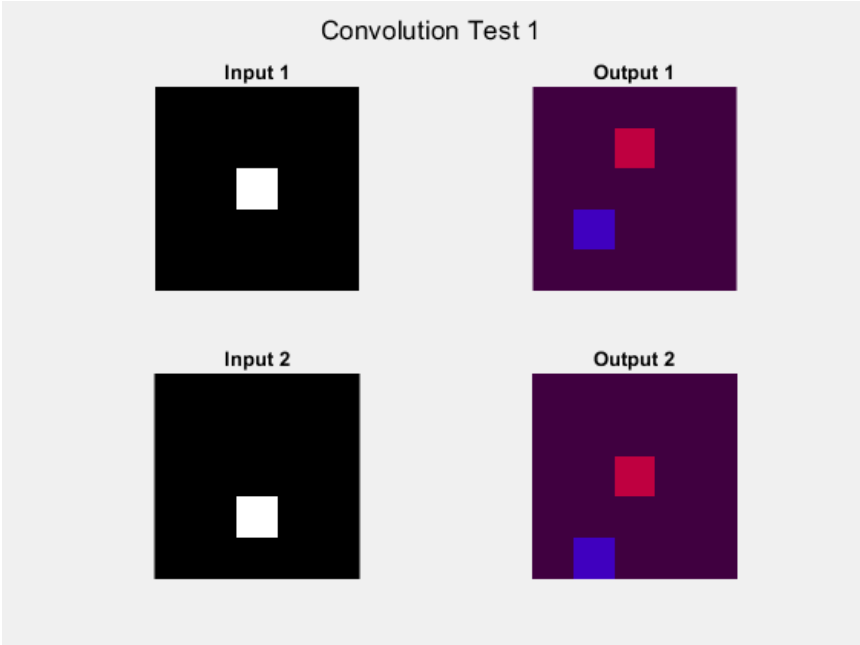
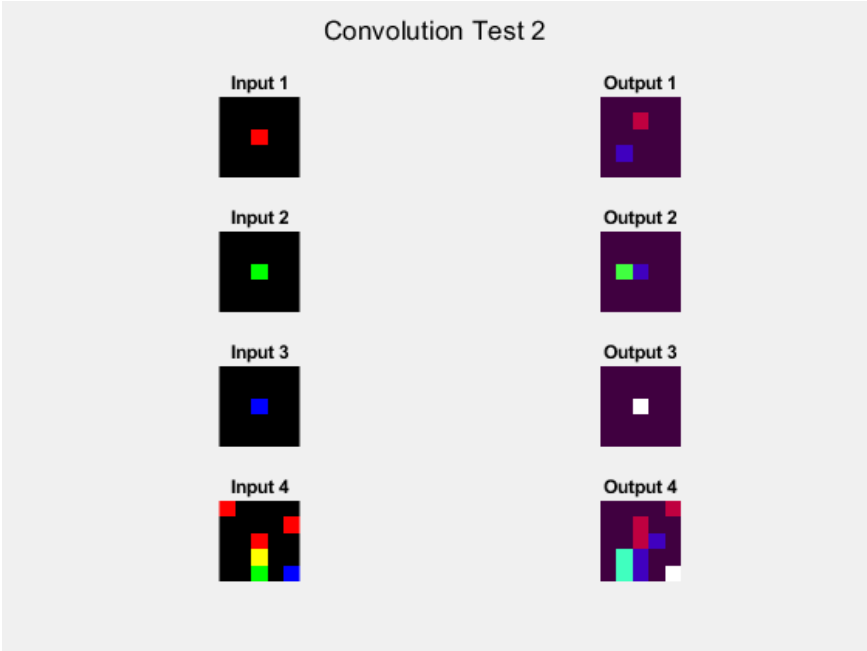
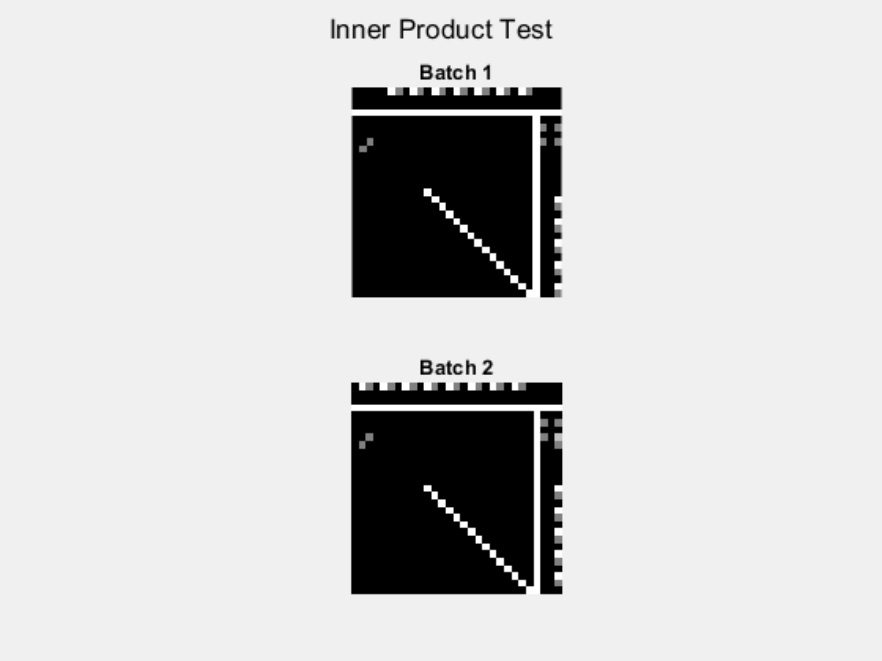
Digit recognition with convolutional neural networks

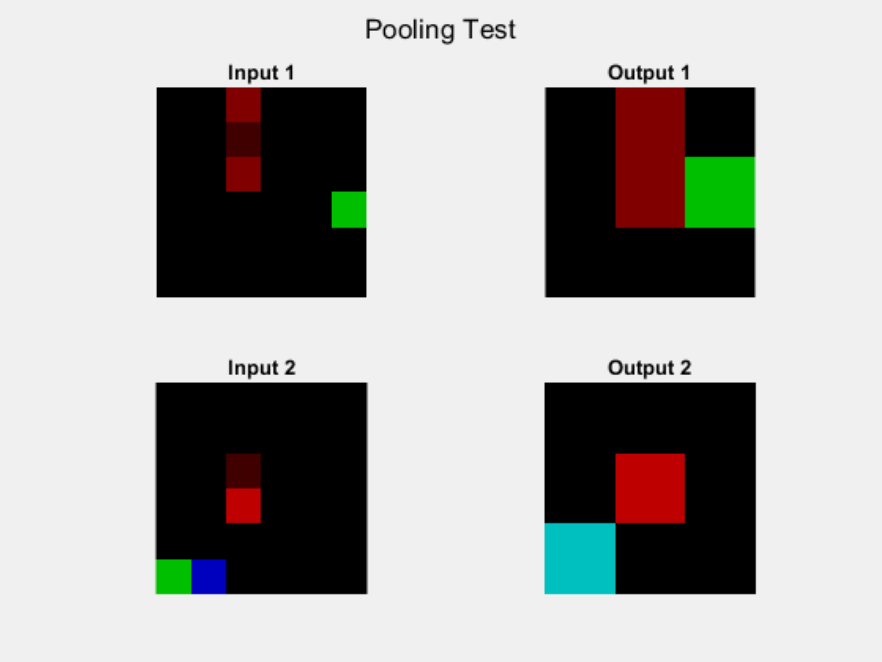
Part 1: Forward Pass

The results of running test_components.m are shown below.









Part 3 Training

Q 3.1 Training

The test accuracy after training for 3000 iterations is 97%. A screenshot of the test result is shown below.

```
iter3000.000000
cost = 0.049833 training_percent = 0.990000
test accuracy: 0.970000
```

Q 3.2 Test the network

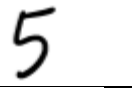
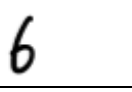
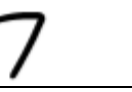
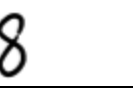
The confusion matrix on test data is shown on the image below. From the confusion matrix we can see that the class 1 and class 7 pair sometimes confuse the model, and also class 4 and class 9. These two pairs are misleading since if the handwriting is not standardized, 7 sometimes looks alike 1 and 4 sometimes looks alike 9. Even so, the chance that the model is getting confused by these cases is still low since there are only one or two cases that these digits being misclassified.

```
>> test_network
   39    0    0    0    0    1    1    0    0    0
    0   69    0    0    0    0    1    1    0    0
    0    0   47    0    0    0    0    1    0    0
    0    0    0   50    0    0    0    0    0    0
    0    0    0    0   51    0    1    0    0    2
    0    0    0    2    0   47    1    0    0    0
    0    0    0    0    0    0   52    0    0    0
    0    2    1    2    0    0    0   36    0    0
    1    0    1    0    0    0    1    0   36    1
    0    0    0    0    1    0    0    1    0   51
```

Q 3.3 Real-world testing

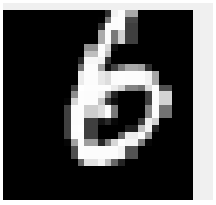
I wrote and scanned ten digits and put them through the model. The results are shown on the chart below. Nine of them are successfully recognized except for digit 9 which has around 80% probability to be digit 4. It is also interesting that among the correctly classified digits, all of them are recognized with at least 95% probability, except for digit 4 which has 11% probability to be digit 9. This experiment also proved our conclusion at Q3.2 that digit 4 and 9 are more likely to be confused as is shown on the confusion matrix.

0	1	2	3	4
0.9590	0.0001	0.0000	0.0000	0.0000
0.0000	0.9867	0.0000	0.0000	0.0000
0.0133	0.0007	0.9985	0.0000	0.0025
0.0000	0.0001	0.0012	1.0000	0.0000
0.0002	0.0002	0.0000	0.0000	0.8828
0.0000	0.0007	0.0001	0.0000	0.0000
0.0040	0.0045	0.0000	0.0000	0.0029
0.0011	0.0014	0.0000	0.0000	0.0002
0.0000	0.0057	0.0002	0.0000	0.0002
0.0223	0.0001	0.0000	0.0000	0.1114
correct	correct	correct	correct	correct

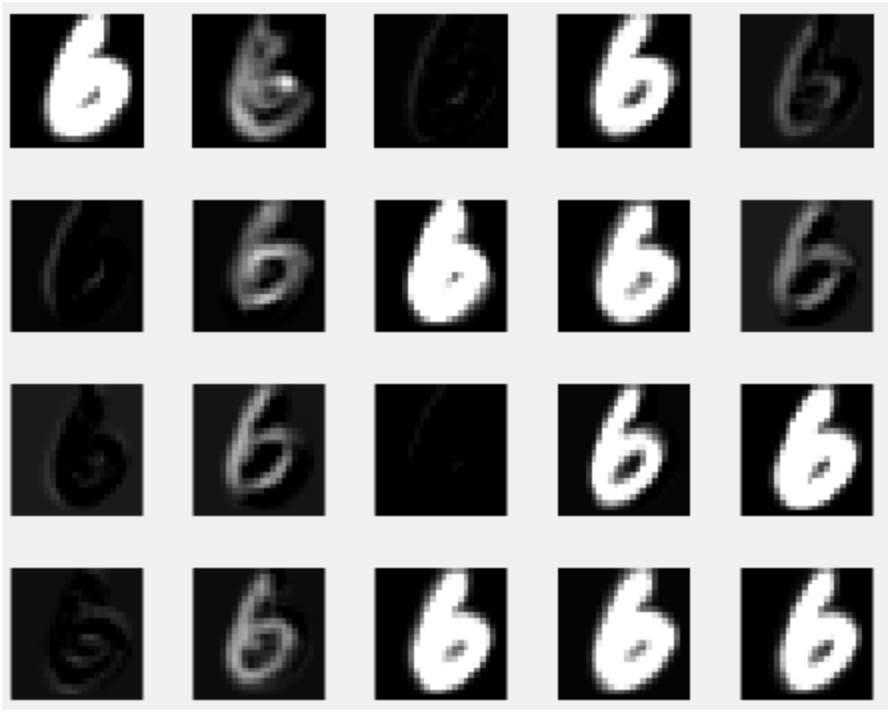
				
0.0000	0.0066	0.0000	0.0000	0.0000
0.0000	0.0000	0.0004	0.0000	0.0000
0.0000	0.0063	0.0010	0.0005	0.0000
0.0009	0.0000	0.0394	0.0009	0.0001
0.0001	0.0008	0.0000	0.0000	0.7999
0.9881	0.0095	0.0000	0.0002	0.0000
0.0000	0.9624	0.0000	0.0000	0.0000
0.0000	0.0000	0.9587	0.0000	0.0000
0.0020	0.0143	0.0000	0.9985	0.0038
0.0087	0.0001	0.0004	0.0000	0.1962
correct	correct	correct	correct	wrong

Part 4 Visualization

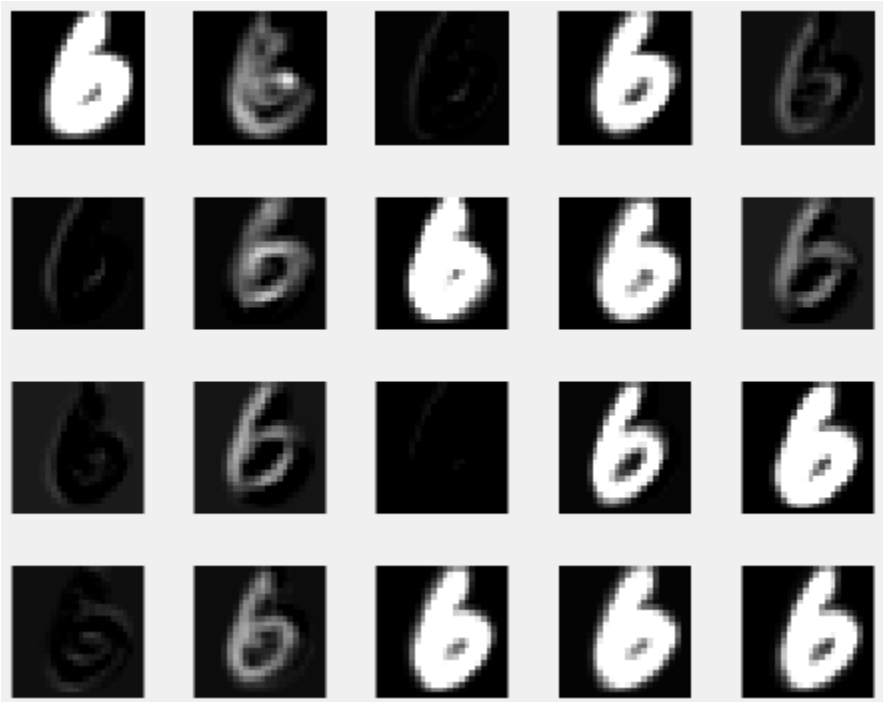
Q 4.1



Original input image



Feature maps of the second layer



Feature maps of the third layer

Q 4.2

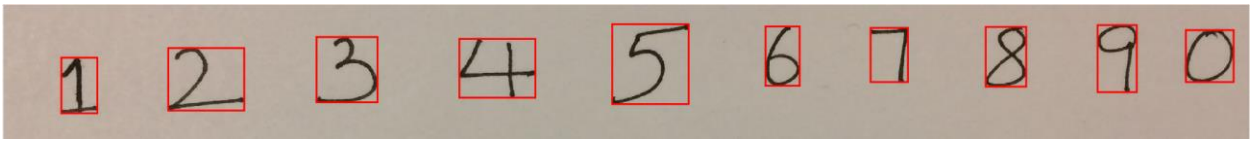
We can observe the contour of the original input image from the feature maps. But the differences are that different feature maps extract different information from the original image. For example on the feature maps we can see some of them looks like edge detection from different angle, and some of them are sharpening or blurring the input image.

The feature maps of the second layer and the third layer looks the same. Because the third layer is a ReLU layer, and what it does is make the negative entries of the output of the second layer zero. When we use imshow to display the output of the second layer, the function does the same thing as the ReLU layer as it cannot display negative value, which makes the visualization of the two layers the same.

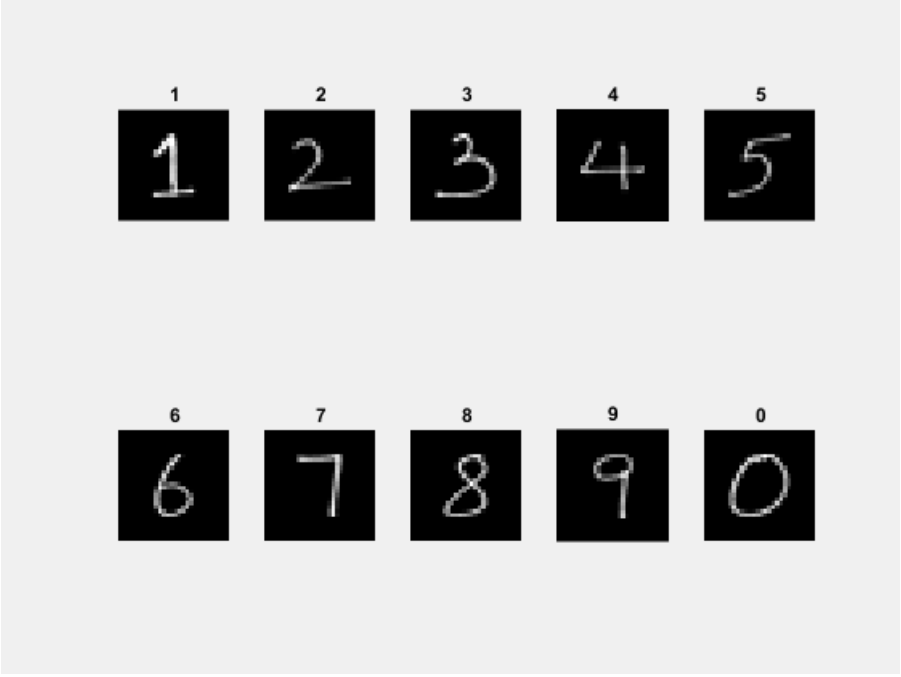
Part 5 Image Classification

In this part I present two visualizations for each test image, which are the visualization of the found bounding boxes on the test image, and the extracted preprocessed images taken by the bounding boxes, with their recognition output from the model on the top of every image. To get better result for different test images, I applied different preprocess method on the extracted images including different threshold for binarizing the image, different method of resizing the image and different padding.

Image 1



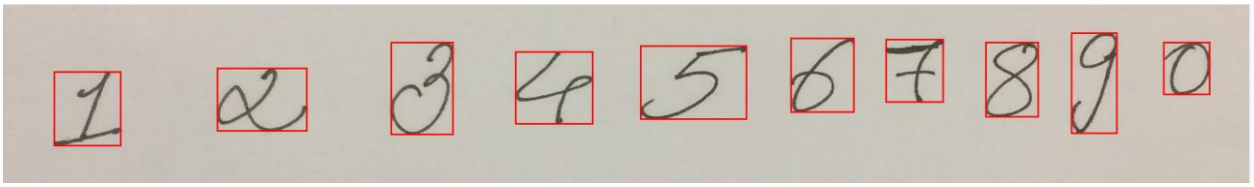
Found bounding boxes on the test image 1



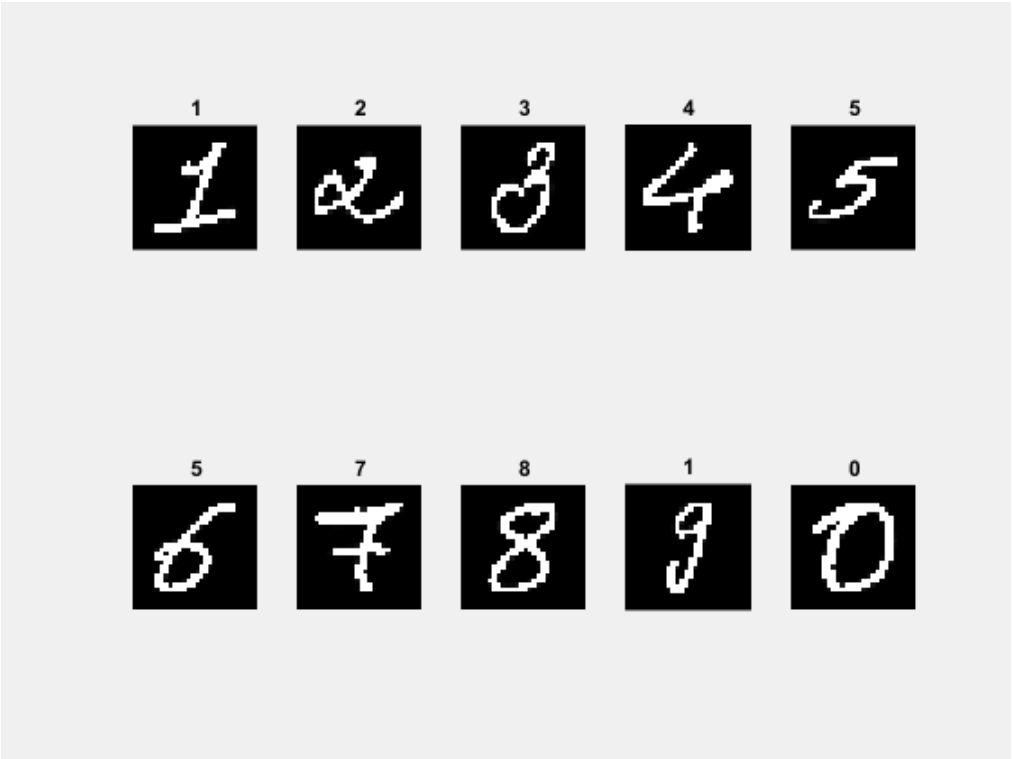
Extracted numbers with the recognition by the model on top of every image

The model successfully classified 10 digits out of 10 on test image 1, accuracy 100%.

Image 2



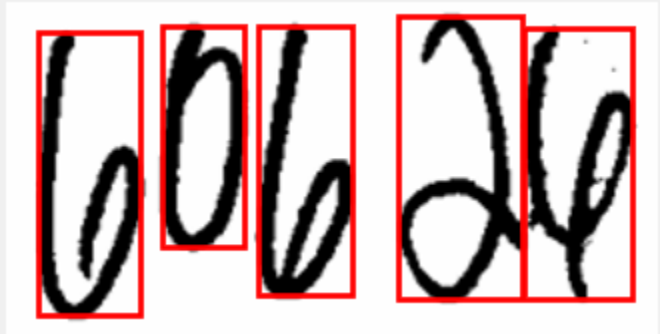
Found bounding boxes on the test image 2



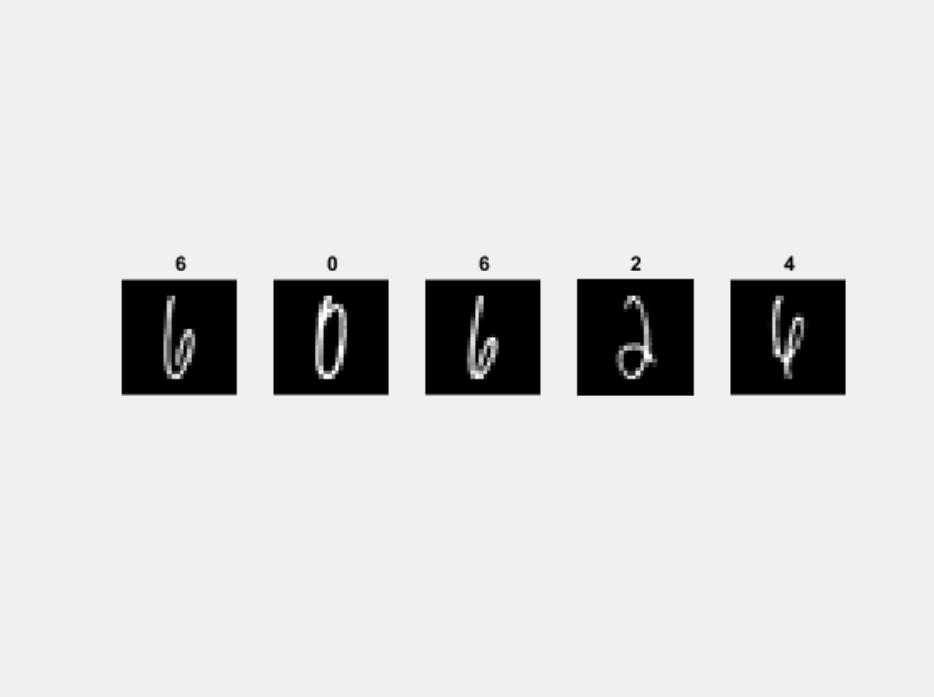
Extracted numbers with the recognition by the model on top of every image

The model successfully classified 8 digits out of 10 on test image 2, accuracy 80%.

Image 3



Found bounding boxes on the test image 3

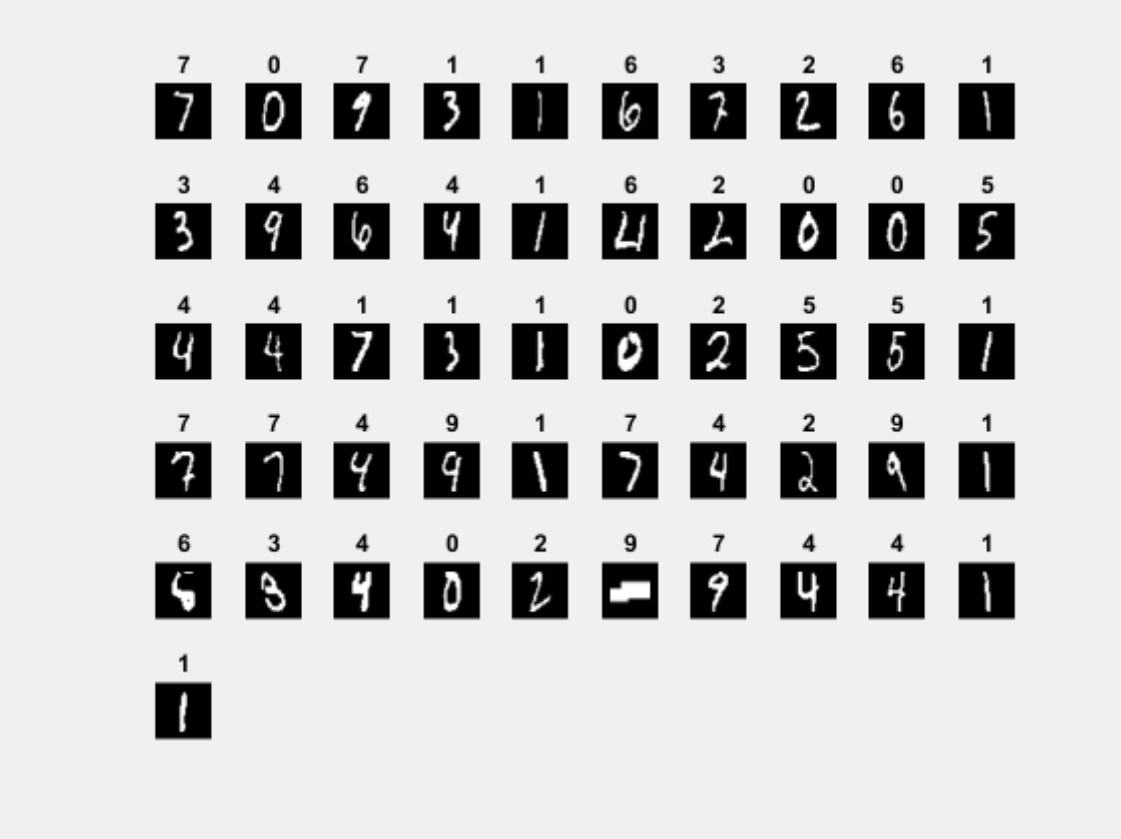


Extracted numbers with the recognition by the model on top of every image

The model successfully classified 5 digits out of 5 on test image 3, accuracy 100%.

Image 4

Found bounding boxes on the test image 4



Extracted numbers with the recognition by the model on top of every image

The model successfully classified 41 digits out of 50 on test image 4, accuracy 82%. Note that there is a mistake on the bounding box that it takes two parts of one digit as two separate digits, as they are not fully connected.