Assignmnet 2

2017741047_허정범

과제목표

CIFAR10 라는 데이터 셋을 이용하여 10가지 클래스들(plane, car, bird, cat, deer, dog, frog, horse ship, truck)을 구분하고 각 클래스들의 정확도를 확인하는 것입니다.

구현방법

```
# Convolucion neural network (two convolution layers)
class ConvNet(nn.Module):
    def __init__(self, num_classes=10):
        super(ConvNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 6, kernel_size=5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(6, 16, kernel_size=5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(5 * 5 * 16, 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
```

class ConvNet는 2개의 layer 로 구성되어있다.

```
#Hyper parameters
num_epochs = 10
num_classes = 10
batch_size = 50
learning_rate = 0.001

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)
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4,
                                         shuffle=False)
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

Hyper parameters

epoch만큼 밑에 나오는 forward backward 과정을 반복하게 되는데 10을 줬으므로 10번 반복할 것이다.

classes 는 10개이므로 10을 주었다.

batch size는 sample 데이터 중 한 번에 네트워크에 넘겨주는 데이터의 수를 말합니다.

learning rate를 0.001주 었습니다.

```
# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
#Train the model
total_step = len(trainloader)
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(trainloader):
        images = images.to(device)
        labels = labels.to(device)

        #Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)

        #Backward and optimize
        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()))
```

다운 받은 정규화된 데이터 셋을 이용하여 트레이닝을 하게 된다. epoch을 10을 줬으므로 10번 반복하게 되어있다. 반복을 하면서 차이를 점점 줄여나가는 것이다.

```
Epoch [10/10], Step [12100/12500], Loss: 1.7225
Epoch [10/10], Step [12200/12500], Loss: 1.8797
Epoch [10/10], Step [12300/12500], Loss: 2.0063
Epoch [10/10], Step [12400/12500], Loss: 0.9005
Epoch [10/10], Step [12500/12500], Loss: 0.4998
Accuracy of plane : 57 %
Accuracy of   car : 69 %
Accuracy of  bird : 52 %
Accuracy of   cat : 44 %
Accuracy of  deer : 50 %
Accuracy of   dog : 40 %
Accuracy of  frog : 65 %
Accuracy of horse : 66 %
Accuracy of  ship : 79 %
Accuracy of truck : 61 %
```

그렇게 프로그램을 실행하면 이렇게 클래스 안에있는 사진들에 대해서 정확도가 각각 출력을 하게 된다.