Cmpt 412

Project 1

Hao Ran Wei

301354701

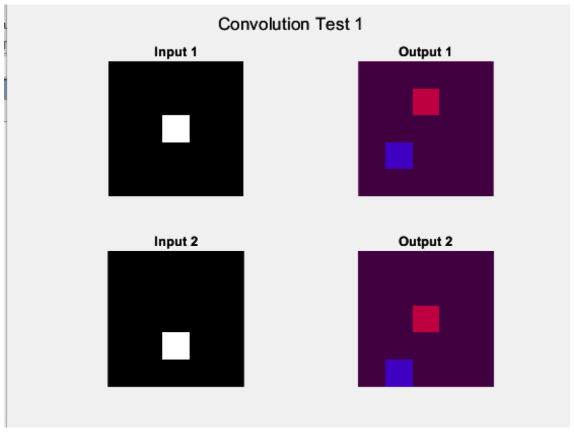
Project pre-requirements:

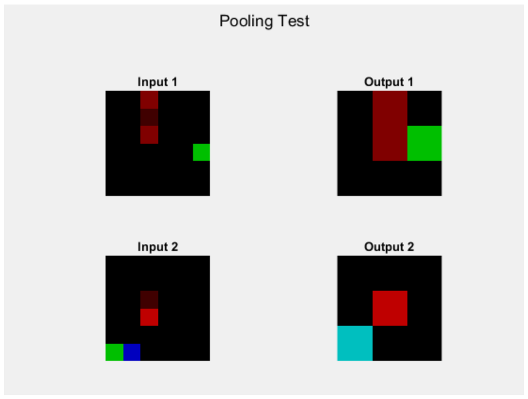
Matlab addons must be installed:

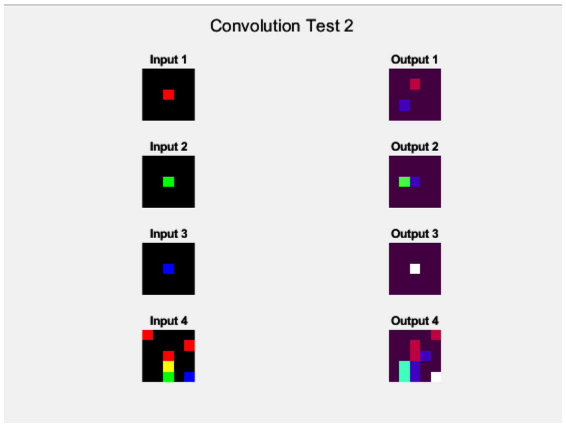
- Deep Learning Toolbox version 14.5
- Image Processing Toolbox version 11.6

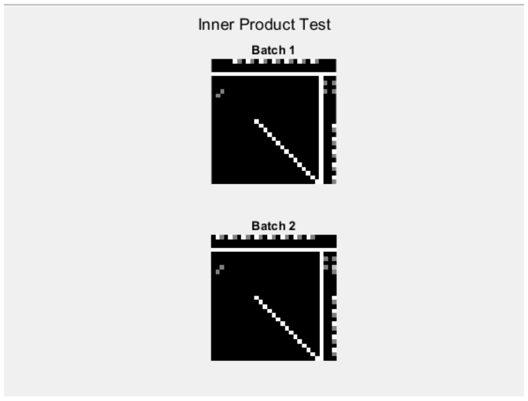
Q1.1 - Q 1.3

Running test_components.m results in the following:



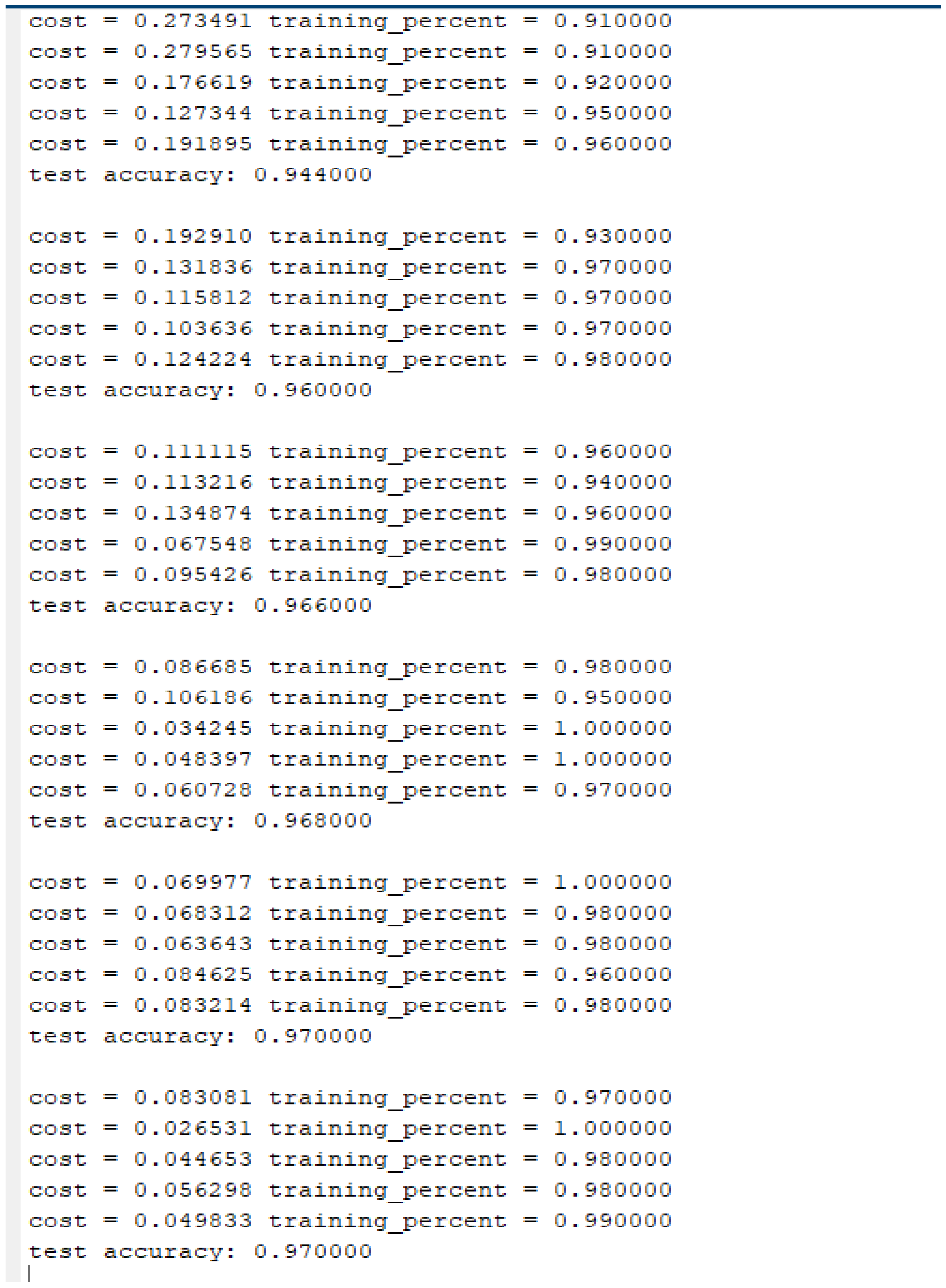






Q 3.1 Training - 1 pts

Running train_lenet.m results in the following:

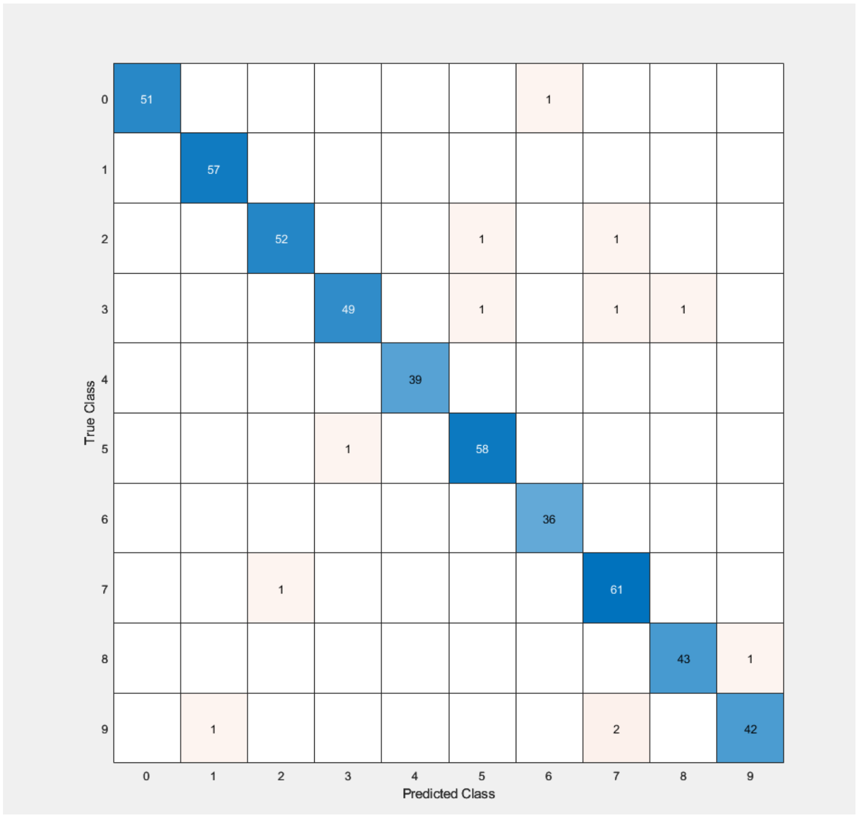


```
cost = 0.273491 training_percent = 0.910000
cost = 0.279565 training_percent = 0.910000
cost = 0.176619 training_percent = 0.920000
cost = 0.127344 training_percent = 0.950000
cost = 0.191895 training_percent = 0.960000
test accuracy: 0.944000

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
```

Notice after 1000 iterations, the test accuracy is steadily above 96 percent. After this, the accuracy consistency plateau compared to the increase in accuracy from 92 percent to 96 percent at iterations 500 and 1000, respectively. This is most likely due to the limitations of neuro networks.

Q 3.2 Test the network - 1 Pts

Running test_network.m results in the following confusion matrix chart:

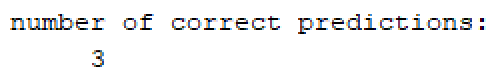


Where the diagonal is the number of times the prediction matches the true. According to this graph, our algorithm is accurate as the blue diagonal has the most values. I.e. most of the prediction is correct. Looking at this graph, when the value of the input picture is 0 and 4, this algorithm has a 100% prediction rate of getting it right. Whereas every other value from 1-9 has a couple of missed predictions. Another thing of note is that numbers 1, 2, 3,6 and 8 also have a reasonably accurate prediction rate, with only a single prediction being incorrect.

The top two most confused predictions are 7 and 5; this is most likely due to the shape of 5 being reasonably similar to 2 and 3 with its stroke shape with the curvature of the lower part of 5 may match with 2 and 3. Whereas 7 shares a lot of similar parts to 9,2 and 3 with its lack of defining features. Since parts of 9 can be mistaken for a 7 if you draw the circle part of the 9 small enough. As for 7 matching up with 2 and 3 it depends on how exaggerated the curves on 2 and 3 are, as it may be mistaken for a straight line which might trick the network into matching it up with a 7.

Q 3.3 Real-world testing - 1 Pts

Run test_samples.m in /matlab folder to see the number of image hits. Results in the following image:

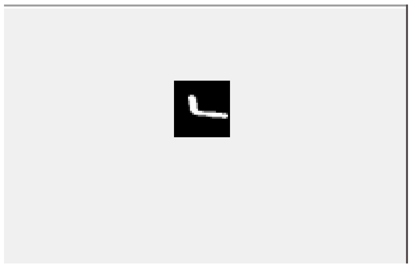


```
number of correct predictions:
     3
```

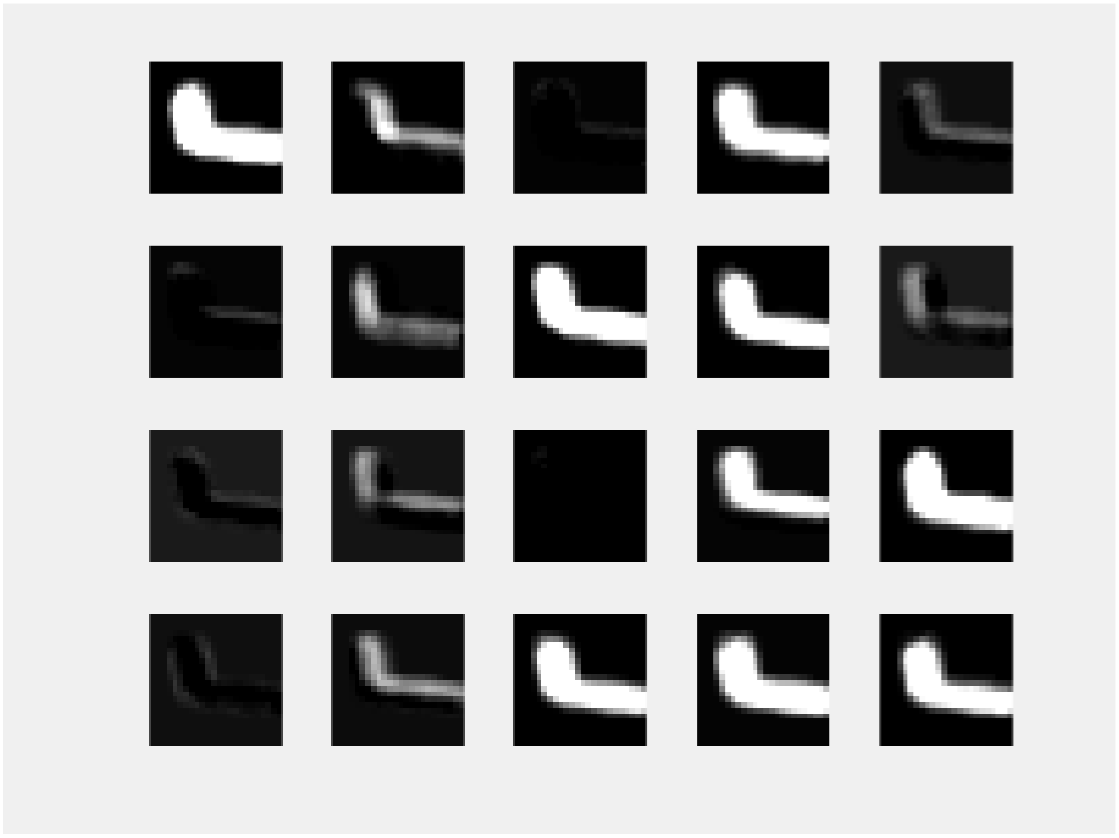
The sample images are located in the folder `../images/samples/`. Labelled from 1-5 containing images cropped to 28x28 images for more accessible conversions. Notice how the image is already in grayscale values, so there is no need to convert from RGB to grayscale in the following samples. Also, note that in real-life images, the hit rate is 2/5, which is 40 percent accuracy, drastically different from the hit rate during training. This might be due to the difference in the format in which the images are trained

Q 4.1 - 1 Pts

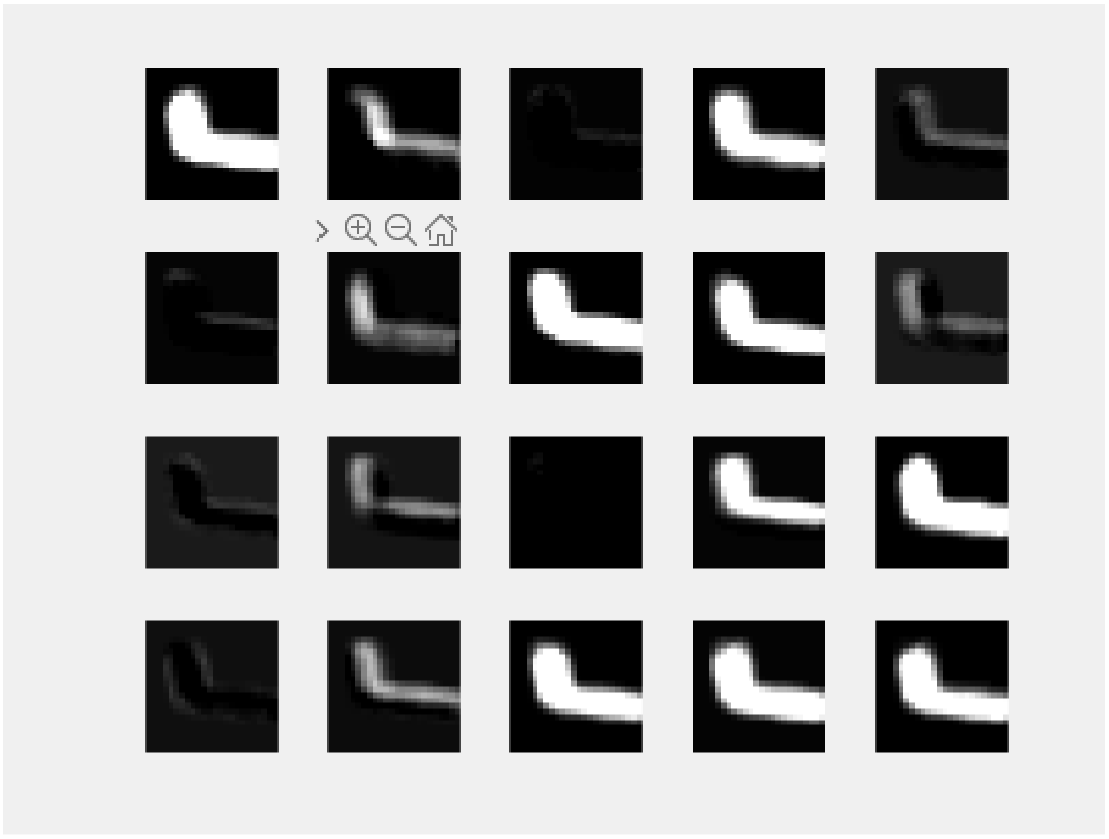
Running vis_data.m gives the following output or something similar:
Original image:



Output of Conv layer:



Output of ReLU layer:



Q 4.2 - 1 Pts

The difference between the original image and the conv layer seems to certain parts of the image are filtered out by weights. This is most likely due to the formula:

$$f(X, W, b) = X * W + b$$

Which explains the resulting images in Conv and how fuzzy they look compared to the original.

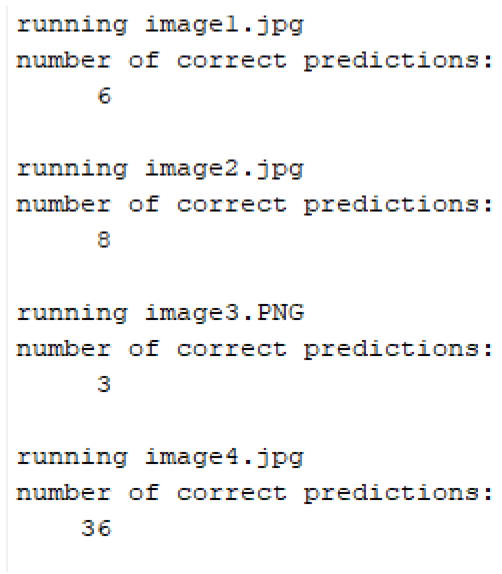
There is minimal to no change in the resulting images from Conv to ReLU output since the function for ReLU is just the following formula:

$$f(x) = \max(x, 0)$$

Which just outputs all positive values of X. However, since the original image is just grayscale. ie black and white. Who's values are already little to no negative so the resulting output would naturally change very little. This explains why there is little to no difference between the two images. At least from the perspective of the human eye.

Part 5 Image Classification - 2 Pts

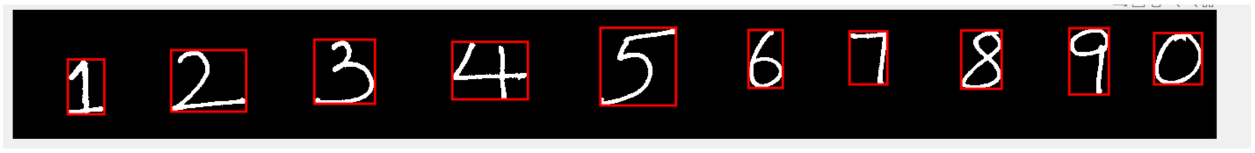
Running ec.m in the /ec file would result in the following outputs:

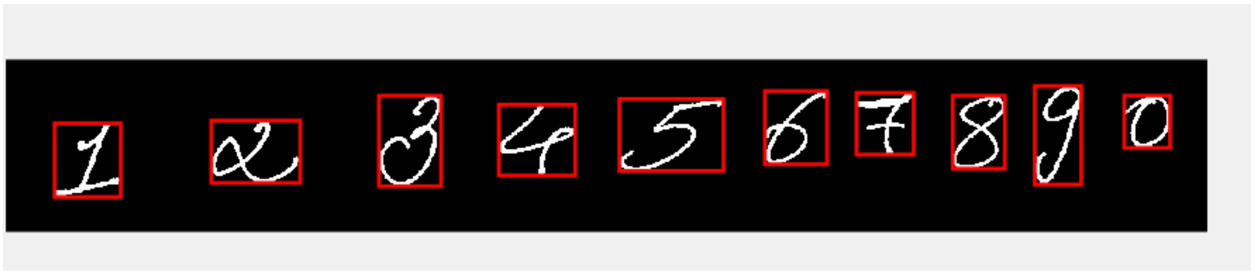


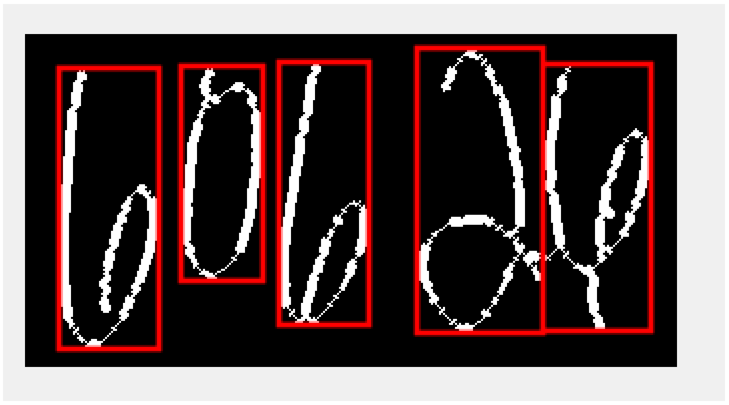
```
running image1.jpg
number of correct predictions:
     6

running image2.jpg
number of correct predictions:
     8

running image3.PNG
number of correct predictions:
     3

running image4.jpg
number of correct predictions:
    36
```

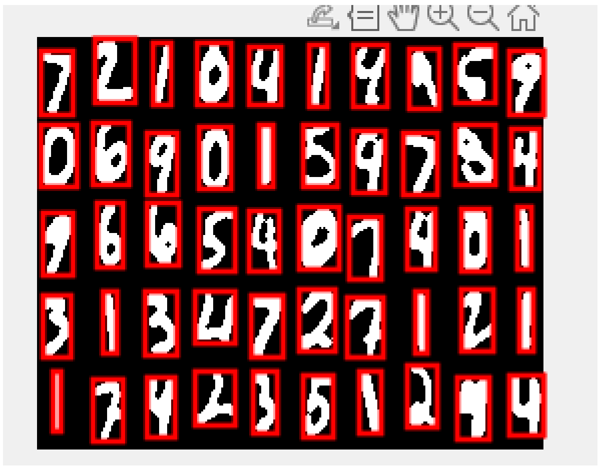
Which represent the number of correct matchups in each of the images1 to images4, respectively.

Also, the following figures are generated where the number inside of the red square will be the image recognized by the algorithm:









Note: more figures are generated than needed to allow the bounding box to show correctly in the image. Also, note that the CNN is trained with the transposed version of the image, you can see this in section 4.1. So after resizing each squire is transposed before put into the CNN function.