Open Software Project

Assignment 10 Technical Report

1976235 오진솔

1. VGG

```
C:\Users\ohjs2\Anaconda3\envs\PyTorch_env\python.exe D:/오진솔/2021/오픈SW프로젝트/Lab/Lab14/main.py
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ../osproj/data/cifar-10-python.tar.gz
100.0%
Extracting ../osproj/data/cifar-10-python.tar.gz to ../osproj/data/
Epoch [1/1], Step [100/500] Loss: 0.1927
Epoch [1/1], Step [200/500] Loss: 0.1812
Epoch [1/1], Step [300/500] Loss: 0.1887
Epoch [1/1], Step [400/500] Loss: 0.1876
Epoch [1/1], Step [500/500] Loss: 0.1881
Accuracy of the model on the test images: 85.21 %

Process finished with exit code 0
```

accuracy : 85.21%

2. ResNet50

```
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)
```

1X1, 3X3, 1X1 conv 를 순서대로 적용하여 bottleneck building block 을 만든다.
stride = 2 인 경우에는 이미지의 크기가 절반으로 줄어들기 때문에 downsample = true 이고
stride = 1 인 경우에는 이미지의 크기가 바뀌지 않게 때문에 downsample = false 이다.

```
self.layer1 = nn.Sequential(
    nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3),
        # Hint : Through this conv-layer, the input image size is halved.
        #       Consider stride, kernel size, padding and input & output
channel sizes.
    nn.BatchNorm2d(64),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
)
```

Layer 1

7X7 conv, channel = 64, stride = 2

3X3 max pool, stride = 2 를 순서대로 적용한다.

이미지의 크기는 32X32 에서 8X8 로 변화하고

채널 개수는 3 개에서 64 개로 변화한다.

```
self.layer2 = nn.Sequential(
    ResidualBlock(in_channels=64, middle_channels = 64, out_channels = 256,
                                                        downsample = False),
    ResidualBlock(in_channels=256, middle_channels = 64, out_channels = 256,
                                                        downsample = False),
    ResidualBlock(in_channels=256, middle_channels=64, out_channels=256,
                                                        downsample = True)
)
```

Layer 2

이미지의 크기는 8X8 에서 4X4 로 변화하고

채널 개수는 64 개에서 256 개로 변화한다.

```
self.layer3 = nn.Sequential(
    ########################################
    ############ fill in here (20 points)
    ####### you can refer to the 'layer2' above
    ResidualBlock(in_channels=256, middle_channels=128, out_channels=512,
                                                        downsample=False),
    ResidualBlock(in_channels=512, middle_channels=128, out_channels=512,
                                                        downsample=False),
    ResidualBlock(in_channels=512, middle_channels=128, out_channels=512,
                                                        downsample=False),
    ResidualBlock(in_channels=512, middle_channels=128, out_channels=512,
                                                         downsample=True)
    ########################################
)
```

Layer 3

이미지의 크기는 4X4 에서 2X2 로 변화하고

채널 개수는 256 개에서 512 개로 변화한다.

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

Layer 4

이미지의 크기는 변화가 없고

채널 개수는 512 개에서 1024 개로 변화한다.

```
self.fc = nn.Linear(1024, 10)
self.avgpool = nn.AvgPool2d(2, 2)
```

fc 부분 잘 모르겠습니다.

그러나 (1024, 10) 외의 값을 넣었을 때는 전부 오류가 뜨고,

size 를 출력해 보았을 때 [100,10]이 나오는 것을 보아 결론적으로 크기가 1000 이 되는 것 같아서 일단 이렇게 작성했습니다.

2*2, stride 2 average pooling 을 사용해 크기를 줄인다.

result

```
C:\Users\ohjs2\Anaconda3\envs\PyTorch_env\python.exe "C:\Program Files\JetBrains\PyCharm Comm
Connected to pydev debugger (build 211.7142.13)
Epoch [1/1], Step [100/500] Loss: 0.2745
Epoch [1/1], Step [200/500] Loss: 0.2827
Epoch [1/1], Step [300/500] Loss: 0.2903
Epoch [1/1], Step [400/500] Loss: 0.2963
Epoch [1/1], Step [500/500] Loss: 0.3011
Accuracy of the model on the test images: 82.7 %

Process finished with exit code 0
```

accuracy : 82.7%