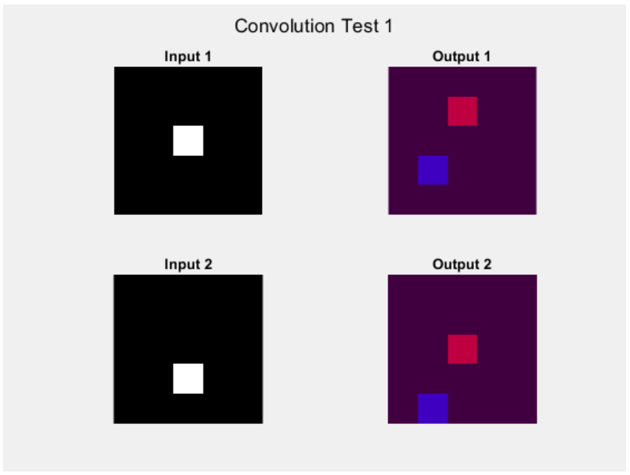
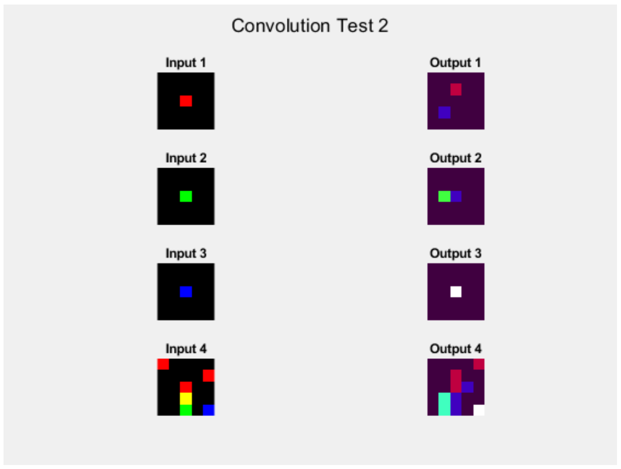
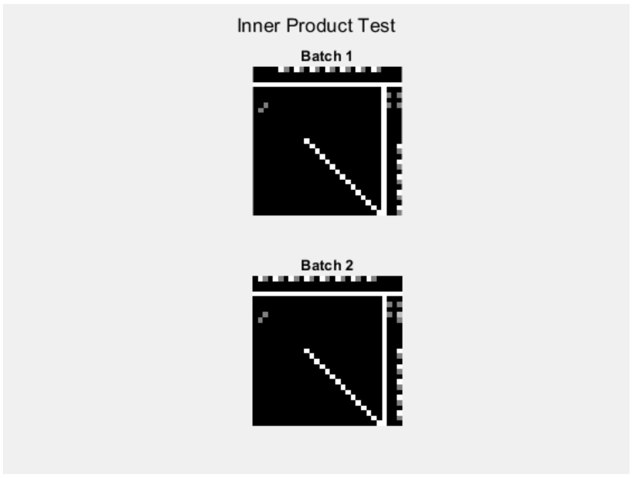
Vaibhav Saini
301386847
CMPT 412

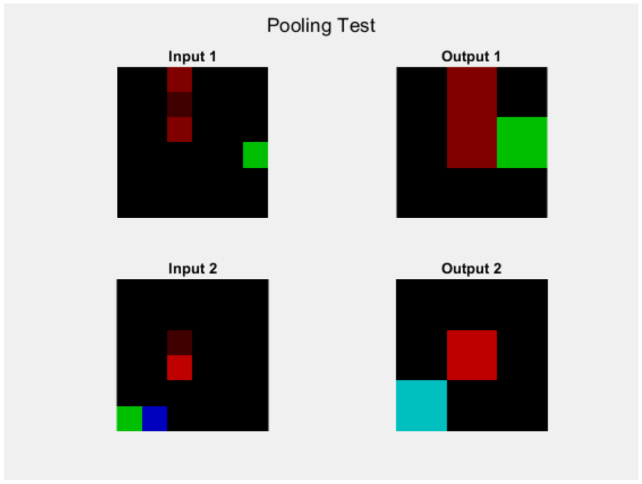
Project 1: Digit recognition with convolutional neural networks

Part 1: Forward Pass









Part 3: Training

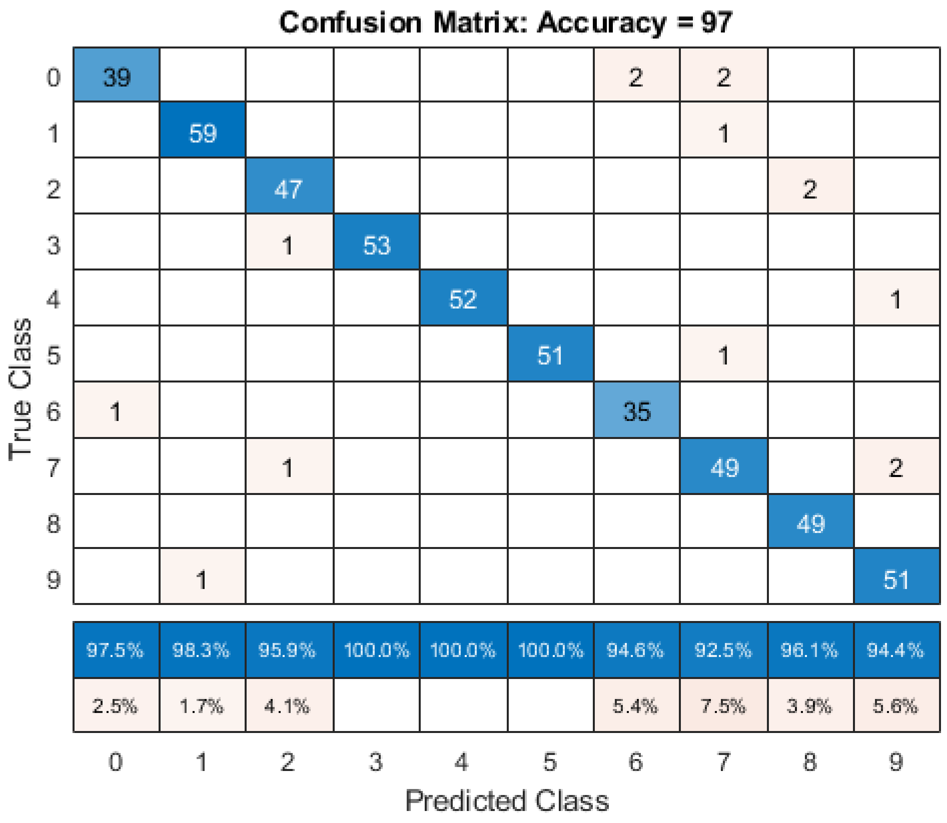
Q3.1

Training the network for 3000 more iterations, we see the test accuracy for every 500 iterations such that the final test accuracy for the network over 6000 iterations is reported to be 97.2%

```
test accuracy: 0.950000                test accuracy: 0.968000
test accuracy: 0.962000                test accuracy: 0.972000
```

test accuracy: 0.964000

test accuracy: 0.972000

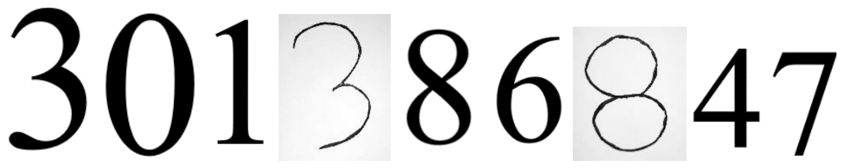


Confusion Matrix: Accuracy = 97

True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	39						2	2		
1		59						1		
2			47						2	
3			1	53						
4					52					1
5						51		1		
6	1						35			
7			1					49		2
8									49	
9		1								51
	97.5%	98.3%	95.9%	100.0%	100.0%	100.0%	94.6%	92.5%	96.1%	94.4%
	2.5%	1.7%	4.1%				5.4%	7.5%	3.9%	5.6%

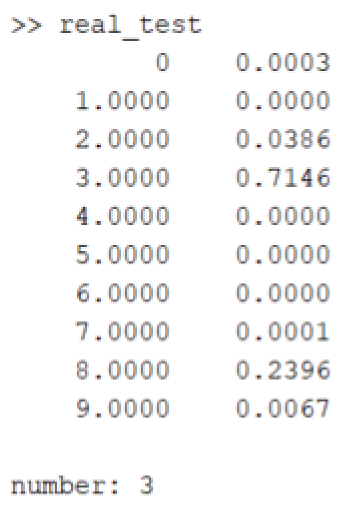
While the predictions made by the convnet are 97% accurate, it can still make mistakes between pairs such as 0 and 6 or 7 and 9. Looking at the numbers it's understandable why such misclassification might occur as both 0 and 6 contain a closed circle so its easier for the network to get confused. Similarly, 7 and 9 have a have a straight line in their shape and that might be cause for the network to misclassify them.

Q3.3 For real world testing the network, I wrote my SFU student id in a word document and cropped the Id to single digits, also to make things interesting I decided to change any repeated digits in my SFU id to Handwritten such that the sample images look like:



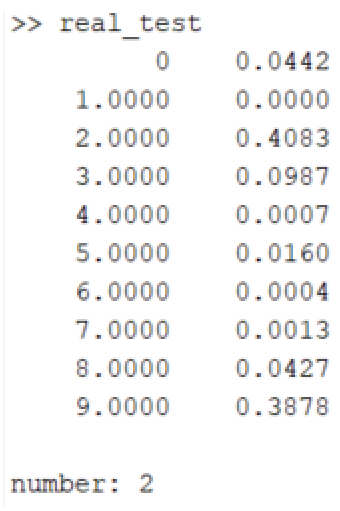
The prediction scores for each image are:

Number: 3



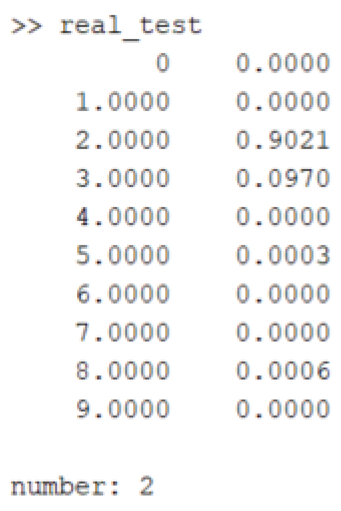
```
>> real_test
         0    0.0003
    1.0000    0.0000
    2.0000    0.0386
    3.0000    0.7146
    4.0000    0.0000
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0001
    8.0000    0.2396
    9.0000    0.0067

number: 3
```

Number: 0



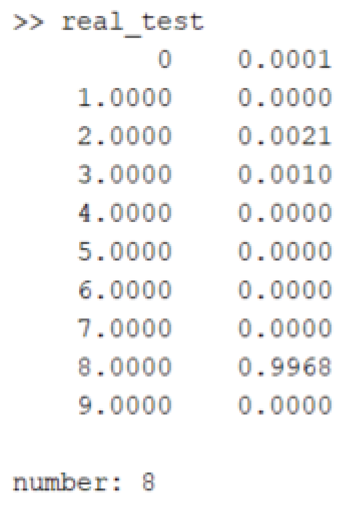
```
>> real_test
         0    0.0442
    1.0000    0.0000
    2.0000    0.4083
    3.0000    0.0987
    4.0000    0.0007
    5.0000    0.0160
    6.0000    0.0004
    7.0000    0.0013
    8.0000    0.0427
    9.0000    0.3878

number: 2
```

Number: 1



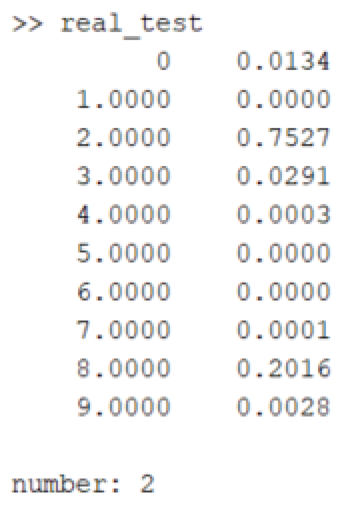
```
>> real_test
         0    0.0000
    1.0000    0.0000
    2.0000    0.9021
    3.0000    0.0970
    4.0000    0.0000
    5.0000    0.0003
    6.0000    0.0000
    7.0000    0.0000
    8.0000    0.0006
    9.0000    0.0000

number: 2
```

Number: 3



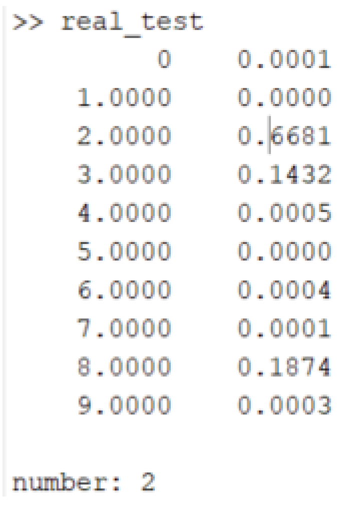
```
>> real_test
         0    0.0001
    1.0000    0.0000
    2.0000    0.0021
    3.0000    0.0010
    4.0000    0.0000
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0000
    8.0000    0.9968
    9.0000    0.0000

number: 8
```

Number: 8



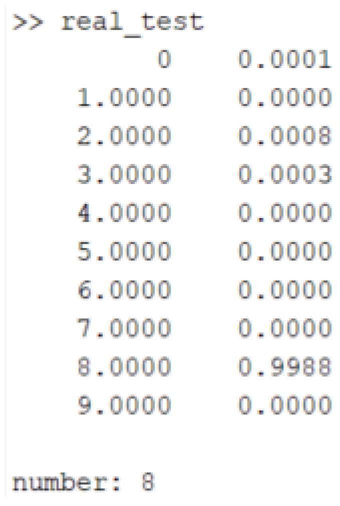
```
>> real_test
         0    0.0134
    1.0000    0.0000
    2.0000    0.7527
    3.0000    0.0291
    4.0000    0.0003
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0001
    8.0000    0.2016
    9.0000    0.0028

number: 2
```

Number: 6



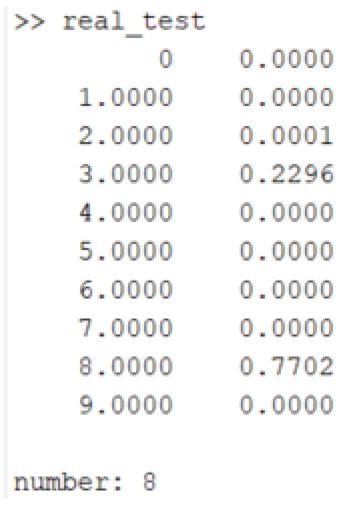
```
>> real_test
         0    0.0001
    1.0000    0.0000
    2.0000    0.6681
    3.0000    0.1432
    4.0000    0.0005
    5.0000    0.0000
    6.0000    0.0004
    7.0000    0.0001
    8.0000    0.1874
    9.0000    0.0003

number: 2
```

Number: 8



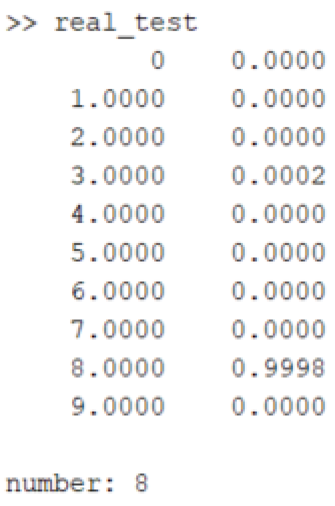
```
>> real_test
         0    0.0001
    1.0000    0.0000
    2.0000    0.0008
    3.0000    0.0003
    4.0000    0.0000
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0000
    8.0000    0.9988
    9.0000    0.0000

number: 8
```

Number: 4



```
>> real_test
         0    0.0000
    1.0000    0.0000
    2.0000    0.0001
    3.0000    0.2296
    4.0000    0.0000
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0000
    8.0000    0.7702
    9.0000    0.0000

number: 8
```

Number: 7



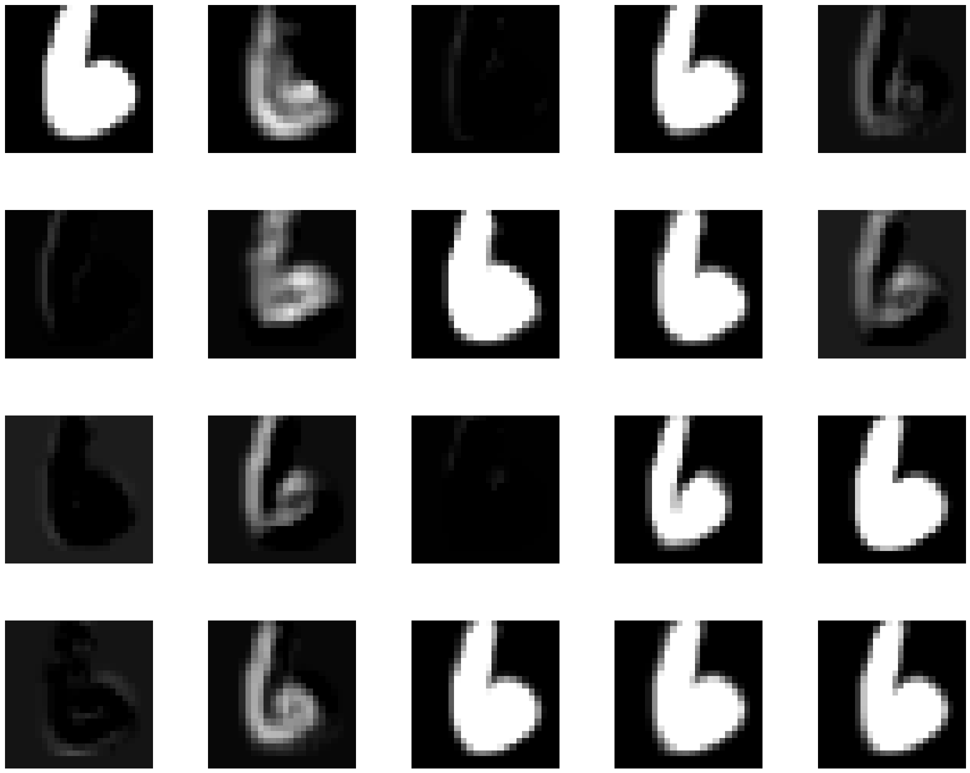
```
>> real_test
         0    0.0000
    1.0000    0.0000
    2.0000    0.0000
    3.0000    0.0002
    4.0000    0.0000
    5.0000    0.0000
    6.0000    0.0000
    7.0000    0.0000
    8.0000    0.9998
    9.0000    0.0000

number: 8
```

Based on real world testing the network only got 2 out 9 images correct, which ~23% accuracy, but this is not enough to say that our network is bad and this accuracy can be increased my training the network more and on images similar to the real world images that we used for testing such as images matching real world image font and style.

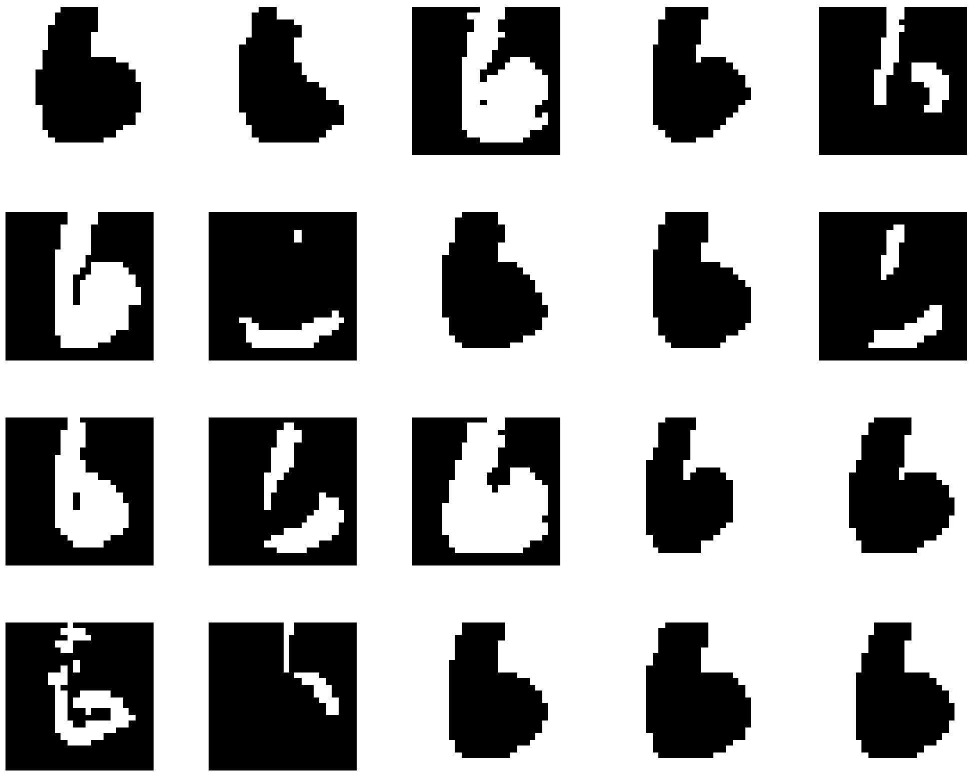
Part 4: Visualization

Input Image:

Conv layer features based on input images:



Activated pixels in Relu layer:



Since Conv layer and Relu layer look identical when plotting using imshow, I decided to invert the feature map before hence effectively highlighting activated pixels in relu layer. Looking at the very first image in the feature maps for conv and relu, we can spot how the network spot the number from it’s background by graduly separating the number by it’s edges and then we can observe the activated pixels because of convolution in relu layer. Also, while some feature in convolution layer may look similar, when looking at same features in relu we can easily spot the different activeted pixels for each feature.

Part 5: Image Classification

After running various images through ec.m code which helps us recognize digits from images and apply our Convolution neutral network on the those digits we get:

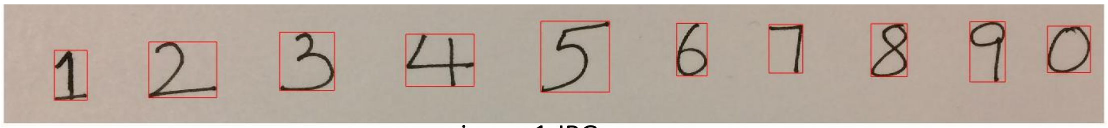


image1.JPG

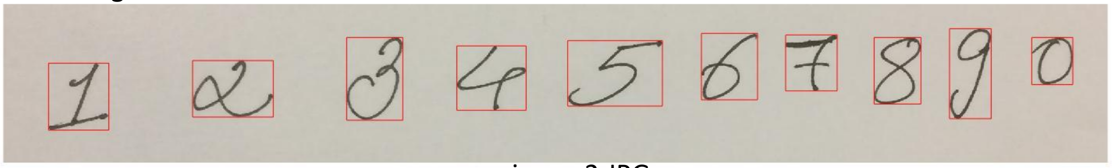


image2.JPG

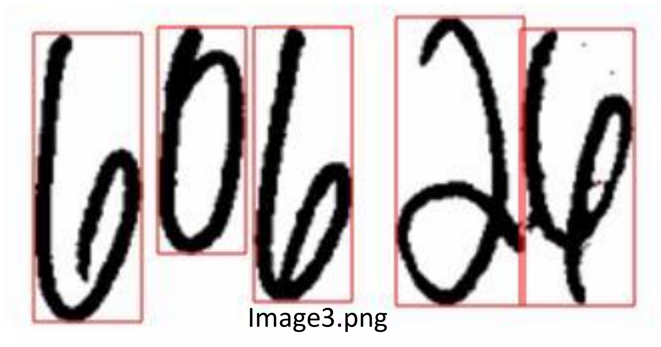


Image3.png

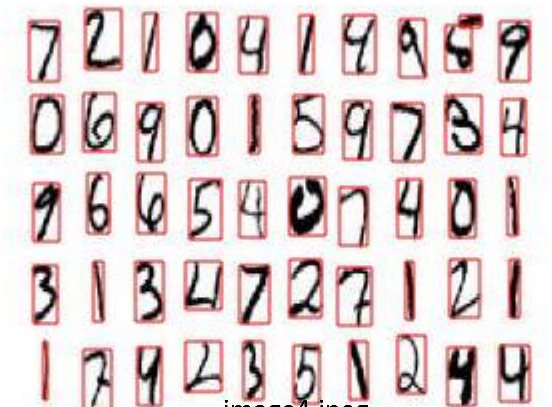
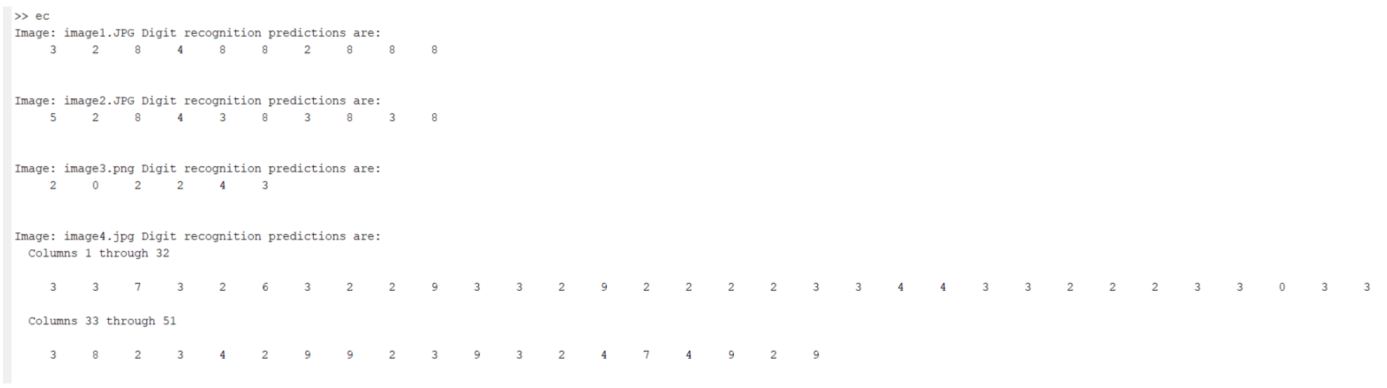


image4.jpeg

From these output images, we can see that the code was quite successful in recognizing digits from the images, while this was not always 100% accurate. As in image3 and image 4 non-digit conponents have also been recognized and hence, also made to through the neutal network.

After the testing each digit in the ConvNet, we get such for each image respectively.



```
>> ec
Image: image1.JPG Digit recognition predictions are:
     3     2     8     4     8     8     2     8     8     8


Image: image2.JPG Digit recognition predictions are:
     5     2     8     4     3     8     3     8     3     8


Image: image3.png Digit recognition predictions are:
     2     0     2     2     4     3


Image: image4.jpg Digit recognition predictions are:
  Columns 1 through 32

     3     3     7     3     2     6     3     2     2     9     3     3     2     9     2     2     2     2     3     3     4     4     3     3     2     2     2     3     3     0     3     3

  Columns 33 through 51

     3     8     2     3     4     2     9     9     2     3     9     3     2     4     7     4     9     2     9
```

Image name	Correct Prediction	Total digits
Image1	3	10
Image2	3	10
Image3	2	6 (1 incorrect recognition)
Image4	3	51((1 incorrect recognition)

Eventhough our network was 97% accurate, it starts to misclassify digits indicating that the network needs to be fine tuned further for more accurate predictions.

Reference:

https://cs231n.github.io/convolutional-networks/

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

https://www.mathworks.com/help/matlab/ref/imresize.html

https://www.mathworks.com/matlabcentral/answers/13499-read-files-from-directory

https://www.mathworks.com/help/matlab/ref/addpath.html

https://www.mathworks.com/matlabcentral/answers/475115-cropping-image-with-bounding-box

https://www.mathworks.com/matlabcentral/answers/158151-how-to-draw-boundingbox-rectangle-transperent-over-an-image

https://stats.stackexchange.com/questions/297678/how-to-calculate-optimal-zero-padding-for-convolutional-neural-networks

https://www.mathworks.com/matlabcentral/answers/829368-pad-matrix-to-a-particular-size