Lab3

November 23, 2020

```python
[1]: import torch
     import torch.nn as nn
     import torch.optim as optim
     import torch.utils.data
     import torch.nn.functional as F
     import torchvision
     from torchvision import transforms
     from PIL import Image
     import matplotlib.pyplot as plt
     import numpy as np
     %matplotlib inline
```

1

```python
[2]: LEARNING_RATE = 0.01
     EPOCHS = 50
     TRAIN_DATA_SIZE = 500
```

```python
[3]: def check_image(path):
         try:
             im = Image.open(path)
             return True
         except:
             return False
```

```python
[4]: img_transforms = transforms.Compose([
         transforms.Resize((64,64)),
         transforms.Grayscale(),
         transforms.ToTensor()
         ])
```

```python
[5]: train_data_path = "./train/"
     train_data = torchvision.datasets.
      ↪ImageFolder(root=train_data_path,transform=img_transforms,␣
      ↪is_valid_file=check_image)
     len(train_data)
```

```
[5]: 811
```

```
[6]: val_data_path = "./val/"
     val_data = torchvision.datasets.
      ↪ImageFolder(root=val_data_path,transform=img_transforms,␣
      ↪is_valid_file=check_image)
```

```
[7]: test_data_path = "./test/"
     test_data = torchvision.datasets.
      ↪ImageFolder(root=test_data_path,transform=img_transforms,␣
      ↪is_valid_file=check_image)
```

```
[8]: batch_size=TRAIN_DATA_SIZE
```

```
[9]: train_data_loader = torch.utils.data.DataLoader(train_data,␣
      ↪batch_size=batch_size)
     val_data_loader  = torch.utils.data.DataLoader(val_data, batch_size=batch_size)
     test_data_loader  = torch.utils.data.DataLoader(test_data,␣
      ↪batch_size=batch_size)
```

```
[10]: class SimpleNet(nn.Module):

          def __init__(self):
              super(SimpleNet, self).__init__()
              self.fc1 = nn.Linear(4096, 84)
              self.fc2 = nn.Linear(84, 50)
              self.fc3 = nn.Linear(50,2)

          def forward(self, x):
              x = x.view(-1, 64*64)
              x = F.relu(self.fc1(x))
              x = F.relu(self.fc2(x))
              x = self.fc3(x)
              return x
```

```
[11]: simplenet = SimpleNet()
```

```
[12]: optimizer = optim.Adam(simplenet.parameters(), lr=LEARNING_RATE)
```

```
[13]: if torch.cuda.is_available():
          device = torch.device("cuda")
      else:
          device = torch.device("cpu")

      simplenet.to(device)
```

/home/blestrong/anaconda3/lib/python3.8/site-packages/torch/cuda/__init__.py:52:

```
UserWarning: CUDA initialization: The NVIDIA driver on your system is too old
(found version 6050). Please update your GPU driver by downloading and
installing a new version from the URL: http://www.nvidia.com/Download/index.aspx
Alternatively, go to: https://pytorch.org to install a PyTorch version that has
been compiled with your version of the CUDA driver. (Triggered internally at
/opt/conda/conda-bld/pytorch_1603729096996/work/c10/cuda/CUDAFunctions.cpp:100.)
  return torch._C._cuda_getDeviceCount() > 0
```

[13]: SimpleNet(
  (fc1): Linear(in_features=4096, out_features=84, bias=True)
  (fc2): Linear(in_features=84, out_features=50, bias=True)
  (fc3): Linear(in_features=50, out_features=2, bias=True)
)

[14]:
```python
def train(model, optimizer, loss_fn, train_loader, val_loader, epochs=20,
↪device="cpu"):
    for epoch in range(epochs):
        training_loss = 0.0
        valid_loss = 0.0
        model.train()
        for batch in train_loader:
            optimizer.zero_grad()
            inputs, targets = batch
            inputs = inputs.to(device)
            targets = targets.to(device)
            output = model(inputs)
            loss = loss_fn(output, targets)
            loss.backward()
            optimizer.step()
            training_loss += loss.data.item() * inputs.size(0)
        training_loss /= len(train_loader.dataset)

        model.eval()
        num_correct = 0
        num_examples = 0
        for batch in val_loader:
            inputs, targets = batch
            inputs = inputs.to(device)
            output = model(inputs)
            targets = targets.to(device)
            loss = loss_fn(output,targets)
            valid_loss += loss.data.item() * inputs.size(0)
            correct = torch.eq(torch.max(F.softmax(output), dim=1)[1], targets).
↪view(-1)
            num_correct += torch.sum(correct).item()
            num_examples += correct.shape[0]
        valid_loss /= len(val_loader.dataset)
```

```
        print('Epoch: {}, Training Loss: {:.2f}, Validation Loss: {:.2f},
→accuracy = {:.2f}'.format(epoch, training_loss,
        valid_loss, num_correct / num_examples))
```

[15]:
```
train(simplenet, optimizer,torch.nn.CrossEntropyLoss(),
→train_data_loader,val_data_loader, epochs=EPOCHS, device=device)
```

```
<ipython-input-14-f38d79bc5988>:28: UserWarning: Implicit dimension choice for
softmax has been deprecated. Change the call to include dim=X as an argument.
  correct = torch.eq(torch.max(F.softmax(output), dim=1)[1], targets).view(-1)

Epoch: 0, Training Loss: 15.11, Validation Loss: 1.17, accuracy = 0.79
Epoch: 1, Training Loss: 0.72, Validation Loss: 2.76, accuracy = 0.21
Epoch: 2, Training Loss: 1.86, Validation Loss: 1.61, accuracy = 0.21
Epoch: 3, Training Loss: 1.23, Validation Loss: 0.73, accuracy = 0.22
Epoch: 4, Training Loss: 0.80, Validation Loss: 0.62, accuracy = 0.79
Epoch: 5, Training Loss: 0.72, Validation Loss: 0.63, accuracy = 0.79
Epoch: 6, Training Loss: 0.69, Validation Loss: 0.67, accuracy = 0.79
Epoch: 7, Training Loss: 0.68, Validation Loss: 0.69, accuracy = 0.46
Epoch: 8, Training Loss: 0.69, Validation Loss: 0.70, accuracy = 0.29
Epoch: 9, Training Loss: 0.94, Validation Loss: 0.74, accuracy = 0.21
Epoch: 10, Training Loss: 0.69, Validation Loss: 0.83, accuracy = 0.21
Epoch: 11, Training Loss: 0.72, Validation Loss: 0.88, accuracy = 0.21
Epoch: 12, Training Loss: 0.74, Validation Loss: 0.88, accuracy = 0.21
Epoch: 13, Training Loss: 0.75, Validation Loss: 0.86, accuracy = 0.21
Epoch: 14, Training Loss: 0.74, Validation Loss: 0.82, accuracy = 0.21
Epoch: 15, Training Loss: 0.72, Validation Loss: 0.78, accuracy = 0.22
Epoch: 16, Training Loss: 0.71, Validation Loss: 0.76, accuracy = 0.24
Epoch: 17, Training Loss: 0.70, Validation Loss: 0.75, accuracy = 0.26
Epoch: 18, Training Loss: 0.69, Validation Loss: 0.75, accuracy = 0.31
Epoch: 19, Training Loss: 0.68, Validation Loss: 0.76, accuracy = 0.30
Epoch: 20, Training Loss: 0.68, Validation Loss: 0.77, accuracy = 0.27
Epoch: 21, Training Loss: 0.68, Validation Loss: 0.79, accuracy = 0.26
Epoch: 22, Training Loss: 0.69, Validation Loss: 0.81, accuracy = 0.26
Epoch: 23, Training Loss: 0.69, Validation Loss: 0.81, accuracy = 0.28
Epoch: 24, Training Loss: 0.69, Validation Loss: 0.81, accuracy = 0.29
Epoch: 25, Training Loss: 0.69, Validation Loss: 0.80, accuracy = 0.30
Epoch: 26, Training Loss: 0.69, Validation Loss: 0.80, accuracy = 0.33
Epoch: 27, Training Loss: 0.68, Validation Loss: 0.79, accuracy = 0.37
Epoch: 28, Training Loss: 0.68, Validation Loss: 0.79, accuracy = 0.37
Epoch: 29, Training Loss: 0.68, Validation Loss: 0.78, accuracy = 0.39
Epoch: 30, Training Loss: 0.67, Validation Loss: 0.78, accuracy = 0.42
Epoch: 31, Training Loss: 0.67, Validation Loss: 0.78, accuracy = 0.42
Epoch: 32, Training Loss: 0.67, Validation Loss: 0.78, accuracy = 0.43
Epoch: 33, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.44
Epoch: 34, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.45
Epoch: 35, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.46
```

```
Epoch: 36, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.46
Epoch: 37, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.47
Epoch: 38, Training Loss: 0.67, Validation Loss: 0.79, accuracy = 0.48
Epoch: 39, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.48
Epoch: 40, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.47
Epoch: 41, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.47
Epoch: 42, Training Loss: 0.66, Validation Loss: 0.78, accuracy = 0.48
Epoch: 43, Training Loss: 0.66, Validation Loss: 0.78, accuracy = 0.52
Epoch: 44, Training Loss: 0.66, Validation Loss: 0.78, accuracy = 0.52
Epoch: 45, Training Loss: 0.66, Validation Loss: 0.78, accuracy = 0.50
Epoch: 46, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.48
Epoch: 47, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.47
Epoch: 48, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.47
Epoch: 49, Training Loss: 0.66, Validation Loss: 0.79, accuracy = 0.53
```

\#

```python
labels = ['cat','fish']

img = Image.open("./val/fish/100_1422.JPG")
img = img_transforms(img).to(device)

plt.imshow(  img.permute(1, 2, 0), cmap ='gray'  )

prediction = F.softmax(simplenet(img))
prediction = prediction.argmax()
print(labels[prediction])
```
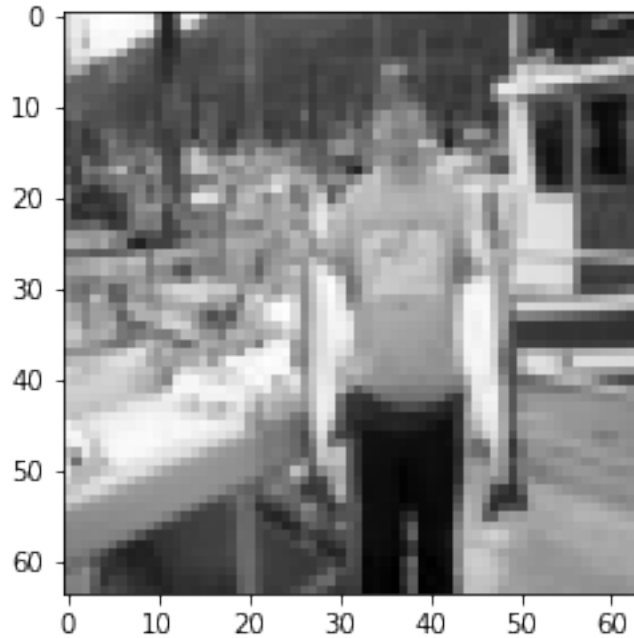
```
fish
```

```
<ipython-input-16-b4ae3b790d49>:8: UserWarning: Implicit dimension choice for
softmax has been deprecated. Change the call to include dim=X as an argument.
  prediction = F.softmax(simplenet(img))
```

```
[17]: num_examples = len(test_data)
      num_correct = 0
      for data in test_data:
          prediction = F.softmax(simplenet(data[0]))
          prediction = prediction.argmax()
          if data[1] == prediction:
              num_correct += 1
      print("Accuracy = {}".format(num_correct/num_examples))
```

```
<ipython-input-17-033a5615afd5>:4: UserWarning: Implicit dimension choice for
softmax has been deprecated. Change the call to include dim=X as an argument.
  prediction = F.softmax(simplenet(data[0]))

Accuracy = 0.675
```

2

```
[18]: torch.save(simplenet.state_dict(), "tmp/simplenet")
```

3

```
[19]: simplenet = SimpleNet()
      simplenet_state_dict = torch.load("tmp/simplenet")
      simplenet.load_state_dict(simplenet_state_dict)
```

[19]: <All keys matched successfully>