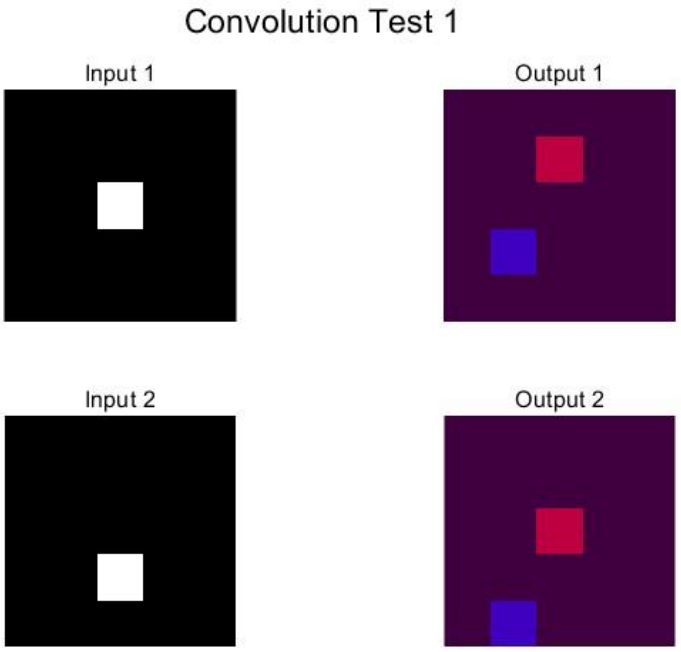
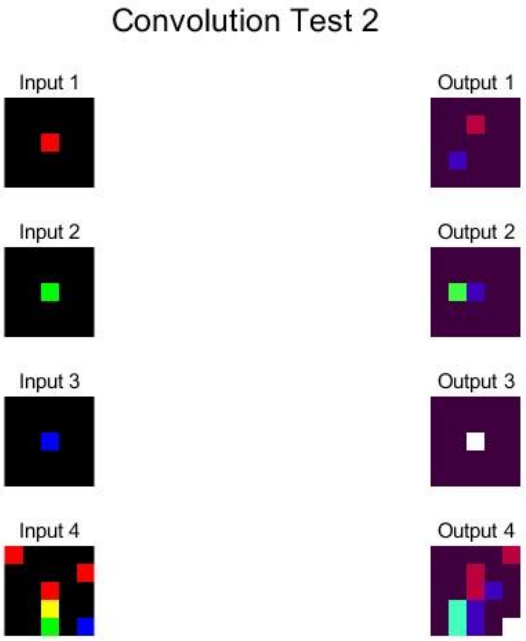
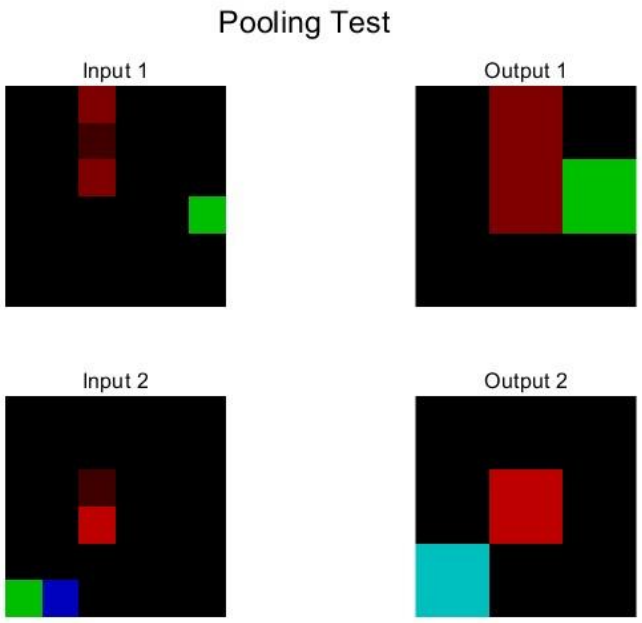
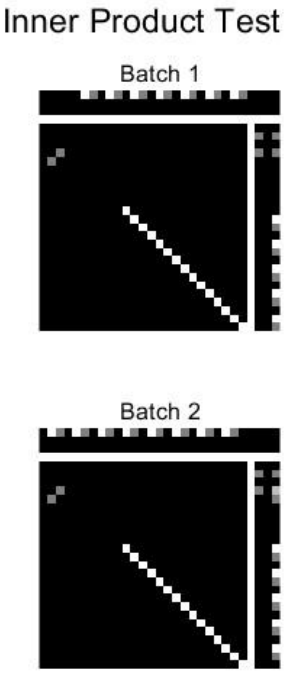
Project 1 Digit recognition with convolutional neural networks
Hankun Wang 301416863
Free-late Days: 1 Days

Part 1: Forward pass:
After implementing inner_product_forward.m, pooling_forward.m and conv_layer_forward.m, I can get images shown below by running test_component.m





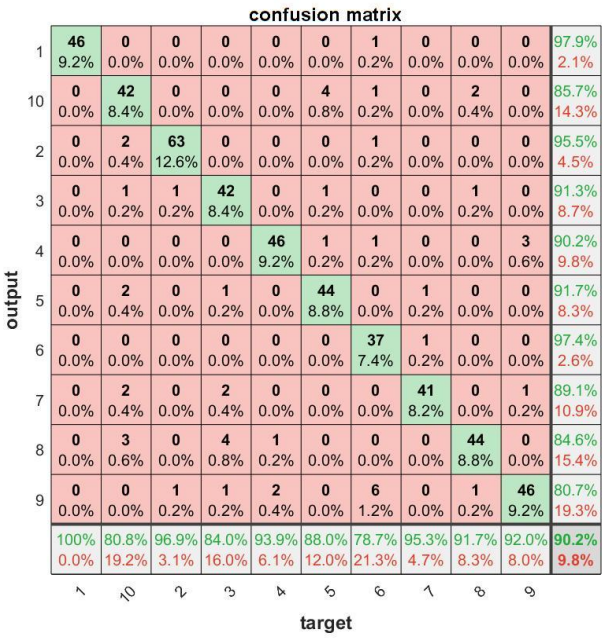

Part3

Q3.1
The result is shown below

```
>> 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
```

Meanwhile, there is also a .txt file that shows the result.

Q3.2
I use function “plotconfusion()” from “deep learning toolbox” and finally get result shown below




As shown above, It can be found that if my input is 6, its accuracy rate is the lowest, which is 78.7%. That is to say, there is a 21.3% that my system will recognize 6 as other numbers. Therefore, 6 is the most confusing number. With the same theory, 10 is the second confused number.

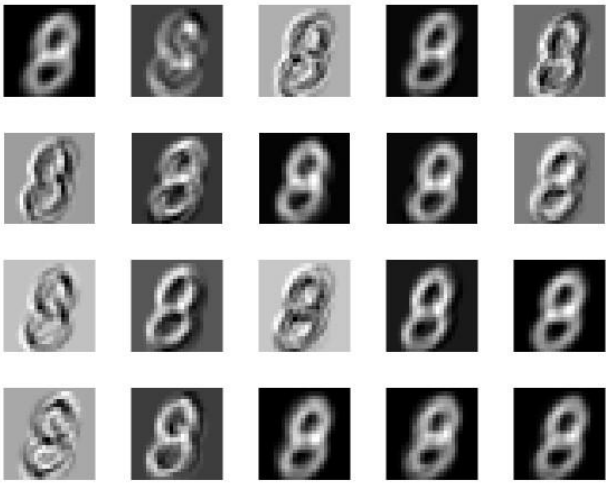
Q3.3

I write a file function called *read_number.m* which has an input parameter that indicates file name and has an output which contains a wrapped-up data that can be put directly in the network. Then I also wrote a file called *test_real_world.m*. It reads 5 images from the directory and processes them into the network. The output, which is output{9}(final FC layer), is stored in *real_world_output.mat*.

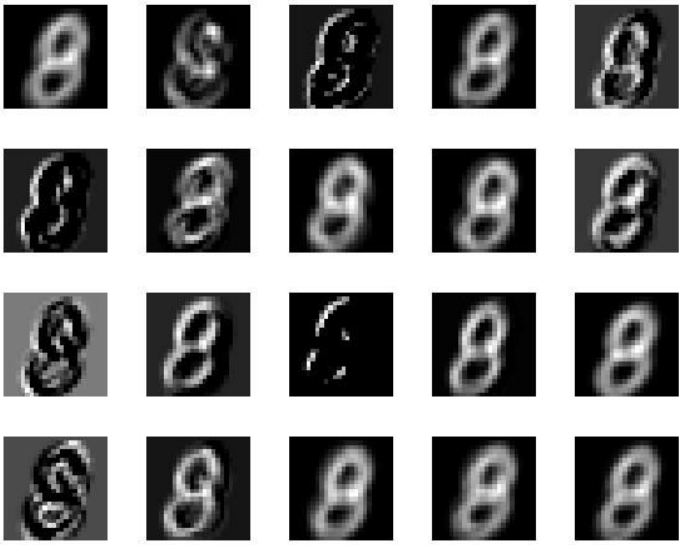
Part 4

Q4.1

The two images are shown below:



This image above is output for layer 2(convolutional layer)



This second image above is output for layer 3(relu layer).

Q4.2

The origin image is shown below:

Feature maps of layer 2, compared with the original image, blurs the edge of the number. Meanwhile, some filters of this layer invert the grey scale  of the image. The background of some output images are in light colour that is close to white.

Feature maps of layer 3, deep the white part of images from the last layer. Since the Relu function will increase the colour value if it is too low (i.e. lower than 0).