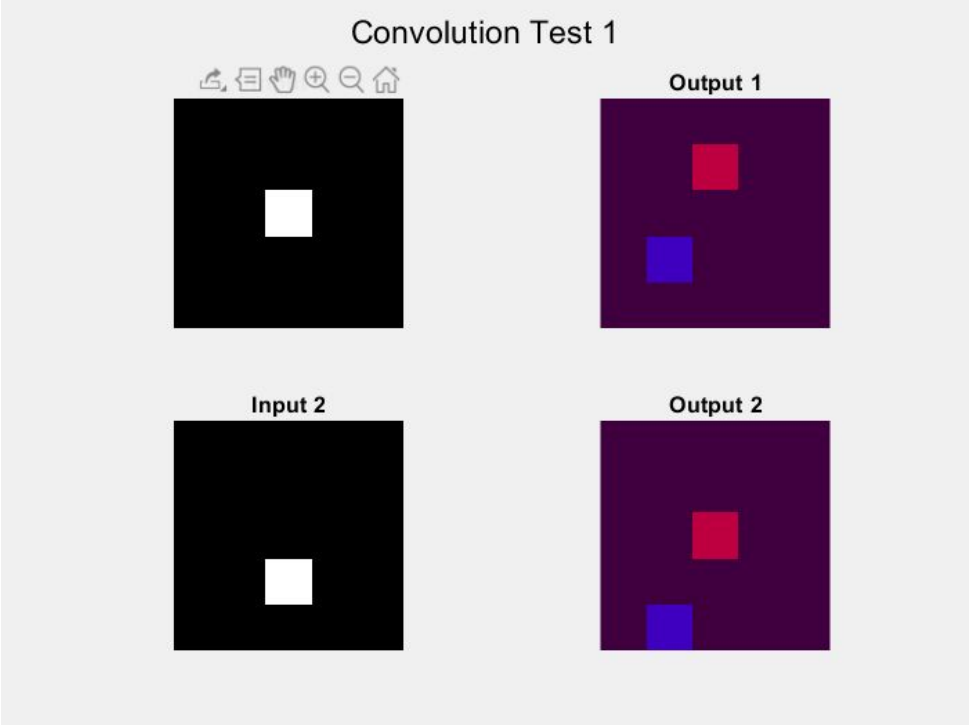
# Project 5
# Digit recognition with convolutional neural networks

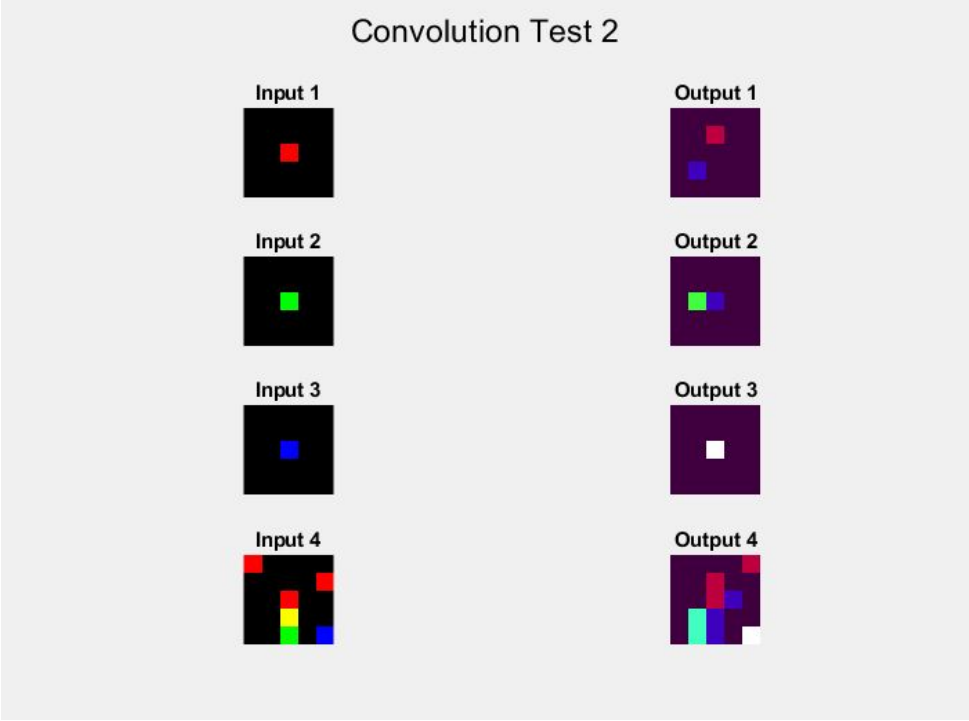
Due date: 23:59 Sunday 3/29th (2020)

Sepideh Sarajian Maralan

# Part 1: Forward Pass

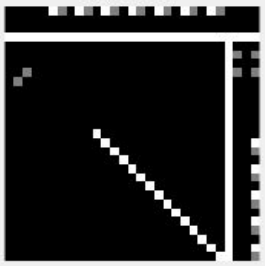
Here are the images of running test_components.m



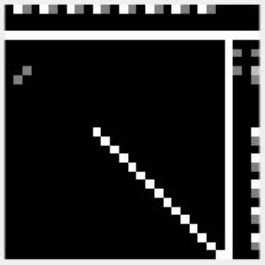



Inner Product Test

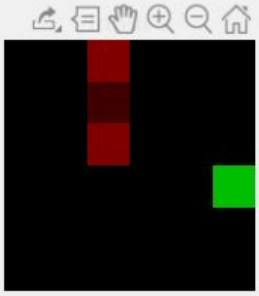
Batch 1



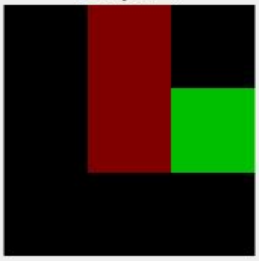
Batch 2



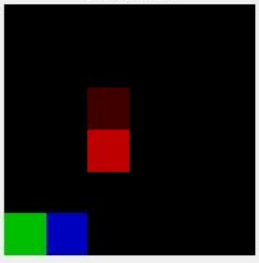
Pooling Test



Output 1



Input 2



Output 2

# Part 3 Training

## Q 3.1 Training - 1 pts

Here is the result of training with train_lenet.m:

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

## Q 3.2 Test the network - 1 Pts

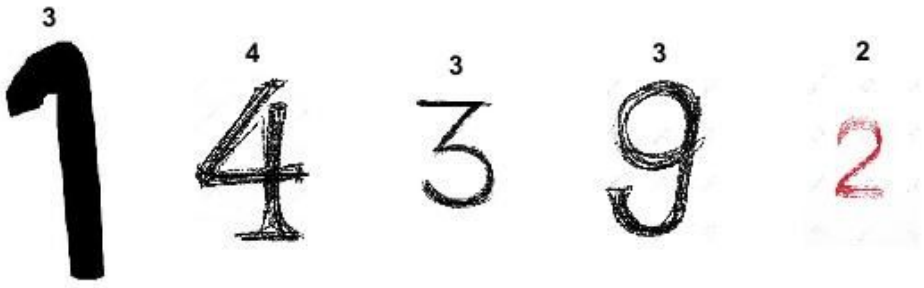
Here is the confusion matrix:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 45 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 46 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 53 | 2 | 0 | 0 | 0 | 4 | 0 | 0 |
| 0 | 0 | 0 | 43 | 0 | 1 | 0 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 49 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 38 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 55 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 46 | 0 | 2 |
| 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 51 | 0 |
| 3 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 45 |

For the first class (zero) one of the images was confused with 8 which kinda makes sense as 0 and 8 have similar shapes.
In the last class (10) three images were confused with 0 which again makes sense as there is a 0 in 10.

### Q 3.3 Real-world testing - 1 Pts

I downloaded 5 images and tried to make them look as close as to the training dataset, I tried grayscaling,  padding, thresholding and getting the complement of the image to give to the network.
The result is that from 5 pictures the network got 3 right, I think with better preprocessing the result can get even better.
Here are images and the number the network predicted on top:

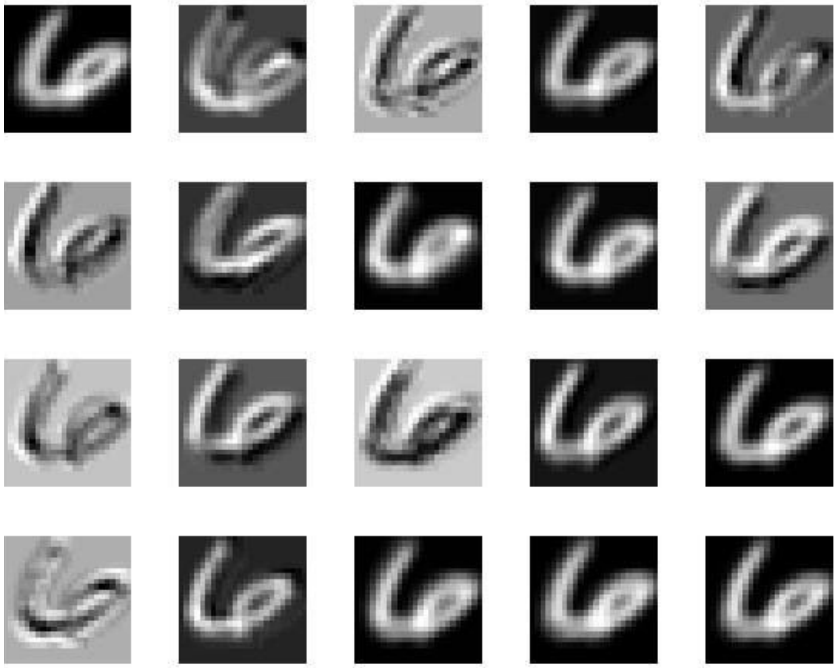



## Part 4 Visualization

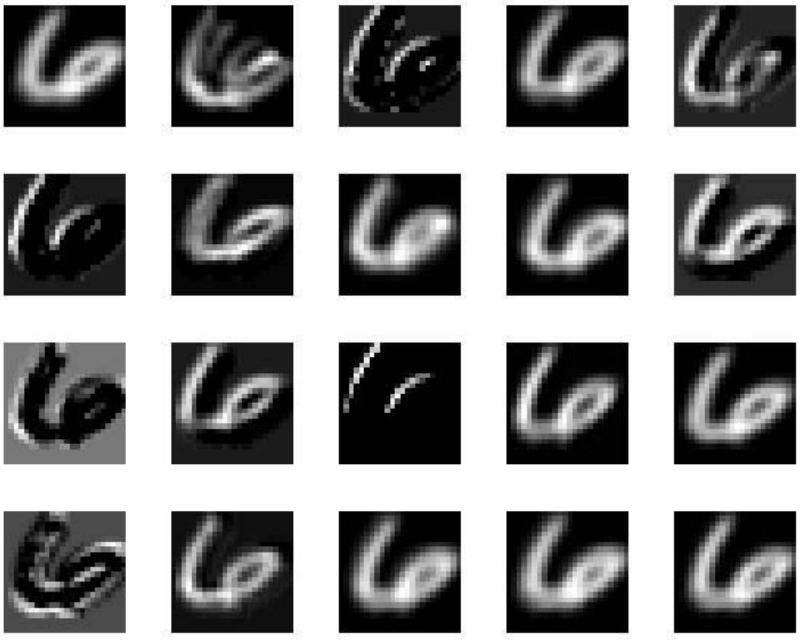
### Q 4.1 - 1 Pts

### Here is the original image:

Here is the 20 images from the second layer (conv):



Here is the 20 images from the third layer (relu):



## Q 4.2 - 1 Pts

In the convolution layer, the images are being filtered and they get more blurry or sharper in some cases.
In the relU layer some parts of the images have been omitted by doing the thresholding.

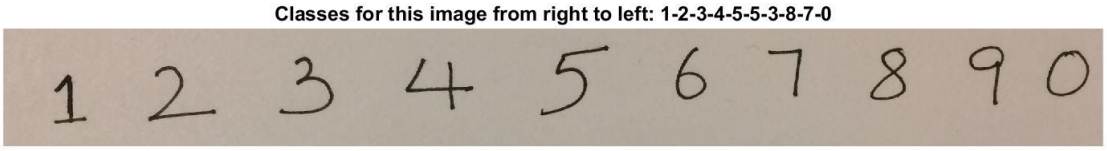
## Part 5 Image Classification - 2 Pts

I threshold the images, found the connected componants and cropped the image by getting the minimum and maximum value of the components and padded the images.

I gave the images to the network and got the following results:
I output the given image with the classes for them in a figure.

Cropped images:

As you can see in the next figure, the classes for 6 classes were predicted correctly.

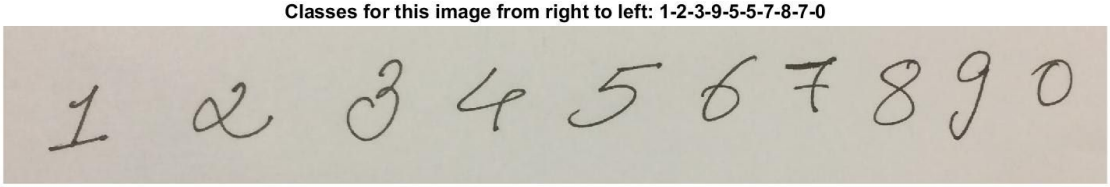



Result: 1-2-3-4-55-3-8-7-0

Cropped images:

As you can see in the next figure, the classes for 7 classes were predicted correctly.
Result: 1-2-3-9-5-5-7-8-7-0

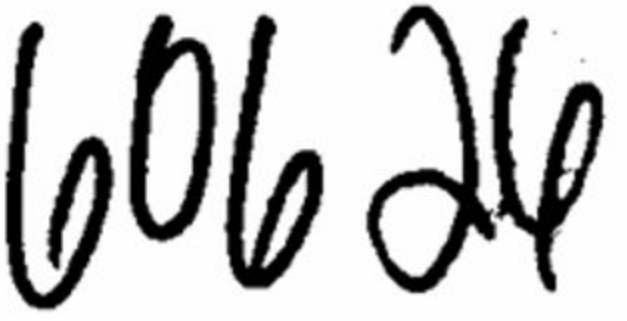
Classes for this image from right to left: 1-2-3-9-5-5-7-8-7-0



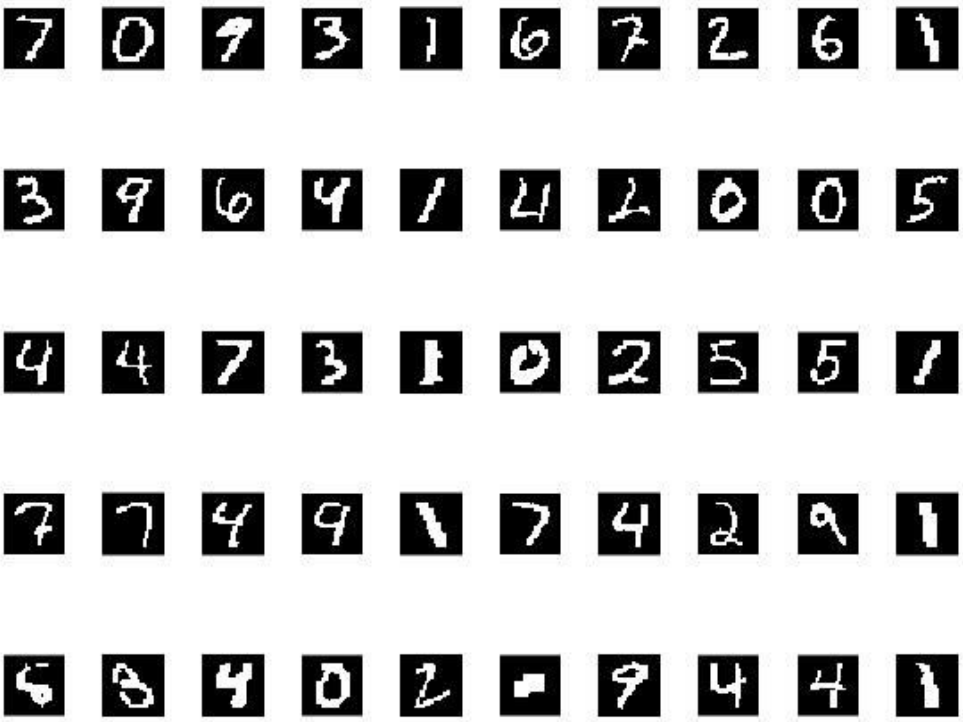
Cropped images:

All of the images were predicted correctly.
Result: 6-0-2-4

Classes for this image from right to left: 6-0-6-2-4



Cropped images:



Result:
7-0-7-3-1-6-7-2-6-1
3-4-6-4-1-4-2-0-0-5
4-4-2-3-8-0-2-5-5-1
7-7-4-4-1-7-4-2-9-1
6-3-4-0-2-9-8-4-4-1

The final result is that I got 42 images right out of 50 images.

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