Assignment3

-2017204045 장지연

목표

나만의 네트워크를 구성해본다. 다양한 데이터들을 사용해서 다양한 학습을 시킴

구현방법

```
class Net(torch.nn.Module):
    def __init__(self,num_classes=2):
        super(Net, 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.layer3 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.fc = nn.Linear(28* 28 * 64, num_classes)
```

데이터를 불러올 때 크기를 다시 조정해주었는데 크기가 224*224*3으로 추정된다.

총 3층을 넣어 주었는데 계산식에 의하면

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

224*224*3->112*112*16->56*56*32->28*28*64가 된다.

```
num_classes=2
num_epochs = 5
batch_size = 100
learning_rate = 0.001
model = Net(num_classes).to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

Hyper parameters

Num_classes 는 벌과 개미가 있기 때문에 2으로 설정하였고 batch_size = 100, num_epochs = 5로 하였다.

Loss and optimizer

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

결과

```
train Loss: 0.6609 Acc: 0.5984
val Loss: 0.6634 Acc: 0.5882
```

그렇게 좋진 않다 잘 못 만든 것 같다. 데이터가 적어서 그런지 내가 잘못 만든 건지 궁금하다.