# 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

Running the command **analyzeNetwork(net)** in the command window, will give a GUI with breakdown of the trainable parameters

ANALYSIS RESULT

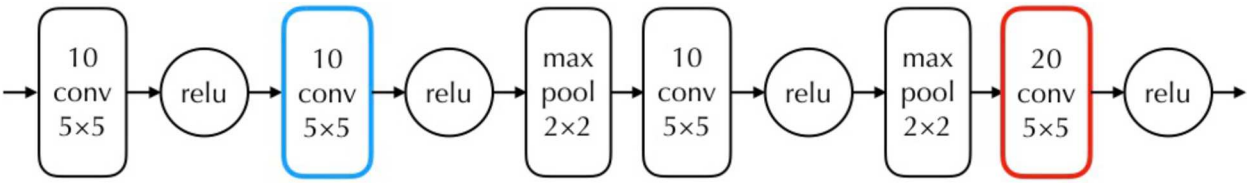
| | Name | Type | Activations | Learnables |
|---|---|---|---|---|
| 1 | imageinput<br>28x28x1 images with 'zerocenter' normalization | Image Input | 28×28×1 | - |
| 2 | conv<br>20 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0] | Convolution | 24×24×20 | Weights 5×5×1×20<br>Bias 1×1×20 |
| 3 | relu<br>ReLU | ReLU | 24×24×20 | - |
| 4 | maxpool<br>2x2 max pooling with stride [2 2] and padding [0 0 0 0] | Max Pooling | 12×12×20 | - |
| 5 | fc<br>10 fully connected layer | Fully Connected | 1×1×10 | Weights 10×2880<br>Bias 10×1 |
| 6 | softmax<br>softmax | Softmax | 1×1×10 | - |
| 7 | classoutput<br>crossentropyex with '0' and 9 other classes | Classification Output | - | - |

(Dimensions x Filters) + Bias

calculated_weight = (20*(5*5) *1)+1 + (10*(2*2) *1)+1

The calculated weight is 542

## Exercise 2.16




blue_layer = (10*5*5*1) +1;

** This implies 250 multiplications and 251 additions

red_layer = (20*5*5*1) +1;

** This implies 500 multiplications and 501 additions.

The Blue layer time consumption is c250

The Blue layer needs c250Npix calculations = 25*250Npix = 6250Npix (input layer is 5x5)

The Red layer time consumption is c500

The Red layer requires c500Npix calculations = 4*500Npix = 2000Npix (input layer is 2x2)

The red layer is like three times faster than the blue layer
It consists of double the number of parameters as compared to 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.

The total number of calculations is 25*90Npix + 9*30Npix = 2520Npix

The two 10 3x3 layers offers about 40 percent faster setup than the ordinary setup.

Also, it has 180 parameters which is about 30 percent lower than ordinary setup.

This should amount to loss of some accuracy in detecting larger objects in a given image. This accounts for lower percentage in classification

**Exercise 2.18**

The better_cnn_classifier.m function file returned 100 %

**Exercise 2.19**

99.6% was classified accurately (the best value after many trial runs)