KW_VIP 2016707015 신승현

과제 목표

이번 과제는 hymenoptera_data의 데이터를 이용하여 개미와 벌을 학습하는 것이 목표이다.

구현 방법

```
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.layer3 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(50176, num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = F.dropout(out, training=self.training)
        out = out.reshape(out.size(0),-1)
        out = self.fc(out)
        return out
```

지난 과제인 CIFAR10의 모델에 Conv layer를 하나 더 추가하였고, 245개의 적은 train dataset이기 때문에 dropout도 이용하였다.

```python
for epoch in range(num_epochs):
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0

        for images, labels in dataloaders[phase]:
            images = images.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()

            with torch.set_grad_enabled(phase == 'train'):
                outputs = model(images)
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)

            if phase == 'train':
                loss.backward()
                optimizer.step()

            running_loss += loss.item() * images.size(0)
            running_corrects += torch.sum(preds == labels.data)

        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]

        print('{} Loss: {:.4f} Acc: {:.4f}'.format(
            phase, epoch_loss, epoch_acc))
```

Train의 경우 모델을 학습하고, val의 경우 모델을 평가하도록 하였고

With torch.set_grad_enabled(phase == 'train')을 이용하여 train의 경우에만 gradient를 계산하도록 하였다.

나머지 부분은 Assignment2와 비슷하다.

결과 화면

```
C:\anaconda\envs\KW_VIP\python.exe C:/Users/신승현/PycharmProjects/pythonProject3/aaaa.py
train Loss: 0.6911 Acc: 0.5533
val Loss: 0.7296 Acc: 0.4641
train Loss: 0.6854 Acc: 0.5492
val Loss: 0.7253 Acc: 0.4837
train Loss: 0.6692 Acc: 0.5697
val Loss: 0.6873 Acc: 0.5098
train Loss: 0.6626 Acc: 0.5656
val Loss: 0.6754 Acc: 0.6275
train Loss: 0.6505 Acc: 0.6107
val Loss: 0.6719 Acc: 0.5621
train Loss: 0.6370 Acc: 0.6230
val Loss: 0.6571 Acc: 0.6340
train Loss: 0.6324 Acc: 0.6189
val Loss: 0.6602 Acc: 0.5817
train Loss: 0.6400 Acc: 0.5943
val Loss: 0.6440 Acc: 0.6405
train Loss: 0.6220 Acc: 0.6270
val Loss: 0.6559 Acc: 0.6275
train Loss: 0.6340 Acc: 0.6107
val Loss: 0.6580 Acc: 0.5882

Process finished with exit code 0
```

아무래도 데이터셋이 별로 없다 보니 정확도는 그리 높지 않은 것을 볼 수 있다.