과목명	KW_VIP	-	-	담당교수	심동규 교수님
학과	전자통신공학과	학번	2017707066	이름	이성민
과제명: Assignment3					

1. 과제설명

Transfer Learning을 이용해서 hymenoptera의 데이터셋의 loss값과 정확도를 출력하는 모델을 구성합니다.

2. 주요소스코드 설명

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

train_model의 메소드 안의 코드이며, 학습과 검사를 하나의 코드로 합쳐놓아서 각 epoch마다 Training accuracy와 Test accuracy를 확인할 수 있습니다.
epoch을 크게해서 한 번 실행하고 나면, 적어도 그 epoch 범위 내에서는 최적의 epoch값을 찾을 수 있습니다.

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

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

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()
```

개인의 CNN모델을 사용하라고 했지만, 아직 직접 구현하지 못해서 일단은 resnet18로 이미 학습된 모델을 불러왔습니다.
마지막 계층을 제외한 신경망의 모든 부분을 고정시킵니다.

```
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)
```

Optimizer 설정하는 부분에서 freeze 시키지 않은 layer에 대해서만 최적화를 진행하도록 optimizer 부분에 parameter 명시 필요합니다.

```
model_conv = train_model(model_conv, criterion, optimizer_conv,
                         exp_lr_scheduler, num_epochs=25
```

train_model() 메소드를 사용해서 학습을 시키기고 결과를 출력합니다.

3. 실행결과 및 설명

```
C:\SPB_Data\.conda\envs\kwkw_vip\python.exe C:/Users/이성민/PycharmProjects/pythonProject9/assignment3.py
cpu
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pth" to C:\SPB_Data/.cache\torch\hub\checkpoints\resnet18-5c106cde.pth
100.0%
Epoch 0/24
----------
train Loss: 0.7428 Acc: 0.6230
val Loss: 0.2472 Acc: 0.9281

Epoch 1/24
```

resnet을 사용해서 다운로드를 받습니다.
이 후 24 epoch만큼 잘 반복되는 것을 확인 할 수 있습니다.

```
Epoch 20/24
----------
train Loss: 0.3028 Acc: 0.8811
val Loss: 0.2000 Acc: 0.9412

Epoch 21/24
----------
train Loss: 0.2576 Acc: 0.8852
val Loss: 0.2033 Acc: 0.9412

Epoch 22/24
----------
train Loss: 0.3391 Acc: 0.8443
val Loss: 0.1863 Acc: 0.9542

Epoch 23/24
----------
train Loss: 0.2988 Acc: 0.8689
val Loss: 0.2242 Acc: 0.9477

Epoch 24/24
----------
train Loss: 0.3657 Acc: 0.8156
val Loss: 0.2177 Acc: 0.9477

Training complete in 10m 52s
Best val Acc: 0.954248

Process finished with exit code 0
```

4. 전체 소스코드

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy

plt.ion()

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])
    ]),
}

data_dir = 'hymenoptera_data'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                          data_transforms[x])
                  for x in ['train', 'val']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4,
                                              shuffle=True)
               for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            running_loss = 0.0
            running_corrects = 0

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

```
                optimizer.zero_grad()

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

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

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)
            if phase == 'train':
                scheduler.step()

            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))

            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())

        print()

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    model.load_state_dict(best_model_wts)
    return model


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

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

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

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=25)
```