Deep Learning - 01
Project #2 (Resnet50)

2018170754 유인혁

The description of code

```
########################################
############## fill in here (20 points)
# Hint : use these functions (conv1x1, conv3x3)
conv1x1(in_channels, middle_channels, stride=2, padding=0),
conv3x3(middle_channels, middle_channels, stride=1, padding=1)
conv1x1(middle_channels, out_channels, stride=1, padding=0)
########################################
```

```
conv1x1(in_channels, middle_channels, stride=1, padding=0),
conv3x3(middle_channels, middle_channels, stride=1, padding=1),
conv1x1(middle_channels, out_channels, stride=1, padding=0)
```

```
self.layer1 = nn.Sequential(
    nn.Conv2d(in_channels=3 , out_channels=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)
```

```
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)
```

If downsample=True, following code is executed.
The code stride=2 makes output size halve.
In the formula $W_{out} = (W_{in} + 2P - F)/S + 1$,
the padding=1 of the 3x3 convolution to keep the size.
On the other hand, if downsample=False, stride=1.
Cifar-10 image enter layer1 as 32*32*3 and is converted to 8*8*64 through 7*7 conv and max pooling.
Layer2, Layer3, and Layer4 have a similar structure, but only the number and variables of Residual Block are different.
After passing through layer1, the image that was 8*8*64 goes through one ResidualBlock to become 8*8*256, and 256 goes into number of in_channels. Subsequent parameters of ResidualBlocks are applied similarly.

Results

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to
170499072/? [00:11<00:00, 17449555.64it/s]
Extracting ../osproj/data/cifar-10-python.tar.gz to ../osproj/data/
Epoch [1/1], Step [100/500] Loss: 0.1847
Epoch [1/1], Step [200/500] Loss: 0.1796
Epoch [1/1], Step [300/500] Loss: 0.1841
Epoch [1/1], Step [400/500] Loss: 0.1862
Epoch [1/1], Step [500/500] Loss: 0.1881
Accuracy of the model on the test images: 86.44 %
```

#VGG (86.44%) train

```
Epoch [1/1], Step [100/500] Loss: 0.2940
Epoch [1/1], Step [200/500] Loss: 0.2839
Epoch [1/1], Step [300/500] Loss: 0.2930
Epoch [1/1], Step [400/500] Loss: 0.2954
Epoch [1/1], Step [500/500] Loss: 0.2991
Accuracy of the model on the test images: 83.03 %
```

#Resnet (83.03%) train

```
Epoch [1/1], Step [100/100] Loss: 0.5869
Accuracy of the model on the test images: 93.326 %
```

VGG (93.326%) test

```
Epoch [1/1], Step [100/100] Loss: 0.5701
Accuracy of the model on the test images: 90.358 %
```

Resnet (90.358%) test

Discussions

When I trained the neural network of the cifar-10 dataset using Resnet50, I got a good understanding of how the CNN was structured. Although Resnet has a deeper structure than VGG, we found that stability was achieved by minimizing the residuals. VGG seems to perform better than Resnet, but I think that adding pooling to Resnet will reduce the number of parameters in Resnet and improve performance.