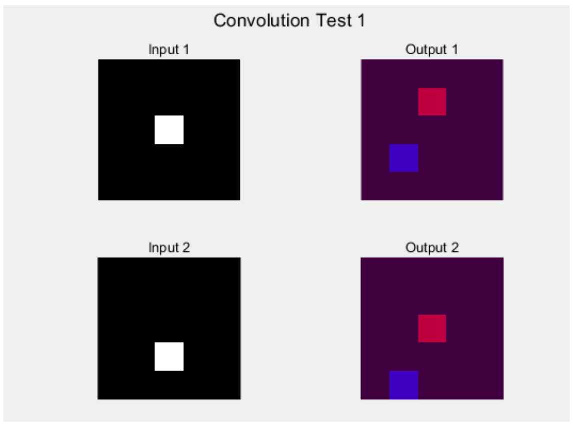
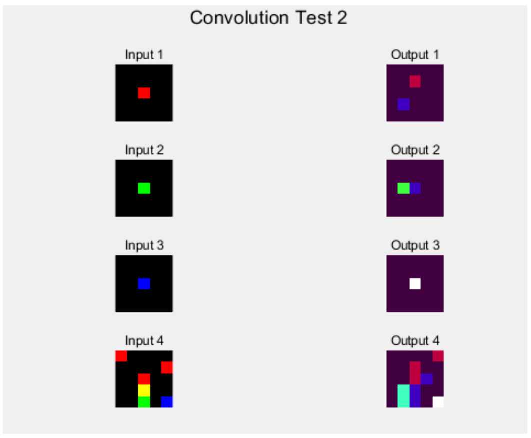
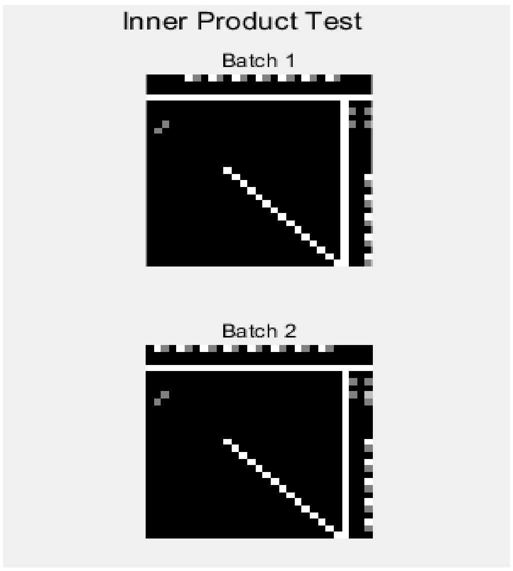
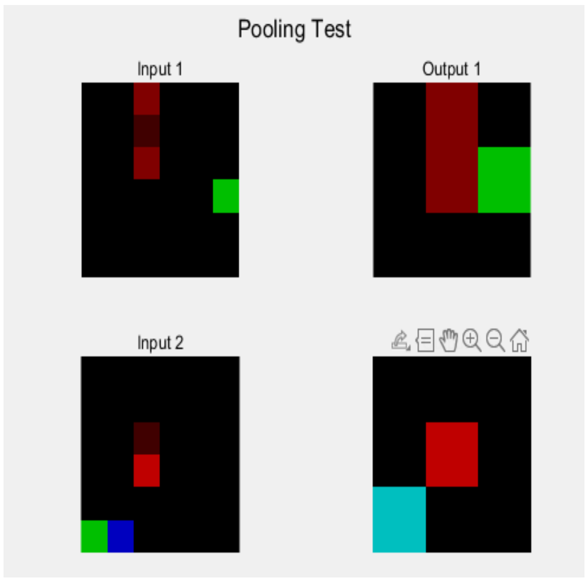
CMPT412
Project1
Xiaohong Xu
301417289
use 1 late_day

Part 1









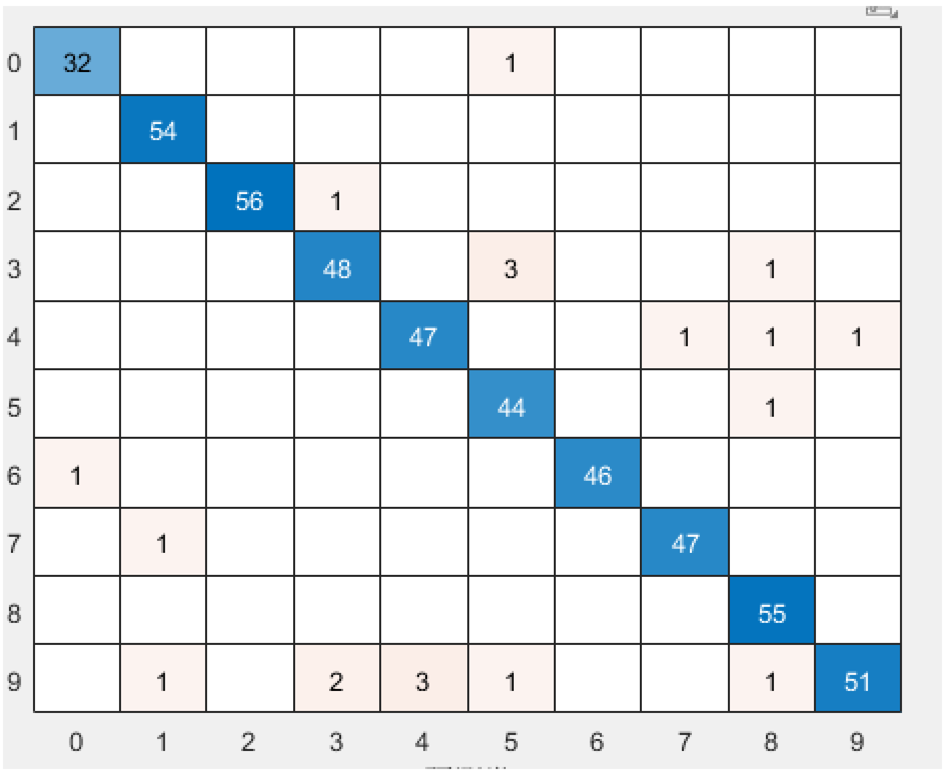
Part 3

3.1

after training for 3000 more iterations.
best accuracy:

```
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
```

3.2

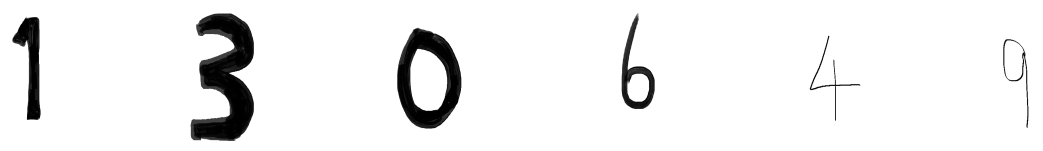


	0	1	2	3	4	5	6	7	8	9
0	32					1				
1		54								
2			56	1						
3				48		3			1	
4					47			1	1	1
5						44			1	
6	1						46			
7		1						47		
8									55	
9		1		2	3	1			1	51

(9,4) is the most confused pair that misses predictions a few times.
The reason might be the digit 4 and 9 both have a hole in the middle so as to miss predict.

3.3

Original



Result:

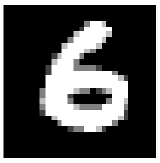
```
>> ec
       1       3       0       6       4       9
       1       3       0       6       4       1
```

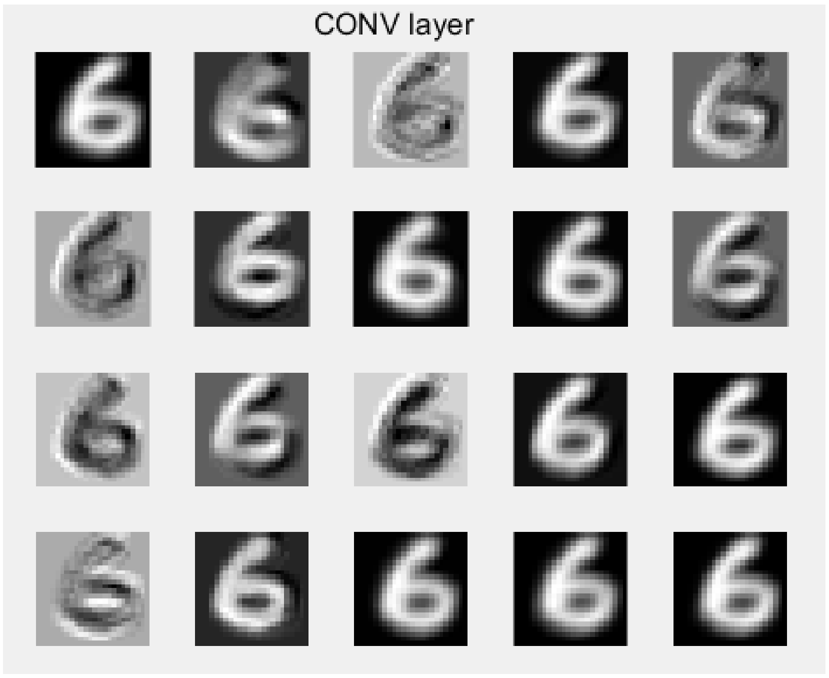
The first line is the expected result.
The second line is the actual result.

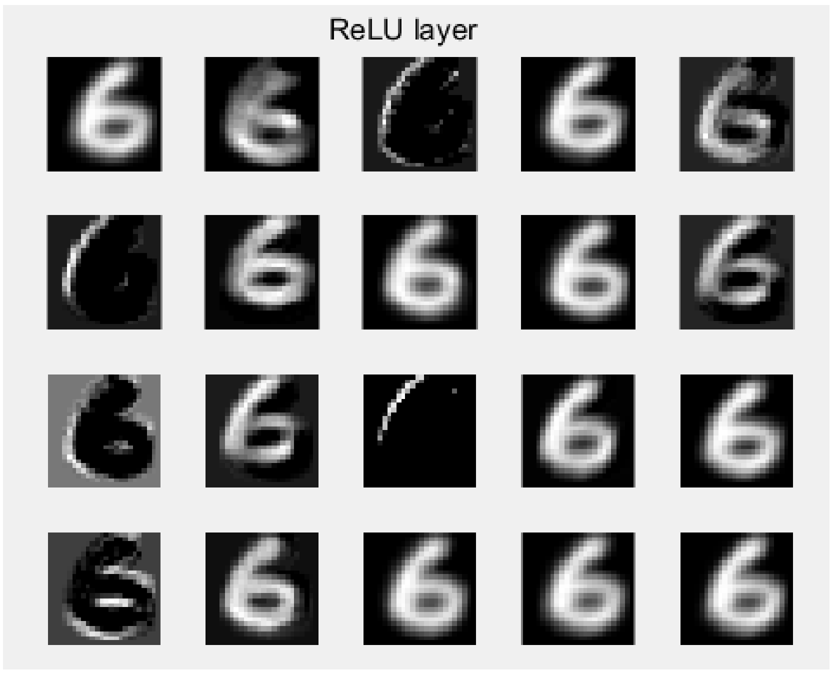
As we can see, the only incorrect number is 9.
5 out of 6 is correct. The reason might be the line is too thin so that the pixels of the digit are less than the dataset.

Part 4

4.1



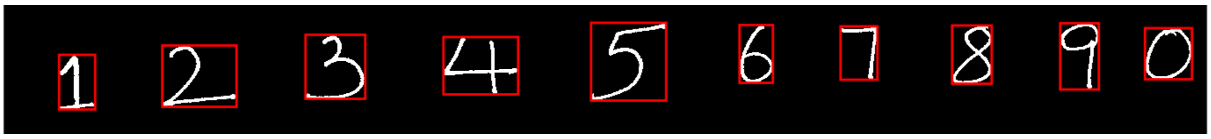




4.2

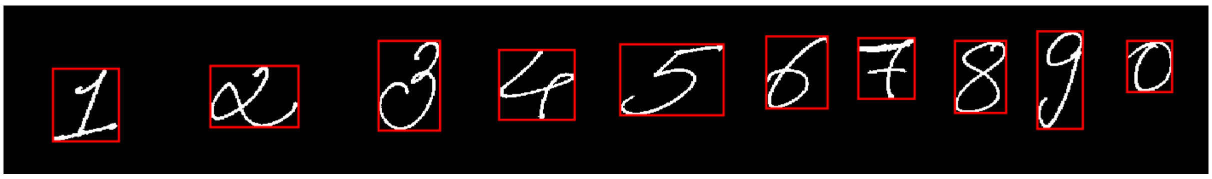
By comparing the original image with the CONV layer and ReLU we see that the network takes the shapes of digits by filtering the original image in different ways.
Then the CONV layer compared with the ReLU layer, ReLU has more black areas because of the nature of ReLU and we set white pixels to 0.

Part 5



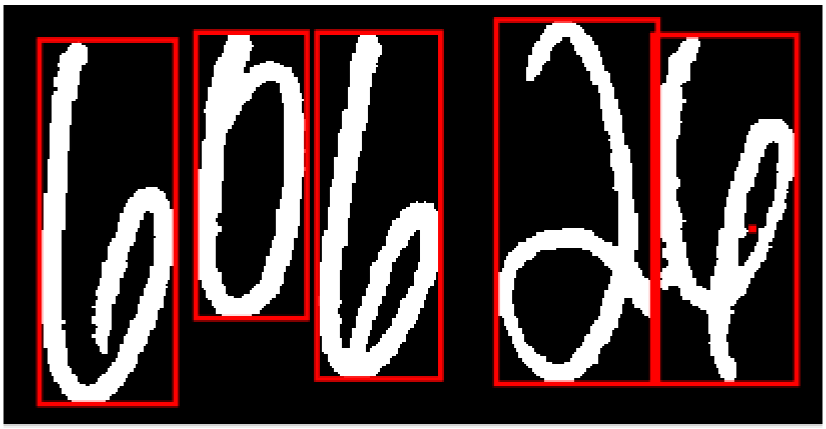
```
>> ec
     1     2     3     4     5     6     7     8     9     0
     1     2     3     4     5     5     3     8     7     0
```

The first line is original .
the second line is the result we get.
7 out of 10 are correct.



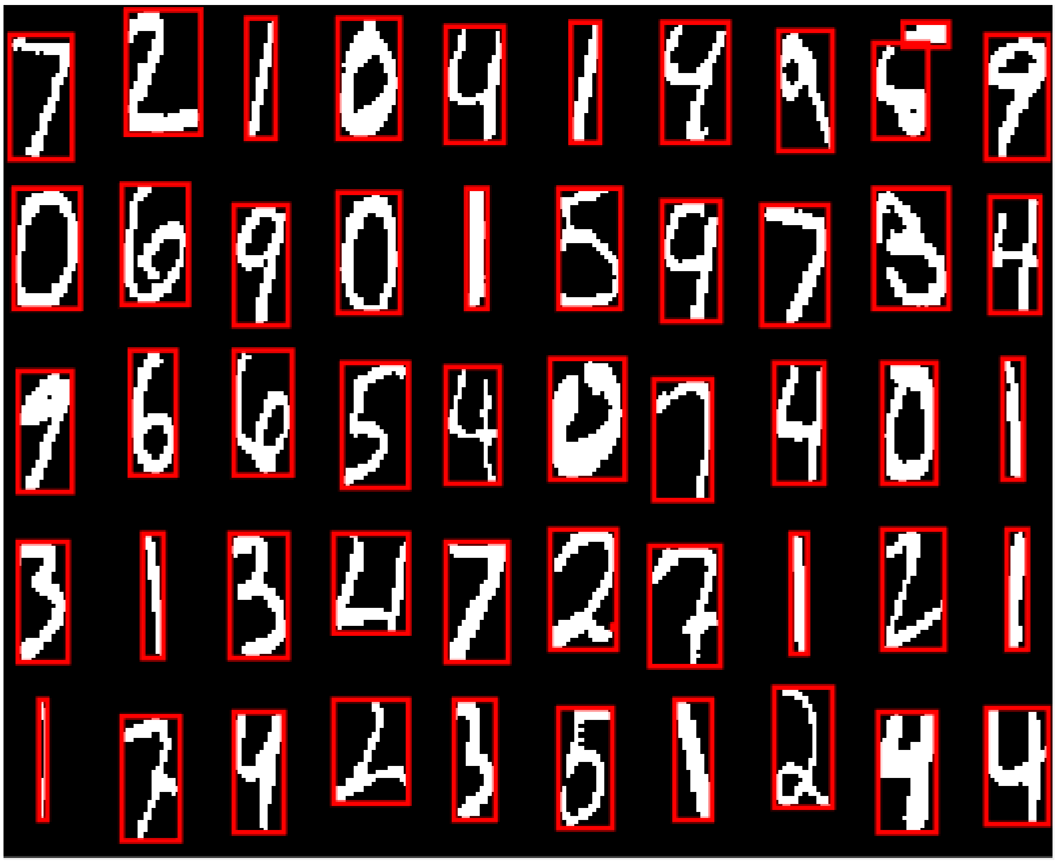
```
 1     2     3     4     5     6     7     8     9     0
 1     2     3     4     5     5     7     8     7     0
```

The first line is original .
The second line is the result we get.
8 out of 10 are correct.



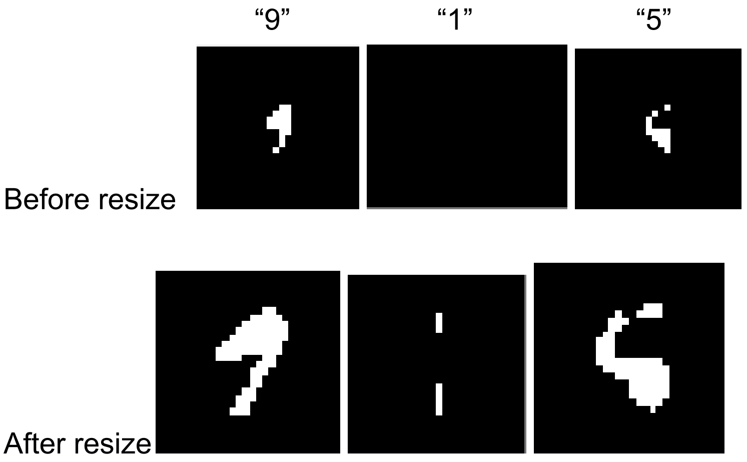
```
6        0        6        2        4
6        0        6        2        6
```

The first line is original .
The second line is the result we get.
4 out of 5 are correct.



```
列 1 至 21

   7     0     7     7     1     6     7     2     6     1     3     9     6     4     1     6     2     0     0     5     4

列 22 至 42

   4     7     3     1     0     2     5     5     1     5     7     7     4     9     1     7     4     2     9     1     4

列 43 至 52

   5     4     0     2     9     9     4     4     1     1
```

After countless tests, I found that the order in which matlab locates bounding boxes is not what we want. Its order is column by column, from left to right, and irregular. For example, the output of five digits in the first column of the output is 70771, the original is 70931 which only has 3 digits correct. The output of five digits in the second column is 67261, and the original is 26617. Although the order of the bounding boxes is different, the second column is all correct, and the recognition result is correct.
Since there are about 50 digits in an image, I resize and enlarge the size of the digits in each bounding box. The results are significantly better.



I have printed all the output and compared each digit single by single by hand and I got 43 correct digits.
so the accuracy is about 86%