

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.

```
110 #-----Strategy1-----
111 model_ft = models.resnet18(pretrained=True)
112 num_ftfs = model_ft.fc.in_features
113 # 여기서 각 출력 샘플의 크기는 2로 설정합니다.
114 # 또는, nn.Linear(num_ftfs, len(class_names))로 일반화할 수 있습니다.
115 model_ft.fc = nn.Linear(num_ftfs, 2)
116
117 model_ft = model_ft.to(device)
118
119 criterion = nn.CrossEntropyLoss()
120
121 # 모든 매개변수들이 최적화되었는지 관찰
122 optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)
123
124 # 7 에폭마다 0.1씩 학습률 감소
125 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
126
127 model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
128                        num_epochs=10)
129 #-----Strategy3-----
130 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_ftfs = 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_fts, 2)
; Classifier를 정의
```

```
129 #-----strategy3-----
130 model_conv = torchvision.models.resnet18(pretrained=True)
131 for param in model_conv.parameters():
132     param.requires_grad = False
133
134 # 새로 생성된 모듈의 매개변수는 기본적으로 requires_grad=True 임
135 num_fts = model_conv.fc.in_features
136
137 model_conv.fc = nn.Linear(num_fts, 2)
138
139 model_conv = model_conv.to(device)
140
141 criterion = nn.CrossEntropyLoss()
142
143 # 이전과는 다르게 마지막 계층의 매개변수들만 최적화되는지 관찰
144 optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)
145
146 # 7 에폭마다 0.1씩 학습률 감소
147 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
148
149 model_conv = train_model(model_conv, criterion, optimizer_conv,
150                           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_fts = model_conv.fc.in_features
; input의 개수를 in_features로 정의, resnet18 속의 input 개수는 총
512개이므로, in_features 대신에 512를 넣어도 원만하게 코드 진행 가
능. model_ft의 epochs를 10으로 설정하여 총 epoch 0~9까지 출력되게
끔 설정.
```

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

3. 결과 화면

가. 결과 화면 캡처

```

189 #-----strategy1-----
190 model_ft = models.resnet18(pretrained=True)
191 num_fts = model_ft.fc.in_features
192 # 여기서 각 층의 층들의 크기는 2로 설정합니다.
193 # 또는, nn.Linear(num_fts, len(class_names))로 일반화할 수 있습니다.
194 model_ft.fc = nn.Linear(num_fts, 2)
195
196 model_ft = model_ft.to(device)
197
198 criterion = nn.CrossEntropyLoss()
199
200 # 모든 매개변수들이 최적화되었는지 관찰
201 optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)
202
203 # 7 에폭마다 0.1씩 학습률 감소
204 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
205
206 model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
207                        num_epochs=10)
208
209 #####-----strategy3-----
210 model_conv = torchvision.models.resnet18(pretrained=True)
211 for param in model_conv.parameters():
212     param.requires_grad = False
  
```

Run: Assignment4

```

train Loss: 0.6610 Acc: 0.6115
val Loss: 0.2521 Acc: 0.9281

Training complete in 10m 3s
Best val Acc: 0.934641

Process finished with exit code 0
  
```

<strategy 1>

```

189 #####-----strategy3-----
190 model_conv = torchvision.models.resnet18(pretrained=True)
191 for param in model_conv.parameters():
192     param.requires_grad = False
193
194 # 새로 생성된 모듈의 매개변수는 기본적으로 requires_grad=True 임
195 num_fts = model_conv.fc.in_features
196
197 model_conv.fc = nn.Linear(num_fts, 2)
198
199 model_conv = model_conv.to(device)
200
201 criterion = nn.CrossEntropyLoss()
202
203 # 이전과는 다르게 마지막 계층의 매개변수들만 최적화되도록 관찰
204 optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)
205
206 # 7 에폭마다 0.1씩 학습률 감소
207 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
208
209 model_conv = train_model(model_conv, criterion, optimizer_conv,
210                          exp_lr_scheduler, num_epochs=10)
211
212 train_model() for epoch in range(num_epochs)
  
```

Run: Assignment4

```

Epoch 9/9
-----
train Loss: 0.3900 Acc: 0.8320
val Loss: 0.1986 Acc: 0.9477

Training complete in 4m 56s
Best val Acc: 0.967320
  
```

<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