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

3/3 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.

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(0.5),
          nn.Linear(128, 10),
        )
```

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

Correct

Test results

Hide output

CORRECT

`Test: has 11 layers`

Testing has 11 layers

Output:

```
Length:  10
```

```
Test: has correct layers
```

```
Testing has correct layers
```

Output:

```
Layers:
Conv2d
ReLU
MaxPool2d
Conv2d
ReLU
MaxPool2d
Flatten
Linear
Dropout
Linear
```

```
Test: has correct number of parameters
```

```
Testing has correct number of parameters
```

Output:

```
Number of parameters:
225034
```

```
Test: has correct output
```

Testing has correct output

`Output:`

```
model(a):  tensor([[ 2.4448,  1.5225,  2.1802,  4.4122,  0.5503,  1.3307, -2.1938,  2.3987,
         -2.9601, -0.4598]])
a[:,:,:5] tensor([[[[ 1.1542e-01,  1.3941e+00, -2.7714e-01,  4.2701e-01,  9.3229e-01,
            2.3895e-01,  3.2457e-01, -3.4939e-01,  1.9670e-02,  2.2480e-01,
            1.9173e+00,  2.2247e-01, -4.6467e-01, -7.6888e-01,  3.0430e-02,
            9.3643e-01,  7.7472e-02,  3.7119e-01, -1.7743e+00, -7.9063e-01,
            3.8385e-02,  6.5397e-01, -8.9731e-02, -1.4515e-01,  3.5549e-01,
           -7.8692e-01, -5.8899e-04, -4.8655e-01],
          [ 1.3405e+00,  1.2712e+00,  3.7503e-01,  8.9544e-01, -4.0794e-01,
            2.1923e+00, -6.6135e-01, -2.2415e+00, -1.4822e+00, -5.5531e-01,
            8.7844e-01,  8.4059e-01,  5.6392e-01,  1.8175e-01,  5.1660e-01,
            3.1636e-01,  7.3709e-01,  2.5617e-02, -2.5206e-01, -4.5273e-01,
            1.5415e+00, -1.5221e+00, -7.3029e-01,  2.2291e-01,  4.1424e-01,
           -3.2722e-01, -1.5834e-01, -4.2584e-01],
          [-5.0879e-01, -6.4091e-01, -2.4095e-01, -2.6180e-01,  9.3761e-01,
           -1.3980e-02, -7.2690e-01, -8.9905e-02,  2.6777e-01,  1.2556e-01,
            1.4992e+00, -8.0736e-01,  3.7684e-01,  2.7557e-01,  1.7586e-01,
           -7.9403e-01,  4.4957e-01,  7.0530e-02,  1.3219e-01,  5.7742e-01,
            1.0787e-01, -1.1266e+00, -3.7247e-01, -1.6174e+00,  6.5828e-01,
           -7.1602e-01, -6.8528e-01,  6.9362e-01],
          [-1.2379e+00,  2.4469e-01,  2.4287e-01, -2.3104e+00, -1.9266e-02,
            4.6654e-01,  6.2467e-01, -1.1184e+00,  8.1859e-01,  6.2088e-01,
            6.2070e-01,  1.7712e-01, -3.1773e-01, -6.1871e-01, -5.3287e-01,
            1.0487e+00,  1.8772e+00,  9.3554e-01, -1.2786e-02, -7.4107e-02,
            1.0031e+00, -9.3705e-01, -1.1818e+00,  8.6155e-01, -1.1396e+00,
            3.6708e-01, -7.9953e-01, -7.9232e-01],
          [ 3.1212e-02, -7.6052e-01, -3.5220e-01,  1.0426e-01,  8.0699e-01,
            9.4311e-01,  1.9873e+00,  1.5439e+00, -9.5529e-02, -1.6095e+00,
           -6.9276e-01, -7.7082e-01,  4.8388e-02, -8.1956e-02,  1.2461e-02,
           -1.0657e+00,  8.7008e-01, -1.5414e+00, -1.3320e+00, -6.0466e-01,
           -1.1799e-01, -5.1236e-01,  1.5981e-01,  7.8021e-01, -2.4048e-01,
           -9.0496e-01, -5.5437e-01, -1.3648e+00]]]])
```

Hide output

Submit You have used 7 of 20 attempts

✓ Correct (3/3 points)

Discussion

Hide Discussion

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

Add a Post

Show all posts ▼ by recent activity ▼

[Staff] There is a grader problem (NameError: name 'Flatten' is not defined).
Dear Staff, It seems something wrong with `Flatten` definition in the grader. I've got the following problem twice: Traceback (most recent call last): Fi... 1 new

? Grader processing too long for 9. Convolutional Neural Networks
I submitted the code, and the grader has already been running for 1 hour, .. and I cannot submit new version of code... is that normal? 23

Useful link for those struggling like me
I found the following link useful https://blog.algorithmia.com/convolutional-neural-nets-in-pytorch/ 4

? flatten and fully connected layers
1. If I understand correctly, fully connected layer is a Linear layer. We are given only one dimension of it (128). What is the other param ? Is it 10 (sa... 3

? what model code needs to be pasted?
I created a class for my model as described in this article [Convolutional Neural Networks Tutorial in PyTorch][1] <br> How do I post the code into gr... 6

Dimensions
Although through the log files I could figure the size of the fully connected layer I'm not sure I'm able to reconstruct it myself... so just checking The i... 3 new

Some interesting read about dropout layer
This article gives a very good explanation about what is dropout and why we need it in our nnet. https://github.com/vinsis/understanding-neuralnet... 1
Community TA

?
name 'Flatten' is not defined
Im calling Flatten() in my code. After a long run, the grader tells me "name 'Flatten' is not defined" it works in the project-files. Do I need to add an im... 2