HW1_p4

September 30, 2022

[6]:
```
# Import PyTorch package
import numpy as np
import torch
import torchvision
```

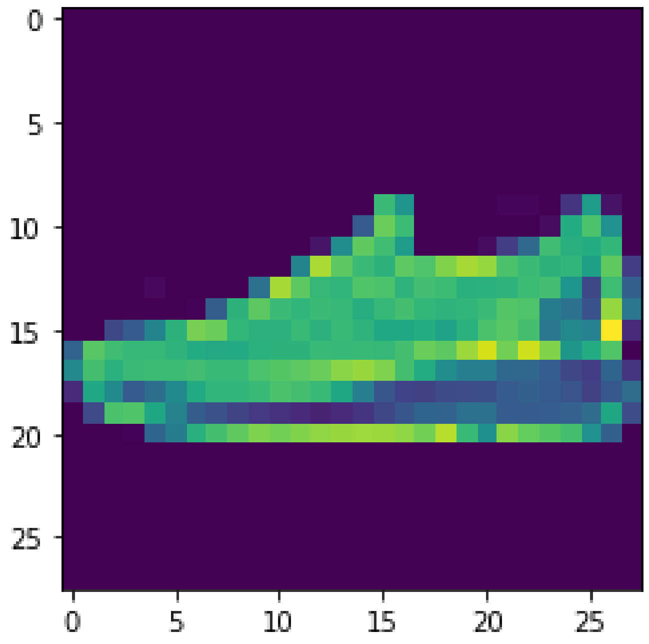
[7]:
```
# Import Dataset
trainingdata = torchvision.datasets.FashionMNIST('./FashionMNIST/
 ↪',train=True,download=True,transform=torchvision.transforms.ToTensor())
testdata = torchvision.datasets.FashionMNIST('./FashionMNIST/
 ↪',train=False,download=True,transform=torchvision.transforms.ToTensor())
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz to ./FashionMNIST/FashionMNIST/raw/train-images-
idx3-ubyte.gz

  0%|          | 0/26421880 [00:00<?, ?it/s]


Extracting ./FashionMNIST/FashionMNIST/raw/train-images-idx3-ubyte.gz to
./FashionMNIST/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz to ./FashionMNIST/FashionMNIST/raw/train-labels-
idx1-ubyte.gz

  0%|          | 0/29515 [00:00<?, ?it/s]


Extracting ./FashionMNIST/FashionMNIST/raw/train-labels-idx1-ubyte.gz to
./FashionMNIST/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to
./FashionMNIST/FashionMNIST/raw/t10k-images-idx3-ubyte.gz
```

```
  0%|          | 0/4422102 [00:00<?, ?it/s]


Extracting ./FashionMNIST/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to
./FashionMNIST/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to
./FashionMNIST/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz
  0%|          | 0/5148 [00:00<?, ?it/s]


Extracting ./FashionMNIST/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to
./FashionMNIST/FashionMNIST/raw
```

[9]:
```python
# Test Downloaded Dataset
image, label = trainingdata[1893]
print(image.shape, label)

import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow(image.squeeze().numpy())
plt.show()
```

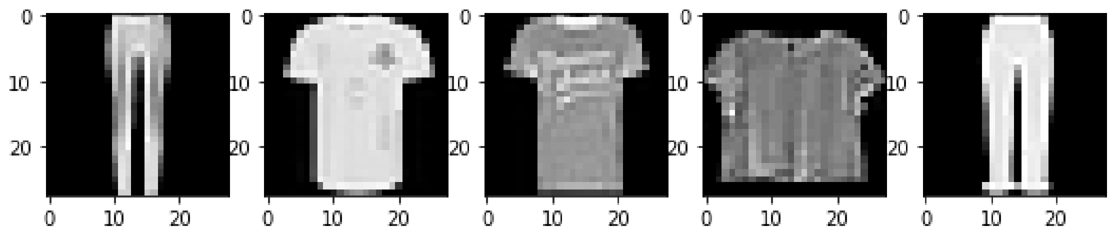
```
torch.Size([1, 28, 28]) 7
```



[10]:
```
# Create data loader for training and testing
trainDataLoader = torch.utils.data.
 ↪DataLoader(trainingdata,batch_size=64,shuffle=True)
testDataLoader = torch.utils.data.
 ↪DataLoader(testdata,batch_size=64,shuffle=False)

print(len(trainDataLoader))
print(len(testDataLoader))
```

```
938
157
```

[11]:
```
# Test Dataloader
images, labels = iter(trainDataLoader).next()

plt.figure(figsize=(10,4))
for index in np.arange(0,5):
  plt.subplot(1,5,index+1)
  plt.imshow(images[index].squeeze().numpy(),cmap=plt.cm.gray)
```



[12]:
```
# Create Network Structure and initialize the parameter
from torch import nn

net_3_layers_perceptron = nn.Sequential(nn.Flatten(),
                    nn.Linear(28 * 28, 256),
                    nn.ReLU(),
                    nn.Linear(256, 128),
                    nn.ReLU(),
                    nn.Linear(128, 64),
                    nn.ReLU(),
                    nn.Linear(64, 10)).cuda()

def init_weights(m):
    if type(m) == nn.Linear:
        nn.init.normal_(m.weight, std=0.01)
```

```
net_3_layers_perceptron.apply(init_weights)
```

[12]:
```
Sequential(
  (0): Flatten(start_dim=1, end_dim=-1)
  (1): Linear(in_features=784, out_features=256, bias=True)
  (2): ReLU()
  (3): Linear(in_features=256, out_features=128, bias=True)
  (4): ReLU()
  (5): Linear(in_features=128, out_features=64, bias=True)
  (6): ReLU()
  (7): Linear(in_features=64, out_features=10, bias=True)
)
```

[13]:
```
# Define the learning rate, epoch nums, optimizer, loss function...
lr = 0.02
epoch_nums = 50
Loss = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(net_3_layers_perceptron.parameters(), lr)
```

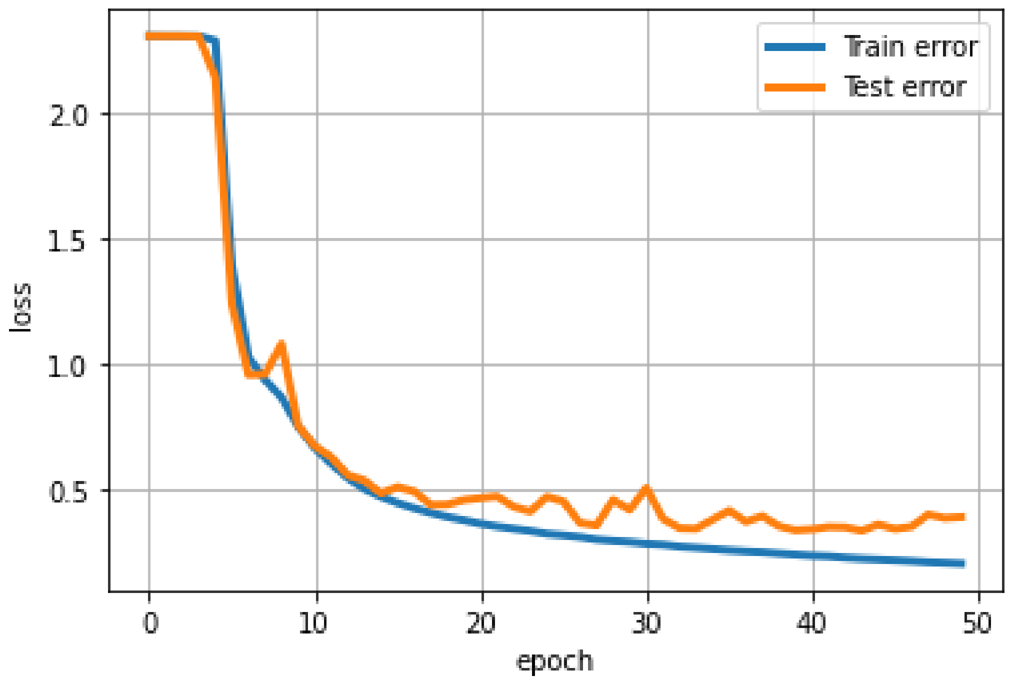
[14]:
```
# Begin training and testing
train_loss_history = []
test_loss_history = []
test_accuracy_history = []

for epoch in range(epoch_nums):
  train_loss = 0.0
  test_loss = 0.0
  test_accuracy = 0.0
  for i, data in enumerate(trainDataLoader):
    images, labels = data
    images = images.cuda()
    labels = labels.cuda()
    optimizer.zero_grad()
    predicted_output = net_3_layers_perceptron(images)
    fit = Loss(predicted_output,labels)
    fit.backward()
    optimizer.step()
    train_loss += fit.item()

  correct_predicted_label_num = 0
  total_num_of_labels = len(testdata)
  for i, data in enumerate(testDataLoader):
    with torch.no_grad():
      images, labels = data
      images = images.cuda()
      labels = labels.cuda()
      predicted_output = net_3_layers_perceptron(images)
```

```
        fit = Loss(predicted_output,labels)
        test_loss += fit.item()

        # Calculate testing accuracy
        predicted_labels = torch.max(predicted_output, 1).indices
        correct_predicted_label_num += torch.eq(predicted_labels, labels).sum()


    test_accuracy = correct_predicted_label_num / total_num_of_labels
    train_loss = train_loss/len(trainDataLoader)
    test_loss = test_loss/len(testDataLoader)
    train_loss_history.append(train_loss)
    test_loss_history.append(test_loss)
    test_accuracy_history.append(test_accuracy)
    print('Epoch %s, Train loss %s, Test loss %s'%(epoch, train_loss, test_loss))
```

```
Epoch 0, Train loss 2.303288939410944, Test loss 2.3026329453583734
Epoch 1, Train loss 2.3026656313999885, Test loss 2.3025010634379783
Epoch 2, Train loss 2.302520084736952, Test loss 2.3023078153087835
Epoch 3, Train loss 2.3020962522482313, Test loss 2.3013176371337503
Epoch 4, Train loss 2.2838296633539423, Test loss 2.1378464182471015
Epoch 5, Train loss 1.3927732301292135, Test loss 1.239315010939434
Epoch 6, Train loss 1.0284173071130251, Test loss 0.957804045100121
Epoch 7, Train loss 0.937849393912724, Test loss 0.9602925390194935
Epoch 8, Train loss 0.8683912374063342, Test loss 1.081169197513799
Epoch 9, Train loss 0.7525679993985305, Test loss 0.7556559291614848
Epoch 10, Train loss 0.6679288638171865, Test loss 0.6756025622984406
Epoch 11, Train loss 0.6047675323003391, Test loss 0.6313826830903436
Epoch 12, Train loss 0.5482314621239329, Test loss 0.5598531329328087
Epoch 13, Train loss 0.5056541453260602, Test loss 0.5387791215803972
Epoch 14, Train loss 0.4740880811169966, Test loss 0.4881954356363625
Epoch 15, Train loss 0.4492566147084429, Test loss 0.5135117249124369
Epoch 16, Train loss 0.42821390169070983, Test loss 0.49530001962260833
Epoch 17, Train loss 0.4102137350101969, Test loss 0.4424129799482929
Epoch 18, Train loss 0.3940752685260671, Test loss 0.4436657093702608
Epoch 19, Train loss 0.3812355605809927, Test loss 0.46166111775644264
Epoch 20, Train loss 0.3669788326535906, Test loss 0.4690746577682009
Epoch 21, Train loss 0.356240924145939, Test loss 0.47580185779340706
Epoch 22, Train loss 0.34739060913607767, Test loss 0.4345349000327906
Epoch 23, Train loss 0.3385332088305879, Test loss 0.41566411685791743
Epoch 24, Train loss 0.3271602399901413, Test loss 0.4743002156732948
Epoch 25, Train loss 0.3214676023593971, Test loss 0.4558868312342152
Epoch 26, Train loss 0.3145685161529446, Test loss 0.37010603250971263
Epoch 27, Train loss 0.3044803959410836, Test loss 0.36022600465139765
Epoch 28, Train loss 0.2988965465331764, Test loss 0.4618303416071424
Epoch 29, Train loss 0.2946172474480387, Test loss 0.4225139598937551
Epoch 30, Train loss 0.28770788703391803, Test loss 0.5094753506646794
Epoch 31, Train loss 0.2840642210350298, Test loss 0.387007164822262555
Epoch 32, Train loss 0.27597395026448696, Test loss 0.34951994553872734
Epoch 33, Train loss 0.27154296161984204, Test loss 0.3474722394518032
Epoch 34, Train loss 0.266581248738237, Test loss 0.3835802800526285
Epoch 35, Train loss 0.26148684940008976, Test loss 0.41975149445852655
Epoch 36, Train loss 0.25747979201995996, Test loss 0.37533049750479924
Epoch 37, Train loss 0.25434230673891395, Test loss 0.3972062335652151
Epoch 38, Train loss 0.24917279453928282, Test loss 0.35734037884101744
Epoch 39, Train loss 0.24438226079222744, Test loss 0.34130788437879767
Epoch 40, Train loss 0.2400833210870147, Test loss 0.3452069767436404
Epoch 41, Train loss 0.2374104316165643, Test loss 0.3544638219532693
Epoch 42, Train loss 0.23199086100149002, Test loss 0.35311419331723715
Epoch 43, Train loss 0.22812451036579445, Test loss 0.3392550607870339
Epoch 44, Train loss 0.22525270053668062, Test loss 0.36403391344152436
Epoch 45, Train loss 0.22210550966706358, Test loss 0.3467123775155681
Epoch 46, Train loss 0.2187631985526095, Test loss 0.35545669277762154
Epoch 47, Train loss 0.21435141906952426, Test loss 0.40347034934979337
Epoch 48, Train loss 0.21154671208833709, Test loss 0.3895075054495198
Epoch 49, Train loss 0.20860832766381535, Test loss 0.3934898465207428
```

[16]:
```
# Plotting tarining and testingloss
plt.plot(range(epoch_nums),train_loss_history,'-',linewidth=3,label='Train␣
 ↪error')
plt.plot(range(epoch_nums),test_loss_history,'-',linewidth=3,label='Test error')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.grid(True)
plt.legend()
```

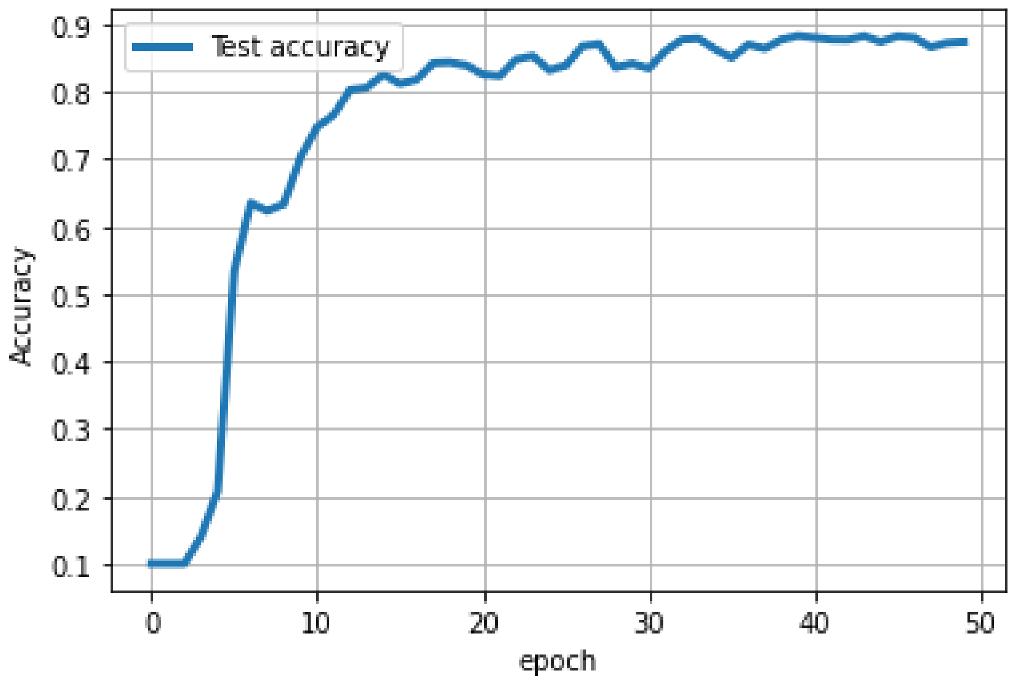
[16]:
```
<matplotlib.legend.Legend at 0x7fe1dd737990>
```



```
[32]: for i in range(epoch_nums):
        item = test_accuracy_history[i].cpu().numpy()
        test_accuracy_history[i] = item
```

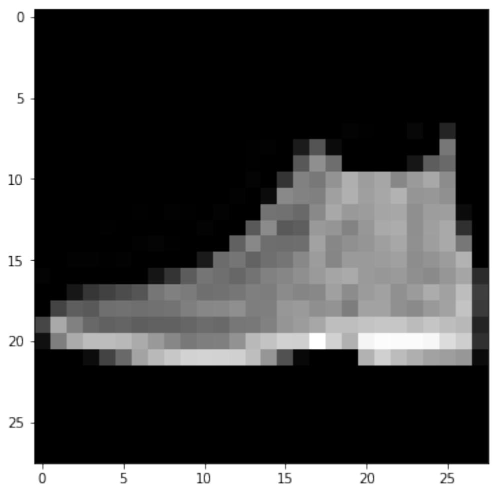
```
[34]: # Plotting testing accuracy
      plt.plot(range(epoch_nums),test_accuracy_history,'-',linewidth=3,label='Test␣
       ↪accuracy')
      plt.xlabel('epoch')
      plt.ylabel('Accuracy')
      plt.grid(True)
      plt.legend()
```

```
[34]: <matplotlib.legend.Legend at 0x7fe1dc05e150>
```

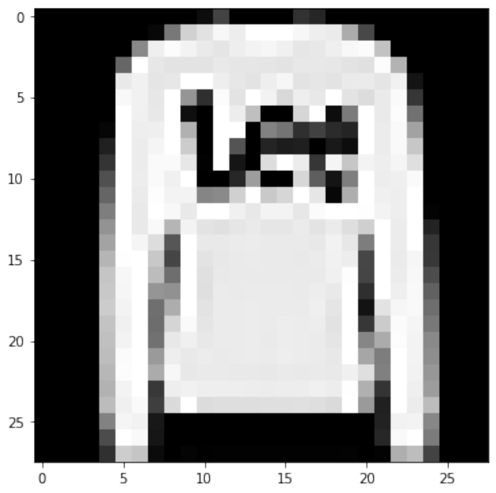


[51]:
```
# Select random 3 images from test dataset and get the predicted labels using␣
↪trained model
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal',␣
↪'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
random_3_dataLoader = torch.utils.data.
↪DataLoader(testdata,batch_size=3,shuffle=False)

random_3_images, random_3_labels = iter(random_3_dataLoader).next()
predicted_output = net_3_layers_perceptron(random_3_images.cuda())
predicted_labels = torch.max(predicted_output, 1)

# Print the prediction probabilities using softmax
softmax_layer = nn.Softmax(dim=1)
predicted_prob = softmax_layer(predicted_output)

# Plot selected images and its label, predicted label and prediction probability
plt.subplots_adjust(top=5)
for index in range(3):
  plt.subplot(3, 1, index+1)
  plt.imshow(random_3_images[index].cpu().squeeze().numpy(),cmap=plt.cm.gray)
  predicted_label = predicted_labels.indices[index]
  plt.title('The correct label is: %s.\n The correct label is: %s, its accuracy␣
↪is: %.5lf.\n' % (class_names[random_3_labels[index]],␣
↪class_names[predicted_label], predicted_prob[index][predicted_label]))
```

The correct label is: Ankle boot.
The correct label is: Ankle boot, its accuracy is: 0.99305.



The correct label is: Pullover.
The correct label is: Pullover, its accuracy is: 0.99924.



The correct label is: Trouser.
The correct label is: Trouser, its accuracy is: 1.00000.