exam 개인학습

Lec 20

Type something...

01) ImageFolder2

01_라이브러리 가져오기

```
[2]  #1. 라이브러리 가져오기
     import torch
     import torch.nn as nn
     import torch.nn.functional as F

     import torch.optim as optim
     from torch.utils.data import DataLoader

     import torchvision
     import torchvision.transforms as transforms
```

02_ 데이터셋 가져오기

```
[4]  # 4. 데이터셋 가져오기
     trans = transforms.Compose([
         transforms.ToTensor()
     ])
     train_data = torchvision.datasets.ImageFolder('/content/drive/MyDrive/deep_learning/pytorch_cnn_example/train_data',
                                                   transform = trans)
```

- Tensor로 변경해주기, ImageFolder로 이미지들 가져오기.

03_학습 모델 만들기

```python
# 5. 학습 모델 만들기
class CNN(torch.nn.Module):
  def __init__(self):
    super(CNN, self).__init__() # 초기화
  # Layer 1
  # image = 3 * 64 * 128
  # Cnn1: in_c = 3, out_c = 6, kernel_size = 5, stride = 1
  # output: 6 * 60 * 124
  # ReLU
  # Maxpool(2)
  # output : 6 * 30 * 62
    self.layer1 = nn.Sequential(
        nn.Conv2d(3, 6, kernel_size = 5, stride = 1),
        nn.ReLU(),
        nn.MaxPool2d(2),
    )
  # Layer 2
  # image = 6 * 30 * 62
  # Cnn1: in_c = 6, out_c = 16, kernel_size = 5, stride = 1
  # output: 16 * 26 * 58
  # ReLU
  # Maxpool(2)
  # output : 16 * 13 * 29
    self.layer2 = nn.Sequential(
        nn.Conv2d(6, 16, kernel_size = 5, stride = 1),
        nn.ReLU(),
        nn.MaxPool2d(2),
    )
  # view : batch * [16, 13, 29] -> batch * [6032]
  # FC1 : (6032, 120)
  # ReLU
  # FC2 : (120, 2) -> binary classification
    self.layer3 = nn.Sequential(
        nn.Linear(16 * 13 * 29, 120),
        nn.ReLU(),
        nn.Linear(120, 2)
```

- `__init__` 와 `forward` 2개를 만들어주면 된다.
- Layer1과 Layer2 모두 CNN layer + ReLU + MaxPooling으로 이루어져 있고, 마지막은 FC1, FC2로 이루어져 있다.

```python
def forward(self, x):
    out = self.layer1(x)
    #print(out.shape)
    out = self.layer2(out)
    #print(out.shape)
    out = out.view(out.shape[0], -1) # batch_size * 6032, view는 forward에서 진행하기
    #print(out.shape)
    out = self.layer3(out)
    #print(out.shape)
    return out
```

04_학습시키기

```
total_batch = len(data_loader)

epochs = 7
for epoch in range(epochs):
  avg_cost = 0.0
  for num, data in enumerate(data_loader):
    imgs, labels = data # 숫자, 이미지, 라벨 순서대로
    imgs = imgs.to(device)
    labels = labels.to(device)

    out = net(imgs)
    loss = loss_func(out, labels)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    avg_cost += loss / total_batch
  print('[Epoch: {}] cost = {}'.format(epoch + 1, avg_cost))
print('Learning Finished!')
```

- 이전과 차이점은 없으므로 생략한다.
- binary classification이지만 softmax classifier를 사용하였다.

05_모델 저장하기(중요)

```
[14] torch.save(net.state_dict(), "/content/drive/MyDrive/deep_learning/pytorch_cnn_example/model/model.pth")

[15] new_net = CNN().to(device)

[16] new_net.load_state_dict(torch.load("/content/drive/MyDrive/deep_learning/pytorch_cnn_example/model/model.pth"))

<ipython-input-16-b237434dc837>:1: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which
  new_net.load_state_dict(torch.load("/content/drive/MyDrive/deep_learning/pytorch_cnn_example/model/model.pth"))
<All keys matched successfully>
```

`torch.save(net.state_dict(), “경로”)`

`new_net = CNN().to(device)`

`new_net.load_state_dict(torch.load("경로")`

- 학습 시킨 상태를 저장하는 방식이다. State_dict()를 특정 폴더에 저장시켜 나중에 `load_state_dict` 로 불러올 수 있다.
- test도 이전과 동일한 방식으로 진행하면 된다.