**Description of code – UNet\_skeleton.py, resnet\_encoder\_unet\_skeleton.py**

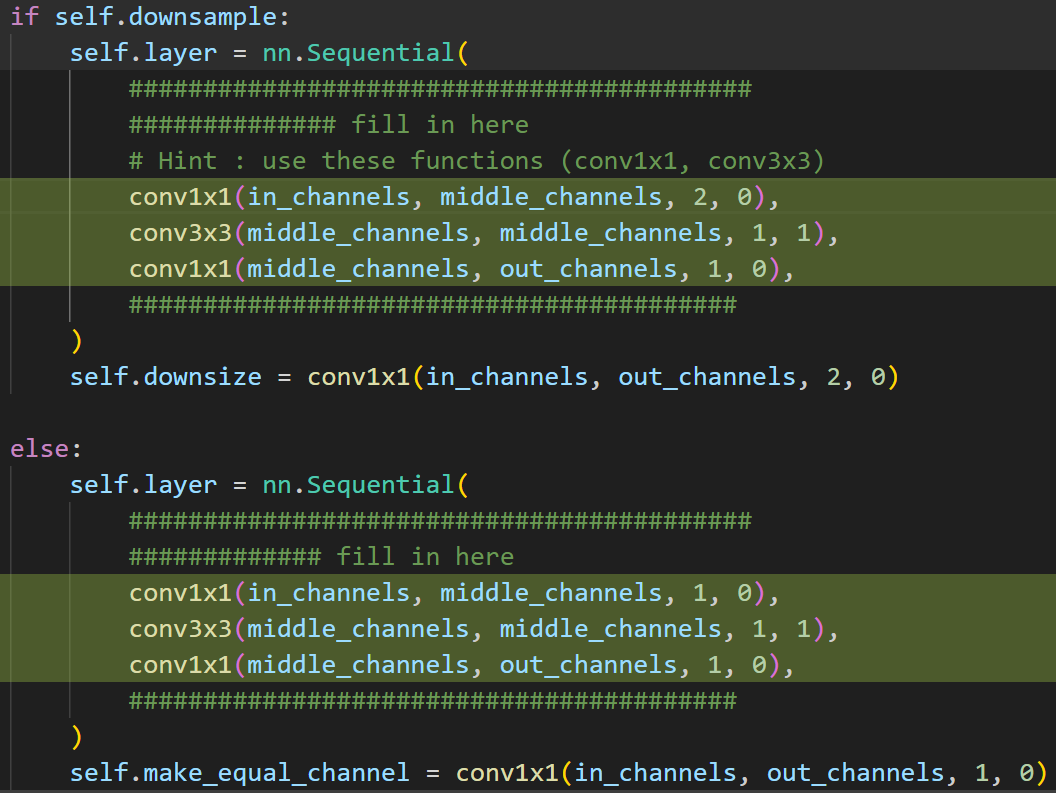
텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

(Above image is captured image of Unet\_skeleton.py with *diff* command)

As required by instruction, I filled in the code. In original U-Net’s encoder part, through four downsampling (max pool) and five convolutional operations, the size of the image was reduced and the size of the channel was increased. And in decoder part, through four times upsampling and four convolutional operations, the size of image was increased and the size of the channel was decreased. And finally, by convolutional opeartion, the channel size become 2. In addition, the result of the contracting path was added to the expanding path of the same level by combining the channel as the axis through concatenation operation.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

(Above image is captured image of resnet\_encoder\_unet\_skeleton.py with *diff* command)

I also filled in this file as required by instruction. First of all, as in project 2, ResNet was implemented, stride and padding were adjusted according to the boolean value of the *downsample*. Also, I coded it as it was while looking at the specifications of layers 2 and 3 required by instruction.

Checking the skeleton code will make for better understandings the structure of ResNet-encoder-Unet. (Overall, it is similar to the original Unet because it has an encoder-decoder structure.)

**Description of code – main\_skeleton.py, modules\_skeleton.py**

This part was written without captured images of the code as a volume(1 page of A4) issue.

In main\_skeleton.py, the model (UNet, ResNet\_encoder\_Unet) was selected to initialize it, and the appropriate checkpoint could be retrieved. In addition, the loss function was set to Cross Entropy Loss, and the optimizer was set to Adam.

In modules\_skeleton.py, There were many utility functions needed in the project (ex. train\_model, accountability\_check, get\_loss\_train, etc.) Among them, the codes of the train\_model, get\_loss\_train, and val\_model functions were filled. In the train\_model, output and loss were obtained, and optimization (update weights) was performed through backpropagation. In get\_loss\_train, the loss in training is calculated. In the val\_model, the validation of the model is in charge. In this step, the accuracy is measured by unused data during training. In addition, there is a part that generate an image composed of single colors to show how the model understands the segmentation.

**Results & Discussions**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명Original U-Net

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명ResNet-encoder-Unet

Original Unet and ResNet\_encoder\_Unet each showed around 75% and 71% accuracy, respectively, in the validation phase. This seems to be a satisfactory figure. For a better accuracy, discussions might be needed on what tasks need to be undertaken, such as deeper models.