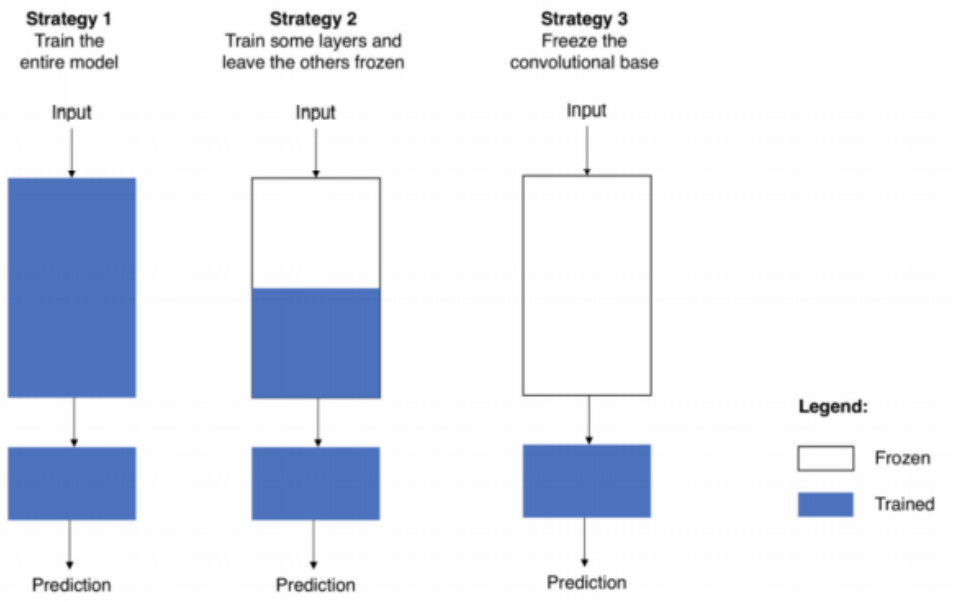
Assignment4

2017204091 심민정

개요

사전에 학습된 모델의 재정의(Fine-tuning)



Fine-tuning의 세 가지 전략

본인 네트워크, Pretrained 된 resnet18 strategy 1 & 3 방법으로 학습한 Val loss, ACC를 비교해보고 training시간도 비교해본다.

구현방법

-본인 네트워크

```
class ConvNet(nn.Module):
    def __init__(self, num_classes=10):
        super(ConvNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer3 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer4 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(14*14*128, num_classes)
```

4개의 conv층을 쌓았다.

-Strategy 1

```
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)
```

미리 학습한 신경망을 초기화 한다.

-strategy 3

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

# 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)
```

마지막에 완전 연결된 계층을 제외한 모든 신경망의 가중치를 고정(freeze) requires_grad==False 하고 마지막 계층만 학습한다.

결과화면

epoch	본인 네트워크		Strategy 1		Strategy 3	
	Val loss	ACC	Val loss	ACC	Val loss	ACC
1	0.6785	0.5425	0.1976	0.9281	0.2501	0.9150
2	0.6727	0.5817	0.2680	0.8758	0.1895	0.9542
3	0.6630	0.5948	0.3146	0.8693	0.2109	0.9477
4	0.6295	0.6340	0.2514	0.8758	0.2119	0.9346
5	0.6378	0.6405	0.2710	0.8824	0.2473	0.9150
6	0.6632	0.6340	0.1825	0.9346	0.1894	0.9346
7	0.6463	0.5948	0.2054	0.9150	0.3255	0.8889
8	0.6380	0.7059	0.2261	0.8824	0.1701	0.9542
9	0.6226	0.7059	0.2373	0.9020	0.1768	0.9542
10	0.5949	0.7124	0.2251	0.9020	0.1978	0.9412

학습시간

본인네트워크: 6m 22s//// Strategy 1: 20m 60s//// Strategy 3: 10m 14s