과목명	KW_VIP	-	-	담당교수	심동규 교수님
학과	전자통신공학과	학번	2017707066	이름	이성민
과제명: Assignment4					

1. 과제설명

Transfer Learning을 이용해서 hymenoptera의 데이터셋의 loss값과 정확도를 출력하는 모델을 구성합니다.

finetuning을 하는데 이미 학습이 되어 있는 resnet18을 이용하여 Strategy1과3 기법으로 학습을 시킨 후 test 결과에 대한 loss값과 Acc 값을 도출해 내고, Assignment3에서 스스로 만든 CNN과 비교합니다.

2. 주요소스코드 설명

```
model = torchvision.models.resnet18(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
```

미리 학습되어 있는 resnet18을 가져옵니다.

model.parameters()의 param.requires_grad는 기본 값이 requires_grad=True로 되어 있기 때문에 false로 바꿔줌으로써 convoloution base 층의 Conv 계층을 모두 freeze 시킵니다.

```
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 2)
```

현재 학습시키고자 하는 카테고리 결과가 2가지 class로 구성되어 있기 때문에 Classifier 설정 부분의 값을 2로 바꿔야 합니다.

model.fc.in_features는 CNN에서 feature extraction 후 fully connected 레이어에 입력되는 데이터 수입니다.

```
optimizer = optim.SGD(model.fc.parameters(), lr=0.001, momentum=0.9)
```

가중치에 대한 최적화는 classifier의 fully connected 부분만 설정합니다.

학습률은 0.001, momentum은 0.9로 설정했습니다.

3. 실행결과 및 설명

Strategy 1에 대한 결과

```
C:\SPB_Data\.conda\envs\kwkw_vip
Epoch 0/14
----------
train Loss: 1.8526 Acc: 0.5820
val Loss: 0.7617 Acc: 0.7778
Epoch 1/14
----------
train Loss: 0.7305 Acc: 0.7172
val Loss: 0.8656 Acc: 0.7255
Epoch 2/14
----------
train Loss: 0.6185 Acc: 0.7746
val Loss: 0.3514 Acc: 0.8758
Epoch 3/14
----------
train Loss: 0.6586 Acc: 0.7705
val Loss: 0.4104 Acc: 0.8431
Epoch 4/14
----------
train Loss: 0.5621 Acc: 0.8074
val Loss: 0.4133 Acc: 0.8366
```

Strategy 3에 대한 결과

```
C:\SPB_Data\.conda\envs\kwkw_vip\pyt
Epoch 0/14
----------
train Loss: 0.6612 Acc: 0.6270
val Loss: 0.2046 Acc: 0.9346
Epoch 1/14
----------
train Loss: 0.4587 Acc: 0.7705
val Loss: 0.1926 Acc: 0.9412
Epoch 2/14
----------
train Loss: 0.4425 Acc: 0.8320
val Loss: 0.2327 Acc: 0.8954
Epoch 3/14
----------
train Loss: 0.5919 Acc: 0.7910
val Loss: 0.1829 Acc: 0.9412
Epoch 4/14
----------
train Loss: 0.6416 Acc: 0.7623
val Loss: 0.1922 Acc: 0.9281
Epoch 5/14
----------
train Loss: 0.7125 Acc: 0.7377
val Loss: 0.2768 Acc: 0.8889
Epoch 6/14
----------
train Loss: 0.5068 Acc: 0.7787
val Loss: 0.3057 Acc: 0.8824
```

```
Epoch 11/14
----------
train Loss: 0.5422 Acc: 0.7623
val Loss: 0.3807 Acc: 0.8758
Epoch 12/14
----------
train Loss: 0.4165 Acc: 0.8484
val Loss: 0.4649 Acc: 0.8366
Epoch 13/14
----------
train Loss: 0.3038 Acc: 0.8852
val Loss: 0.3918 Acc: 0.8824
Epoch 14/14
----------
train Loss: 0.4969 Acc: 0.8115
val Loss: 0.3591 Acc: 0.8758
```

```
Epoch 8/14
----------
train Loss: 0.5138 Acc: 0.7869
val Loss: 0.2903 Acc: 0.9020
Epoch 9/14
----------
train Loss: 0.5219 Acc: 0.7951
val Loss: 0.2230 Acc: 0.9346
Epoch 10/14
----------
train Loss: 0.4876 Acc: 0.8074
val Loss: 0.1943 Acc: 0.9477
Epoch 11/14
----------
train Loss: 0.4083 Acc: 0.8443
val Loss: 0.1977 Acc: 0.9281
Epoch 12/14
----------
train Loss: 0.5266 Acc: 0.8074
val Loss: 0.2497 Acc: 0.9216
Epoch 13/14
----------
train Loss: 0.4398 Acc: 0.8320
val Loss: 0.1986 Acc: 0.9346
Epoch 14/14
----------
train Loss: 0.4254 Acc: 0.8238
val Loss: 0.1838 Acc: 0.9477
```

전체 계층을 모두 수정하는 Strategy1보다는 Classifier을 제외하고 나머지를 freeze시켜서 fine tuning을 하는 Strategy3가 같은 epoch 횟수로 학습시켰을 때 정확도가 더 높은 것을 확인할 수 있습니다.
Strategy1 같은 경우는 전 계층을 수정하기 때문에 dateset의 양이 많아야 하지만 현재 실습에서의 hymenoptera 데이터가 충분하지 않아서 Strategy1 기법에 대한 확실한 효과를 못느낄 수 있다고 생각했습니다.

모델비교

epoch	본인 네트워크		Strategy 1		Strategy 3	
	Val loss	Acc	Val loss	Acc	Val loss	Acc
1	0.6969	0.4575	0.7617	0.7778	0.2046	0.9346
2	0.6841	0.6078	0.8656	0.7255	0.1926	0.9412
3	0.6881	0.5098	0.3514	0.8758	0.2327	0.8954
4	0.6737	0.6405	0.4104	0.8431	0.1829	0.9412
5	0.7948	0.4575	0.4133	0.8366	0.1922	0.9281
6	0.6773	0.5163	0.3697	0.8693	0.2768	0.8889
7	0.6943	0.4706	0.3697	0.8693	0.3057	0.8824
8	0.6623	0.5817	0.3705	0.8954	0.1907	0.9412
9	0.6571	0.6144	0.4304	0.8824	0.2903	0.9020
10	0.6613	0.5686	0.8966	0.7582	0.2230	0.9346
11	0.6519	0.6405	0.7300	0.8235	0.1943	0.9477
12	0.6463	0.6144	0.3010	0.8889	0.1977	0.9281
13	0.6163	0.6516	0.3807	0.8758	0.2497	0.9216
14	0.6796	0.5948	0.4649	0.8366	0.1986	0.9346
15	0.6444	0.6471	0.3918	0.8824	0.1838	0.9477