Image Analysis: Report on LAB 2

LEARNING AND CONVOLUTIONAL NEURAL NETWORKS

Group Number 45

Kashif Shabir

Haitham Babbili

Olalekan Peter Adare

Exercise 2.3

We divided our given data into three (3) parts:

Training data- 50%

Validation Data- 25%

Testing Data- 25%

The used distribution used is 50% 25% 25%, between training, testing, validation. This should give us the best result, and it is guarantees cross-validation of 75%-25%. Since we are going to randomize the data latter on, for selection purposes, it should give us a better result.

Exercise 2.6

We discovered that our given value, **s,** is a sliding-window. We are given values s= {0.01,10, 1,0.1}

We started with s=0.01, then s= 0.1, s=1, and s=10. It works like a scanning filter. We selected our learning rate, **lrate**, to be 0.1. The change in the value of s, with a fixed learning rate, gives different results. We have the pictures saved with our report. The larger the value of s, the lower the recognition quality of the picture, as the iterations increase, as well. With higher s values, we just have pixels, without a recognizable feature or shape.

Exercise 2.7

We made two function files, **process_epoch.m** and **process_epoch2.m**, to test initialize our lab to select data randomly and sequentially, respectively.

Exercise 2.8

Matlab simulation gave maximum accuracy of 98%

Exercise 2.9

With learning rate of 0.001, Matlab simulation still gave maximum accuracy of 98%. We felt it should have been a bit higher.

Exercise 2.14

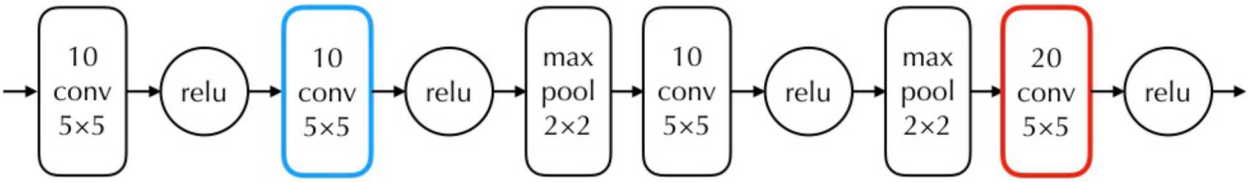
There are 13 layers that are trainable.

	1
1	1x1 ImageInputLayer
2	1x1 Convolution2DLayer
3	1x1 ReLULayer
4	1x1 MaxPooling2DLayer
5	1x1 Convolution2DLayer
6	1x1 ReLULayer
7	1x1 MaxPooling2DLayer
8	1x1 Convolution2DLayer
9	1x1 ReLULayer
10	1x1 DropoutLayer
11	1x1 FullyConnectedLayer
12	1x1 SoftmaxLayer
13	1x1 ClassificationOutputLayer
14	
15	
16	
17	
18	
19	

Calculated weight = 1+(20*(5*5) *1) + 1+(10*(2*2) *1)

Bias + (Dimensions x Filters)

Exercise 2.16



convolution2dLayer(5, 10), the elapsed time is 0.003789 seconds

While for,

convolution2dLayer(5, 20), the elapsed time is 0.003611 seconds

So, we have the same size, 5x5, but with different convolution layers. The time difference may not be much, but in a real-life this can be very significant considering the amount of data involved.

Therefore, the volume of information and computations fed into the blue (**convolution2dLayer(5, 10),** is far lower than what is fed into red, **convolution2dLayer(5, 20).** Therefore, the computation time for the red component is more and the time taken is higher.

Tic-toc approach was used to calculate how fast the set-up works. The parameters of the red layer are more than the blue layer. The red layer appears to be faster in processing than the blue layer.

Exercise 2.17

When we replace 5*5, 10 convolutional boxes with two 3*3, 10 convolutional boxes we need to change the 2nd max pool's dimension from (2,2) to (1,1) and same changes for the last max pool but rest remain same. The accuracy was reduced and it also took more computational time to perform because we added one more layer as compared to the previous scenario. As we increased the layers and we decreased the convolutional size that's why it takes more time and give us less accuracy.

```
Initializing input data normalization.\
|=========================================================================================|\
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |\
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |\
|=========================================================================================|\
|       1 |           1 |       00:00:00 |       14.06% |       2.3035 |          0.1400 |\
|       2 |          50 |       00:00:02 |       50.00% |       1.4981 |          0.0980 |\
|       3 |         100 |       00:00:05 |       75.00% |       0.7615 |          0.0686 |\
|       5 |         150 |       00:00:07 |       84.38% |       0.3698 |          0.0336 |\
|       5 |         185 |       00:00:09 |       95.31% |       0.1047 |          0.0336 |\
|=========================================================================================|\
95.0667% of the validation numbers where classified correctly}
```

Exercise 2.18

```
Initializing input data normalization.\
|=========================================================================================|\
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |\
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |\
|=========================================================================================|\
|       1 |           1 |       00:00:00 |        8.59% |       2.3028 |          0.1400 |\
|       2 |          50 |       00:00:10 |       82.03% |       0.6024 |          0.0980 |\
|       3 |         100 |       00:00:21 |       90.62% |       0.2288 |          0.0686 |\
|       5 |         150 |       00:00:32 |      100.00% |       0.0230 |          0.0336 |\
|       5 |         185 |       00:00:40 |       98.44% |       0.0608 |          0.0336 |\
|=========================================================================================|\
100% of the validation numbers where classified correctly\
`
```

100% (at a few times, we had 100%, others were 98% or 99%)

Exercise 2.19

99.22% was classified accurately (the best value after many trial runs)