Thomas Ehret

Assignment 8

CS519 SP23

Dr. Cao

Convolutional Neural Networks with PyTorch

Per the assignment specifications the CNN diagram is visualized in figure 1

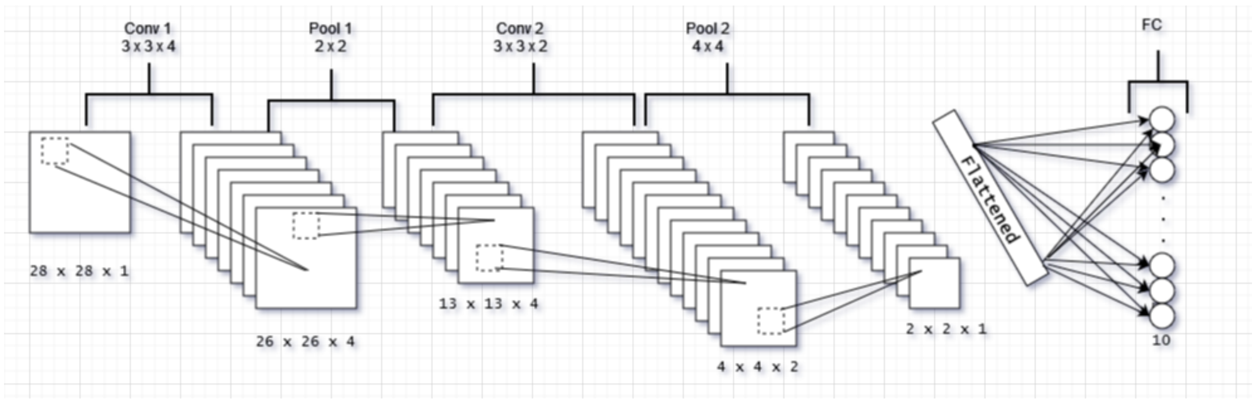


Figure 1

Initial Performance of CNN on MNIST

Time is calculated for each training epoch and then the mean is computed, this is to assure that no outliers are skewing the total computation time. The classification accuracy and total time is computed at the end of each test. Using the default parameters over 10 epochs the results are shown in table 1.

batch_size = 64

num_epochs = 10

learning_rate = 0.001

convolutional_1(1, 4, kernel_size=3, stride=1, padding_mode='zeros')

convolutional_2(4, 2, kernel_size=3, stride=3, padding_mode='zeros')

pool_1(kernel_size=2, stride=2)

pool_2(kernel_size=4, stride=4)

Metric	**CNN**
1. Median Epoch Time	12 s
2. Total Time	68 s
3. Accuracy	38%

Table 1

To check for performance variation over time, the number of epochs was increased from 10 to 50 with only a slight improvement in the accuracy to 43%. Further increases in epochs showed greatly diminishing returns with a linear growth in computation time. Overall, the accuracy increase is at its greatest when epochs is doubled from 5 to10 and then to 20, after that there is little actual gain.

Metric	**CNN**
1. Median Epoch Time	12 s
2. Total Time	638 s
3. Accuracy	43%

Table 2

<u>Further Testing</u>

As a first experiment the learning rate was changed from 0.001 -> 0.01 in increments of 0.001. This had slight gains in accuracy at 0.003 -> 0.005 (53%), but beyond that it had statistically insignificant or worse classification accuracy rate. Decreasing the learning rate from 0.001 -> 0.0001 in increments of -0.0001 increased the accuracy slightly but required many more epochs to produce a similar accuracy result to 5 epochs at 0.001 One could surmise that the learning rate doesn't play an overly significant role in convergence.

Changing the pool function parameters didn't seem to have as much of an effect. In contrast, modifying the kernel and corresponding input/output channels significantly improved the classification performance. Furthermore, convergence was reached in far fewer epochs. Results are displayed in Table 3. Example code testing output for *epochs = 2* is shown in Figure 2

Parameters:

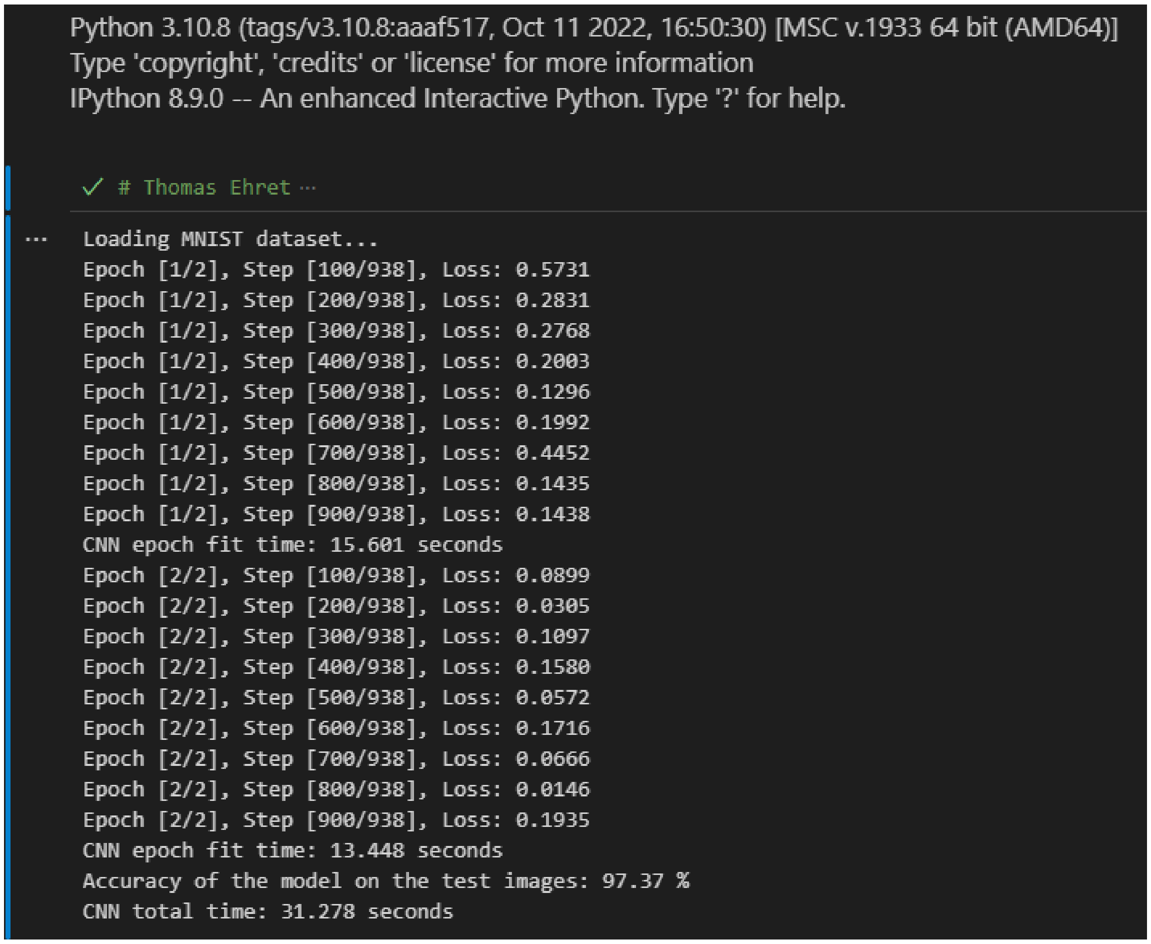
epochs = 2 -> 10

convolutional_1 = (1, 32, kernel_size=5, stride=1, padding=1)

convolutional_2 = (32, 64, kernel_size=5, stride=3, padding=1) *fc =(64*1*1, 128)*

Metric	CNN
1. Median Epoch Time	13 s
2. Total Time	72 s
3. Accuracy	98%

Table 3



```
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.9.0 -- An enhanced Interactive Python. Type '?' for help.

✓ # Thomas Ehret ···

··· Loading MNIST dataset...
Epoch [1/2], Step [100/938], Loss: 0.5731
Epoch [1/2], Step [200/938], Loss: 0.2831
Epoch [1/2], Step [300/938], Loss: 0.2768
Epoch [1/2], Step [400/938], Loss: 0.2003
Epoch [1/2], Step [500/938], Loss: 0.1296
Epoch [1/2], Step [600/938], Loss: 0.1992
Epoch [1/2], Step [700/938], Loss: 0.4452
Epoch [1/2], Step [800/938], Loss: 0.1435
Epoch [1/2], Step [900/938], Loss: 0.1438
CNN epoch fit time: 15.601 seconds
Epoch [2/2], Step [100/938], Loss: 0.0899
Epoch [2/2], Step [200/938], Loss: 0.0305
Epoch [2/2], Step [300/938], Loss: 0.1097
Epoch [2/2], Step [400/938], Loss: 0.1580
Epoch [2/2], Step [500/938], Loss: 0.0572
Epoch [2/2], Step [600/938], Loss: 0.1716
Epoch [2/2], Step [700/938], Loss: 0.0666
Epoch [2/2], Step [800/938], Loss: 0.0146
Epoch [2/2], Step [900/938], Loss: 0.1935
CNN epoch fit time: 13.448 seconds
Accuracy of the model on the test images: 97.37 %
CNN total time: 31.278 seconds
```

Figure 2