9. Convolutional Neural Networks

Project due Apr 5, 2023 08:59 -03 Completed

Next, we are going to apply convolutional neural networks to the same task. These netw
demonstrated great performance on many deep learning tasks, especially in computer

You will be working in the files `part2-mnist/nnet_cnn.py` and `part2-mnist/train` problem

Convolutional Neural Networks

3.0/3.0 points (graded)

We provide skeleton code `part2-mnist/nnet_cnn.py` which includes examples of son layers you will need in this part. Using the PyTorch Documentation, complete the code convolutional neural network with following layers in order:

- A convolutional layer with 32 filters of size 3×3
- A ReLU nonlinearity
- A max pooling layer with size 2×2
- A convolutional layer with 64 filters of size 3×3
- A ReLU nonlinearity
- A max pooling layer with size 2×2
- A flatten layer
- A fully connected layer with 128 neurons
- A dropout layer with drop probability 0.5
- A fully-connected layer with 10 neurons

Note: We are not using a softmax layer because it is already present in the loss: PyTorc `nn.CrossEntropyLoss` combines `nn.LogSoftMax` with `nn.NLLLoss`.

Without GPU acceleration, you will likely find that this network takes quite a long time t we don't expect you to actually train this network until convergence. Implementing the you get approximately 93% **training accuracy** and 98% **validation accuracy** after one should take less than 10 minutes) is enough for this project. If you are curious, you can longer; if implemented correctly, your model should achieve >99% **test accuracy** after you have access to a CUDA compatible GPU, you could even try configuring PyTorch to

```
12          )
13
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

Submit

Previous Next