训练手写数字识别器

> 计算机视觉第六次作业 | 2101212840 游盈萱

代码

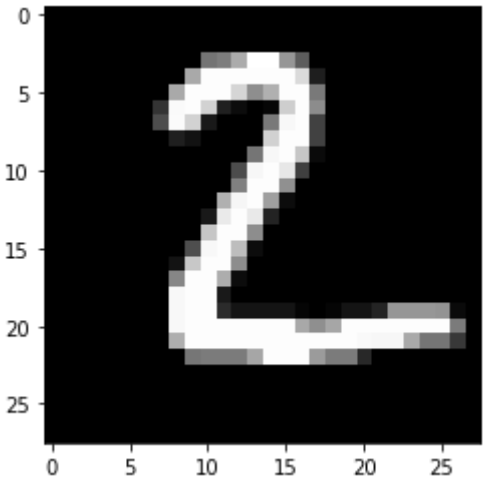
```
import torch
import torch.nn as nn
import torch.utils.data as Data
import torchvision
import torch.nn.functional as F
import numpy as np
from torchvision import transforms as tfs

def train_tf(x):
    im_aug = tfs.Compose([
        tfs.Resize(120),
        tfs.RandomRotation(30),
        tfs.RandomAffine(degrees=0, translate=(3, 3), scale=None, shear=None, resample=False, fillcolor=0),
        tfs.ToTensor(),
    ])
    x = im_aug(x)
    return x


DOWNLOAD_MNIST = True


train_data = torchvision.datasets.MNIST(root='./mnist/', train=True, transform=train_tf,
                                        download=DOWNLOAD_MNIST, )
test_data = torchvision.datasets.MNIST(root='./mnist/', train=False)
print(train_data.train_data.shape)

train_x = torch.unsqueeze(train_data.train_data, dim=1).type(torch.FloatTensor) / 255.
train_y = train_data.train_labels
print(train_x.shape)

test_x = torch.unsqueeze(test_data.test_data, dim=1).type(torch.FloatTensor)[:2000] / 255.  # Tensor on GPU
test_y = test_data.test_labels[:2000]
```

$torch.Size([60000, 28, 28])$
$torch.Size([60000, 1, 28, 28])$

```
test_x.shape
```

$torch.Size([2000, 1, 28, 28])$

```
import matplotlib.pyplot as plt
plt.imshow(test_x[1,0,:,:].numpy(), 'gray')
```




```
test_y[:10]
```

$tensor([7, 2, 1, 0, 4, 1, 4, 9, 5, 9])$

```
class NET(nn.Module):
    def __init__(self):
        super(NET, self).__init__()
        self.conv1 = nn.Sequential(nn.Conv2d(in_channels=1, out_channels=16, kernel_size=5, stride=1, padding=2,), nn.ReLU(),
                                   nn.MaxPool2d(2),)
        self.dropout1 = nn.Dropout(0.3)
        self.conv2 = nn.Sequential(nn.Conv2d(16, 32, 5, 1, 2), nn.ReLU(), nn.MaxPool2d(2),)
        self.dropout2 = nn.Dropout(0.5)
        self.conv3 = nn.Sequential(nn.Conv2d(32, 64, 4, 1, 2), nn.BatchNorm2d(64), nn.ReLU(), nn.MaxPool2d(2))
        self.linear1 = nn.Linear(64 * 4 * 4, 10)
        self.bn = nn.BatchNorm1d(10)

    def forward(self, x):
        x = self.conv1(x)
        self.dropout1
        x = self.conv2(x)
        self.dropout2
        x = self.conv3(x)
        self.dropout2
        x = x.view(x.size(0), -1)
        x = self.linear1(x)

        output = self.bn(x)
        return output


data_size = 20000
batch_size = 32
EPOCH = 3
LR = 0.002

```

```
net = NET()

optimizer = torch.optim.Adam(net.parameters(), lr=LR)
loss_func = nn.CrossEntropyLoss()



for epoch in range(EPOCH):
    random_indx = np.random.permutation(data_size)
    for batch_i in range(data_size // batch_size):
        indx = random_indx[batch_i * batch_size:(batch_i + 1) * batch_size]

        b_x = train_x[indx, :]
        b_y = train_y[indx]

        output = net(b_x)
        loss = loss_func(output, b_y)

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

        if batch_i % 50 == 0:
            test_output = net(test_x)
            pred_y = torch.max(test_output, 1)[1].data.squeeze()
            # pred_y = torch.max(test_output, 1)[1].data.squeeze()
            accuracy = torch.sum(pred_y == test_y).type(torch.FloatTensor) / test_y.size(0)
            print('Epoch: ', epoch, '| train loss: %.4f' % loss.data.cpu().numpy(), '| test accuracy: %.3f' % accuracy)

test_output = net(test_x[:10])
pred_y = torch.max(test_output, 1)[1].data.squeeze()  # move the computation in GPU

print(pred_y, 'prediction number')
print(test_y[:10], 'real number')
```

$Epoch:0|trainloss:2.8338|testaccuracy:0.403$
$Epoch:0|trainloss:0.6470|testaccuracy:0.947$
$Epoch:0|trainloss:0.5726|testaccuracy:0.951$
$Epoch:0|trainloss:0.3787|testaccuracy:0.965$
$Epoch:0|trainloss:0.3894|testaccuracy:0.969$
$Epoch:0|trainloss:0.4790|testaccuracy:0.973$
$Epoch:0|trainloss:0.4183|testaccuracy:0.977$
$Epoch:0|trainloss:0.1873|testaccuracy:0.981$
$Epoch:0|trainloss:0.2982|testaccuracy:0.976$
$Epoch:0|trainloss:0.1809|testaccuracy:0.978$
$Epoch:0|trainloss:0.2417|testaccuracy:0.983$
$Epoch:0|trainloss:0.3181|testaccuracy:0.979$
$Epoch:0|trainloss:0.2491|testaccuracy:0.980$
$Epoch:1|trainloss:0.3246|testaccuracy:0.976$
$Epoch:1|trainloss:0.2366|testaccuracy:0.982$
$Epoch:1|trainloss:0.1431|testaccuracy:0.982$
$Epoch:1|trainloss:0.1077|testaccuracy:0.984$
$Epoch:1|trainloss:0.2690|testaccuracy:0.979$
$Epoch:1|trainloss:0.0928|testaccuracy:0.982$
$Epoch:1|trainloss:0.3645|testaccuracy:0.984$
$Epoch:1|trainloss:0.3114|testaccuracy:0.982$
$Epoch:1|trainloss:0.1202|testaccuracy:0.985$
$Epoch:1|trainloss:0.1885|testaccuracy:0.980$
$Epoch:1|trainloss:0.1152|testaccuracy:0.979$
$Epoch:1|trainloss:0.2172|testaccuracy:0.985$
$Epoch:1|trainloss:0.0573|testaccuracy:0.984$
$Epoch:2|trainloss:0.1718|testaccuracy:0.984$
$Epoch:2|trainloss:0.1892|testaccuracy:0.984$
$Epoch:2|trainloss:0.0446|testaccuracy:0.987$
$Epoch:2|trainloss:0.0530|testaccuracy:0.985$
$Epoch:2|trainloss:0.1286|testaccuracy:0.989$
$Epoch:2|trainloss:0.1181|testaccuracy:0.983$
$Epoch:2|trainloss:0.0529|testaccuracy:0.984$
$Epoch:2|trainloss:0.0915|testaccuracy:0.984$
$Epoch:2|trainloss:0.2105|testaccuracy:0.984$
$Epoch:2|trainloss:0.0503|testaccuracy:0.987$
$Epoch:2|trainloss:0.0714|testaccuracy:0.987$
$Epoch:2|trainloss:0.0461|testaccuracy:0.986$
$Epoch:2|trainloss:0.0802|testaccuracy:0.988$
$tensor([7,2,1,0,4,1,4,9,6,9])prediction number$
$tensor([7,2,1,0,4,1,4,9,5,9])real number$