산업 인공지능 - 실습 8

딥러닝 프로그래밍

2021 Spring

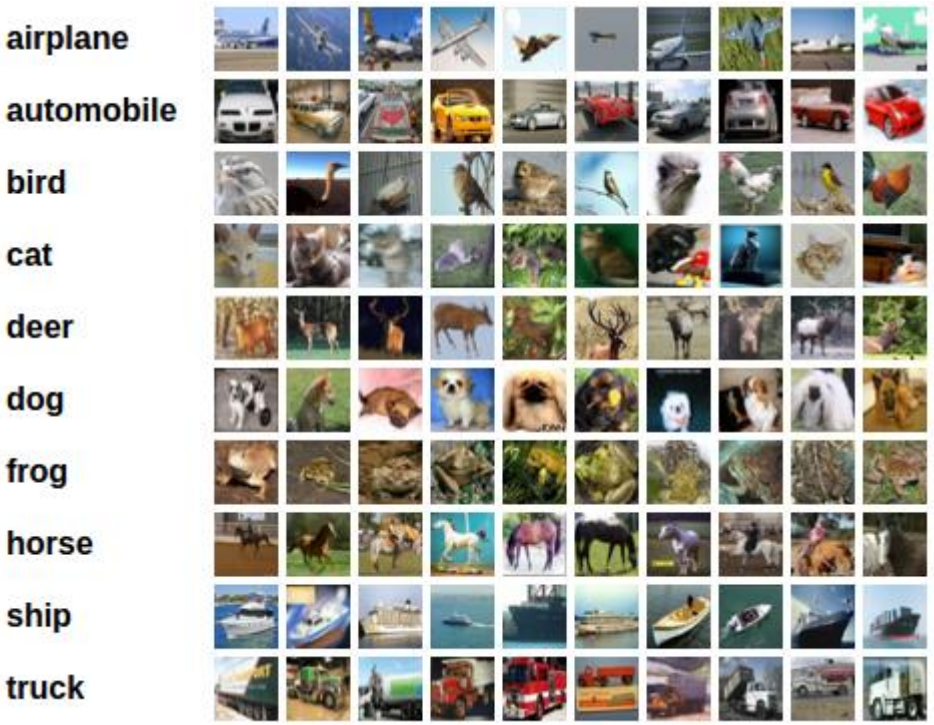
학습 내용

- CNN 모델을 구성하여 CIFAR10 데이터를 학습하도록 하는 프로그램 코드를 살펴본다.
- 학습된 ResNet 모델을 이용한 전이 학습 프로그래밍에 대해서 알아본다.

[실습] CIFAR10 데이터의 인식

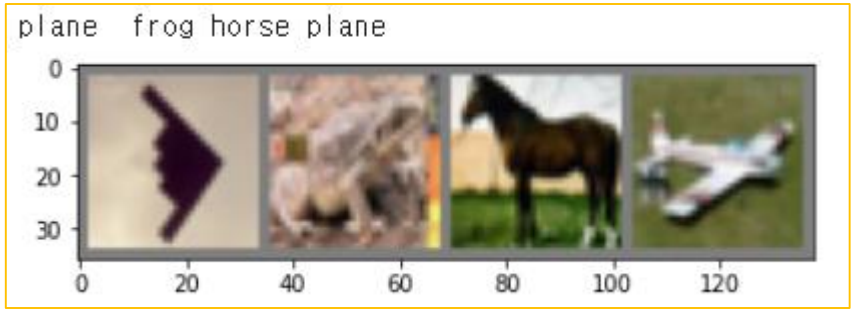
❖ CIFAR 10 데이터

- 3 채널의 32x32 크기 10종의 이미지 데이터
- 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'



❖ Colab에서 PyTorch 사용 실습

```
%matplotlib inline
import torch
import torchvision
import torchvision.transforms as transforms

transform = transforms.Compose(
    [transforms.ToTensor(), transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4, shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

import matplotlib.pyplot as plt
import numpy as np

def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy( )
    plt.imshow(np.transpose(npimg, (1, 2, 0)))

dataiter = iter(trainloader)
images, labels = dataiter.next()

imshow(torchvision.utils.make_grid(images))
print(' '.join('%5s' % classes[labels[j]] for j in range(4)))
```



```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net( )

import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters( ), lr=0.001, momentum=0.9)
```

```
for epoch in range(10):  # 에포크 수

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data   # 학습 데이터
        optimizer.zero_grad( )
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward( )
        optimizer.step( )

        running_loss += loss.item()
        if i % 2000 == 1999:    # 매 2000 mini-batch 별로 출력
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

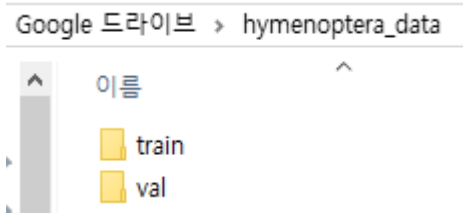
```
[9,  4000] loss: 0.723
[9,  6000] loss: 0.739
[9,  8000] loss: 0.783
[9, 10000] loss: 0.794
[9, 12000] loss: 0.799
[10,  2000] loss: 0.673
[10,  4000] loss: 0.708
[10,  6000] loss: 0.742
[10,  8000] loss: 0.748
[10, 10000] loss: 0.765
[10, 12000] loss: 0.772
Finished Training
```

[실습] ResNet을 이용한 전이 학습

```
%matplotlib inline
from __future__ import print_function, division

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( )   # interactive mode
```

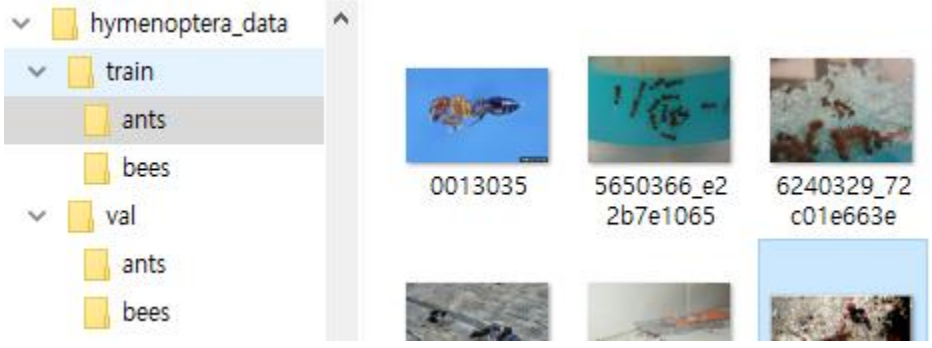
https://download.pytorch.org/tutorial/hymenoptera_data.zip 파일 다운로드
Google Drive의 hymenoptera_data 폴더 만들어 저장

```
from google.colab import drive
drive.mount('/content/drive')
```

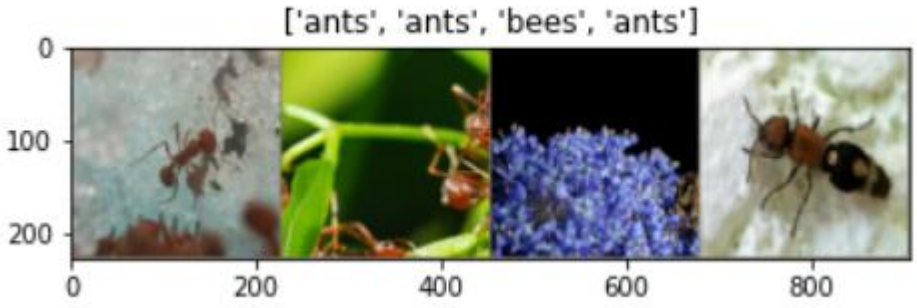


https://colab.research.google.com/github/pytorch/

```
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 = '/content/drive/My Drive/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,
                  num_workers=4) 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")
```



```
def imshow(inp, title=None):
    inp = inp.numpy( ).transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)

inputs, classes = next(iter(dataloaders['train']))
out = torchvision.utils.make_grid(inputs)

imshow(out, title=[class_names[x] for x in classes])
```



```
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':
                scheduler.step( )
                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)
            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
```

```
def visualize_model(model, num_images=6):
    was_training = model.training
    model.eval( )
    images_so_far = 0
    fig = plt.figure( )

    with torch.no_grad( ):
        for i, (inputs, labels) in enumerate(dataloaders['val']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images//2, 2, images_so_far)
                ax.axis('off')
                ax.set_title('predicted: {}'.format(class_names[preds[j]]))
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

```
model_ft = models.resnet18(pretrained=True)   # 사전 학습된 ResNet18 가져오기
num_ftrs = model_ft.fc.in_features  # 모델에서 feature extraction 후 FC 층에 입력되는 특징수
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)  # cpu나 GPU에 model_ft를 할당
criterion = nn.CrossEntropyLoss( )

# 모든 파라미터를 학습
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

#  매 7 에포크 마다 학습율 0.1배 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=25)
```

```
Epoch 1/24
----------
train Loss: 0.4804 Acc: 0.8484
val Loss: 0.2399 Acc: 0.9150
Epoch 2/24
----------
train Loss: 0.4829 Acc: 0.7992
val Loss: 0.3913 Acc: 0.8497
    :

Epoch 24/24
----------
train Loss: 0.3148 Acc: 0.8525
val Loss: 0.2276 Acc: 0.9346

Training complete in 34m 15s
Best val Acc: 0.941176
```

visualize_model(model_ft)

predicted: bees

predicted: bees

predicted: bees

predicted: ants

predicted: bees

predicted: ants

```
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters( ):
    param.requires_grad = False    # 사전 학습된 모델의 가중치를 상수로 고정. 학습시키지 않음

# 새로 생성된 모듈의 파라미터는 기본적으로 requires_grad=True
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)

model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()

# 마지막 층의 파라미터만 학습
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)
```

```
  :
Epoch 23/24
----------
train Loss: 0.3178 Acc: 0.8648
val Loss: 0.1526 Acc: 0.9477

Epoch 24/24
----------
train Loss: 0.3728 Acc: 0.8361
val Loss: 0.1544 Acc: 0.9412

Training complete in 15m 9s
Best val Acc: 0.954248
```

```
visualize_model(model_conv)

plt.ioff( )
plt.show( )
```

predicted: bees

predicted: ants

predicted: bees

predicted: bees

predicted: ants

predicted: bees

실습

1. Colab을 사용하여 AlexNet과 ResNet에 대한 전이 학습를 하는 실습을 해보시오.
2. PyTorch에서 제공하는 다른 모델을 이용한 전이 학습을 해보시오.