2017741047_허정범_로봇학부

->개요

imagenet의 일부인 개미와 벌에 관한 카테고리를 자신이 만든 네트워크에 학습시켜서 이미지를 분류하고, train Loss, val Loss, Accuracy를 출력하는 것이 목표입니다.

->구현방법

ConvNet은 Convolutional layer 2개, Fully Conneted layer 2개로 만들었습니다.
epochs 값은 25로 해줘서 25번 반복하게 만들어 줬습니다.

```
num_epochs=25
num_classes=2
batch_size=64
learning_rate=0.001
```

```
class ConvNet(nn.Module):
    def __init__(self, num_classes=2):
        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.fc=nn.Linear(32*56*56, num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc(out)
        return out
```

input 사이즈는 224 이고 MaxPool 2개를 거치면서 1/4 이 된다. 그리고 마지막 output 사이즈가 56이 되게 된다. 그리고 output 체널은 32이기 때문에 32*56*56이 된다.

```python
#데이터 불러오기.
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

데이터를 불러오고

```python
for epoch in range(num_epochs):#에폭수만큼 반복. 25번 반복해서 트레이닝
    print('Epoch {}/{}'.format(epoch, num_epochs - 1))
    print('-' * 10)
    for phase in ['train', 'val']:# 각 에폭은 학습 단계와 검증 단계를 갖습니다.
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0

            # 데이터를 반복
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)

                # 매개변수 경사도를 0으로 설정
            optimizer.zero_grad()

                # 순전파
                # 학습 시에만 연산 기록을 추적
            with torch.set_grad_enabled(phase == 'train'):
                outputs = model(inputs)
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)

                    # 학습단계의 경우 역전파 + 최적화
            if phase == 'train':
                loss.backward()
                optimizer.step()
```

```
Epoch 22/24
----------
train Loss: 0.6195 Acc: 0.6680
val Loss: 0.6944 Acc: 0.5948
Epoch 23/24
----------
train Loss: 0.6773 Acc: 0.5943
val Loss: 0.6976 Acc: 0.5948
Epoch 24/24
----------
train Loss: 0.6387 Acc: 0.6352
val Loss: 0.6484 Acc: 0.6144

Process finished with exit code 0
```