9. Convolutional Neural Networks

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

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

Convolutional Neural Networks

3.0/3.0 points (graded)

We provide skeleton code `part2-mnist/nnet_cnn.py` which includes examples of some (**not all**) of the new layers you will need in this part. Using the PyTorch Documentation, complete the code to implement a 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: PyTorch's `nn.CrossEntropyLoss` combines `nn.LogSoftMax` with `nn.NLLLoss`.

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

After you successfully implement the above architecture, copy+paste your model code into the codebox below for grading.

Grader note:: If you get a NameEror “Flatten" not found, make sure to unindent your code.

Available Functions: You have access to the `torch.nn` module as `nn` and to the `Flatten` layer as `Flatten`; No need to import anything.

```
model = nn.Sequential(
              nn.Conv2d(1, 32, (3, 3)),
              nn.ReLU(),
              nn.MaxPool2d((2, 2)),
              nn.Conv2d(32, 64, (3, 3)),
              nn.ReLU(),
```

```
            nn.MaxPool2d((2, 2)),
            Flatten(),
            nn.Linear(1600, 128),
            nn.Dropout(p = 0.5),
            nn.Linear(128, 10)
          )
```

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

Correct

```
model = nn.Sequential(
          nn.Conv2d(1, 32, (3, 3)),
          nn.ReLU(),
          nn.MaxPool2d((2, 2)),
          nn.Conv2d(32, 64, (3, 3)),
          nn.ReLU(),
          nn.MaxPool2d((2, 2)),
          Flatten(),
          nn.Linear(1600, 128),
          nn.Dropout(0.5),
          nn.Linear(128, 10),
        )
```

Test results

See full output

```
CORRECT
```

See full output

Submit You have used 2 of 20 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 9. Convolutional Neural Networks