Assignment 2

2017204091 심민정

개요

CIFAR10 데이터 셋 이미지를 plane, car, bird, cat, deer, dog, frog, horse, ship, truck 10개의 클래스로 분류하고 각 클래스 별 정확도를 출력해본다.

구현방법

```
# Convloutional neural network
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(8*8*32, 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 Convnet의 CNN모델은 convolutional layer 2개와 fully connected layer 1개 층으로 이루어져 있다. Mnist 예제와는 다르게 color임으로 채널을 3으로 변경해주었다.

```
# Hyper parameters
num_epochs=5
num_classes=10
batch_size=10
learning_rate=0.001
```

Hyper parameters를 다음과 같이 설정하였다.

```
#dataset
transform=transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5,0.8,0.5),(0.5,0.5,0.5))])

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

test_set=torchvision.datasets.CIFAR10(root='/data',train=False,
                                       download=True, transform=transform)

test_loader=torch.utils.data.DataLoader(test_set,batch_size=batch_size,
                                        shuffle=False)
model = ConvNet(num_classes).to(device)
```

정규화한 data를 사용하였다.

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

손실함수와 optimizer를 정의하였다. Learing_rate는 위에서 0.001로 설정하였다.

```python
# Train the model
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)

        # 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()))
```

신경망을 학습한다.

데이터 셋을 epoch만큼 forward, backward, 최적화 과정을 반복하고 loss값을 출력해본다. Batch_size=10과 class 수가 10개라서 한 step당 100씩 진행되는 것을 알 수 있다.

```
Epoch [5/5], Step [4400/5000], Loss: 1.0965
Epoch [5/5], Step [4500/5000], Loss: 1.1326
Epoch [5/5], Step [4600/5000], Loss: 0.5838
Epoch [5/5], Step [4700/5000], Loss: 2.1366
Epoch [5/5], Step [4800/5000], Loss: 1.3429
Epoch [5/5], Step [4900/5000], Loss: 0.9365
Epoch [5/5], Step [5000/5000], Loss: 0.7900
```

```python
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))
```

각 10개의 클래스 별 분류 정확도를 출력하기 위해 list를 생성하였다.

```
#Test the model
model.eval()
with torch.no_grad():
    correct=0
    total=0
    for images,labels in test_loader:
        images=images.to(device)
        labels=labels.to(device)
        outputs=model(images)
        _, predicted=torch.max(outputs.data,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]))
```

10개의 클래스에 중 가장 높은 값을 가지는 인덱스를 가져오고 클래스별 정확도를 출력한다.

결과화면

```
Accuracy of plane : 52 %
Accuracy of   car : 61 %
Accuracy of  bird : 38 %
Accuracy of   cat : 44 %
Accuracy of  deer : 38 %
Accuracy of   dog : 34 %
Accuracy of  frog : 72 %
Accuracy of horse : 62 %
Accuracy of  ship : 69 %
Accuracy of truck : 51 %
```