Assignment#4

2017706106
현민기

1. 개요 - 과제 목표

가. -Fine-tuning

•실제로 충분한 크기의 데이터셋을 갖추기는 상대적으로 드물기 때문에, (무작위 초기화를 통해) 바닥부터(from scratch) 전체 Convolutional Network를 학습 어려움

•매우 큰 데이터셋 (예. 100가지 Category에 대해 120만개의 이미지가 포함된 ImageNet)에서 Convolutional Network를 미리 학습(Pretrain)한 후, 이 Convolutional Network를 관심있는 작업을 위한 초기화, 고정 특징 추출기로 사용

2. 구현 방법

가.

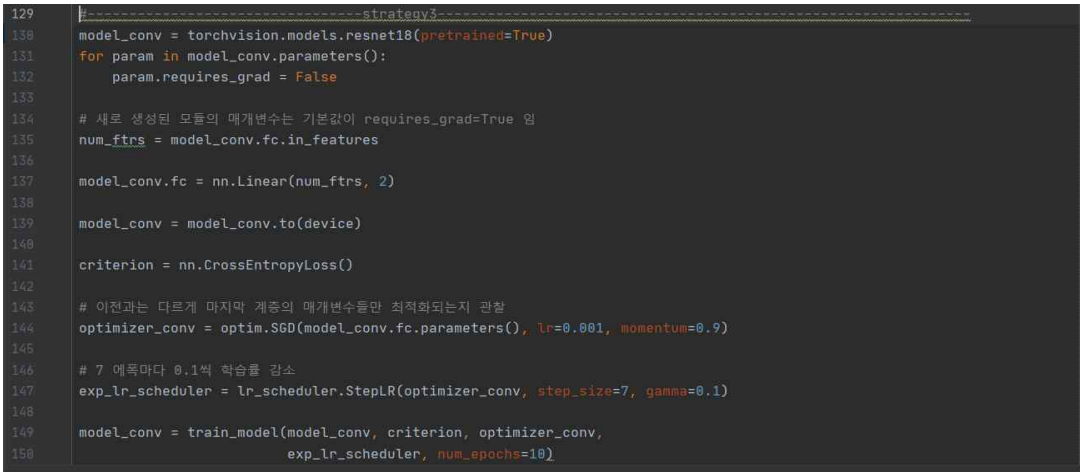
```
#----------------------strategy1-----------------------------------------------------------------------------
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
# 여기서 각 출력 샘플의 크기는 2로 설정합니다.
# 또는, nn.Linear(num_ftrs, len (class_names))로 일반화할 수 있습니다.
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# 모든 매개변수들이 최적화되었는지 관찰
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

# 7 에폭마다 0.1씩 학습률 감소
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)

model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                       num_epochs=10)
"""#---------------------------------strategy3-----------------------------------------------------------------
model_conv = torchvision.models.resnet18(pretrained=True)
```

<strategy 1>;

model_ft = models.resnet18(pretrained=True)
; pretrained 된 resnet18을 불러옴. Freeze를 따로 시키지 않으므로, parameter를 freeze시키는 코드가 없음.

num_ftrs = model_ft.fc.in_features
; input의 개수를 in_features로 정의, resnet18 속의 input 개수는 총 512개이므로, in_features 대신에 512를 넣어도 원만하게 코드 진행 가능. model_ft의 epochs를 10으로 설정하여 총 epoch 0~9까지 출력되게끔 설정.

model_ft.fc=nn.Linear.(num_ftrs, 2)
; Classifier를 정의



```
#--------------------------------strategy3----------------------------------------------------------------------
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# 새로 생성된 모듈의 매개변수는 기본값이 requires_grad=True 임
num_ftrs = model_conv.fc.in_features

model_conv.fc = nn.Linear(num_ftrs, 2)

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

# 이전과는 다르게 마지막 계층의 매개변수들만 최적화되는지 관찰
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

# 7 에폭마다 0.1씩 학습률 감소
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)

model_conv = train_model(model_conv, criterion, optimizer_conv,
                         exp_lr_scheduler, num_epochs=10)
```

<strategy 3>;

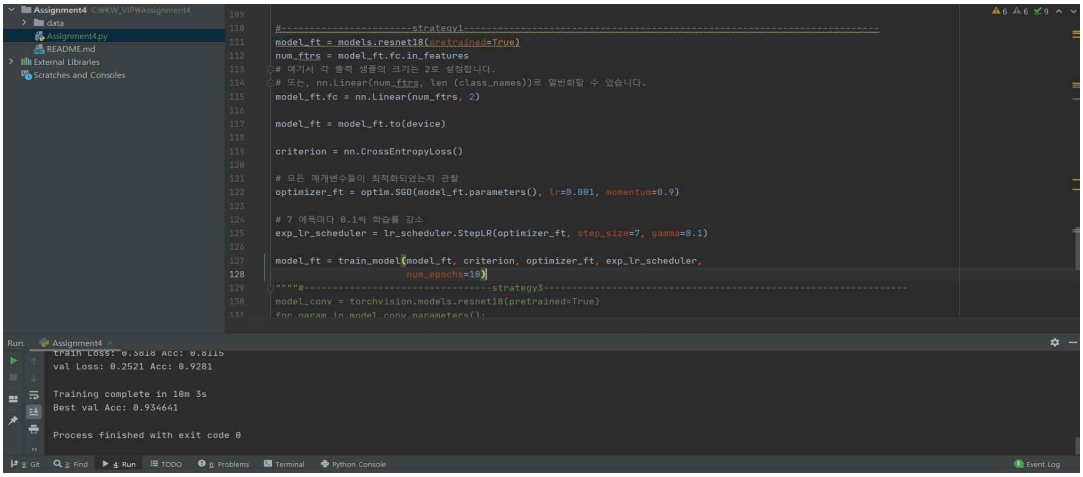
model_conv = models.resnet18(pretrained=True)
; pretrained 된 resnet18을 불러옴. Convolution network part 모두 Freeze 시키므로, parameter를 freeze 시키는
for param in model ~ grad=False 의 구문 존재.

num_ftrs = model_conv.fc.in_features
; input의 개수를 in_features로 정의, resnet18 속의 input 개수는 총 512개이므로, in_features 대신에 512를 넣어도 원만하게 코드 진행 가능. model_ft의 epochs를 10으로 설정하여 총 epoch 0~9까지 출력되게끔 설정.

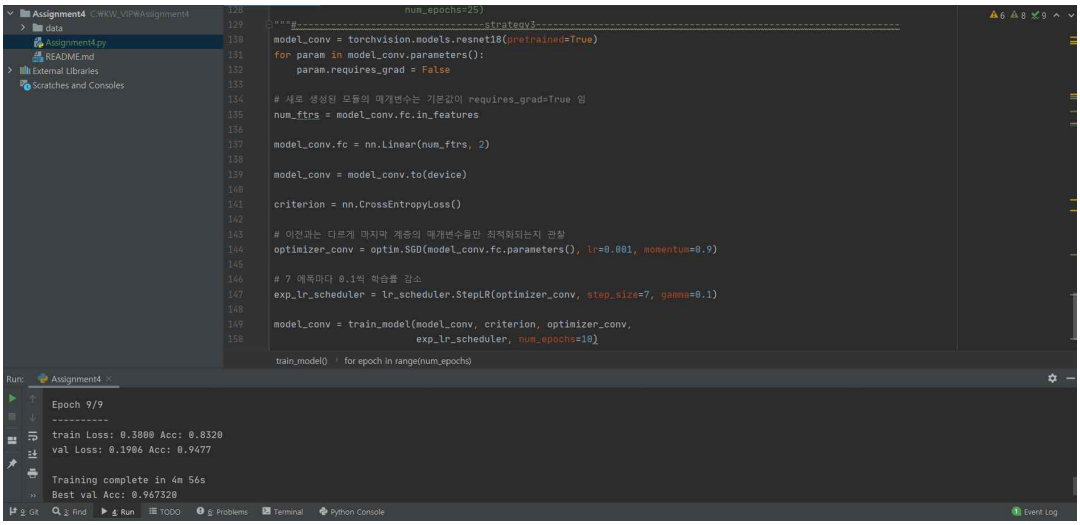
model_conv.fc=nn.Linear.(num_ftrs, 2)
; Classifier를 정의

3. 결과 화면

가. 결과 화면 캡쳐



<strategy 1>



<Strategy 3>

나. 결과표

	본인 네트워크		Strategy 1		Strategy 3	
	Val loss	Acc	Val loss	Acc	Val loss	Acc
0	0.6758	**0.5490**	0.3222	**0.8497**	0.3068	**0.8954**
1	0.6864	**0.5817**	0.2549	**0.9216**	0.2061	**0.9542**
2	0.6861	**0.5882**	0.3729	**0.8562**	0.2202	**0.9216**
3	0.6962	**0.4575**	0.2058	**0.9281**	0.1834	**0.9542**
4	0.7012	**0.4837**	0.2516	**0.9346**	0.3167	**0.8497**
5	0.6859	**0.5163**	0.2885	**0.9020**	0.2069	**0.9412**
6	0.6849	**0.5882**	0.2185	**0.9216**	0.1875	**0.9477**
7	0.6853	**0.5686**	0.2145	**0.9281**	0.1702	**0.9673**
8	0.6875	**0.5098**	0.2122	**0.9281**	0.1745	**0.9608**
9	0.6942	**0.5033**	0.2521	**0.9281**	0.1906	**0.9477**