

CMPT412 Project 1 Report

Ritika Goyal (301401516)

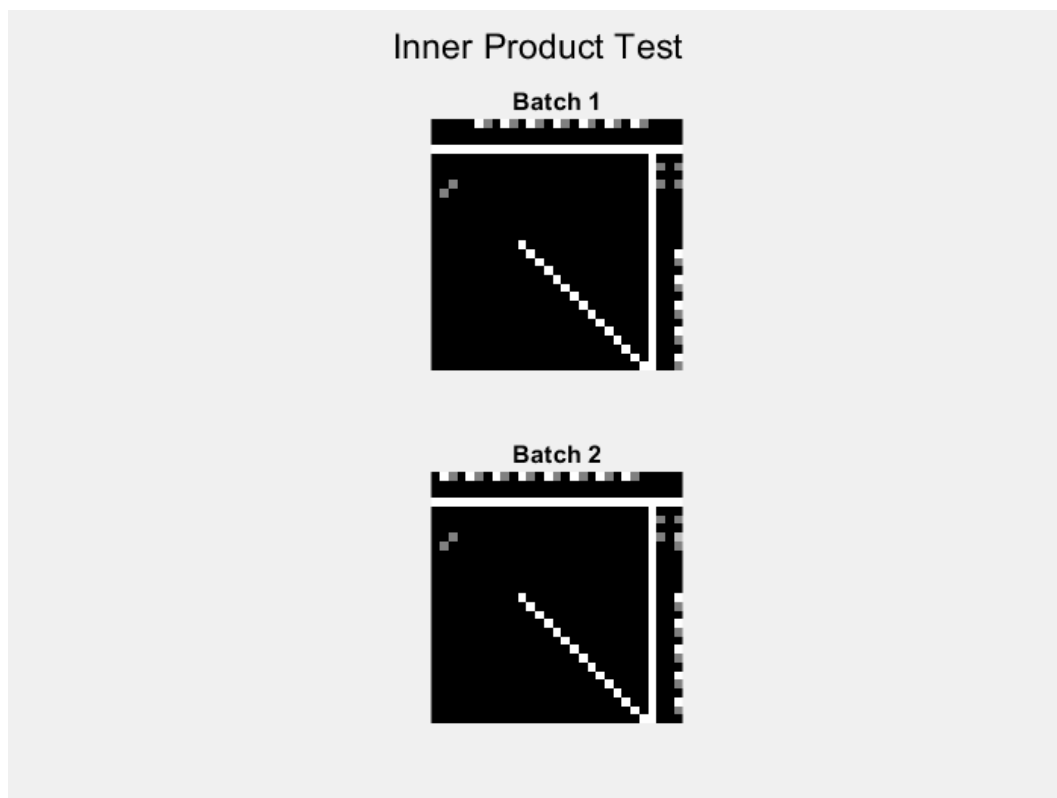
Part 1: Forward Pass

Q 1.1 Inner Product Layer:

In this layer, the output is calculated using $Wx + b$, where W is weight matrix, x is input, and b is bias.

First the outputs were specified and then all the images were traversed in a batch and output was calculated.

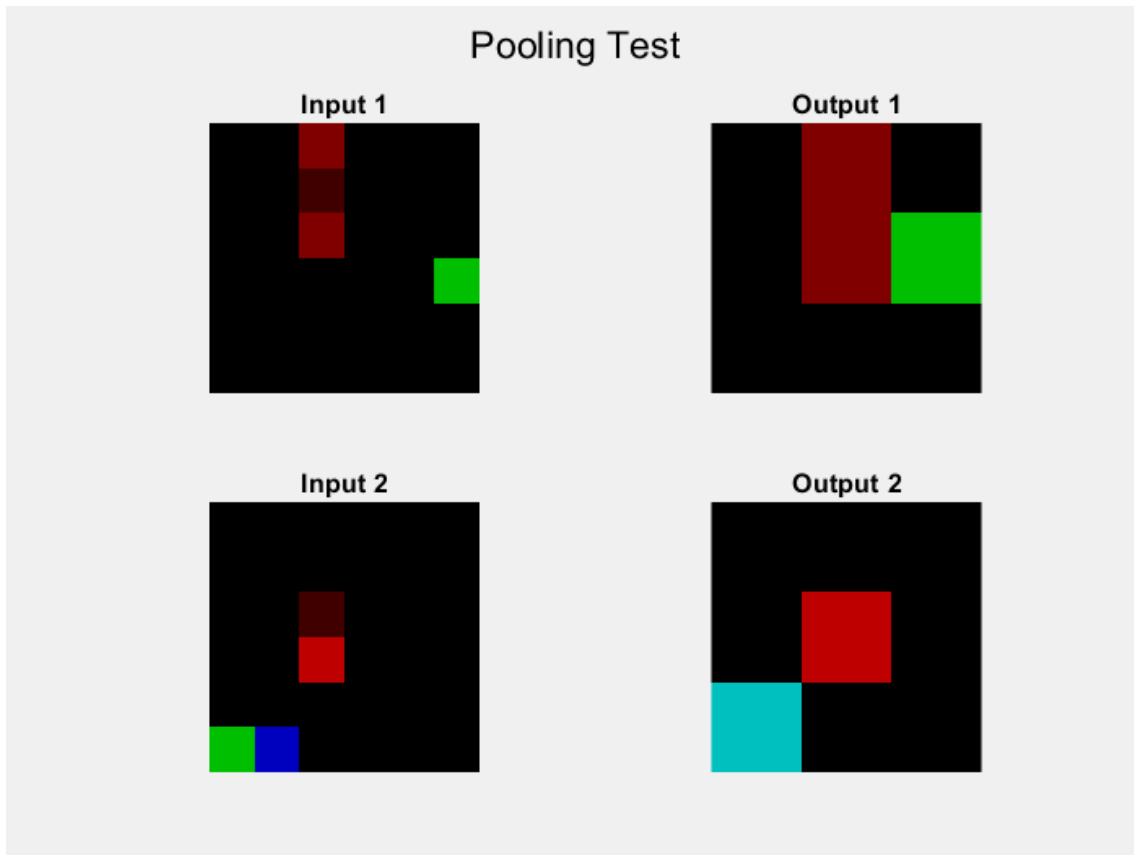
When test_components.m was executed, the inner product was visualized by following image.



Q 1.2: Pooling Layer:

In the pooling layer, the max pooling was used which reduces the size of feature maps by finding max value in each kernel.

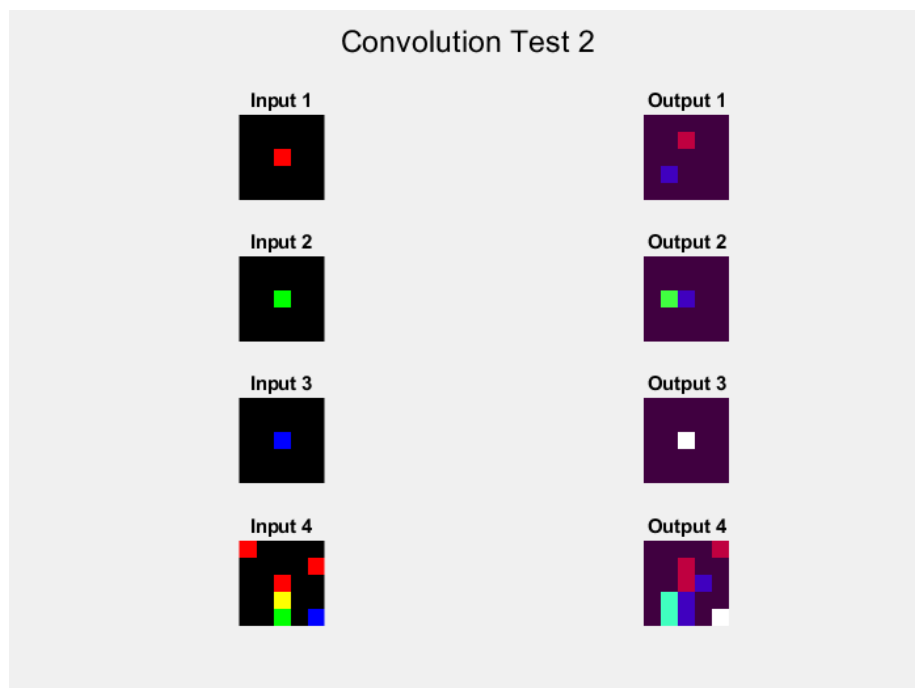
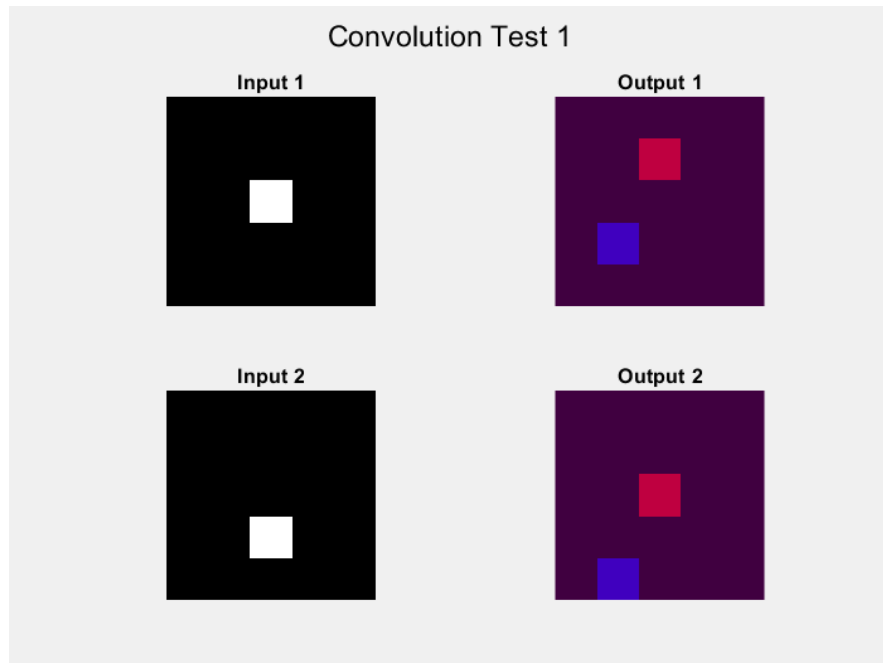
When test_components.m was executed, the polling was visualized by following image.



Q 1.3 Convolution Layer:

Convolution is performed using $k \times k$ filters and a $W \times H$ image. After convolution, feature map is obtained. Convolution layers reduces parameters and exploits spatial structure.

When test_components.m was executed, the convolution was visualized by following image.



Q 1.4 ReLU:

In Relu, $f(x) = \max(x, 0)$ was used. If input is greater than or equal to 0, then output will be x (equal to input) but if input is less than 0, then the output will be 0 after this layer.

PART 2: BACK PROPAGATION

Q 2.1 ReLU:

The negative values are replaced by 0 and positive values are kept in matrix which are then multiplied element wise with output and result is stored in input_od.

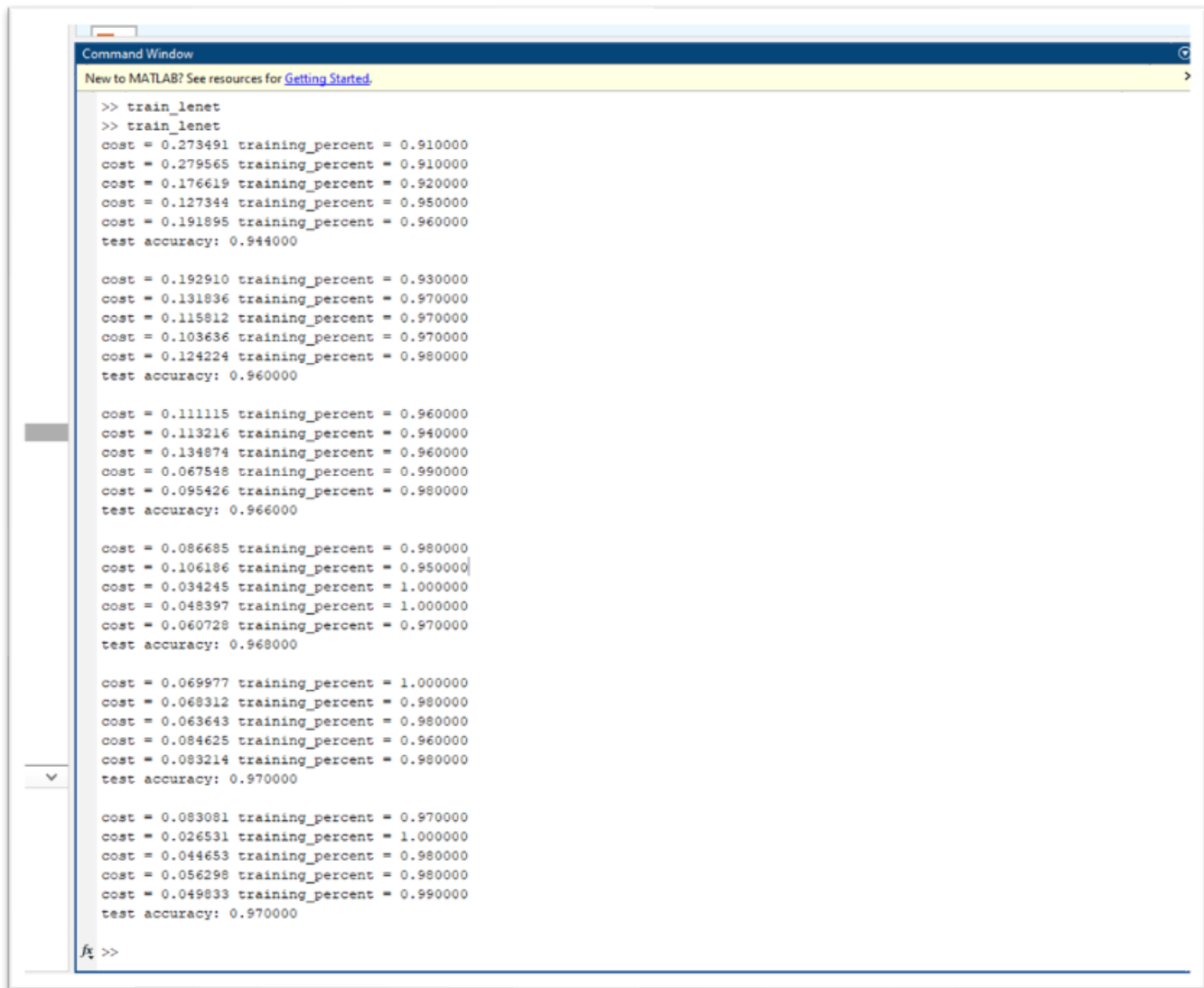
Q 2.2 Inner :Product Layer:

The differentiation output.diff is calculated.

PART 3: TRAINING

Q 3.1: Training:

After running `train_lenet.m`, the test accuracy of the CNN network is 97%.



```
Command Window
New to MATLAB? See resources for Getting Started.

>> train_lenet
>> train_lenet
cost = 0.273491 training_percent = 0.910000
cost = 0.279565 training_percent = 0.910000
cost = 0.176619 training_percent = 0.920000
cost = 0.127344 training_percent = 0.950000
cost = 0.191895 training_percent = 0.960000
test accuracy: 0.944000

cost = 0.192910 training_percent = 0.930000
cost = 0.131836 training_percent = 0.970000
cost = 0.115812 training_percent = 0.970000
cost = 0.103636 training_percent = 0.970000
cost = 0.124224 training_percent = 0.980000
test accuracy: 0.960000

cost = 0.111115 training_percent = 0.960000
cost = 0.113216 training_percent = 0.940000
cost = 0.134874 training_percent = 0.960000
cost = 0.067548 training_percent = 0.990000
cost = 0.095426 training_percent = 0.980000
test accuracy: 0.966000

cost = 0.086685 training_percent = 0.980000
cost = 0.106186 training_percent = 0.950000
cost = 0.034245 training_percent = 1.000000
cost = 0.048397 training_percent = 1.000000
cost = 0.060728 training_percent = 0.970000
test accuracy: 0.968000

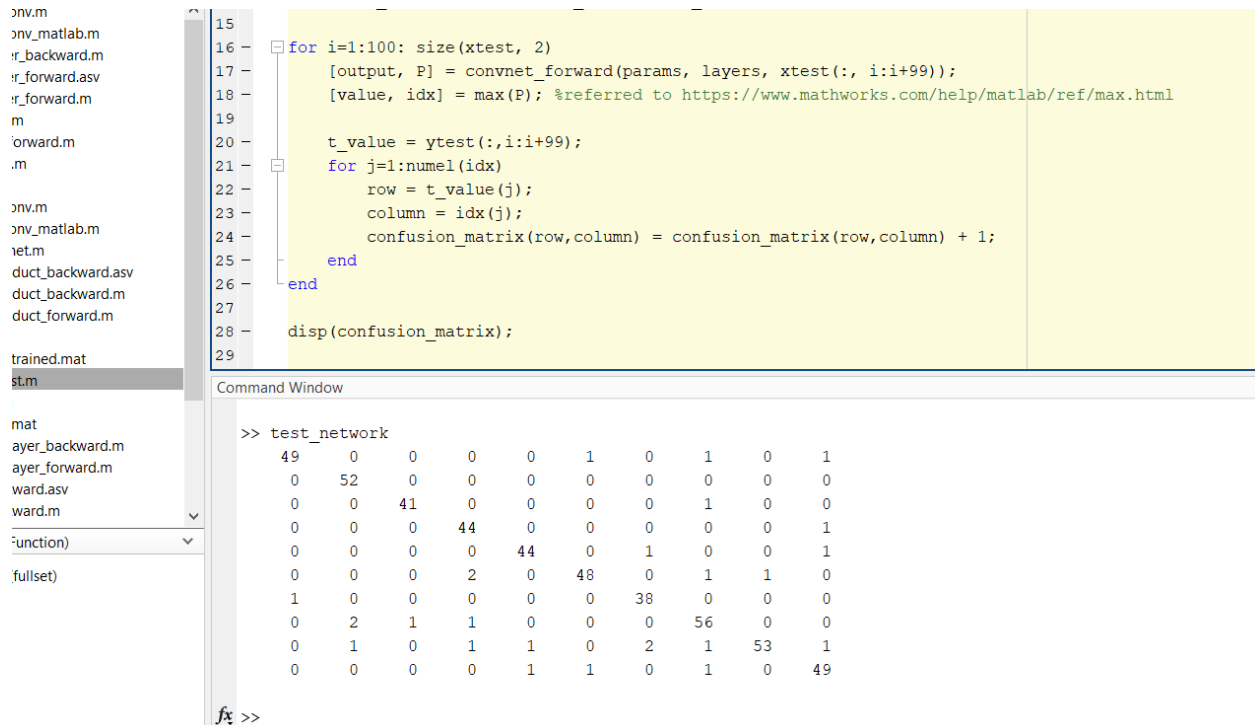
cost = 0.069977 training_percent = 1.000000
cost = 0.068312 training_percent = 0.980000
cost = 0.063643 training_percent = 0.980000
cost = 0.084625 training_percent = 0.960000
cost = 0.083214 training_percent = 0.980000
test accuracy: 0.970000

cost = 0.083081 training_percent = 0.970000
cost = 0.026531 training_percent = 1.000000
cost = 0.044653 training_percent = 0.980000
cost = 0.056298 training_percent = 0.980000
cost = 0.049833 training_percent = 0.990000
test accuracy: 0.970000

fx >>
```

Q 3.2: TEST THE NETWORK

When test_network.m was executed, the following confusion matrix was obtained:



```
15
16 for i=1:100: size(xtest, 2)
17     [output, P] = convnet_forward(params, layers, xtest(:, i:i+99));
18     [value, idx] = max(P); %referred to https://www.mathworks.com/help/matlab/ref/max.html
19
20     t_value = ytest(:,i:i+99);
21     for j=1:numel(idx)
22         row = t_value(j);
23         column = idx(j);
24         confusion_matrix(row,column) = confusion_matrix(row,column) + 1;
25     end
26 end
27
28 disp(confusion_matrix);
29
```

Command Window

```
>> test_network
49    0    0    0    0    1    0    1    0    1
0   52    0    0    0    0    0    0    0    0
0    0   41    0    0    0    0    1    0    0
0    0    0   44    0    0    0    0    0    1
0    0    0    0   44    0    1    0    0    1
0    0    0    2    0   48    0    1    1    0
1    0    0    0    0    0   38    0    0    0
0    2    1    1    0    0    0   56    0    0
0    1    0    1    1    0    2    1   53    1
0    0    0    0    1    1    0    1    0   49
```

fx >>

It is obvious that if each time the test_network.m will be executed, the confusion_matrix would be different. According to above confusion matrix, the two most confused pairs were (2,7) and (4,9). Since these numbers can be easily mistaken by each other, so this confusion is justified.

Q 3.3 REAL WORLD TESTING

I downloaded 5 images and saved them in images/real_images. These images were passed through the network and some of them were correctly predicted.



Output -> 3



Output -> 3



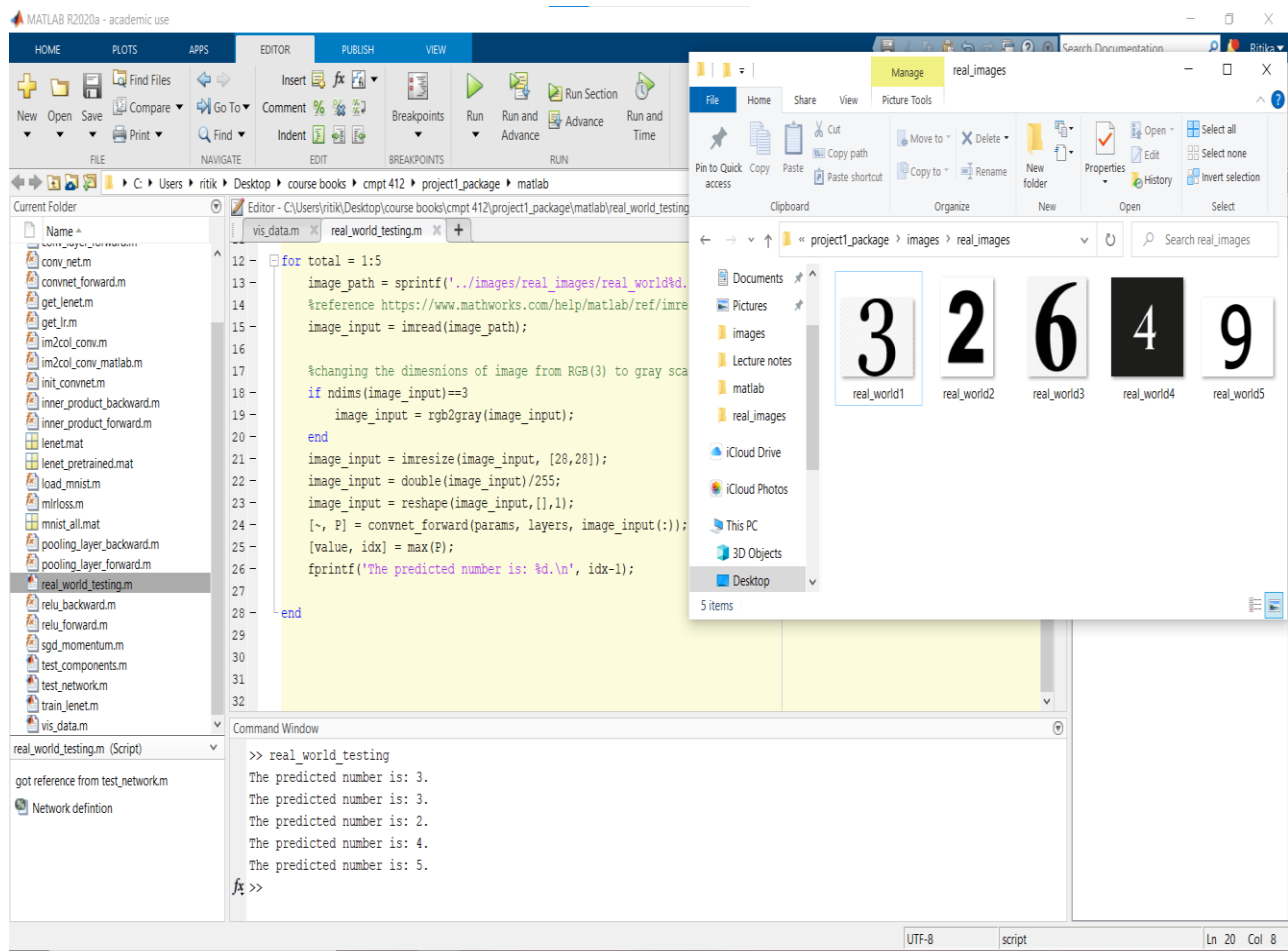
Output -> 2



Output ->4

9

Output->5

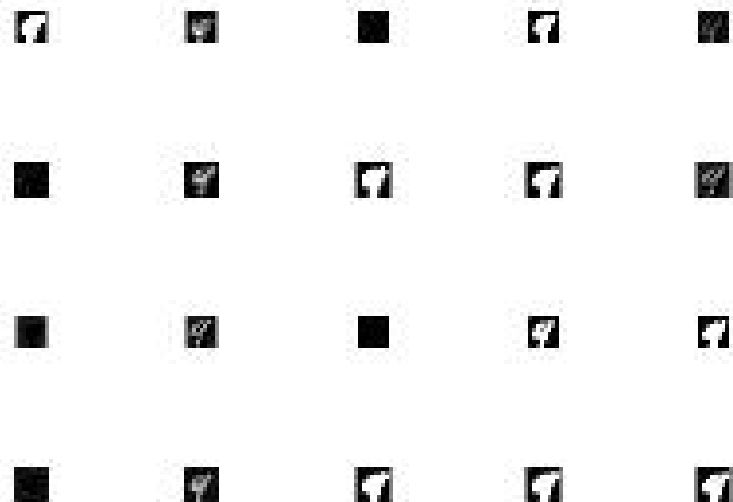


Part 4: VISUALIZATION

Q 4.1:

After running vis_data.m, following outputs were obtained for convolution and Relu Layer.

CONV Layer:



ReLU Layer:



Q 4.2:

The layer filters are applied for the original image which is shown in feature map. The original image was digit 9. After passing the original image from both the layers, it is seen that both the layers are different. The size of the image changes from Conv to Relu. Relu adds more darkness to the image because Relu focuses to increase the non-linearity of images.

PART 5: IMAGE CLASSIFICATION

First the image is converted to greyscale by using threshold function. Then the different connected components were calculated and bounding box was assigned to components. After fixing the padding and resizing it, the image was passed through the network.