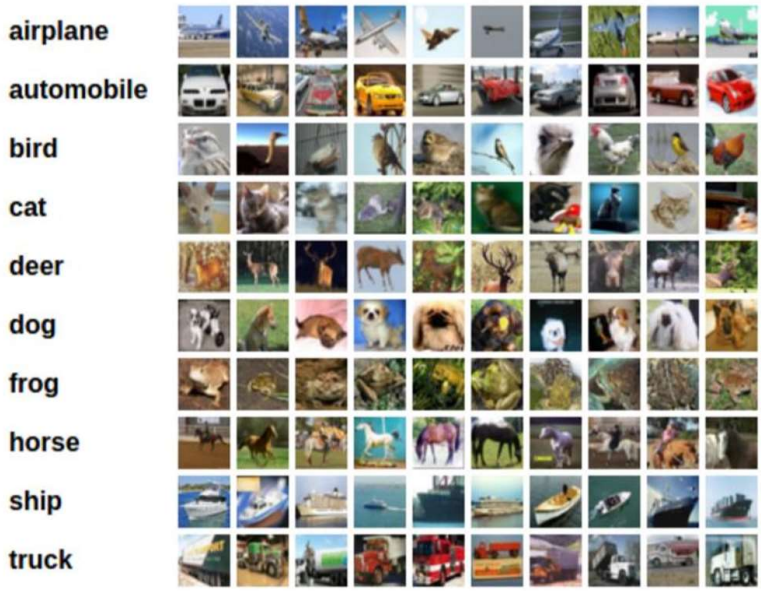
과목명	KW_VIP	-	-	담당교수	심동규 교수님
학과	전자통신공학과	학번	2017707066	이름	이성민
과제명: Assignment2					

1. 과제설명

CIFAR10 데이터셋을 이용해서 각 클래스별(사진 종류별) 정확도를 출력합니다.

2. 주요소스코드 설명

```
classes = ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```



CNN모델을 통과해서 나온 CIFAR10의 결과의 순서는 위와 같이 airplane부터 truck까지 순으로 list에 각각의 확률이 나타나므로 미리 classes라는 배열형식으로 사진의 클래스명을 미리 정의해둡니다.

```
self.layer1 = nn.Sequential(
            nn.Conv2d(3,16,kernel_size=5,stride=1,padding=2),
```

class ConvNet(nn.Module): 클래스부분 안의 코드이며 Mnist데이터셋과 다르게 CIFAR10은 RGB형태로 3가지 색상요소로 데이터가 구성되어있기 때문에 Mnist에서 1이었던 입력값을 3으로 바꿔야 합니다.

```
self.fc=nn.Linear(2048,num_classes)
```

class ConvNet(nn.Module): 클래스부분 안의 코드이며 Mnist에서 self.fc=nn.Linear(7*7*32,num_classes)로 코드를 한 결과 다음과 같은 오류가 발생했습니다.

```
RuntimeError: size mismatch, m1: [4 x 2048], m2: [1568 x 10] at ..\aten\src\TH/generic/THTensorMath.cpp:41
```

m1: [a x b], m2: [c x d]에서 b와 c를 같게 맞춰줘야 하므로 7*7*32대신에 2048을 넣어서 오류를 고쳐주고 실행을 시키면 됩니다.

```
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))

with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        c = (predicted == labels).squeeze()
        for i in range(4):
            label = labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1
```

클래스마다 정확도를 출력해주는 부분의 코드이며 바로 위의 설명에서 오류를 보면 [4x2048]과 [2048x10]이 행렬 연산을 하게 되어 [4x10]으로 결과가 나올 것이므로 test_loader에 사진 4장에 대한 결과가 나온다는 것을 유추할 수 있고 따라서 이중 for문 안에 range(4)로 설정해서 사진들에 대한 정확도를 판단합니다.

3. 실행결과 및 설명

우선 CIFAR10을 설치하는 코드에 의해서 처음에만 다음과 같이 다운로드가 진행됩니다.

```
C:\SPB_Data\.conda\envs\kwkw_vip\python.exe C:/Users/이성민/PycharmProjects/pythonProject9/main.py
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz
3.0%
```

처음 다운로드가 완료된 후 부터는 저장이 되었으므로 다음과 같이 출력이 됩니다.

```
C:\SPB_Data\.conda\envs\kwkw_vip\python.exe C:/Users/이성민/PycharmProjects/pythonProject9/main.py
Files already downloaded and verified
Files already downloaded and verified
```

CNN 모델에 대한 epoch 값을 5로 주고 학습을 진행중인 모습입니다.

```
Epoch [2/5, Step [4800/12500], Loss: 2.4828
Epoch [2/5, Step [4900/12500], Loss: 1.4979
Epoch [2/5, Step [5000/12500], Loss: 1.9780
Epoch [2/5, Step [5100/12500], Loss: 1.4145
Epoch [2/5, Step [5200/12500], Loss: 0.6588
Epoch [2/5, Step [5300/12500], Loss: 1.4641
Epoch [2/5, Step [5400/12500], Loss: 1.2764
Epoch [2/5, Step [5500/12500], Loss: 0.9018
Epoch [2/5, Step [5600/12500], Loss: 2.3926
```

각 클래스(사진 종류)별로 정확도가 출력된 모습입니다.

```
Test Accuracy of the model on the 10000 test images: 10.15 %
Accuracy of airplane : 65 %
Accuracy of automobile : 74 %
Accuracy of  bird : 56 %
Accuracy of   cat : 41 %
Accuracy of  deer : 54 %
Accuracy of   dog : 53 %
Accuracy of  frog : 63 %
Accuracy of horse : 63 %
Accuracy of  ship : 78 %
Accuracy of truck : 73 %

Process finished with exit code 0
```

epoch을 5번 밖에 반복하지 않아서 정확도가 낮게 나왔습니다.

4. 전체 소스코드

```
import torch
import torch.nn as nn
import torchvision
import torchvision.transforms as transforms

device = 'cpu'

num_epochs = 5
num_classes = 10
batch_size = 100
learning_rate = 0.001

classes = ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

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.fc=nn.Linear(2048,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

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

train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                             download=True, transform=transform)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=4,shuffle=True)

test_dataset = torchvision.datasets.CIFAR10(root='./data',train=False,
                                             download=True, transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=4,shuffle=False)

model = ConvNet(num_classes).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

total_step = len(train_loader)
for epoch in range(num_epochs):
    for i, (images,labels) in enumerate(train_loader):
        images = images.to(device)
        labels = labels.to(device)

        outputs = model(images)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
```

```
            print('Epoch [{}/{}, Step [{}/{}], Loss: {:.4f}'
                  .format(epoch +  1, num_epochs, i+ 1,total_step,loss.item()))

torch.save(model.state_dict(), 'model.ckpt')

model.eval()
with torch.no_grad():
    correct = 0
    total = 0
    for imagees, labels in test_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    print('Test Accuracy of the model on the 10000 test images: {} %'.format(100*correct/total))

class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))

with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        c = (predicted == labels).squeeze()
        for i in range(4):
            label = labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1

for i in range(10):
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] / class_total[i]))
```