Lab 4: Convolutional Neural Networks: LeNet5

Bareera Mirza

Start your lab4 by downloading the attached files, reading through the instruction file(Lab4.pdf) and the programming file(lab4.ipynb). **You are expected to submit one .ipynb file that contains all your code, results and the answers to the following questions**:

1. Implement LeNet in PyTorch with tanh activation and Adam Optimizer, achieve greater than 95% accuracy on the MNIST classification task.

1. Import necessary libraries
    a. import torch,torch.nn as nn, torchvision, torchvision.transforms as transforms
2. Using torchvision , we will load the dataset
3. MNIST data can't be used as it is for the LeNet5 architecture. The LeNet5 architecture accepts the input to be 32x32 and the MNIST images are 28x28 so we resize it

```
# Use the following code to load and normalize the dataset

train_loader = torch.utils.data.DataLoader(

 torchvision.datasets.MNIST('/files/', train=True, download=True,

 transform=torchvision.transforms.Compose([torchvision.transforms.Resize((32, 32)),

                              torchvision.transforms.ToTensor(),

                              torchvision.transforms.Normalize(

                                (0.5,), (0.5,))
```

4. We will used testloader to impede performance
5. Next we define num_epoch=6, batch size=64 and learning rate=0.001
6. Then we will define architecture LeNET5.
    a. First we initiate it by _init_
    b. Then we give layers convolutional layers using the `nn.Conv2D` function with the appropriate kernel size and the input/output channels. We also apply max pooling using `nn.MaxPool2D` function. I used sequential but we can do separately as well

    c. Then we define fully connected layer (fc)
7. #Setting the loss function: cost = nn.CrossEntropyLoss()
8. #Setting the optimizer with the model parameters and learning rate: optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
9. #this is defined to print how many steps are remaining when training: total_step = len(train_loader)
10. Then we train the model:
    a. We start by iterating through the number of epochs, and then the batches in our training data.
    b. In the forward pass, we make predictions using our model and calculate loss based on those predictions and our actual labels.
    c. Next, we do the backward pass where we actually update our weights to improve our model
    d. We then set the gradients to zero before every update using `optimizer.zero_grad()` function.
    e. Then, we calculate the new gradients using the `loss.backward()` function.
    f. And finally, we update the weights with the `optimizer.step()` function.
    g. Next we test the model and here I have got 98.6% accuracy
    h. Then I have got probabilities and guesses for prediction

2. What values did you use for following hyperparameters when training LeNet that meets the accuracy requirement?

1) Learning Rate: 0.001

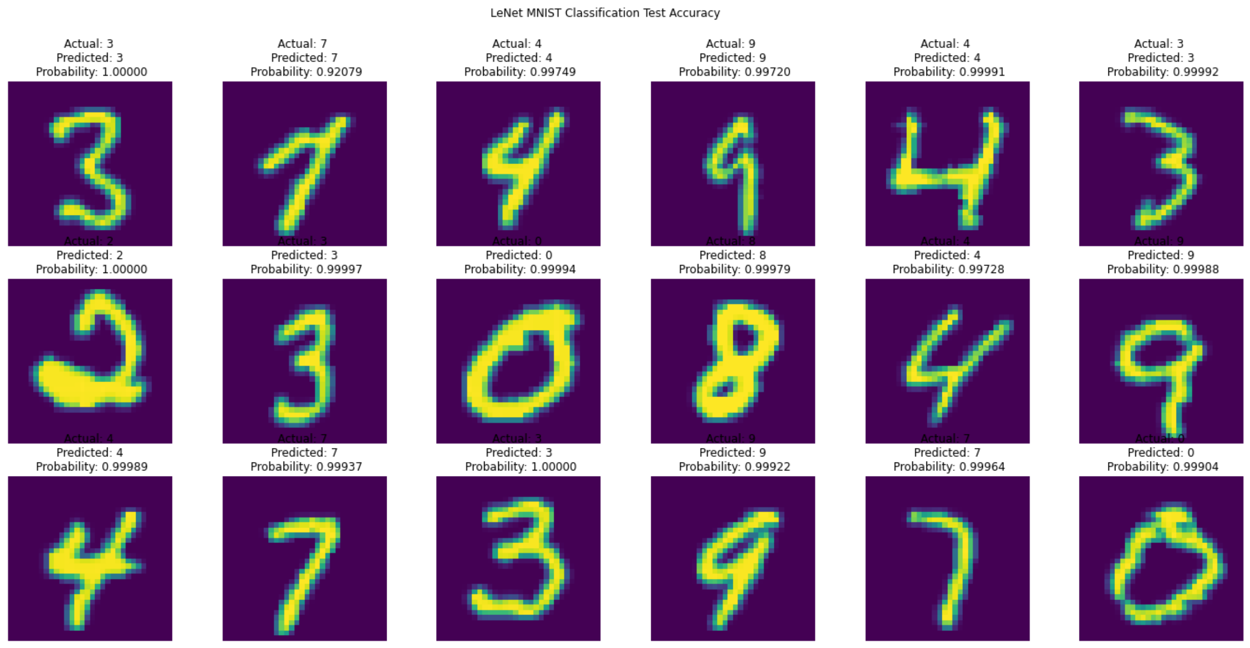
2) Batch Size: 64

3) Dropout (if any) after your convolutional layers

4) Any other hyperparameters you set manually (regularization, normalization, etc.)

I have used BatchNormal2D to increase the accuracy

3. Please plot the first 10 test images with a title that shows the probability of the predicted class.


LeNet MNIST Classification Test Accuracy
Actual: 3 Predicted: 3 Probability: 1.00000
Actual: 7 Predicted: 7 Probability: 0.92079
Actual: 4 Predicted: 4 Probability: 0.99749
Actual: 9 Predicted: 9 Probability: 0.99720
Actual: 4 Predicted: 4 Probability: 0.99991
Actual: 3 Predicted: 3 Probability: 0.99992
Actual: 2 Predicted: 2 Probability: 1.00000
Actual: 3 Predicted: 3 Probability: 0.99997
Actual: 0 Predicted: 0 Probability: 0.99994
Actual: 8 Predicted: 8 Probability: 0.99979
Actual: 4 Predicted: 4 Probability: 0.99728
Actual: 9 Predicted: 9 Probability: 0.99988
Actual: 4 Predicted: 4 Probability: 0.99989
Actual: 7 Predicted: 7 Probability: 0.99937
Actual: 3 Predicted: 3 Probability: 1.00000
Actual: 9 Predicted: 9 Probability: 0.99922
Actual: 7 Predicted: 7 Probability: 0.99964
Actual: 0 Predicted: 0 Probability: 0.99904