Assignment #2

-2017204045 장지연

개요

목표 : CIFAR10을 사용하여 Pytorch의 CNN을 공부&구현해보는 것이다.

구현 방법

CIFAR10 은 32x32 픽셀의 60000개 컬러 이미지가 포함되어 있으며, 각 이미지는 10개의 클래스로 라벨링이 되어있다. 머신러닝 연구에 널리 사용되는 dataset 중 하나이다.

Convolutional 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
```

컬러사진이라 layer1에서 input channel size를 3으로 하였다. 나머지는 MNIST의 코드와 똑같이 convolutional layer2개와 fully connected layer 1개로 구성하였다. Size를 구하기 위해서 아래의 공식을 참고하여 계산하였다.

input image size -> 28x28x1 (width x height x channel)

$$\frac{W-F+2*P}{S}+1$$

W: image width

F: filter width

P: padding size

S: Stride number

If there is max pooling layer after convolution filter,

$$\frac{W-F}{S}+1$$

W: input width

F: filter width

S: Stride number

layer1 = 32*32*3

ㄴ Conv2d : (32 – 3 + 2*1) / 1 + 1 = 32

ㄴ Maxpool2D : 16*16*16

Layer2 = 32*32*16

ㄴ Conv2d : (16 – 3 + 2*1) / 1 + 1 = 8

ㄴ Maxpool2D : 8*8*32

=>Output : 8 * 8 * 32

Hyper parameters

Num_classes 는 10가지의 종류가 있기 때문에 10으로 설정하였고 batch_size = 100, num_epochs = 5로 하였다.

Loss and optimizer

classification에서 많이 사용하는 loss function인 crossentropyloss을 사용하였고, optimizer은 SGD(확률적 경사 하강법)을 사용하였다.

Train the model

```
total_step = len(trainloader)
for epoch in range(num_epochs):
    for i, data in enumerate(trainloader):
        inputs, labels = data
        labels = labels.to(device)
        inputs = inputs.to(device)

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

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1)%500 ==0:
            print('Epoch [{}/{}], Step [{}/{}], Loss: {:.4f}'
                  .format(epoch + 1, num_epochs, i+1, total_step, loss.item()))
```

결과 화면

```
Accuracy of plane : 71 %
Accuracy of   car : 71 %
Accuracy of  bird : 50 %
Accuracy of   cat : 43 %
Accuracy of  deer : 55 %
Accuracy of   dog : 47 %
Accuracy of  frog : 80 %
Accuracy of horse : 58 %
Accuracy of  ship : 75 %
Accuracy of truck : 55 %
```

강아지와 고양이를 잘 구별은 못하는 결과가 나왔다.