CMPT 412
Project 1
Digit recognition with convolutional neural networks
Marco Hiu Yeung Lai
301356237

Part 1: Forward Pass

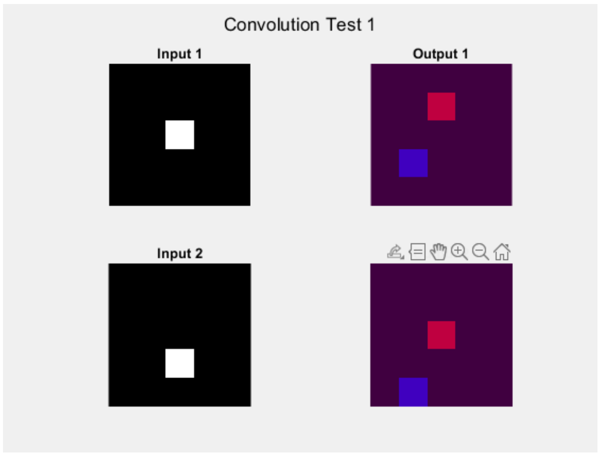
1.1 Inner Product Layer: Loop the batch and apply the formula f(x) = W x + b where W is the weight and b is the bias which can be called from the param(param.b and param.w).

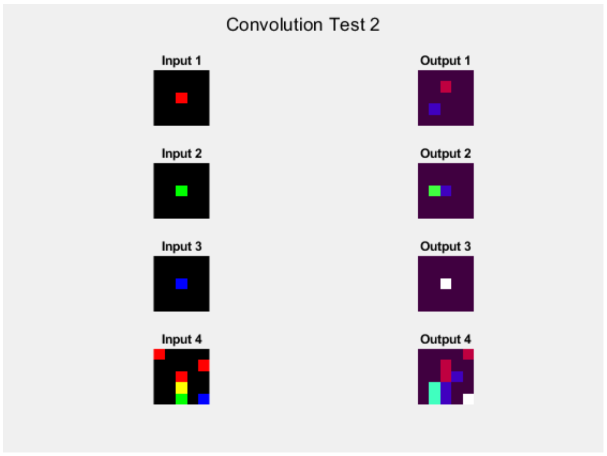
1.2 Pooling Layer: Loop through the pixel with the input height and width of the given stride.

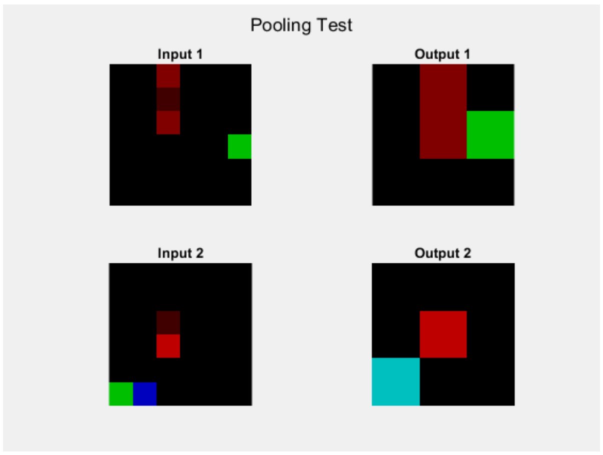
1.3 Convolution Layer: Use the given function im2col to create a vector of pixels and multiply by the weight.

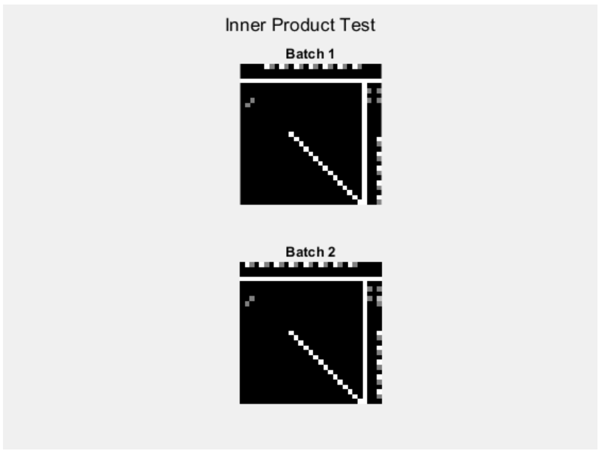
1.4 ReLU: Simply just return the max(input.data, 0).

Figures generated from test_components.m:









Part 2: Back Propagation

2.1 ReLu

h_i = f_i(w_i, h_(i - 1))
Input_od = dl/dh_(i -1)
= dl/(dh_i) * (dh_i)/ (dh_(i - 1))

For the forward layer we got:

dl/dh_i = input.data if x > 0, else 0

The derivative of relu is 1 when x > 0 else 0

Therefore, the relu_backward can be done by multiplying the output.diff and input.data > 0

2.2 Inner Product Layer

Since h_i = wx + b

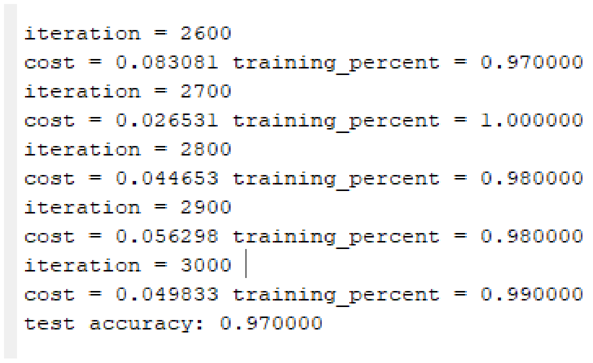
dl/dw = dl/d_i * dh_i/dw_i = output.diff * input.data
dl/dh_(i-1) = dl/dh_i * dh_i/dh_(i-1) = output.diff * param.w
dl/db = dl/dh_i = dl/dh_i * dh_i/db = output.diff * 1

From the equations above we can get param_grad.w = (output.diff * input.data')' and param_grad.b is just the sum of the output.diff. For more details, please refer to the implementation in relu_backward.m and inner_product_backward.m.

Part 3: Training

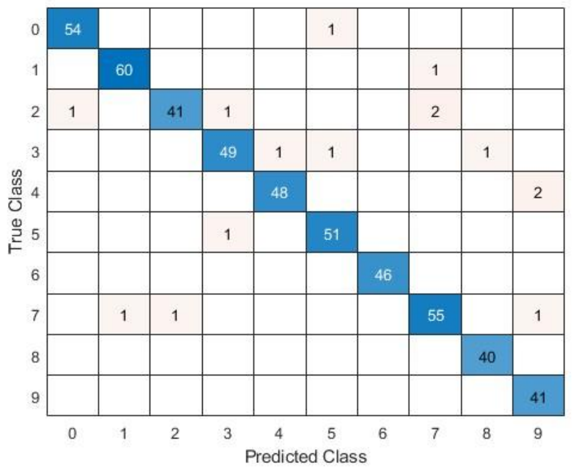
3.1 Training

For 3000 iterations, the training percent is 99% and test accuracy is 97%



```
iteration = 2600
cost = 0.083081 training_percent = 0.970000
iteration = 2700
cost = 0.026531 training_percent = 1.000000
iteration = 2800
cost = 0.044653 training_percent = 0.980000
iteration = 2900
cost = 0.056298 training_percent = 0.980000
iteration = 3000
cost = 0.049833 training_percent = 0.990000
test accuracy: 0.970000
```

3.2 Test the network



True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	54					1				
1		60						1		
2	1		41	1				2		
3				49	1	1			1	
4					48					2
5				1		51				
6							46			
7		1	1					55		1
8									40	
9										41

The top two confusing digits are 2 and 4. 2 and 7 have a similar pattern except for the bottom line of 2 especially when the detection of stroke and semi-circle were not sensitive. 4 and 9 have a similar pattern as well especially when the written “4” is closed on top, there is a chance that it can be detected as a circle and therefore predicted as 9.

3.3 Real-world testing

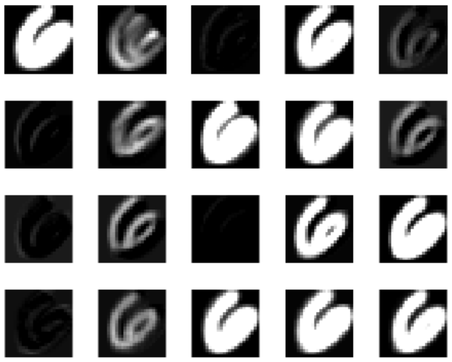
For this part, I used Paint 3D to generate 28*28 pixels with text digits 1 to 5.
Images used:

The network predicted all correctly except 4. 4 is predicted as 2 with a greater chance, I believe it is because the upper part of the digit “4” shares a larger portion in the image, both downward and the right stokes are too short.

Part 4: Visualization

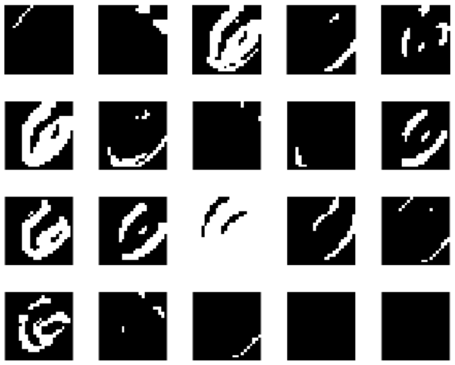
Input:

CONV Layer



Most of the digits are clearly viewed except for the middle picture in the third column. It is because the network did not identify properly. The corresponding image in the ReLU one is completely black as well(flipped).

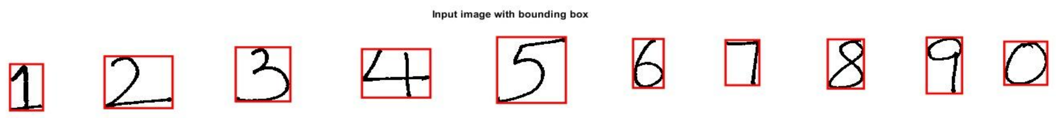
ReLU Layer

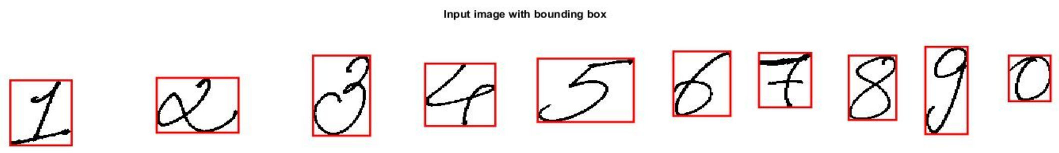


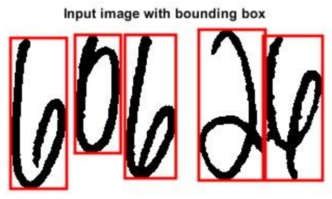
For the ReLU part, the pixel values are supposed to be 0 if pixel values are less than 0 and 1 if greater than 0. However, because the matlab imshow function renders negative pixel values as black, if I were to display the output relu function, it would have looked exactly like CONV Layer. Therefore I changed the pixel values that are less than 0 to 1 and greater than 0 to 0. The figure is showing the affected pixels where the fewer white pixels represent the white digits since they are flipped.

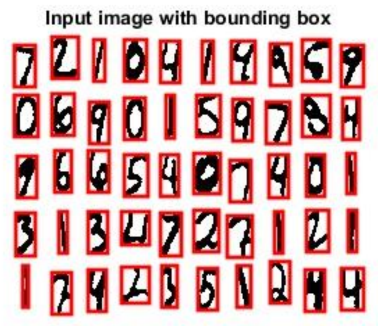
Part 5: Image Classification

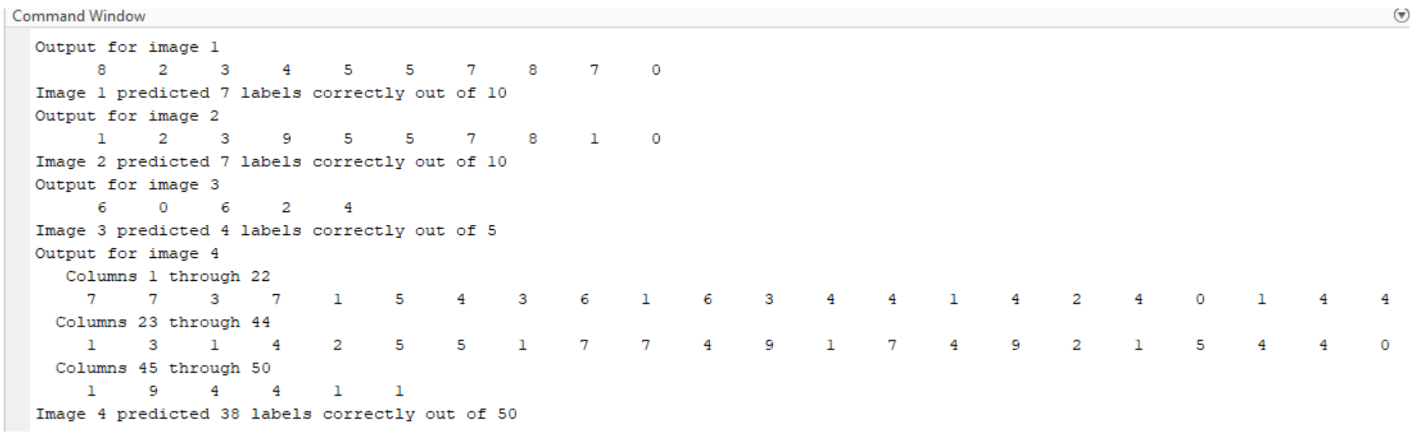
For this part, I first load all the images and convert them to grayscale, binarize, and remove small-scaled objects. Then I use bwlabel and regionprops to get the connected components. However, the prediction correctness is very low if I passed the images directly to the network. Therefore I padded images based on the ratio and fix with the height and width. For the first image, after I resize and indicate it is box, the correct prediction raised from 5 to 7 out of 10.











```
Output for image 1
     8     2     3     4     5     5     7     8     7     0
Image 1 predicted 7 labels correctly out of 10
Output for image 2
     1     2     3     9     5     5     7     8     1     0
Image 2 predicted 7 labels correctly out of 10
Output for image 3
     6     0     6     2     4
Image 3 predicted 4 labels correctly out of 5
Output for image 4
   Columns 1 through 22
    7     7     3     7     1     5     4     3     6     1     6     3     4     4     1     4     2     4     0     1     4     4
  Columns 23 through 44
    1     3     1     4     2     5     5     1     7     7     4     9     1     7     4     9     2     1     5     4     4     0
  Columns 45 through 50
    1     9     4     4     1     1
Image 4 predicted 38 labels correctly out of 50
```

Both image1 and image2 got 7/10 correct which is 70%, image3 4/5 which is 80%, and image 4 38/50 which is 76%. Overall 56/75 which is 75% accuracy.