KW_VIP 2016707015 신승현

과제 목표

이번 과제는 CIFAR10 dataset을 다운받고 Convolution layer로 구성된 모델에 학습시키는 것이 목표이다.

CIFAR10은 airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck으로 구성되어 있다.

구현 방법

모델의 layer를 쌓기 위해 nn.Module를 상속하는 클래스를 생성하였고 Convolution layer 2개, Maxpooling layer 2개를 번갈아 가면서 쌓았고 마지막에는 Fully connected layer를 사용하여 하나의 값을 출력하게 하였다.

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

입력 데이터 값을 -1~1의 범위로 정규화 하기위해 transform 함수를 다음과 같이 정의하였다.

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

train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=batch_size,
                                           shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=batch_size,
                                          shuffle=False)
```

이 코드로 CIFAR10 데이터를 다운 받고 DataLoader을 통해 배치 사이즈만큼 나누어 입력 이미지와 이에 대한 라벨링을 train_loader 라는 변수에 넣었다.

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

모델은 10개의 카테고리중 하나를 설정하는 classification 이므로 loss를 CrossEntropy로 설정하였고

배치 사이즈마다 weight를 업데이트 하기위해 SGD optimizer를 사용하였다.

```
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))
with torch.no_grad():
    for images, labels in test_loader:
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        c = (predicted == labels)
        for i in range(batch_size):
            class_label = labels[i]
            class_correct[class_label] += c[i].item()
            class_total[class_label] += 1
```

Test_loader에는 전체 입력 개수 / 배치 사이즈 만큼의 수가 있고

outputs에는 10개의 카테고리에 대한 확률이 들어있으므로 torch.max를 통해 가장 큰 값을 뽑아낸다.

그 후 라벨링과 예측한 값이 같으면 c=True가 되고, item()을 이용하여 True=>1, False=>0로 바꿔준다.

배치 사이즈만큼 반복하면서 class_correct에 c값을 입력해주고 total에는 총 개수를 저장한다.

```
for i in range(10):
    print('Accuracy of {} : {} %'
          .format(classes[i], 100*class_correct[i] / class_total[i]))
```

각 라벨링에 대한 정확도를 출력해 준다.

결과 화면

```
Accuracy of plane : 57.7 %
Accuracy of car : 73.2 %
Accuracy of bird : 58.2 %
Accuracy of cat : 61.5 %
Accuracy of deer : 46.2 %
Accuracy of dog : 32.0 %
Accuracy of frog : 63.8 %
Accuracy of horse : 68.9 %
Accuracy of ship : 82.7 %
Accuracy of truck : 67.1 %

Process finished with exit code 0
```

다음과 같이 배가 제일 정확도가 높은 것을 볼 수 있다.