_loss: 0.2
363
Epoch 93/100
11/11 [==============================] - 8s 709ms/step - loss: 0.1049 - val_loss: 0.1
889
Epoch 94/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1016 - val_loss: 0.1
778
Epoch 95/100
11/11 [==============================] - 8s 711ms/step - loss: 0.1064 - val_loss: 0.1
878
Epoch 96/100
11/11 [==============================] - 8s 710ms/step - loss: 0.1158 - val_loss: 0.2
826
Epoch 97/100
11/11 [==============================] - 8s 712ms/step - loss: 0.1001 - val_loss: 0.1
887
Epoch 98/100
11/11 [==============================] - 8s 710ms/step - loss: 0.0977 - val_loss: 0.1
867
Epoch 99/100
11/11 [==============================] - 8s 714ms/step - loss: 0.1020 - val_loss: 0.1
931
Epoch 100/100
11/11 [==============================] - 8s 707ms/step - loss: 0.1000 - val_loss: 0.1
886
```

In [48]:

```python
model_save_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_model.h5'
weights_save_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_weights.h5

unet.save(model_save_path)  #학습한 모델을 저장해 주세요.
print(f'Model saved to {model_save_path}')

unet.save_weights(weights_save_path)
print(f'Model weights saved to {weights_save_path}')

import pandas as pd
history_df = pd.DataFrame(history.history)
history_csv_path = 'unet_history.csv'
history_df.to_csv(history_csv_path, index=False)
print(f'Training history saved to {history_csv_path}')
```

```
Model saved to /aiffel/aiffel/semantic_segmentation/unet_model.h5
Model weights saved to /aiffel/aiffel/semantic_segmentation/unet_weights.h5
Training history saved to unet_history.csv
```

In [ ]:

```
unet_plus = build_unet_plus()
unet_plus.compile(optimizer = Adam(1e-4), loss = 'binary_crossentropy')
history = unet_plus.fit(
     train_generator,
     validation_data=test_generator,
     steps_per_epoch=len(train_generator),
     epochs=100,
 )
```

```
Epoch 1/100
11/11 [==============================] - 21s 2s/step - loss: 0.5075 - val_loss: 0.700
6
Epoch 2/100
11/11 [==============================] - 18s 2s/step - loss: 0.3875 - val_loss: 0.733
5
Epoch 3/100
11/11 [==============================] - 18s 2s/step - loss: 0.3179 - val_loss: 0.789
8
Epoch 4/100
11/11 [==============================] - 19s 2s/step - loss: 0.3035 - val_loss: 0.937
2
Epoch 5/100
11/11 [==============================] - 19s 2s/step - loss: 0.2698 - val_loss: 1.019
3
Epoch 6/100
11/11 [==============================] - 18s 2s/step - loss: 0.2716 - val_loss: 1.126
9
Epoch 7/100
11/11 [==============================] - 18s 2s/step - loss: 0.2250 - val_loss: 1.232
0
Epoch 8/100
11/11 [==============================] - 18s 2s/step - loss: 0.2320 - val_loss: 1.489
3
Epoch 9/100
11/11 [==============================] - 19s 2s/step - loss: 0.2407 - val_loss: 0.983
6
Epoch 10/100
11/11 [==============================] - 18s 2s/step - loss: 0.2336 - val_loss: 0.867
1
Epoch 11/100
11/11 [==============================] - 18s 2s/step - loss: 0.2127 - val_loss: 0.837
5
Epoch 12/100
11/11 [==============================] - 18s 2s/step - loss: 0.2115 - val_loss: 0.877
5
Epoch 13/100
11/11 [==============================] - 18s 2s/step - loss: 0.2095 - val_loss: 0.688
0
Epoch 14/100
11/11 [==============================] - 18s 2s/step - loss: 0.1947 - val_loss: 0.909
0
Epoch 15/100
11/11 [==============================] - 18s 2s/step - loss: 0.1763 - val_loss: 0.722
2
Epoch 16/100
11/11 [==============================] - 18s 2s/step - loss: 0.1765 - val_loss: 0.777
2
Epoch 17/100
11/11 [==============================] - 18s 2s/step - loss: 0.1784 - val_loss: 0.880
```

```
4
Epoch 18/100
11/11 [==============================] - 18s 2s/step - loss: 0.1782 - val_loss: 1.058
4
Epoch 19/100
11/11 [==============================] - 18s 2s/step - loss: 0.1706 - val_loss: 0.803
7
Epoch 20/100
11/11 [==============================] - 18s 2s/step - loss: 0.1626 - val_loss: 0.783
4
Epoch 21/100
11/11 [==============================] - 18s 2s/step - loss: 0.1640 - val_loss: 0.925
4
Epoch 22/100
11/11 [==============================] - 18s 2s/step - loss: 0.1638 - val_loss: 0.826
2
Epoch 23/100
11/11 [==============================] - 18s 2s/step - loss: 0.1563 - val_loss: 0.680
7
Epoch 24/100
11/11 [==============================] - 18s 2s/step - loss: 0.1621 - val_loss: 1.197
1
Epoch 25/100
11/11 [==============================] - 18s 2s/step - loss: 0.1785 - val_loss: 0.745
8
Epoch 26/100
11/11 [==============================] - 18s 2s/step - loss: 0.1619 - val_loss: 0.476
9
Epoch 27/100
11/11 [==============================] - 18s 2s/step - loss: 0.1619 - val_loss: 0.421
7
Epoch 28/100
11/11 [==============================] - 18s 2s/step - loss: 0.1660 - val_loss: 0.768
5
Epoch 29/100
11/11 [==============================] - 18s 2s/step - loss: 0.1553 - val_loss: 0.701
0
Epoch 30/100
11/11 [==============================] - 18s 2s/step - loss: 0.1526 - val_loss: 0.419
5
Epoch 31/100
11/11 [==============================] - 18s 2s/step - loss: 0.1828 - val_loss: 0.693
7
Epoch 32/100
11/11 [==============================] - 18s 2s/step - loss: 0.1545 - val_loss: 0.381
9
Epoch 33/100
11/11 [==============================] - 18s 2s/step - loss: 0.1512 - val_loss: 0.450
7
Epoch 34/100
11/11 [==============================] - 18s 2s/step - loss: 0.1394 - val_loss: 0.307
7
Epoch 35/100
11/11 [==============================] - 18s 2s/step - loss: 0.1571 - val_loss: 0.300
8
Epoch 36/100
11/11 [==============================] - 18s 2s/step - loss: 0.1359 - val_loss: 0.342
8
Epoch 37/100
11/11 [==============================] - 18s 2s/step - loss: 0.1549 - val_loss: 0.318
0
Epoch 38/100
11/11 [==============================] - 18s 2s/step - loss: 0.1471 - val_loss: 0.344
3
```

```
Epoch 39/100
11/11 [==============================] - 18s 2s/step - loss: 0.1587 - val_loss: 0.317
8
Epoch 40/100
11/11 [==============================] - 18s 2s/step - loss: 0.1571 - val_loss: 0.265
1
Epoch 41/100
11/11 [==============================] - 18s 2s/step - loss: 0.1438 - val_loss: 0.291
7
Epoch 42/100
11/11 [==============================] - 18s 2s/step - loss: 0.1468 - val_loss: 0.224
0
Epoch 43/100
11/11 [==============================] - 18s 2s/step - loss: 0.1300 - val_loss: 0.250
0
Epoch 44/100
11/11 [==============================] - 18s 2s/step - loss: 0.1357 - val_loss: 0.247
5
Epoch 45/100
11/11 [==============================] - 18s 2s/step - loss: 0.1351 - val_loss: 0.245
9
Epoch 46/100
11/11 [==============================] - 18s 2s/step - loss: 0.1325 - val_loss: 0.232
9
Epoch 47/100
11/11 [==============================] - 18s 2s/step - loss: 0.1377 - val_loss: 0.210
6
Epoch 48/100
11/11 [==============================] - 18s 2s/step - loss: 0.1342 - val_loss: 0.232
7
Epoch 49/100
11/11 [==============================] - 18s 2s/step - loss: 0.1289 - val_loss: 0.214
5
Epoch 50/100
11/11 [==============================] - 18s 2s/step - loss: 0.1288 - val_loss: 0.225
4
Epoch 51/100
11/11 [==============================] - 18s 2s/step - loss: 0.1326 - val_loss: 0.268
1
Epoch 52/100
11/11 [==============================] - 18s 2s/step - loss: 0.1353 - val_loss: 0.203
5
Epoch 53/100
11/11 [==============================] - 18s 2s/step - loss: 0.1312 - val_loss: 0.208
8
Epoch 54/100
11/11 [==============================] - 19s 2s/step - loss: 0.1582 - val_loss: 0.302
9
Epoch 55/100
11/11 [==============================] - 18s 2s/step - loss: 0.1378 - val_loss: 0.226
3
Epoch 56/100
11/11 [==============================] - 18s 2s/step - loss: 0.1335 - val_loss: 0.210
8
Epoch 57/100
11/11 [==============================] - 18s 2s/step - loss: 0.1213 - val_loss: 0.204
1
Epoch 58/100
11/11 [==============================] - 18s 2s/step - loss: 0.1287 - val_loss: 0.180
5
Epoch 59/100
11/11 [==============================] - 18s 2s/step - loss: 0.1204 - val_loss: 0.193
8
Epoch 60/100
```

```
11/11 [==============================] - 18s 2s/step - loss: 0.1291 - val_loss: 0.211
0
Epoch 61/100
11/11 [==============================] - 18s 2s/step - loss: 0.1311 - val_loss: 0.198
4
Epoch 62/100
11/11 [==============================] - 18s 2s/step - loss: 0.1203 - val_loss: 0.180
3
Epoch 63/100
11/11 [==============================] - 18s 2s/step - loss: 0.1263 - val_loss: 0.172
8
Epoch 64/100
11/11 [==============================] - 18s 2s/step - loss: 0.1184 - val_loss: 0.196
6
Epoch 65/100
11/11 [==============================] - 18s 2s/step - loss: 0.1213 - val_loss: 0.183
6
Epoch 66/100
11/11 [==============================] - 18s 2s/step - loss: 0.1304 - val_loss: 0.206
7
Epoch 67/100
11/11 [==============================] - 18s 2s/step - loss: 0.1283 - val_loss: 0.214
7
Epoch 68/100
11/11 [==============================] - 18s 2s/step - loss: 0.1307 - val_loss: 0.242
5
Epoch 69/100
11/11 [==============================] - 19s 2s/step - loss: 0.1327 - val_loss: 0.223
2
Epoch 70/100
11/11 [==============================] - 18s 2s/step - loss: 0.1165 - val_loss: 0.238
1
Epoch 71/100
11/11 [==============================] - 18s 2s/step - loss: 0.1115 - val_loss: 0.197
7
Epoch 72/100
11/11 [==============================] - 18s 2s/step - loss: 0.1202 - val_loss: 0.209
9
Epoch 73/100
11/11 [==============================] - 18s 2s/step - loss: 0.1118 - val_loss: 0.194
9
Epoch 74/100
11/11 [==============================] - 18s 2s/step - loss: 0.1230 - val_loss: 0.202
3
Epoch 75/100
11/11 [==============================] - 18s 2s/step - loss: 0.1163 - val_loss: 0.196
5
Epoch 76/100
11/11 [==============================] - 18s 2s/step - loss: 0.1114 - val_loss: 0.229
0
Epoch 77/100
11/11 [==============================] - 18s 2s/step - loss: 0.1065 - val_loss: 0.189
7
Epoch 78/100
11/11 [==============================] - 18s 2s/step - loss: 0.1120 - val_loss: 0.197
5
Epoch 79/100
11/11 [==============================] - 18s 2s/step - loss: 0.1084 - val_loss: 0.250
3
Epoch 80/100
11/11 [==============================] - 18s 2s/step - loss: 0.1038 - val_loss: 0.212
1
Epoch 81/100
11/11 [==============================] - 18s 2s/step - loss: 0.1059 - val_loss: 0.190
```

```
4
Epoch 82/100
11/11 [==============================] - 18s 2s/step - loss: 0.1041 - val_loss: 0.210
5
Epoch 83/100
11/11 [==============================] - 18s 2s/step - loss: 0.1140 - val_loss: 0.190
5
Epoch 84/100
 1/11 [=>............................] - ETA: 19s - loss: 0.1318
```

In [42]:

```
model_save_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_plus_model.h
weights_save_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_plus_weigh

unet_plus.save(model_path)  #학습한 모델을 저장해 주세요.
print(f'Model saved to {model_save_path}')

unet_plus.save_weights(weights_save_path)
print(f'Model weights saved to {weights_save_path}')

import pandas as pd
history_df = pd.DataFrame(history.history)
history_csv_path = 'unet_plus_history.csv'
history_df.to_csv(history_csv_path, index=False)
print(f'Training history saved to {history_csv_path}')
```

```
Model saved to /aiffel/aiffel/semantic_segmentation/unet_plus_model.h5
Model weights saved to /aiffel/aiffel/semantic_segmentation/unet_plus_weights.h5
Training history saved to unet_plus_history.csv
```

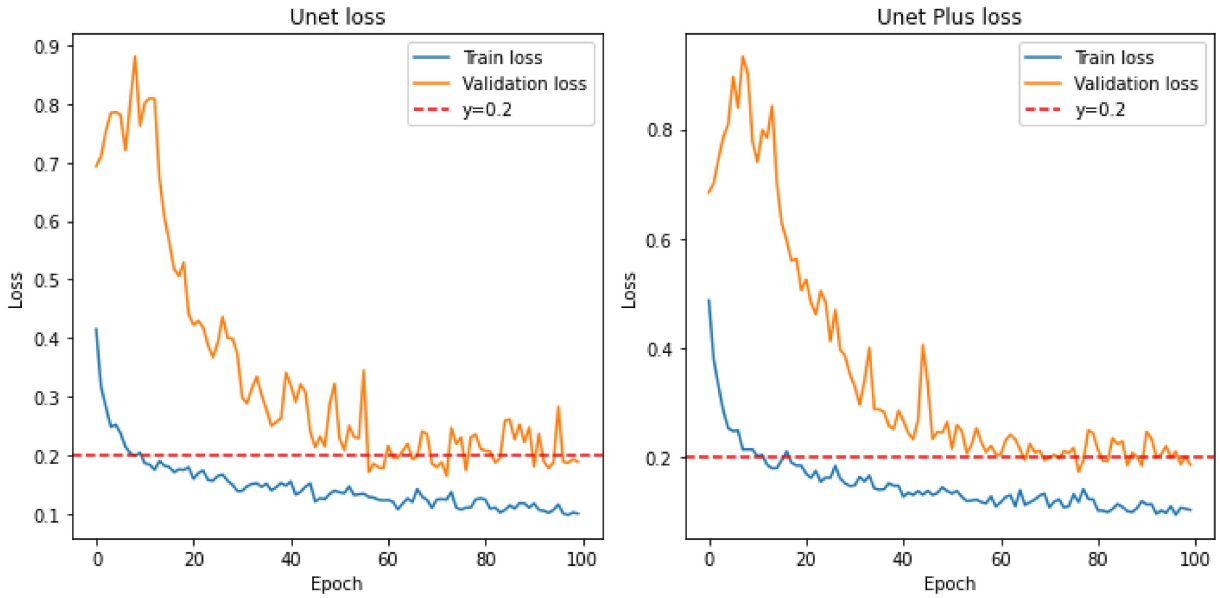
In [99]:

```
unet_csv_path = 'unet_history.csv'
unet_plus_csv_path = 'unet_plus_history.csv'
unet_history = pd.read_csv(unet_csv_path)
unet_plus_history = pd.read_csv(unet_plus_csv_path)

# Plot training & validation loss values
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(unet_history['loss'], label='Train loss')
plt.plot(unet_history['val_loss'], label='Validation loss')
plt.axhline(y=0.2, color='r', linestyle='--', label='y=0.2')  # Add horizontal line
plt.title('Unet loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(unet_plus_history['loss'], label='Train loss')
plt.plot(unet_plus_history['val_loss'], label='Validation loss')
plt.axhline(y=0.2, color='r', linestyle='--', label='y=0.2')  # Add horizontal line
plt.title('Unet Plus loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



In [103…

```
# 미리 준비한 모델을 불러오려면 아래 주석을 해제하세요
model_path = dir_path + '/seg_model_unet.h5'
unet_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_model.h5'
unet_plus_path = os.getenv('HOME')+'/aiffel/semantic_segmentation/unet_plus_model.h5
model = tf.keras.models.load_model(model_path)
unet = tf.keras.models.load_model(unet_path)
unet_plus = tf.keras.models.load_model(unet_plus_path)
```

모델 시각화

In [73]:

```
def get_output(model, preproc, image_path, output_path):
    origin_img = imread(image_path)
    data = {"image":origin_img}
    processed = preproc(**data)
    output = model(np.expand_dims(processed["image"]/255,axis=0))
    output = (output[0].numpy()>0.5).astype(np.uint8).squeeze(-1)*255  #0.5라는 thres
    output = Image.fromarray(output)
    background = Image.fromarray(origin_img).convert('RGBA')
    output = output.resize((origin_img.shape[1], origin_img.shape[0])).convert('RGBA
    output = Image.blend(background, output, alpha=0.5)
    return output
```

In [105…

```
def compare_models(model1, model2, model3, preproc, image_path, output_path):
    # 모델1의 결과
    output1 = get_output(model1, preproc, image_path, output_path)

    # 모델2의 결과
    output2 = get_output(model2, preproc, image_path, output_path)

    # 모델3의 결과
    output3 = get_output(model3, preproc, image_path, output_path)

    # 두 이미지를 좌우로 붙이기
    width1, height1 = output1.size
    width2, height2 = output2.size
    width3, height3 = output3.size
    combined_width = width1 + width2 + width3
    combined_height = max(height1, height2, height3)

    combined_image = Image.new('RGBA', (combined_width, combined_height))
```

```
    combined_image.paste(output1, (0, 0))
    combined_image.paste(output2, (width1, 0))
    combined_image.paste(output3, (width1+width2, 0))

    combined_image.show()
    return combined_image
```

In [117…

```
# 완성한 뒤에는 시각화한 결과를 눈으로 확인해봅시다!

for i in range(20):# i값을 바꾸면 테스트용 파일이 달라집니다.
    compare_models(
        model, unet, unet_plus,
        test_preproc,
        image_path=dir_path + f'/image_2/00{str(i).zfill(4)}_10.png',
        output_path=dir_path + f'./result_{str(i).zfill(3)}.png'
    )
```

맨 왼쪽은 주어진 unet model, 가운데는 새로만든 unet, 그리고 맨 왼쪽은 새로 만든 unet++ 입니다.
기존 모델과 비교해봤을때 새로만든 unet과 unet++가 더 나은 결과를 내는 것으로 보여집니다.
하지만 새로만든 unet과 unet++에서의 차이점을 찾아내기는 힘든것 같습니다.

평가하기

In [94]:

```
def calculate_iou_score(target, prediction, string):
    intersection = np.logical_and(target, prediction).sum()# intersection을 구하는 코
    union = np.logical_or(target, prediction).sum() # Union을 구하는 코드를 작성해주세
    iou_score = float(intersection) / float(union) # iou 스코어를 구하되 결과값을 floa
    print(f"{string}'s IoU : {iou_score} ")
    return iou_score
```

In [112…

```
def get_output_label(model, preproc, image_path, output_path, label_path):
    origin_img = imread(image_path)
    data = {"image":origin_img}
    processed = preproc(**data)
    output = model(np.expand_dims(processed["image"]/255,axis=0))
    output = (output[0].numpy()>=0.5).astype(np.uint8).squeeze(-1)*255  #0.5라는 thre
    prediction = output/255   # 도로로 판단한 영역

    output = Image.fromarray(output)
```

```
    background = Image.fromarray(origin_img).convert('RGBA')
    output = output.resize((origin_img.shape[1], origin_img.shape[0])).convert('RGBA
    output = Image.blend(background, output, alpha=0.5)
#     output.show()   # 도로로 판단한 영역을 시각화!

    if label_path:
        label_img = imread(label_path)
        label_data = {"image":label_img}
        label_processed = preproc(**label_data)
        label_processed = label_processed["image"]
        target = (label_processed == 7).astype(np.uint8)*1   # 라벨에서 도로로 기재된

        return output, prediction, target
    else:
        return output, prediction, _
```

In [115…

```
def compare_results(model1, model2,model3, preproc, image_path, output_path, label_p
    # 모델1의 결과
    output1, prediction1, target1 = get_output_label(model1, preproc, image_path, ou

    # 모델2의 결과
    output2, prediction2, target2 = get_output_label(model2, preproc, image_path, ou

    # 모델3의 결과
    output3, prediction3, target3 = get_output_label(model3, preproc, image_path, ou

    # 두 이미지를 좌우로 붙이기
    width1, height1 = output1.size
    width2, height2 = output2.size
    width3, height3 = output3.size
    combined_width = width1 + width2+ width3
    combined_height = max(height1, height2, height3)

    combined_image = Image.new('RGBA', (combined_width, combined_height))
    combined_image.paste(output1, (0, 0))
    combined_image.paste(output2, (width1, 0))
    combined_image.paste(output3, (width1+width2, 0))
    combined_image.show()

    return target1, prediction1, target2, prediction2, target3, prediction3
```

In [116…

```
# 완성한 뒤에는 시각화한 결과를 눈으로 확인해봅시다!
for i in range(20):   # i값을 바꾸면 테스트용 파일이 달라집니다.
    t1, p1, t2, p2, t3, p3 = compare_results(
         model, unet, unet_plus,
         test_preproc,
         image_path=dir_path + f'/image_2/00{str(i).zfill(4)}_10.png',
         output_path=dir_path + f'./result_{str(i).zfill(3)}.png',
         label_path=dir_path + f'/semantic/00{str(i).zfill(4)}_10.png'
     )

    calculate_iou_score(t1, p1, "Defualt U-Net")
    calculate_iou_score(t2, p2, "U-Net")
    calculate_iou_score(t3, p3, "U-Net ++")
```

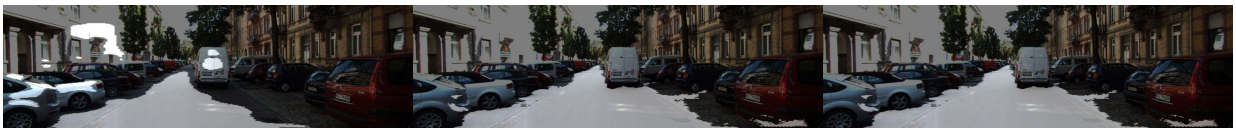
```
Defualt U-Net's IoU : 0.1691919191919192
U-Net's IoU : 0.8946866809625923
```

U-Net ++'s IoU : 0.8946866809625923

Defualt U-Net's IoU : 0.41964089410040306
U-Net's IoU : 0.9022878228782287
U-Net ++'s IoU : 0.9022878228782287

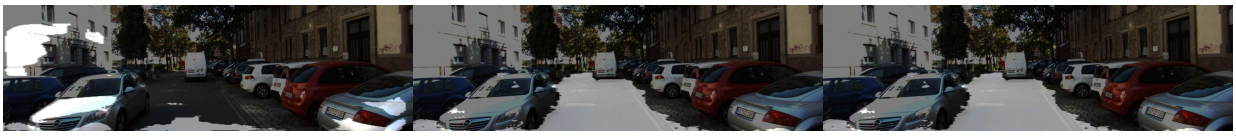
Defualt U-Net's IoU : 0.5840873634945398
U-Net's IoU : 0.7365366446214064
U-Net ++'s IoU : 0.7365366446214064

Defualt U-Net's IoU : 0.5712116368286445
U-Net's IoU : 0.5411802853437094
U-Net ++'s IoU : 0.5411802853437094



Defualt U-Net's IoU : 0.5602958516454069
U-Net's IoU : 0.7262690644206693
U-Net ++'s IoU : 0.7262690644206693

Defualt U-Net's IoU : 0.6470318061775175
U-Net's IoU : 0.7993355481727574
U-Net ++'s IoU : 0.7993355481727574



Defualt U-Net's IoU : 0.09676714722586283
U-Net's IoU : 0.7660073751257124
U-Net ++'s IoU : 0.7660073751257124

Defualt U-Net's IoU : 0.6822262118491921
U-Net's IoU : 0.8482063445004447
U-Net ++'s IoU : 0.8482063445004447

Defualt U-Net's IoU : 0.3827360245934795
U-Net's IoU : 0.8692522652214413
U-Net ++'s IoU : 0.8692522652214413

Defualt U-Net's IoU : 0.4179015612595925
U-Net's IoU : 0.792665262048986
U-Net ++'s IoU : 0.792665262048986

위의 그림들과 같이 맨 왼쪽이 기존 모델, 새로 만든 Unet그리고 Unet++.
Default의 IoU는 현저히 떨어졌고, 새로 만든 U-Net과 U-Net++는 IoU가 똑같이 꽤나 높았습니다.
하지만 어떻게 U-Net과 U-Net++의 IoU가 완전히 똑같은 값이 나오는지는 의문.

회고

- 배운점 : Unet과 Unet++의 구조를 배움
- 아쉬운점: Unet과 Unet++가 왜 똑같이 나온건지 아니면 어떤 부분이 잘못된 것일지 제대로 파악하지 못했다는 점.
- 어려운점: 개념을 공부할때 unet과 unet++의 skip connection 부부을 구현하기 헷갈렸습니다.
- 느낀점: unet과 unet++를 배우게 되어 재밌었습니다.