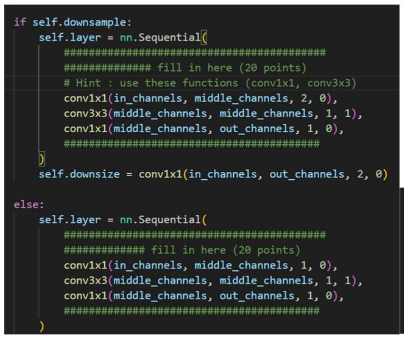
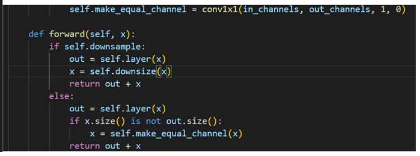
Description of code

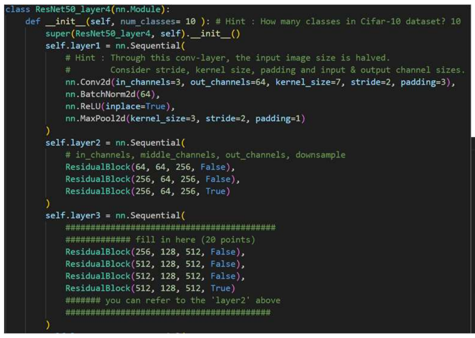


```
if self.downsample:
    self.layer = nn.Sequential(
        ##########################################
        ############# fill in here (20 points)
        # Hint : use these functions (conv1x1, conv3x3)
        conv1x1(in_channels, middle_channels, 2, 0),
        conv3x3(middle_channels, middle_channels, 1, 1),
        conv1x1(middle_channels, out_channels, 1, 0),
        #########################################
    )
    self.downsize = conv1x1(in_channels, out_channels, 2, 0)

else:
    self.layer = nn.Sequential(
        ##########################################
        ############# fill in here (20 points)
        conv1x1(in_channels, middle_channels, 1, 0),
        conv3x3(middle_channels, middle_channels, 1, 1),
        conv1x1(middle_channels, out_channels, 1, 0),
        #########################################
    )
```

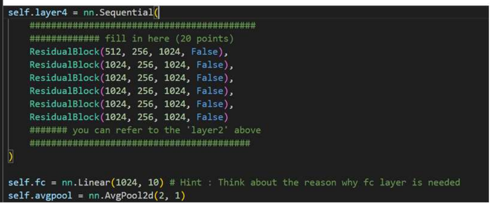


```
            self.make_equal_channel = conv1x1(in_channels, out_channels, 1, 0)

    def forward(self, x):
        if self.downsample:
            out = self.layer(x)
            x = self.downsize(x)
            return out + x
        else:
            out = self.layer(x)
            if x.size() is not out.size():
                x = self.make_equal_channel(x)
            return out + x
```

This code is about Residual blocks. There are two types of resNet's residual blocks: downsampling and normal. When checking the specification pdf 14 page of Project 2(especially layer 2 and 3), downsampling was performed by giving 2 to the stride attribute to the first 1X1 convolution layer. So if the downsample property is true, the first 1x1 convolution layer stride is given as 2. In addition, I decided the stride and padding of 3x3 and second 1x1 convolution layer, focusing on the fact that the size of the input should be maintained. (Give stride and padding as 1, 1 and 1, 0 respectively.) And the part that adds the residual value is in the forward method. (Not my implementation.)



```
class ResNet50_layer4(nn.Module):
    def __init__(self, num_classes= 10 ): # Hint : How many classes in Cifar-10 dataset? 10
        super(ResNet50_layer4, self).__init__()
        self.layer1 = nn.Sequential(
            # Hint : Through this conv-layer, the input image size is halved.
            #        Consider stride, kernel size, padding and input & output channel sizes.
            nn.Conv2d(in_channels=3, out_channels=64, kernel_size=7, stride=2, padding=3),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
        )
        self.layer2 = nn.Sequential(
            # in_channels, middle_channels, out_channels, downsample
            ResidualBlock(64, 64, 256, False),
            ResidualBlock(256, 64, 256, False),
            ResidualBlock(256, 64, 256, True)
        )
        self.layer3 = nn.Sequential(
            ##########################################
            ############# fill in here (20 points)
            ResidualBlock(256, 128, 512, False),
            ResidualBlock(512, 128, 512, False),
            ResidualBlock(512, 128, 512, False),
            ResidualBlock(512, 128, 512, True)
            ####### you can refer to the 'layer2' above
            #########################################
        )
```



```
self.layer4 = nn.Sequential(
    ##########################################
    ############# fill in here (20 points)
    ResidualBlock(512, 256, 1024, False),
    ResidualBlock(1024, 256, 1024, False),
    ResidualBlock(1024, 256, 1024, False),
    ResidualBlock(1024, 256, 1024, False),
    ResidualBlock(1024, 256, 1024, False),
    ResidualBlock(1024, 256, 1024, False)
    ####### you can refer to the 'layer2' above
    #########################################
)

self.fc = nn.Linear(1024, 10) # Hint : Think about the reason why fc layer is needed
self.avgpool = nn.AvgPool2d(2, 1)
```

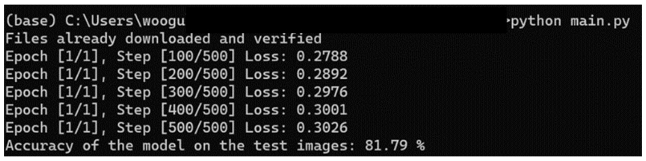
This code is about Resnet50 Class. And I implemented init method. First, since the CIFAR-10 dataset has 10 classes, num_classes was set to 10. Layer 1, 2, 3, and 4 were all made by referring to the project2 specification 14 page.

In layer1, the stride and padding of the 7x7 convolution layer and the max pool layer could be determined based on the fact that layer1's output image size is 8x8 though the input(CIFAR-10 dataset) size is 32x32. And also, already the comments mentioned that the image size is halved in convolution layer, so the stride and padding of the conv layer were set to 2 and 3, respectively, and in the case of the max pool layer, it was set to 2 and 1.

In layers 2, 3, and 4, it was implemented with caution in downsampling (if the stride of the first 1x1 convolution layer in block is 2). It was implemented using a pre-made ResidualBlock class, and blocks were stacked, noting that the output_channel of the previous layer becomes the input_channel of the current layer.

And using the average pool, the 2x2 image size should be made 1x1. (called Global Average Pooling) In addition, the FC Layer connected 1024,10 because 10 classes must be explained through 1024 channels.

Results & Discussions



```
(base) C:\Users\woogu[redacted]>python main.py
Files already downloaded and verified
Epoch [1/1], Step [100/500] Loss: 0.2788
Epoch [1/1], Step [200/500] Loss: 0.2892
Epoch [1/1], Step [300/500] Loss: 0.2976
Epoch [1/1], Step [400/500] Loss: 0.3001
Epoch [1/1], Step [500/500] Loss: 0.3026
Accuracy of the model on the test images: 81.79 %
```

Checking the results, we can see that the accuracy is about 81%, which is quite satisfying the specification. But as for the loss going up, It can be thought that the loss fluctuates while learning has been completed to some extent because we've already done 285 epochs of learning.