Project 1

Ngawang (Choenden) Kyirong

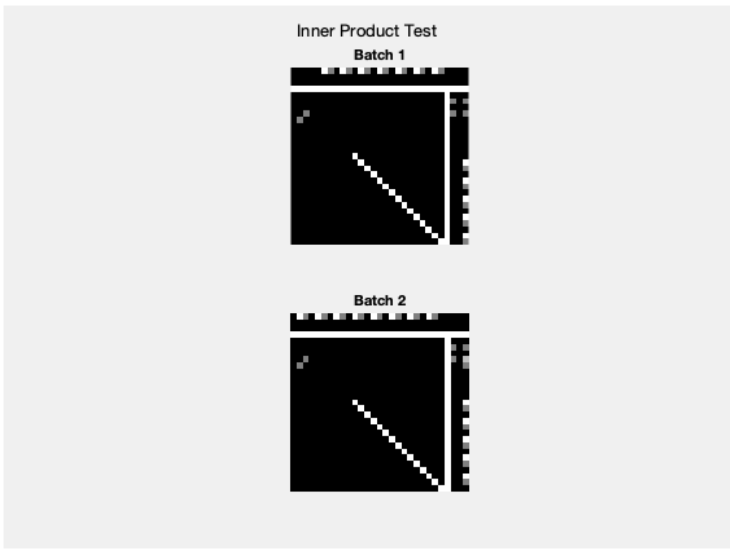
ckyirong - 301312227

4 Late Days Used

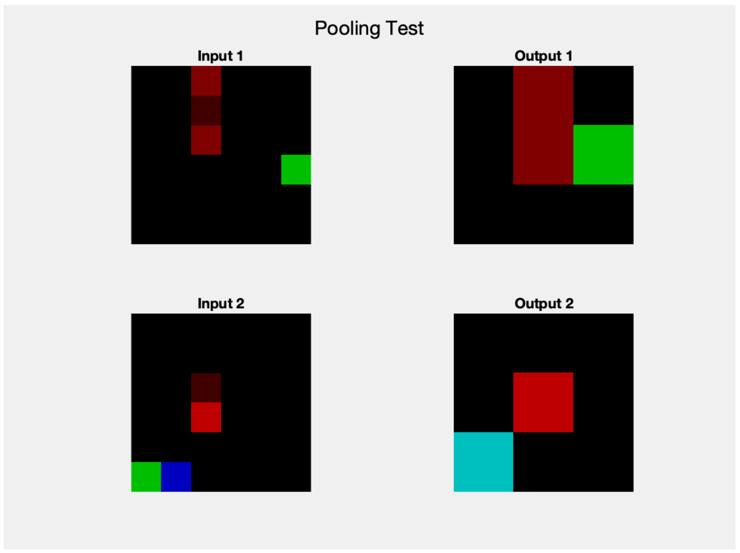
Here are my results for section **1**. The code for section and 1 and 2 can be used as reference.

As a brief summary, for max pooling I grabbed each individual pixel (forming a grid) over the images and had if statements to detect the edge, or bottom corner. For convolution i did not do it this way as it seemed Exhaustive so instead i took advantage of matlabs ability do do calculations in 3-d so i performed convolutions For each channel of the image on the 3 filters, 3 at a time.

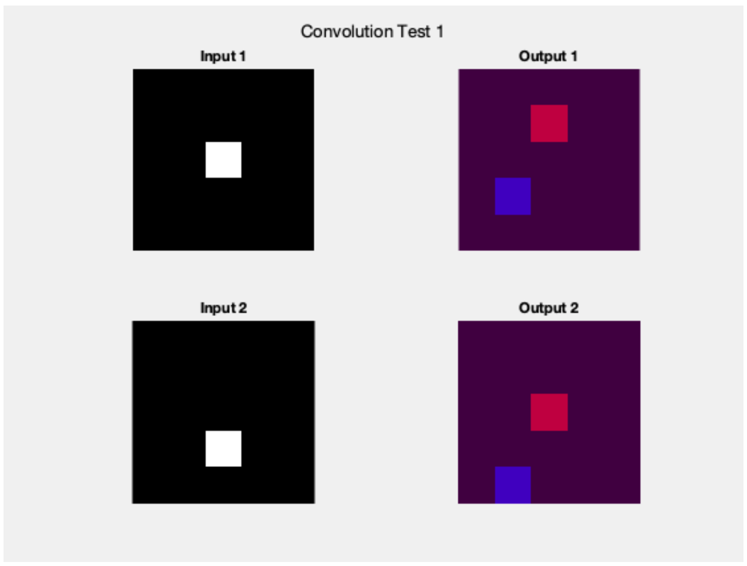
1.1

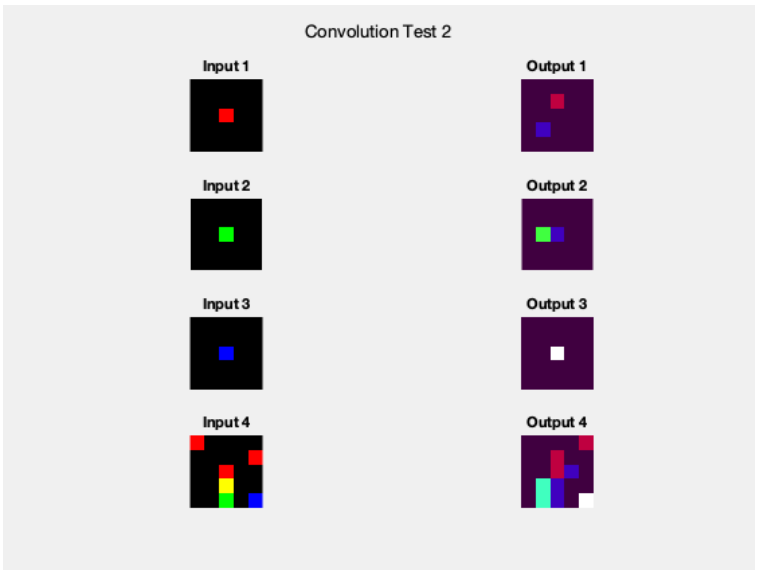


1.2

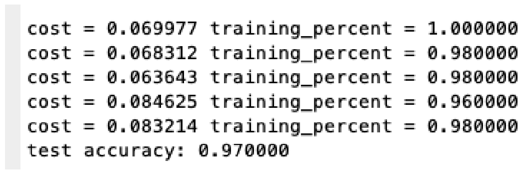


1.3





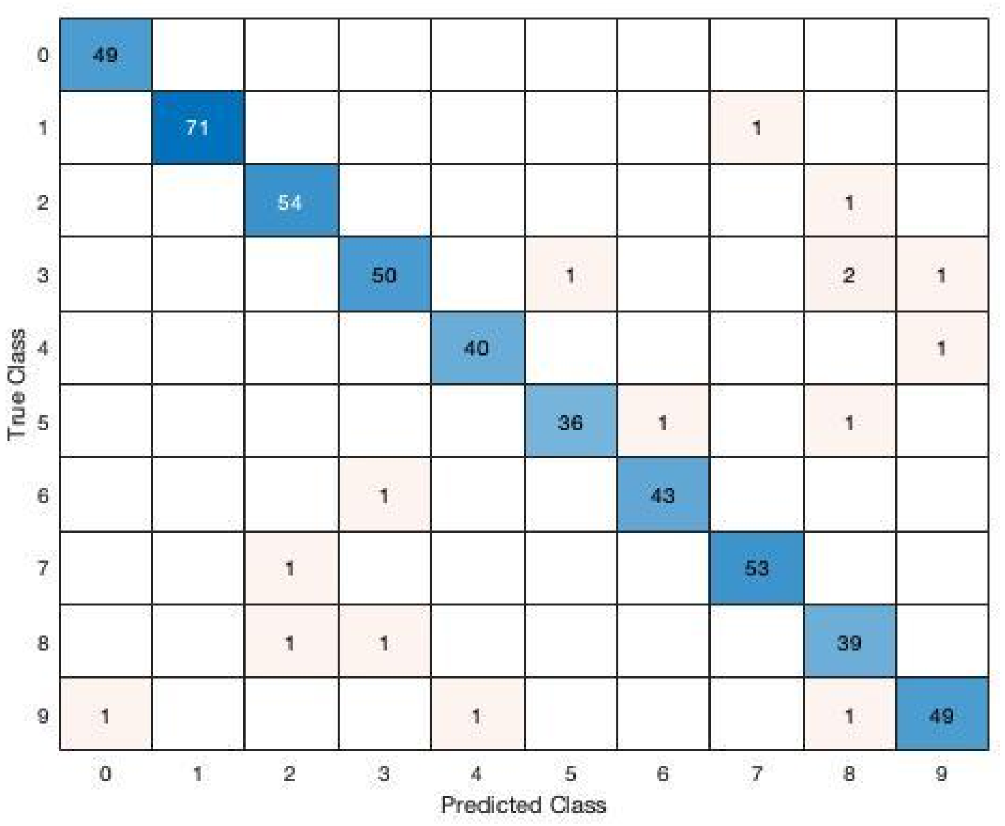
3.1
Due to my inefficient code, train_lenet took around 5 hours to complete. Here is an image of the last line printed when training the model:



```
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
```

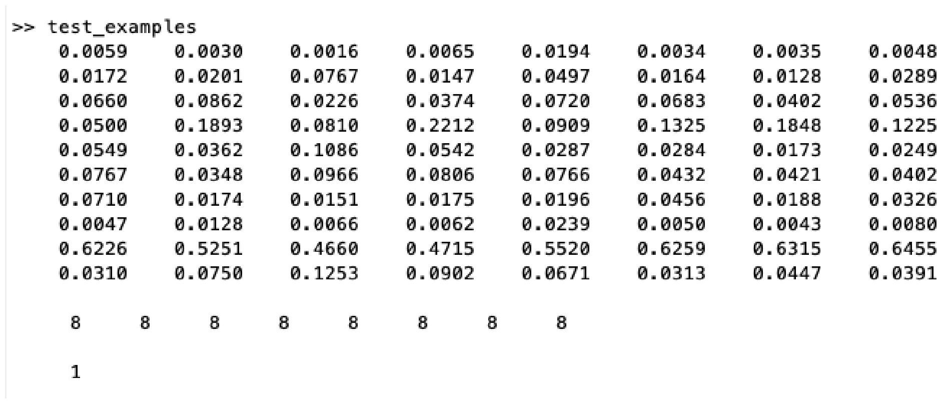
3.2
Here is the confusion matrix generated using: `confusionmat(ytest, predictions);` and `confusionchart(C, [0:9])`

The most confused pair seems to be 3 and 8. This is possibly due to the fact that the number themselves look very similar only differing in a connecting line along the left side. I think this throws off the network since they are quite similar in strokes. The rest of the pairs are all tied for second.

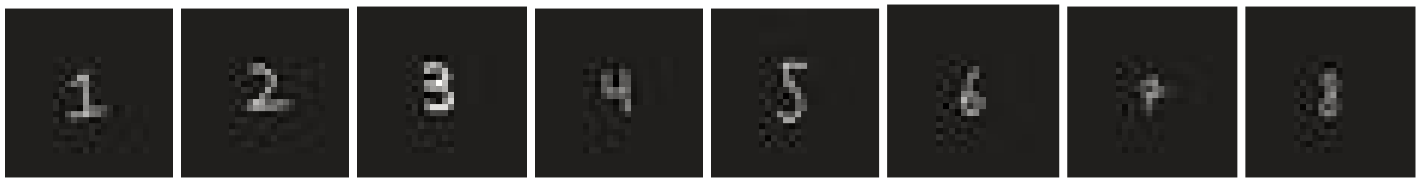


	0	1	2	3	4	5	6	7	8	9
0	49									
1		71						1		
2			54						1	
3				50		1			2	1
4					40					1
5						36	1		1	
6				1			43			
7			1					53		
8			1	1					39	
9	1				1				1	49

3.3 My model performed quite poorly on predicting real world data. My images may be too blurry as i used my ipad to draw a number, took a screenshot, cropped the photo to center my number, and then resized using an online tool to 28x28. The model predicted the number “8” for every single input. This is either a preprocessing issue or the model has completely overfit to the number 8. However, with the MNIST data this is not a problem.

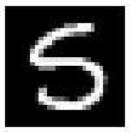


```
>> test_examples
    0.0059    0.0030    0.0016    0.0065    0.0194    0.0034    0.0035    0.0048
    0.0172    0.0201    0.0767    0.0147    0.0497    0.0164    0.0128    0.0289
    0.0660    0.0862    0.0226    0.0374    0.0720    0.0683    0.0402    0.0536
    0.0500    0.1893    0.0810    0.2212    0.0909    0.1325    0.1848    0.1225
    0.0549    0.0362    0.1086    0.0542    0.0287    0.0284    0.0173    0.0249
    0.0767    0.0348    0.0966    0.0806    0.0766    0.0432    0.0421    0.0402
    0.0710    0.0174    0.0151    0.0175    0.0196    0.0456    0.0188    0.0326
    0.0047    0.0128    0.0066    0.0062    0.0239    0.0050    0.0043    0.0080
    0.6226    0.5251    0.4660    0.4715    0.5520    0.6259    0.6315    0.6455
    0.0310    0.0750    0.1253    0.0902    0.0671    0.0313    0.0447    0.0391

     8     8     8     8     8     8     8     8

     1
```

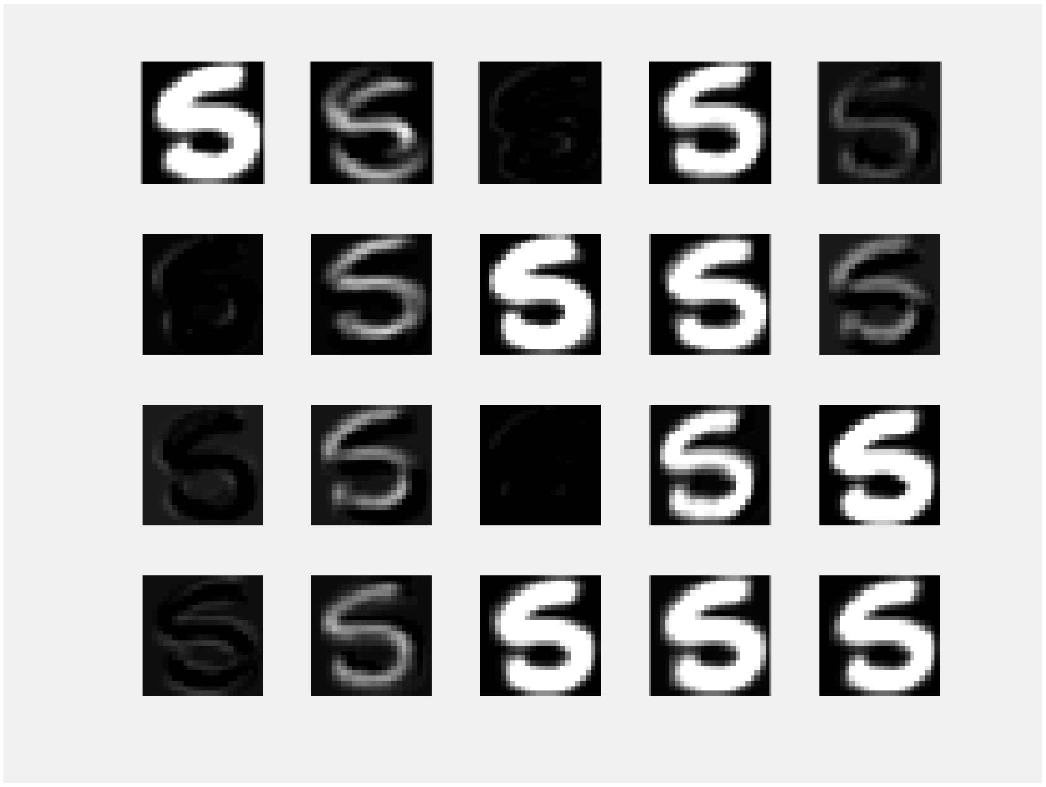


As we can see above, the images are quite blurry but it’s still odd that the model is only predicting 8. From the confusion matrix we can also see that our model predicted the number 8 incorrectly more than any other number.

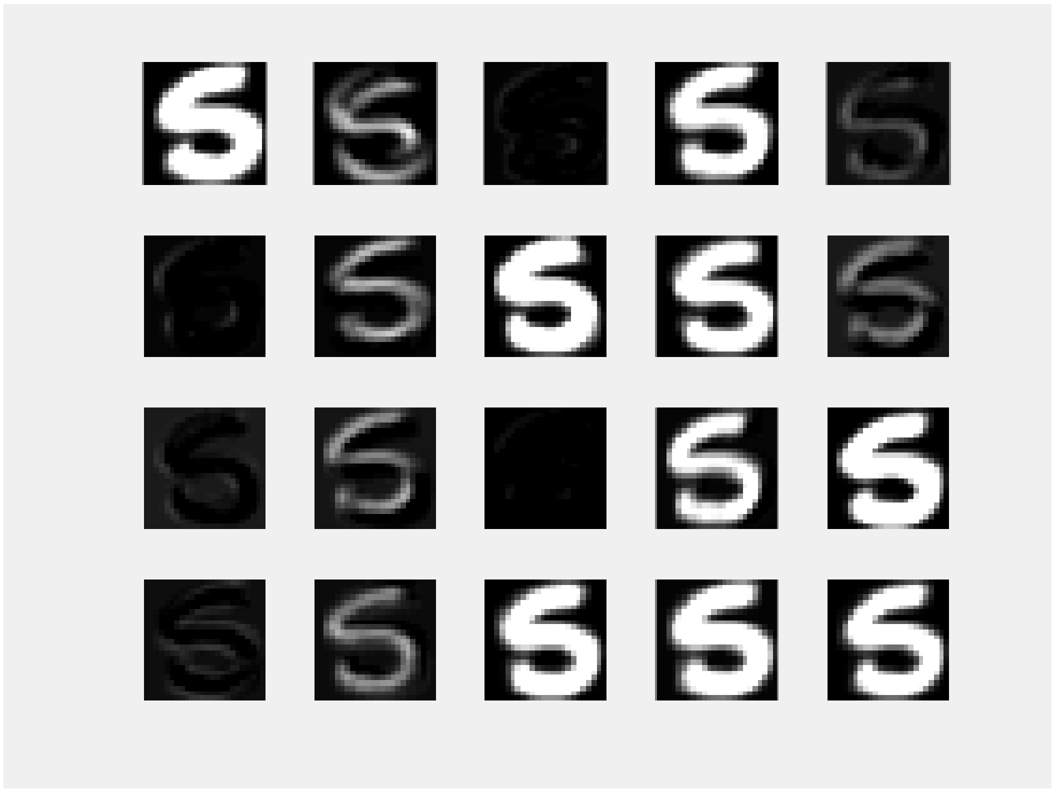
4.1
Orignal Number:



CONV Images:



RELU Images:



4.2

The first Conv layer seems to learn edges of the digits which can be seen by the outlines of each image. Also, they seem to be retaining the full image/shape of the digit. However, some images are completely dark- this could mean that they weren’t even activated there at all. Non activated layers could indicate that the features were not properly.

The RELU outputs look exactly the same as the Conv layer outputs. In this scenario this may make sense sense all RELU does is max(0, x). In matlab, imshow() function seems to display negative values as black anyways. Since 0 is black in imshow() then all negative values are 0 in RELU which means the images will be identical due to the implementation.