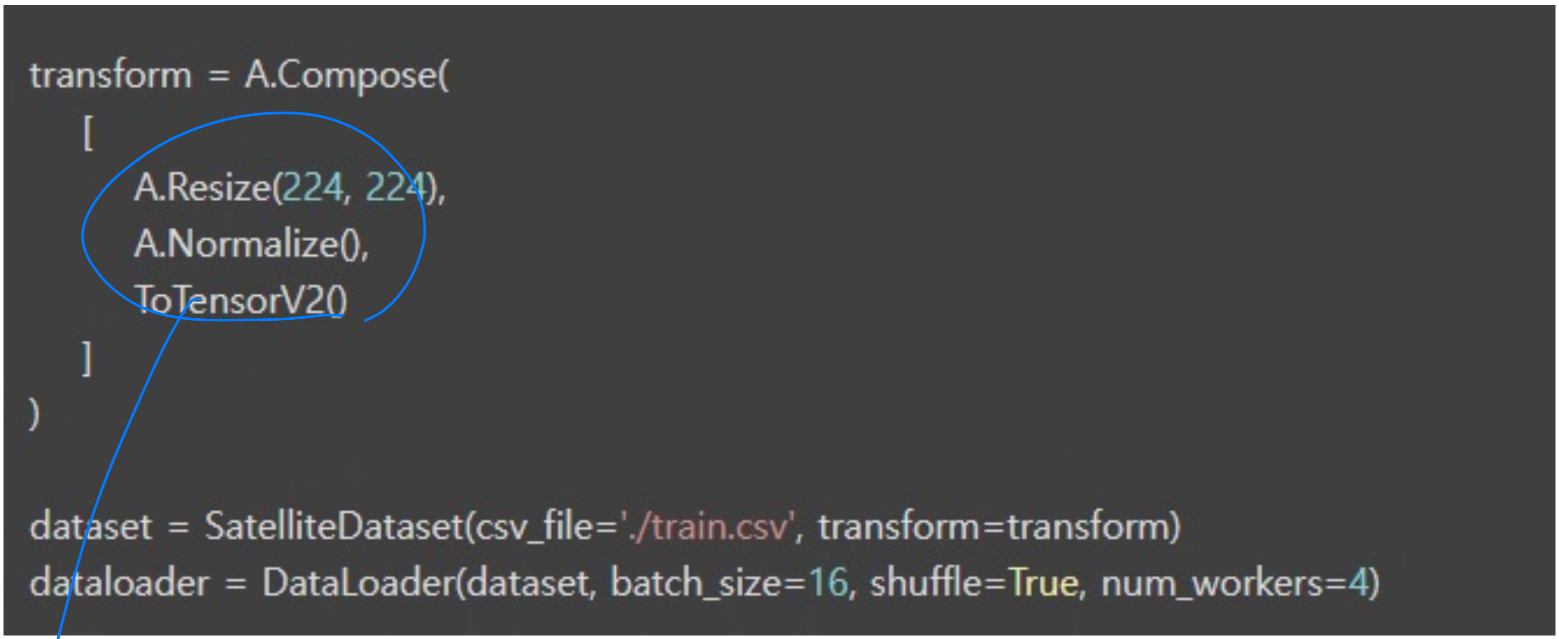


```
transform = A.Compose(
    [
        A.Resize(224, 224),
        A.Normalize(),
        ToTensorV2()
    ]
)

dataset = SatelliteDataset(csv_file='./train.csv', transform=transform)
dataloader = DataLoader(dataset, batch_size=16, shuffle=True, num_workers=4)
```

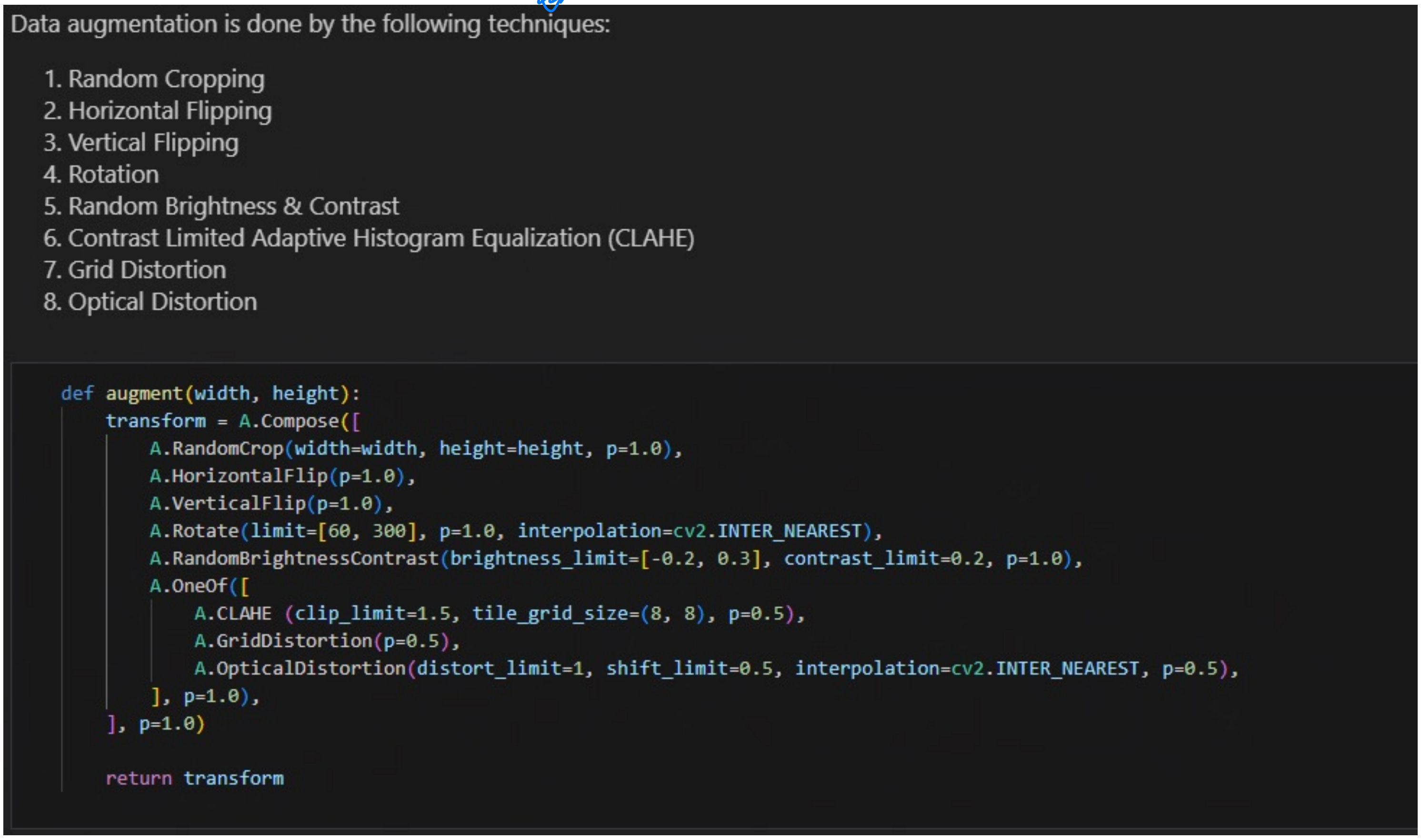
이미지 크기 { train data : 1024×1024
test_data : 224×224

이미지 전처리 작업.

현재 코드 : resize → 표준화 → tensor으로..

train data set이 부족하다면? → 이미지 증강. $\boxed{\text{수정 대상}}$

(Albumentations 라이브러리 사용)

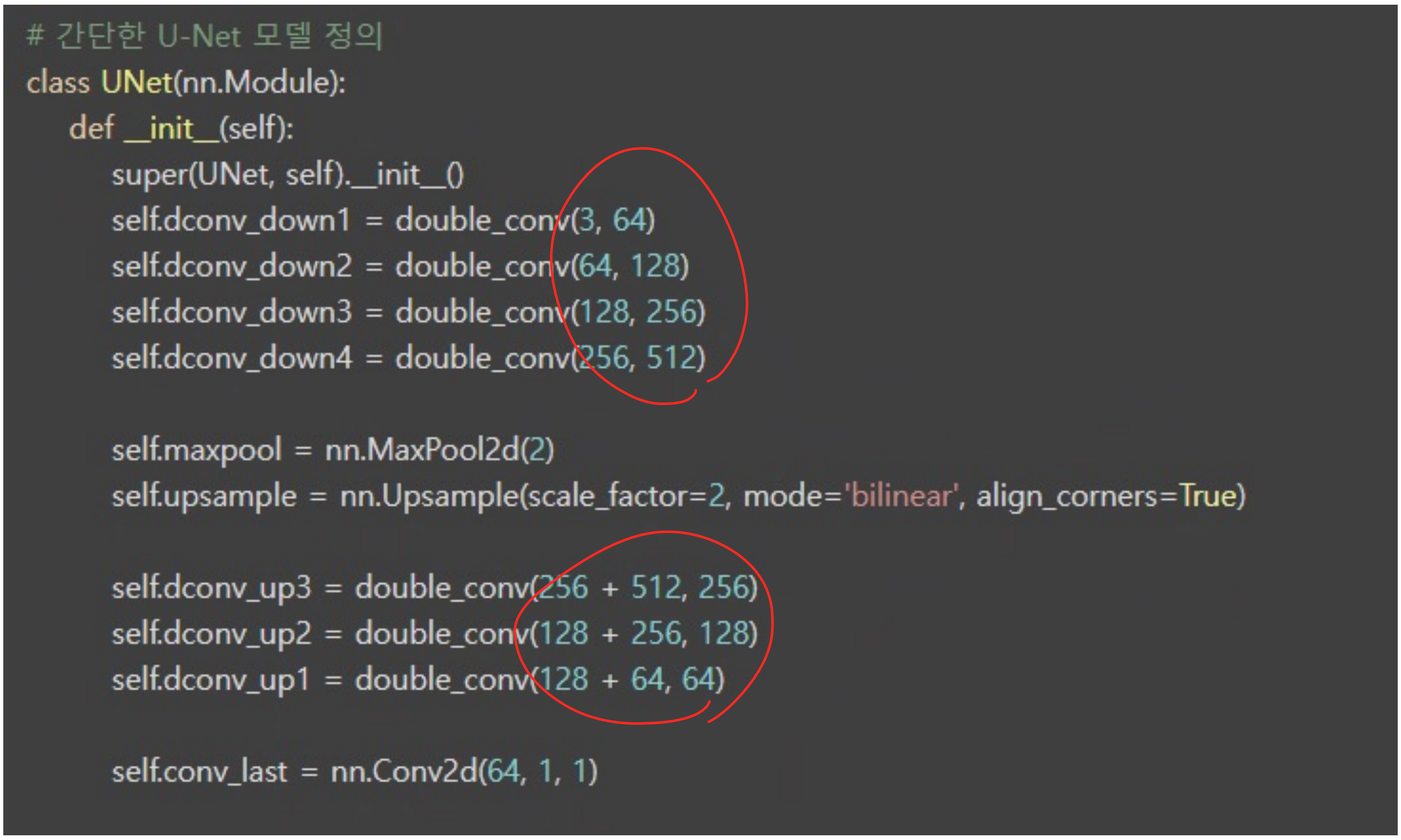


Data augmentation is done by the following techniques:

1. Random Cropping
2. Horizontal Flipping
3. Vertical Flipping
4. Rotation
5. Random Brightness & Contrast
6. Contrast Limited Adaptive Histogram Equalization (CLAHE)
7. Grid Distortion
8. Optical Distortion

```
def augment(width, height):
    transform = A.Compose([
        A.RandomCrop(width=width, height=height, p=1.0),
        A.HorizontalFlip(p=1.0),
        A.VerticalFlip(p=1.0),
        A.Rotate(limit=[60, 300], p=1.0, interpolation=cv2.INTER_NEAREST),
        A.RandomBrightnessContrast(brightness_limit=[-0.2, 0.3], contrast_limit=0.2, p=1.0),
        A.OneOf([
            A.CLAHE (clip_limit=1.5, tile_grid_size=(8, 8), p=0.5),
            A.GridDistortion(p=0.5),
            A.OpticalDistortion(distort_limit=1, shift_limit=0.5, interpolation=cv2.INTER_NEAREST, p=0.5),
        ], p=1.0),
    ], p=1.0)

    return transform
```

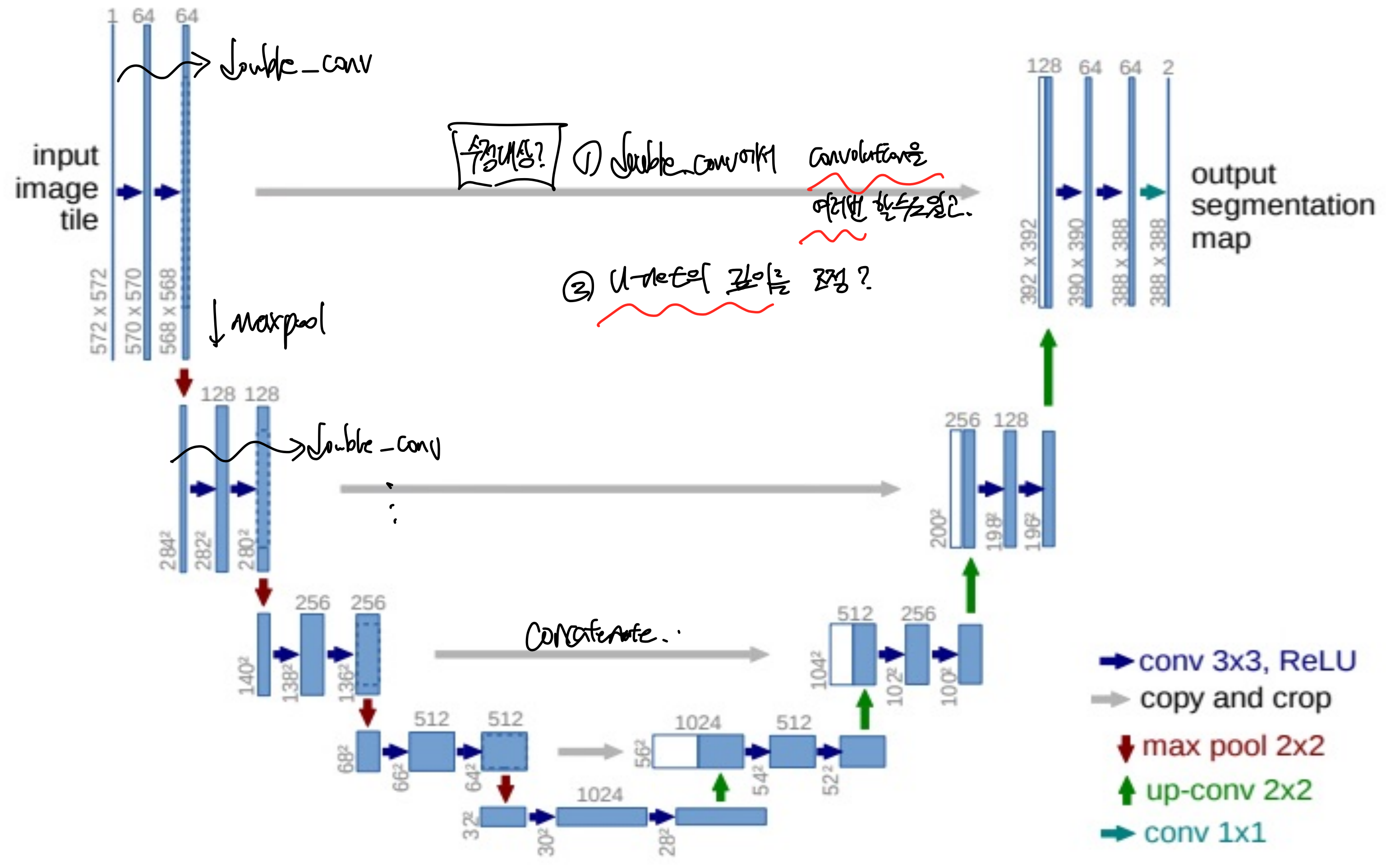
출처: https://github.com/ayushdabra/dubai-satellite-imagery-segmentation



```python
# 간단한 U-Net 모델 정의
class UNet(nn.Module):
    def __init__(self):
        super(UNet, self).__init__()
        self.dconv_down1 = double_conv(3, 64)
        self.dconv_down2 = double_conv(64, 128)
        self.dconv_down3 = double_conv(128, 256)
        self.dconv_down4 = double_conv(256, 512)

        self.maxpool = nn.MaxPool2d(2)
        self.upsample = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)

        self.dconv_up3 = double_conv(256 + 512, 256)
        self.dconv_up2 = double_conv(128 + 256, 128)
        self.dconv_up1 = double_conv(128 + 64, 64)

        self.conv_last = nn.Conv2d(64, 1, 1)
```

double_conv 함수의 인자 : (입력 채널 수, 출력 채널 수)

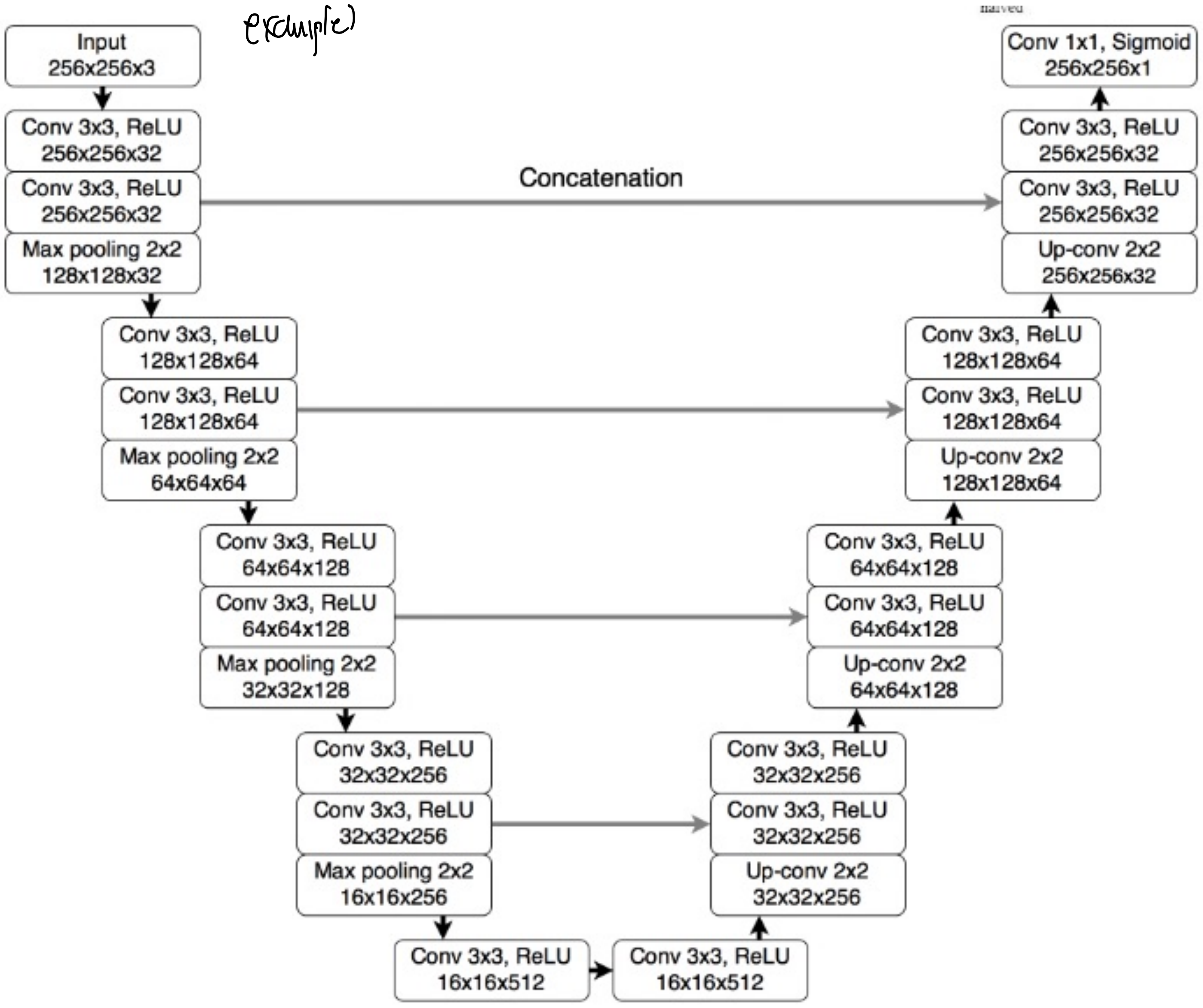
$\boxed{\text{수정대상?}}$ U-net의 인코딩-디코딩 부분을 구현하게 해주는 인자들.

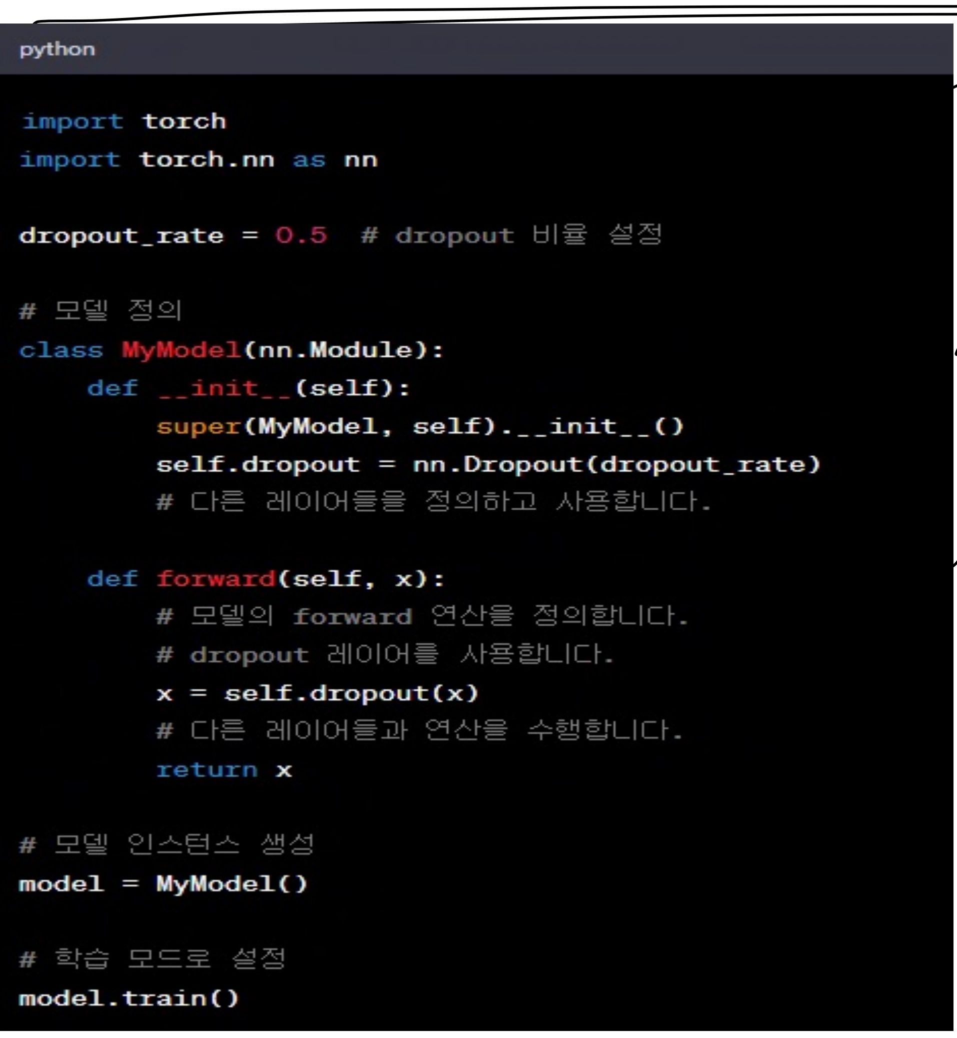


$\boxed{\text{수정대상?}}$ ① double_conv에서 convolution을 여러번 할수도있고.

② U-net의 깊이를 조정?

example)




python

```python
import torch
import torch.nn as nn

dropout_rate = 0.5  # dropout 비율 설정

# 모델 정의
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()
        self.dropout = nn.Dropout(dropout_rate)
        # 다른 레이어들을 정의하고 사용합니다.

    def forward(self, x):
        # 모델의 forward 연산을 정의합니다.
        # dropout 레이어를 사용합니다.
        x = self.dropout(x)
        # 다른 레이어들과 연산을 수행합니다.
        return x

# 모델 인스턴스 생성
model = MyModel()

# 학습 모드로 설정
model.train()
```

→ "Chat GPT"

model.train() : default 0.5로 dropout 되는 듯함

→ ~처럼 구현할 수도.

dropout 비율에 따라 학습 수준이 달라짐

$\boxed{\text{측정 대상}}$

적절한 dropout 비율? 0.2~0.5?

작은 (모델 · data set) → 0.2

큰 (모델 · 복잡 data set) → 0.5.