KW_VIP 2016707015 신승현

과제목표

이번 과제는 내가 쌓은 모델과 resnet18을 전이학습으로 학습하였을 때 나오는 Loss와 Accuracy를 비교하는 과제이다.

구현 방법

```
class Strategy1(nn.Module):
    def __init__(self, num_classes=2):
        super(Strategy1, self).__init__()
        self.model = models.resnet18(pretrained=True)
        model_fc = self.model.fc.in_features
        self.model.fc = nn.Linear(model_fc, num_classes)

    def forward(self, x):
        return self.model(x)
```

```
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=512, out_features=1000, bias=True)
```

Resnet18의 경우 마지막 fully connected layer의 output이 1000개이고, 우리의 데이터셋은 2개의 class를 분류하는 것이므로 output을 2개로 바꿔주었다.

또한 Strategy1의 경우 전체 파라미터를 학습하는 것 이기 때문에 freeze를 하지 않았다.

```
C:\anaconda\envs\KW_VIP\python.exe C:/Users/신승현/PycharmProjects/pythonProject3/asdf.py
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pth" to C:\Users\신승
100.0%
train Loss: 0.5971 Acc: 0.7008
val Loss: 0.3773 Acc: 0.8954
train Loss: 0.4908 Acc: 0.7869
val Loss: 0.3250 Acc: 0.8954
train Loss: 0.4423 Acc: 0.7869
val Loss: 0.2573 Acc: 0.9412
train Loss: 0.3646 Acc: 0.8566
val Loss: 0.2346 Acc: 0.9346
train Loss: 0.3924 Acc: 0.8074
val Loss: 0.2317 Acc: 0.9281
train Loss: 0.4029 Acc: 0.8197
val Loss: 0.2131 Acc: 0.9346
train Loss: 0.2969 Acc: 0.8852
val Loss: 0.1879 Acc: 0.9412
train Loss: 0.3631 Acc: 0.8115
val Loss: 0.1855 Acc: 0.9346
train Loss: 0.3205 Acc: 0.8730
val Loss: 0.2158 Acc: 0.9150
train Loss: 0.3121 Acc: 0.8730
val Loss: 0.2049 Acc: 0.9412

Process finished with exit code 0
```

그 결과 다음과 같이 나왔다.

```python
class Strategy3(nn.Module):
    def __init__(self, num_classes=2):
        super(Strategy3, self).__init__()
        self.model = models.resnet18(pretrained=True)

        for param in self.model.parameters():
            param.requires_grad = False

        model_fc = self.model.fc.in_features
        self.model.fc = nn.Linear(model_fc, num_classes)

    def forward(self, x):
        return self.model(x)
```

Strategy3의 경우 classifier만 새로 학습하도록 모든 파라미터를 requires_grad = False로 하고

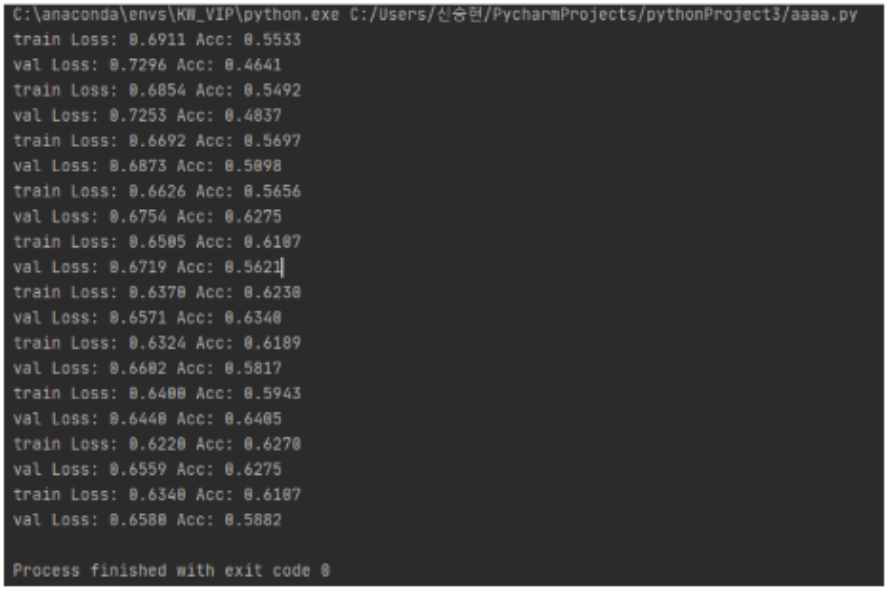
Fully connected layer를 변환해 주었다. 새로 변환될 경우 자동으로 requires_grad = True가 된다.

```
C:\anaconda\envs\KW_VIP\python.exe C:/Users/신승헌/PycharmProjects/pythonProject3/zxcv.py
train Loss: 0.6943 Acc: 0.5861
val Loss: 0.5267 Acc: 0.7451
train Loss: 0.5905 Acc: 0.6721
val Loss: 0.4449 Acc: 0.8431
train Loss: 0.5214 Acc: 0.7705
val Loss: 0.3726 Acc: 0.9020
train Loss: 0.5011 Acc: 0.7541
val Loss: 0.3315 Acc: 0.8954
train Loss: 0.5173 Acc: 0.7377
val Loss: 0.3154 Acc: 0.9281
train Loss: 0.4597 Acc: 0.7664
val Loss: 0.3142 Acc: 0.8954
train Loss: 0.4789 Acc: 0.7910
val Loss: 0.2880 Acc: 0.9085
train Loss: 0.4211 Acc: 0.8197
val Loss: 0.2678 Acc: 0.9150
train Loss: 0.4467 Acc: 0.7869
val Loss: 0.2497 Acc: 0.9216
train Loss: 0.4034 Acc: 0.8279
val Loss: 0.2346 Acc: 0.9412

Process finished with exit code 0
```

그 결과 다음과 같이 나왔다.

내가 쌓은 모델의 경우 Assignment3에 있다.



결과는 다음과 같다.

Epoch	My network		Strategy1		Strategy3	
	Val loss	Acc	Val loss	Acc	Val loss	Acc
1	0.7296	0.4641	0.3773	0.8954	0.5267	0.7451
2	0.7253	0.4837	0.3250	0.8954	0.4449	0.8431
3	0.6873	0.5098	0.2573	0.9412	0.3726	0.9020
4	0.6754	0.6275	0.2346	0.9346	0.3315	0.8954
5	0.6719	0.5621	0.2317	0.9281	0.3154	0.9281
6	0.6571	0.6340	0.2131	0.9346	0.3142	0.8954
7	0.6602	0.5817	0.1879	0.9412	0.2880	0.9085
8	0.6440	0.6405	0.1855	0.9346	0.2678	0.9150
9	0.6559	0.6275	0.2158	0.9150	0.2497	0.9216
10	0.6588	0.5882	0.2049	0.9412	0.2346	0.9412

확실히 학습된 모델을 사용할 경우 학습을 처음 시작할 때부터 정확도가 높은 것을 볼 수 있다.

strategy3의 경우 초반의 정확도가 다소 낮은데 학습이 진행됨에 따라 올라가는 것을 볼 수 있다.

Resnet18의 학습 데이터와 우리의 학습 데이터가 매우 비슷하다는 것을 볼 수 있다.