INFO 284, Assignment 2

Bli014

April 2020

1 Introduction

This is my report for assignment 2 for the course INFO284. My model is saved in my google drive folder which can be found here. This is needed for task 3, in this folder you will find both my model and my ipynb file for this assignment.

2 Task one

For my classifier I decided to use three convolutional layers. This is to make the classifier as accurate as possible. The source code can be seen in the image underneath. The image also contains comments.

```
class NnDigitsClassifier(nn.Module):
    def __init__(self):
        super(NnDigitsClassifier, self).__init__()

        # Task 1: Implementing layers for classification. Uses sequential layering to gather conv-layer and pooling-layer in one.
        # The first layer.
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        # The second layer.
        self.layer2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        # The third layer.
        self.layer3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        self.drop_out = nn.Dropout(0.1)
        self.fc1 = nn.Linear(3 * 3 * 128, 100)
        self.fc2 = nn.Linear(100, NUM_CLASSES)


    # Defines the function "forward" which sends the information through the different layers.
    def forward(self, x):
      out = self.layer1(x)
      out = self.layer2(out)
      out = self.layer3(out)

      out = out.reshape(out.size(0), -1)

      out = self.drop_out(out)
      out = self.fc1(out)
      out = self.fc2(out)
      return out
```

Figure 1: Source code for the classifier

3 Task two

I tried changing my learning rate, my number of epochs, and the batch size to play around and see what kind of results I got. I found that having a large batch size (around 500) significantly reduced the accuracy by around 10 percent. As seen below, having a learning rate of 0.01 and 0.002 yielded worse results than

0.001, so that is why O choose 0.001 as the learning rate for this assignment I also ran a few tests with batch size of 30 and 100, with no large difference in accuracy between them as seen under. These images are examples of the different parameters I tested; the final result will be shown later.

```
# Task 2: Training the network
loss_list = []
acc_list = []
for epoch in range(num_epochs):
  for i, (images, labels) in enumerate(trainloader):

    # Runs the forward method.
    outputs = model(images)
    loss = criterion(outputs, labels)
    loss_list.append(loss.item())

    # Runs the Adam optimization.
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()


    # Tracks the accuracy.
    total = labels.size(0)
    _, x_predicted = torch.max(outputs.data, 1)
    correct = (x_predicted == labels).sum().item()
    acc_list.append(correct / total)


  print('Epoch [{}/{}] | Loss: {:.4f} | Accuracy: {:.2f}%'.format(epoch +1, num_epochs, loss.item(), (correct /total) * 100))
```

Figure 2: Trainer source code

```
Epoch [1/25], Loss: 0.3854, Accuracy: 84.00%
Epoch [2/25], Loss: 0.3295, Accuracy: 91.00%
Epoch [3/25], Loss: 0.3362, Accuracy: 88.00%
Epoch [4/25], Loss: 0.1772, Accuracy: 91.00%
Epoch [5/25], Loss: 0.1951, Accuracy: 93.00%
Epoch [6/25], Loss: 0.2354, Accuracy: 91.00%
Epoch [7/25], Loss: 0.2163, Accuracy: 93.00%
Epoch [8/25], Loss: 0.1946, Accuracy: 92.00%
Epoch [9/25], Loss: 0.1891, Accuracy: 91.00%
Epoch [10/25], Loss: 0.1787, Accuracy: 92.00%
Epoch [11/25], Loss: 0.3086, Accuracy: 91.00%
Epoch [12/25], Loss: 0.1470, Accuracy: 93.00%
Epoch [13/25], Loss: 0.1988, Accuracy: 92.00%
Epoch [14/25], Loss: 0.1349, Accuracy: 96.00%
Epoch [15/25], Loss: 0.1501, Accuracy: 96.00%
Epoch [16/25], Loss: 0.1609, Accuracy: 96.00%
Epoch [17/25], Loss: 0.0614, Accuracy: 98.00%
Epoch [18/25], Loss: 0.1200, Accuracy: 95.00%
Epoch [19/25], Loss: 0.1324, Accuracy: 94.00%
Epoch [20/25], Loss: 0.0917, Accuracy: 96.00%
Epoch [21/25], Loss: 0.1076, Accuracy: 98.00%
Epoch [22/25], Loss: 0.0427, Accuracy: 99.00%
Epoch [23/25], Loss: 0.0889, Accuracy: 96.00%
Epoch [24/25], Loss: 0.0750, Accuracy: 96.00%
Epoch [25/25], Loss: 0.0469, Accuracy: 99.00%
```

(a) Training set w/batch size = 100

```
Epoch [1/25] | Loss: 0.2424 | Accuracy: 93.33%
Epoch [2/25] | Loss: 0.4784 | Accuracy: 83.33%
Epoch [3/25] | Loss: 0.2008 | Accuracy: 93.33%
Epoch [4/25] | Loss: 0.3310 | Accuracy: 90.00%
Epoch [5/25] | Loss: 0.3330 | Accuracy: 90.00%
Epoch [6/25] | Loss: 0.1075 | Accuracy: 96.67%
Epoch [7/25] | Loss: 0.3066 | Accuracy: 86.67%
Epoch [8/25] | Loss: 0.0394 | Accuracy: 100.00%
Epoch [9/25] | Loss: 0.1859 | Accuracy: 93.33%
Epoch [10/25] | Loss: 0.1851 | Accuracy: 90.00%
Epoch [11/25] | Loss: 0.1184 | Accuracy: 93.33%
Epoch [12/25] | Loss: 0.0682 | Accuracy: 100.00%
Epoch [13/25] | Loss: 0.1301 | Accuracy: 90.00%
Epoch [14/25] | Loss: 0.1069 | Accuracy: 96.67%
Epoch [15/25] | Loss: 0.1922 | Accuracy: 96.67%
Epoch [16/25] | Loss: 0.1536 | Accuracy: 93.33%
Epoch [17/25] | Loss: 0.0518 | Accuracy: 96.67%
Epoch [18/25] | Loss: 0.1574 | Accuracy: 93.33%
Epoch [19/25] | Loss: 0.1234 | Accuracy: 93.33%
Epoch [20/25] | Loss: 0.0872 | Accuracy: 96.67%
Epoch [21/25] | Loss: 0.0816 | Accuracy: 96.67%
Epoch [22/25] | Loss: 0.0292 | Accuracy: 100.00%
Epoch [23/25] | Loss: 0.0406 | Accuracy: 96.67%
Epoch [24/25] | Loss: 0.1130 | Accuracy: 96.67%
Epoch [25/25] | Loss: 0.0339 | Accuracy: 100.00%
```

(b) Training set w/batch size = 30

```
Accuracy: 79.66%
```

(a) Test set w/Learning rate = 0.01

```
Accuracy: 11.96%
```

(b) Test Set w/Learning Rate = 0.002

4 Task four

After trying different parameters, I settled on a batch size of 30, a learning rate of 0.001, and an epoch size of 10. After computing 10 epochs on the training set, the accuracy is hovering between 83 percent and 100 percent. This can be seen in the picture below, as well as a confusion matrix. The result with a batch size of 30, a learning rate of 0.001 and an epoch amount of 10 the final result yielded 91 percent.

```
Accuracy: 91.06%
```

Figure 5: Final Training set accuracy

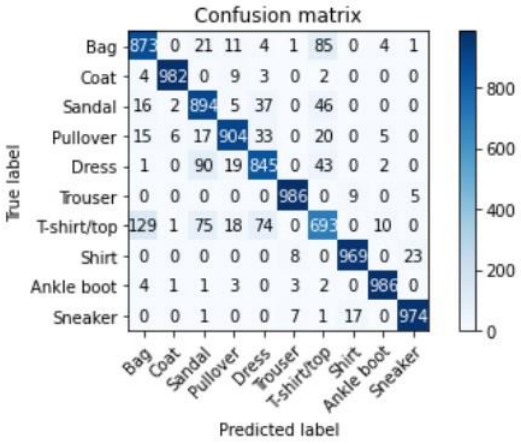

Figure 6: Confusion matrix