

CMPT412 – Assignment 1

Report

(Using 1 free late day)

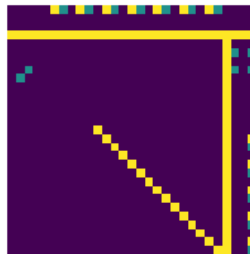
Name: Canh Nhat Minh Le
Student Number: 301384865
Email: cnle@sfu.ca

Q1. Result:

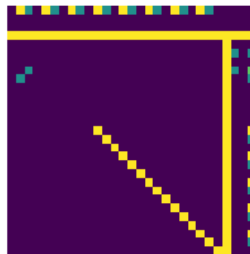
1.1. Inner Product Test:

Inner Product Test

Batch 1



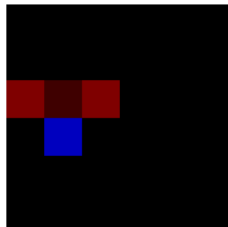
Batch 2



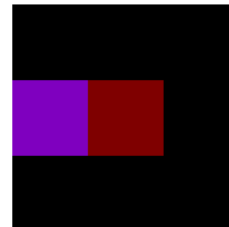
1.2: Pooling Test:

Pooling Test

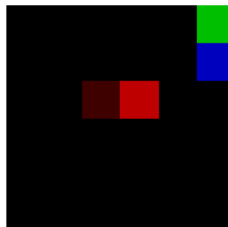
Input 1



Output 1



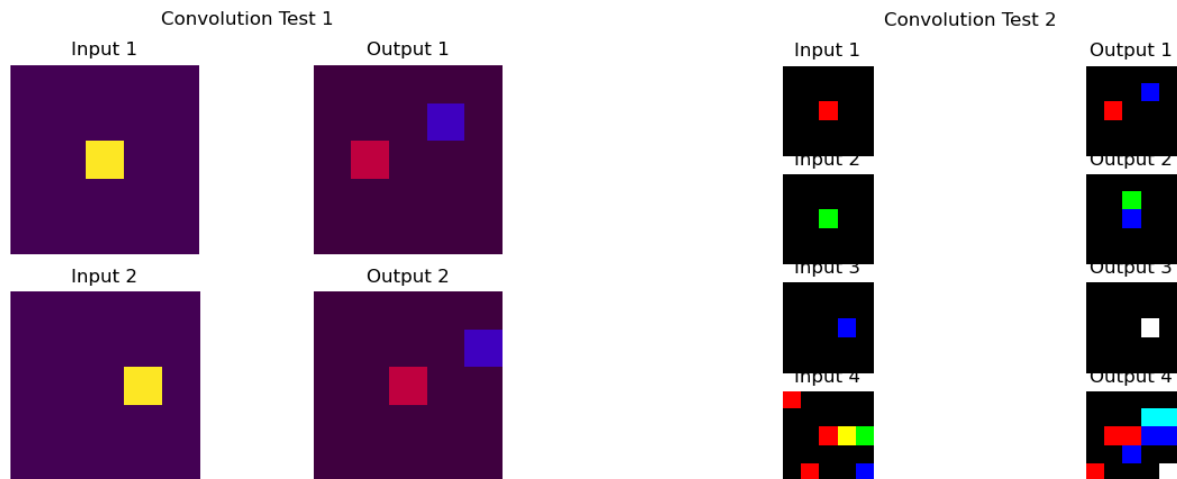
Input 2



Output 2



1.3: Convolution Test:



Q3.

3.1: A full log of training:

cost = 2.3025850929940455 training_percent = 0.03

0

test accuracy: 0.104

cost = 3.785015929490838 training_percent = 0.62

cost = 2.3838366512406584 training_percent = 0.92

cost = 0.7045031274267878 training_percent = 0.94

cost = 0.5960525416316059 training_percent = 0.94

cost = 0.4982438150513 training_percent = 0.95

cost = 0.07301611183325665 training_percent = 0.97

cost = 0.03608708034467231 training_percent = 0.98

cost = 0.056427447927317236 training_percent = 0.98

cost = 0.007703436582670364 training_percent = 0.99

cost = 0.12197846267866844 training_percent = 0.98

500

test accuracy: 0.956

cost = 0.09278740353636787 training_percent = 0.98

cost = 0.09260318841447132 training_percent = 0.98

cost = 0.12683349217827786 training_percent = 0.99

cost = 0.03238492328365088 training_percent = 0.99

cost = 8.824844434715568e-06 training_percent = 1.0

cost = 1.1812592573849802e-08 training_percent = 1.0

cost = 2.3422211767009587e-06 training_percent = 1.0

cost = 2.7301787211577347e-07 training_percent = 1.0

cost = 0.0017924667708612248 training_percent = 1.0
cost = 5.732722311600216e-06 training_percent = 1.0
1000

test accuracy: 0.972

cost = 0.0011079556096477473 training_percent = 1.0
cost = 0.005783923586280409 training_percent = 1.0
cost = 9.070705525815472e-06 training_percent = 1.0
cost = 1.9305886351441283e-06 training_percent = 1.0
cost = 2.097627840692941e-06 training_percent = 1.0
cost = 0.007492962765634258 training_percent = 0.99
cost = 1.4593859321726517e-08 training_percent = 1.0
cost = 8.314452744093253e-08 training_percent = 1.0
cost = 9.282511102751405e-05 training_percent = 1.0
cost = 7.916033979913916e-09 training_percent = 1.0
1500

test accuracy: 0.97

cost = 9.735797414752594e-09 training_percent = 1.0
cost = 3.6153375472433736e-07 training_percent = 1.0
cost = 7.605881875093993e-09 training_percent = 1.0
cost = 9.271334044913283e-08 training_percent = 1.0
cost = 0.000419485324188713 training_percent = 1.0
cost = 1.785958048223906e-12 training_percent = 1.0
cost = 6.047664525799374e-08 training_percent = 1.0
cost = 1.2069087107021663e-08 training_percent = 1.0
cost = 6.164401867555524e-09 training_percent = 1.0

Test accuracy is 97%

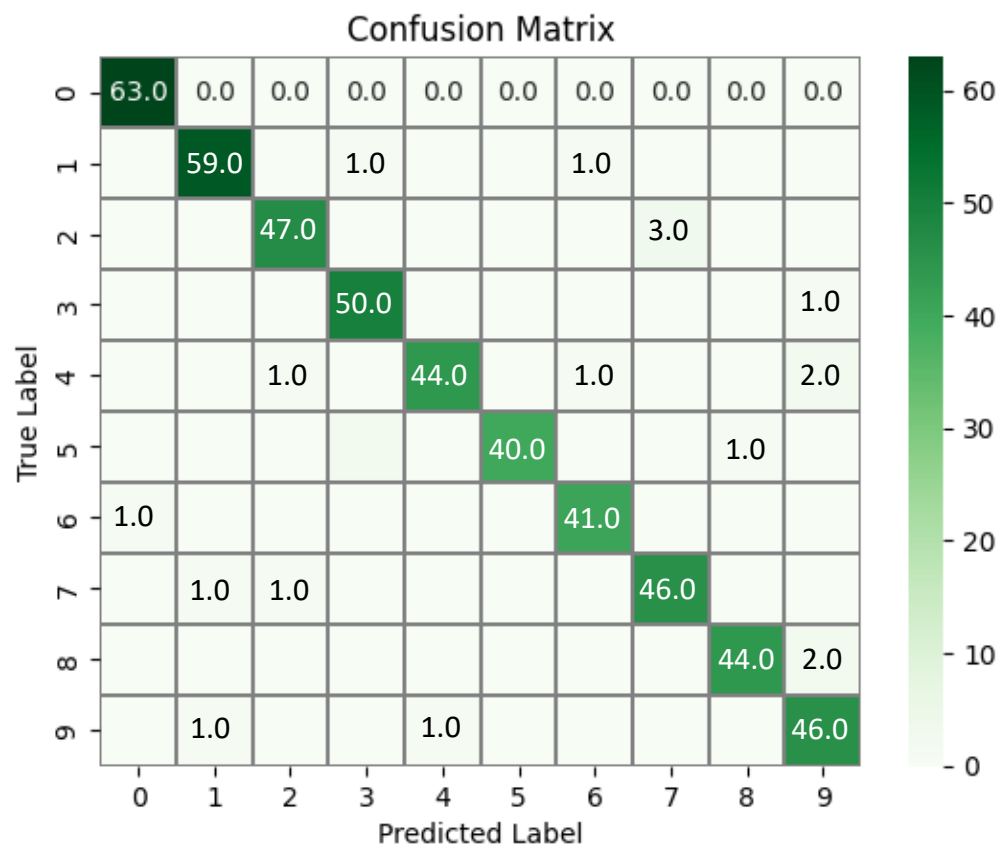
3.2:

Confusion Matrix Obtained:

```

[[63  0  0  0  0  0  0  0  0  0]
 [ 0 59  0  1  0  0  1  0  0  0]
 [ 0  0 47  0  0  0  0  3  0  0]
 [ 0  0  0 50  0  0  0  0  0  1]
 [ 0  0  1  0 44  0  1  0  0  2]
 [ 0  0  0  2  0 40  0  0  1  0]
 [ 1  0  0  0  0  0 41  0  0  0]
 [ 0  1  1  0  0  0  0 46  0  0]
 [ 0  0  0  0  0  0  0  0 44  2]
 [ 0  1  0  0  1  0  0  0  0 46]]

```



Based on this matrix, it appears that the model find it difficult to predict digit 2 the most, constantly mistaken it with number 7. This might be because both 2's and 7's tends to have an right cross and down to the left stroke. Model also mistaken 4's and 9's since they have a loop, same with 8's and 9's

3.3:

I sketched these numbers on an iPad and crop them out in square

0 1 2 3 4 6 9 1

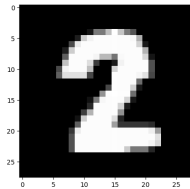
These numbers are put in an array with the following order:

Input = [2,3,1,0,4,6,1,9]

Predicted_result = [2,3,5,0,4,6,1,1]

Q4.1:

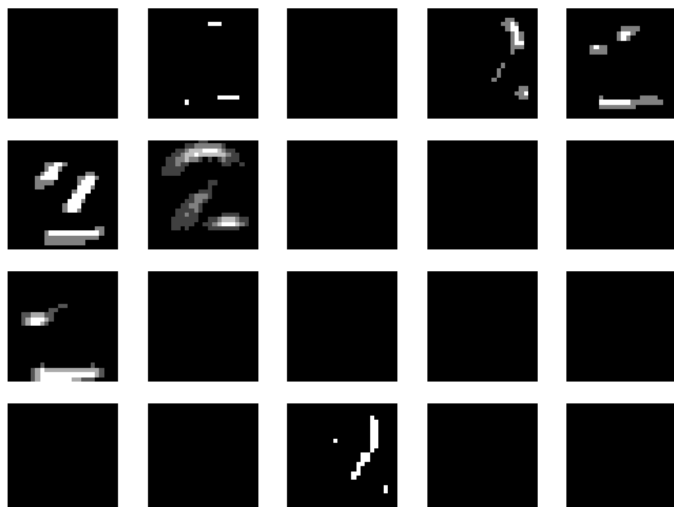
Input:



Convolution layer:



ReLU Layer:



4.2:

Convolution layer: In comparison to the input, a lot of other features that are not present in the input are displayed. We can see in some images that there are certain highlights on the number 2, and some other emphasize the shadow behind the number, which indicates that the original image are 3D.

ReLU: The ReLU layer resulted in some completely dark images. This is because ReLU pass convert all the negative values to 0

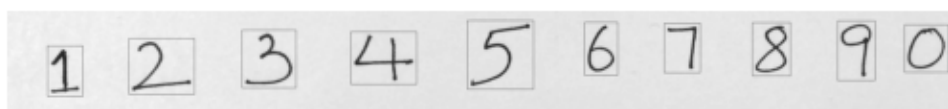
Question 5:

In the ec.py code, four samples are loaded into an array called "image_files". I then extracted each sample by indexing (0 = image4; 1 = image1; 2 = image2; 3 = image3) and continues to process each sample.

I utilized a series of built-in cv2 functions in order to achieve the bounding of each digit in the samples. I first convert all the samples to grayscale, and then used cv2.threshold to set thresholds. I also added extra padding to each of the bounded digit because I found that giving more space surrounding the digit increased prediction accuracy. Because the spaces between digits vary in each sample, the padding is also different.

Below are the results:

For the **first** sample:

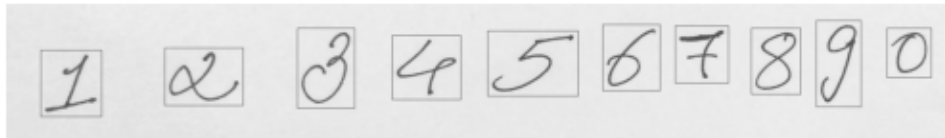


The prediction result:

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

The model predicted 2 correct labels out of 10 digits, which gives the rate of accuracy 20%

Second sample:

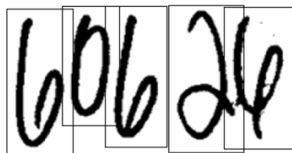
A row of ten boxes, each containing a handwritten digit from 1 to 0 in sequence.

Prediction result:

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

The model predicted 2 correct labels out of 10 digits, which gives the rate of accuracy 20%

Third sample:

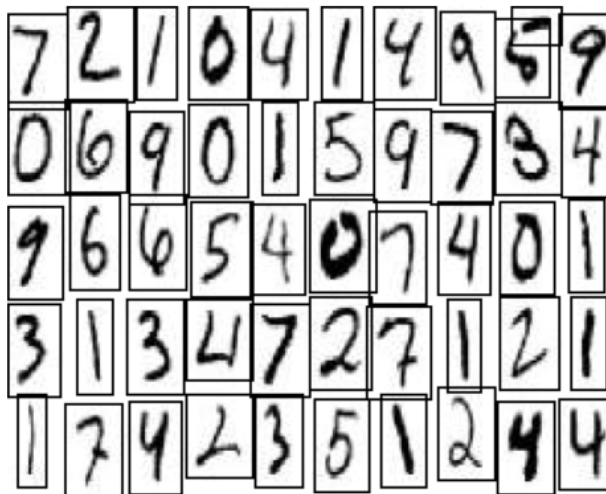
A row of five boxes, each containing a handwritten digit: 6, 0, 6, 2, and 4.

Prediction result:

6	6	6	0	2
6	6	6	0	2

The model predicted 6 correct labels out of 6 digits, which gives the rate of accuracy 100%

Fourth sample:



Prediction result:



The model predicted 31 correct labels out of 50 digits

Based on the rate of accuracy from part 3.3 and part 5, I notice that the model predicts better when the digits are not too cursive, and rather look simple.