CMPT 412 Project 1

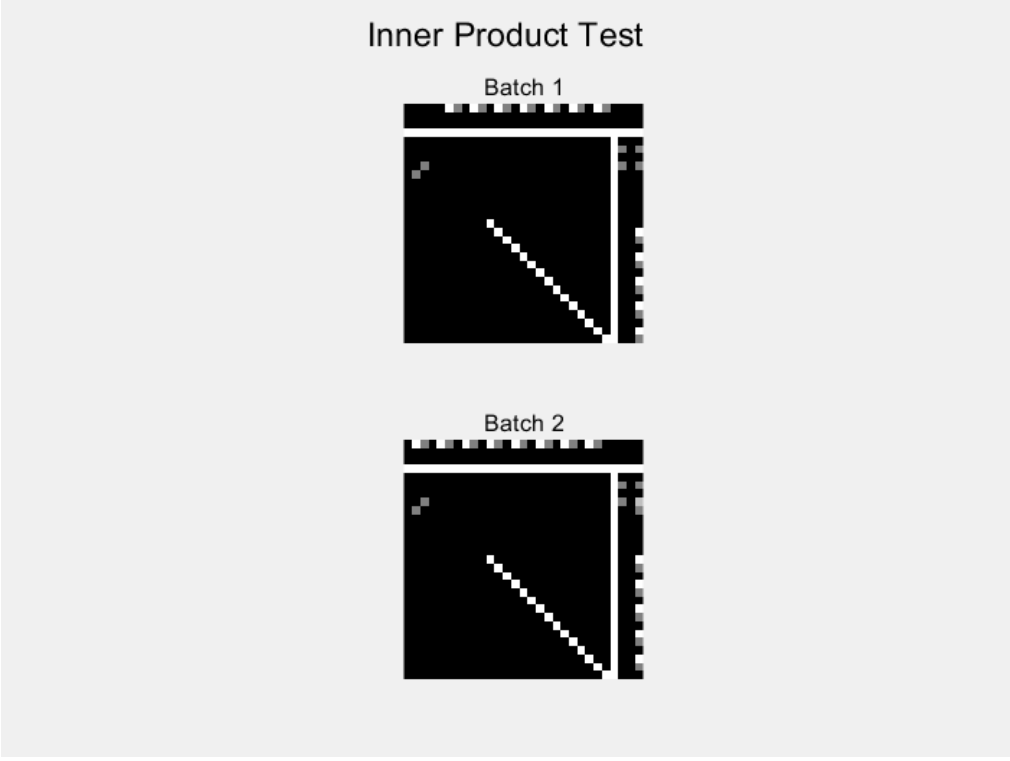
Huiyi Zou 301355563

Part 1: Forward Pass

Q 1.1 Inner Product Layer

For this layer, we have $f(x) = Wx + b$. I transpose the param.w and param.b to interchange the row and column for matrix calculation.

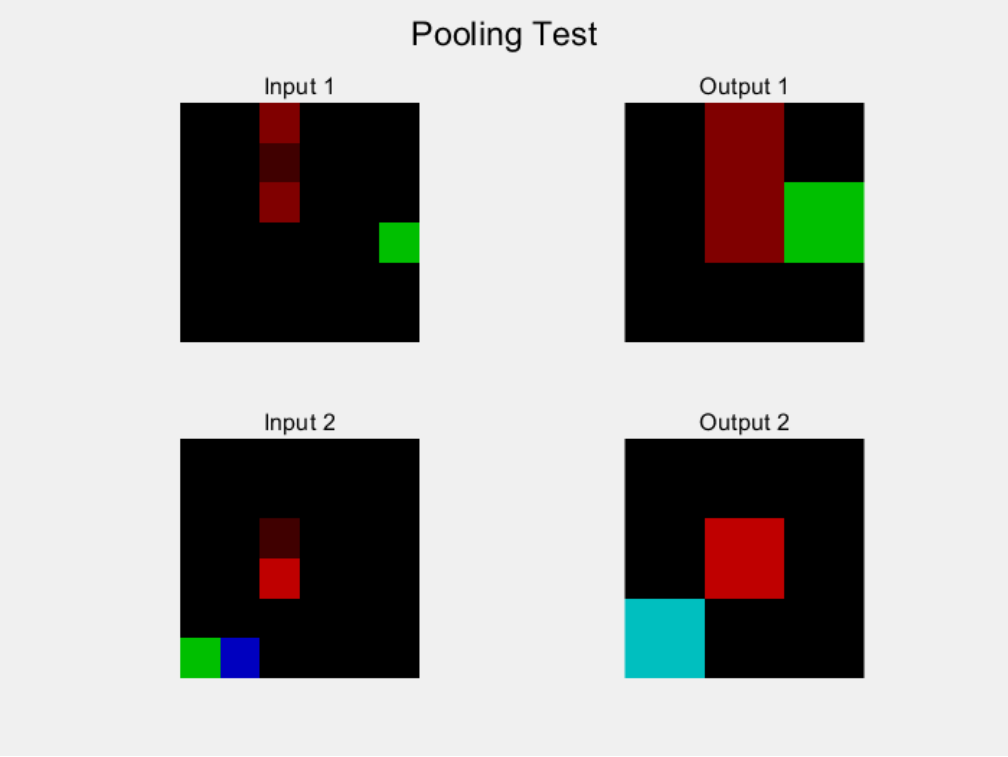
```
% Replace the following line with your implementation.
output.height = input.height;
output.width = input.width;
output.channel = input.channel;
output.batch_size = k;

% output f(x) = W * x + b
output.data = param.w' * input.data + param.b';
```



Q 1.2 Pooling Layer

For this layer, I use loops to form kernels and find the max value in each kernel to reduce the size of the feature maps.

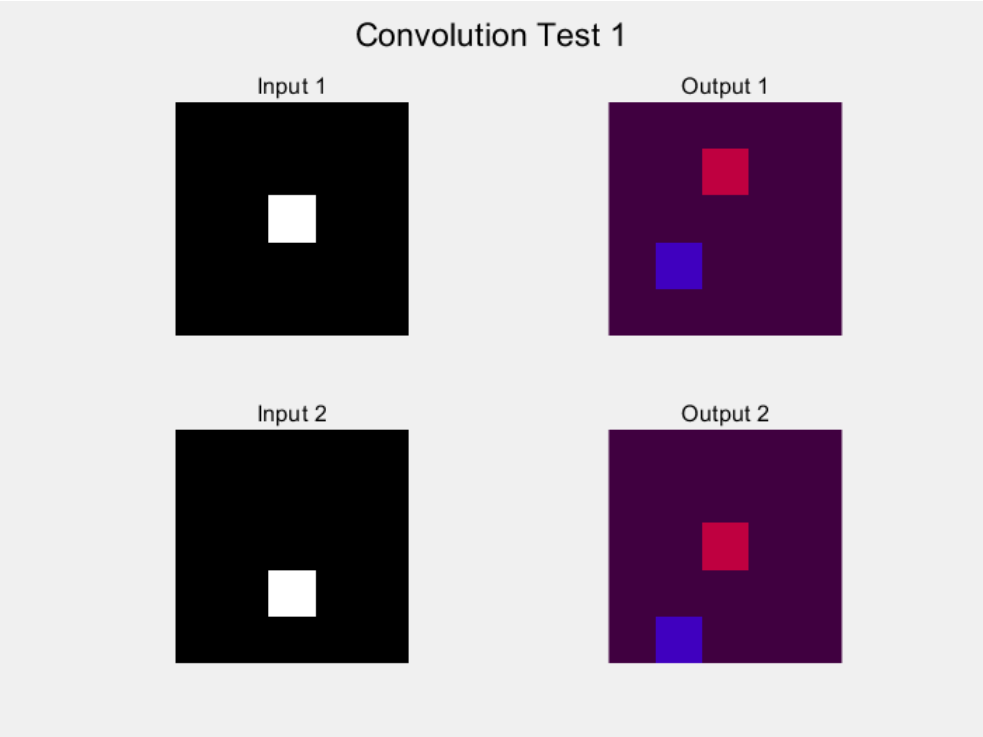
```
% Replace the following line with your implementation.
output.data = zeros([h_out, w_out, c, batch_size]);
data = reshape(input.data, [h_in, w_in, c, batch_size]);
data = padarray(data, [pad pad], 0, 'both');

% form each kernel and find the max value in the kernel
for i = 1:h_out
    row = (i-1) * stride;
    for j = 1:w_out
        col = (j-1) * stride;
        kernel = data(row+1:row+k, col+1:col+k, :, :);
        output.data(i, j, :, :) = max(max(kernel));
    end
end

output.data = reshape(output.data, [h_out*w_out*c, batch_size]);
```

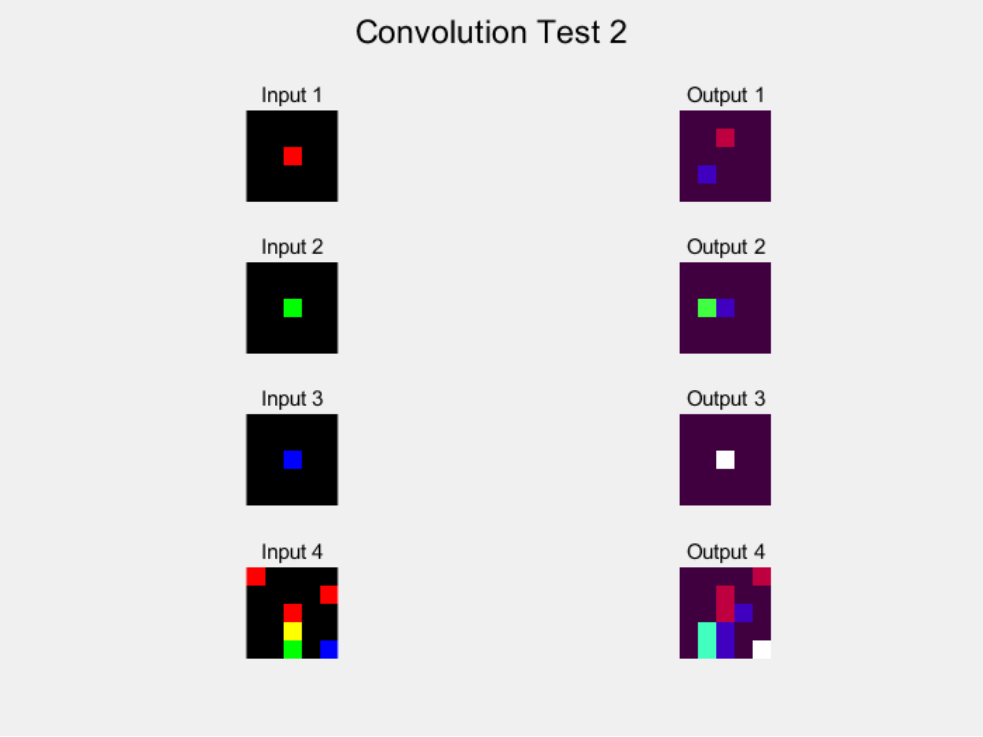


Q 1.3 Convolution Layer

In this layer, we have $f(X, W, b) = X * W + b$. For each batch size, I use im2col_covn() to rearrange the matrix, reshape and transpose it for matrix calculation.

```
%% Fill in the code
% Iterate over the each image in the batch, compute response,
% Fill in the output datastructure with data, and the shape.
output.height = h_out;
output.width = w_out;
output.channel = num;
output.batch_size = batch_size;

output.data = zeros([h_out*w_out, num, batch_size]);

for b = 1:batch_size
    input_n.data = input.data(:, b);

    % rearrange matrix blocks into columns
    col = im2col_conv(input_n, layer, h_out, w_out);

    % reshape the size for the calculation
    col = reshape(col, [k*k*c, h_out*w_out]);

    % the convonlution operation: f(X,W,b) = X * W + b
    output.data(:, :, b) = col'*param.w + param.b;
end

output.data = reshape(output.data, [h_out*w_out*num, batch_size]);
```


Convolution Test 1
Input 1
Output 1
Input 2
Output 2


Convolution Test 2
Input 1
Output 1
Input 2
Output 2
Input 3
Output 3
Input 4
Output 4

Q 1.4 ReLU

The ReLU function is $f(x) = \max(x, 0)$, which means if $x \geq 0$, it will be x; if $x < 0$, it will be 0. Here, output.data(output.data<0)=0 find all output.data which are less than 0 and set 0.

```
% Replace the following line with your implementation.
output.data = input.data;

% replace all negative numbers by 0
output.data(output.data<0) = 0;
```

Part 2 Back Propagation

Q 2.1 ReLU

We have $h_i = \max(h_{i-1}, 0)$

Thus, if $h_{i-1} \geq 0, h_i = h_{i-1}, \frac{dhi}{dh_{i-1}} = 1$;

if $h_{i-1} < 0, h_i = 0, \frac{dhi}{dh_{i-1}} = 0$.

Since $\frac{dl}{dh_{i-1}} = \frac{dl}{dh_i}\frac{dhi}{dh_{i-1}}$, if $h_{i-1} \geq 0$, $\frac{dl}{dh_{i-1}} = \frac{dl}{dh_i}$; if $h_{i-1} < 0$, $\frac{dl}{dh_{i-1}} = 0$

```
% Replace the following line with your implementation.
% hi = max(hi-1, 0)
% dhi/dhi-1 = 1 or 0
% dl/dhi-1 = (dl/dhi)(dhi/dhi-1)
input_od = output.diff;
input_od(input.data<0) = 0;
```

Q 2.2 Inner Product Layer

We have $h_i = w_i * h_{i-1} + b_i$

Then $\frac{dh_i}{dw_i} = h_{i-1}$, $\frac{dh_i}{d_{hi-1}} = wi$, $\frac{dh_i}{db_i} = 1$

Thus, $\frac{dl}{dw_i} = \left(\frac{dl}{dh_i}\right)\left(\frac{dh_i}{dw_i}\right) = \left(\frac{dl}{dh_i}\right)(h_{i-1})$

$\frac{dl}{db_i} = \left(\frac{dl}{dh_i}\right)\left(\frac{dh_i}{db_i}\right) = \left(\frac{dl}{dh_i}\right)(1)$

$\frac{dl}{dh_{i-1}} = \left(\frac{dl}{dh_i}\right)\left(\frac{dh_i}{dh_{i-1}}\right) = \left(\frac{dl}{dh_i}\right)(w_i)$

We need to take transpose of matrices for calculation.

```
% Replace the following lines with your implementation.

% for this layer: hi = wi * hi-1 + bi
% dhi/dwi = hi-1; dhi/dhi-1 = wi; dhi/dbi = 1
% dl/dwi = (dl/dhi)(dhi/dwi) = (dl/dhi)(hi-1)
% dl/dbi = (dl/dhi)(dhi/dbi) = (dl/dhi)(1)
% dl/dhi-1 = (dl/dhi)(dhi/dhi-1) = (dl/dhi)(wi)

param_grad.w = (output.diff * input.data')';
param_grad.b = (output.diff * ones(1, input.batch_size)')';
input_od = (output.diff' * param.w')';
```

Part 3 Training

Q 3.1 Training

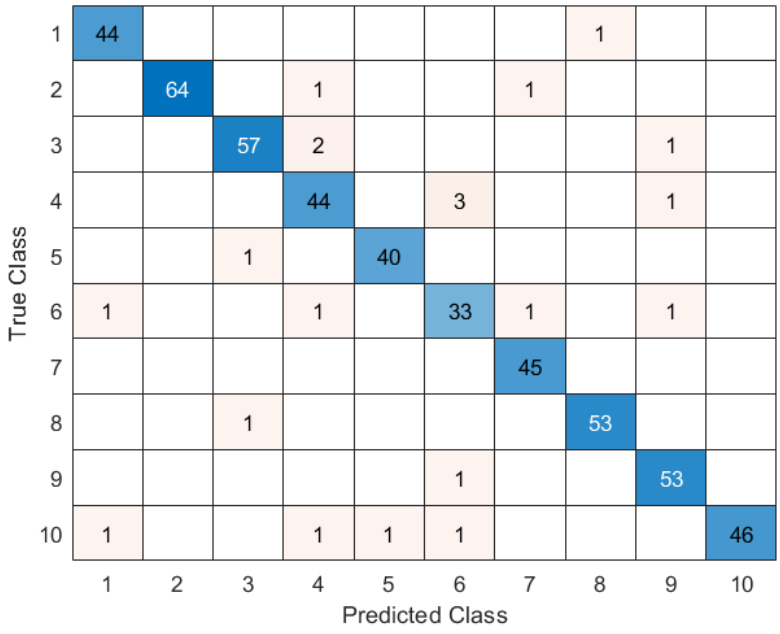
The test accuracy is 97%

```
>> 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

fx >>
```

Q 3.2 Test the network

I use *confusionmat()* to get confusion matrix and *confusionchart()* to show the result.

```
%% Testing the network
% Modify the code to get the confusion matrix
ypred = zeros(1, size(xtest, 2));
for i=1:100:size(xtest, 2)
    [output, P] = convnet_forward(params, layers, xtest(:, i:i+99));

    % get the largest value in each column with their row index (label)
    [M, ypred(i:i+99)] = max(P);
end
% use confusionmat() to get confusion matrix
C = confusionmat(ytest, ypred);
confusionchart(C)
disp('The confusion matrix is')
disp(C)
```



There are 3 times that the actual class is 4 but incorrectly predicted as class 6, which means the digit 3 is predicted as digit 5.

There are 2 times that the actual class is 3 but incorrectly predicted as class 4, which means the digit 2 is predicted as digit 3.

Those errors are acceptable. Those digits are similar.

Q 3.3 Real-world testing

The code is in the file real_world.m. I drew 6 images of digit using Photoshop and saved them in the folder real_world_images. They are 0, 1, 2, 3, 5, 8.

The 6 real-world examples are all correctly predicted.

Example 1:

digit 0

The result:

```
Actual label:
     1

Predicted label:
     1
```

Example 2:

digit 1

The result:

```
Actual label:
     2

Predicted label:
     2
```

Example 3:

digit 2

The result:

```
Actual label:
     3

Predicted label:
     3
```

Example 4:

digit 3

The result:

```
Actual label:
     4

Predicted label:
     4
```

Example 5:

digit 5

The result:

```
Actual label:
     6

Predicted label:
     6
```

Example 6:

digit 8

The result:

```
Actual label:
     9

Predicted label:
     9
```

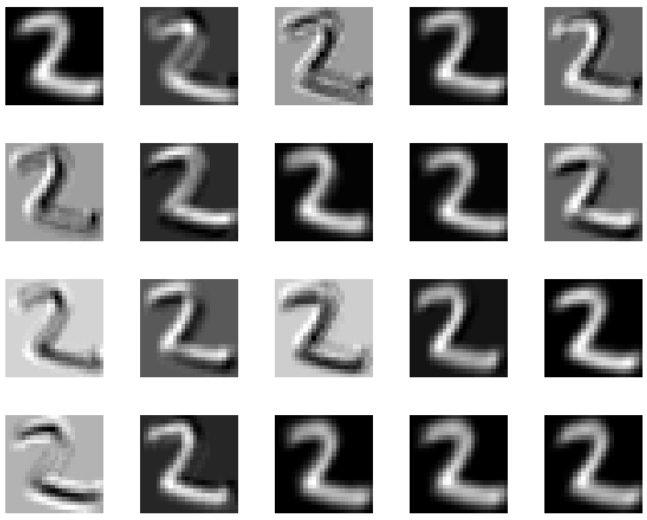
Part 4 Visualization

Q 4.1

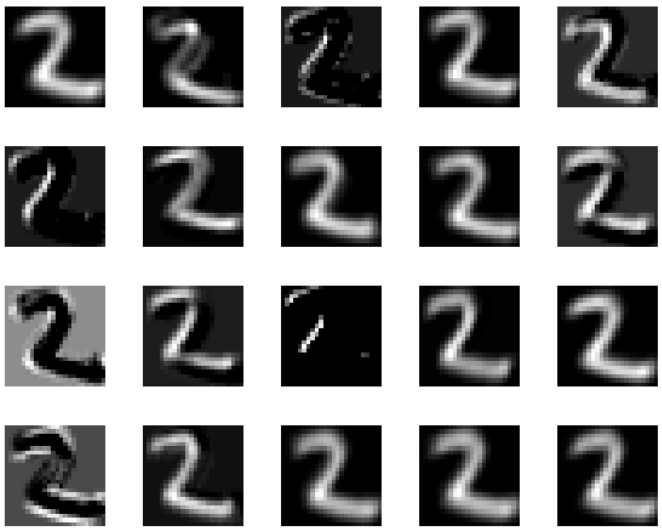
```
% Fill in your code here to plot the features.
% get the output from the layer 2 and the layer 3
output_2 = reshape(output{2}.data, 24, 24, 20);
output_3 = reshape(output{3}.data, 24, 24, 20);

% show CONV layer
figure('NumberTitle', 'off', 'Name', 'CONV layer')
for i = 1:20
    subplot(4, 5, i);
    % scaling the display based on the range of pixel values in output_2
    imshow(output_2(:, :, i)', []);
end

% show RELU layer
figure('NumberTitle', 'off', 'Name', 'RELU layer')
for i = 1:20
    subplot(4, 5, i);
    % scaling the display based on the range of pixel values in output_3
    imshow(output_3(:, :, i)', []);
end
```

The original image 

The CONV layer



The RELU layer



Q 4.2

The feature maps show the result after applying the layer filters to the original image.

We can see that feature maps of the CONV layer have different versions of the digit 2. They focus on different aspects with highlighted features, which depend on their filter weights. Using the Relu layer is to increase the non-linearity in our images. We can see that the feature maps of the RELU layer have a darker color and sharper contrast. The Relu layer fixes the negative value computed from the convolutional layer.

Part 5 Image Classification

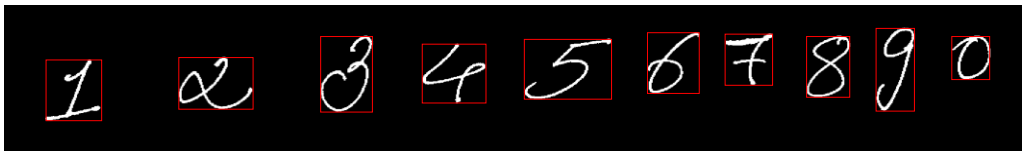
The code ec.m in the folder matlab/

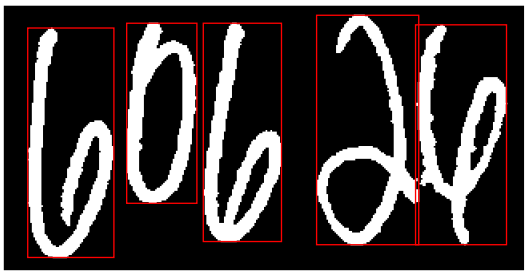
1. I use the *rgb2gray()* function to convert RGB image to grayscale. Since the provided images have a light background and dark foreground, I reverse the color of the images. Then the background color is removed through the morphological opening to form an image with a darker background.

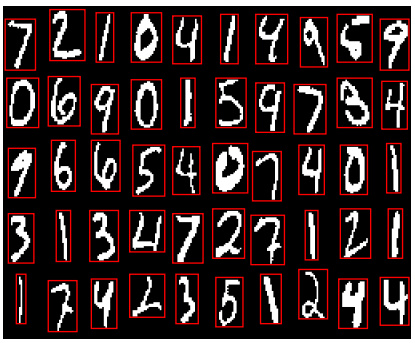
   For this step, I referred to this documentation:

   https://www.mathworks.com/help/images/correcting-nonuniform-illumination.html

2. The *bwlabel()* function finds the connected components and sets labels for them. There is a digit 5 which is unconnected in image4, so I use *imdilate()* function to make it be connected. The *regionprops()* function is used to create the bounding boxes. I place the boxes around each digit using *rectangle()* function.







3. I extract sub-images by the positions of bounding boxes and pad them to squares by their length and width. I also pad zeros with suitable padsize around the sub-images to make sure the digits in the middle of the sub-images. The padsize was manually chosen for each image. Then I resize those images, convert them range, transpose and reshape them, and pass them through the network.

Image 1:

The result:

image1.jpg

Actual label:

2 3 4 5 6 7 8 9 10 1

Predicted label:

2 3 4 5 6 7 8 9 10 1

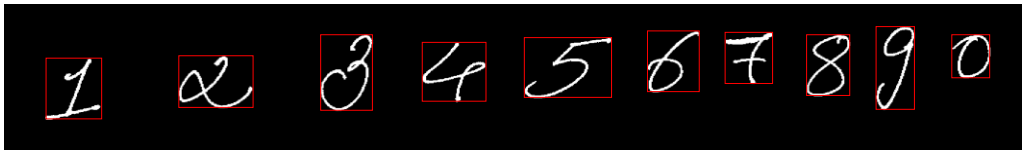
The total number of digits: 10

The number of correct predicted label: 10

The prediction accuracy for the current image: 1

Got 100% correct prediction for this image.

Image 2



The result:

image2.jpg

Actual label:

2 3 4 5 6 7 8 9 10 1

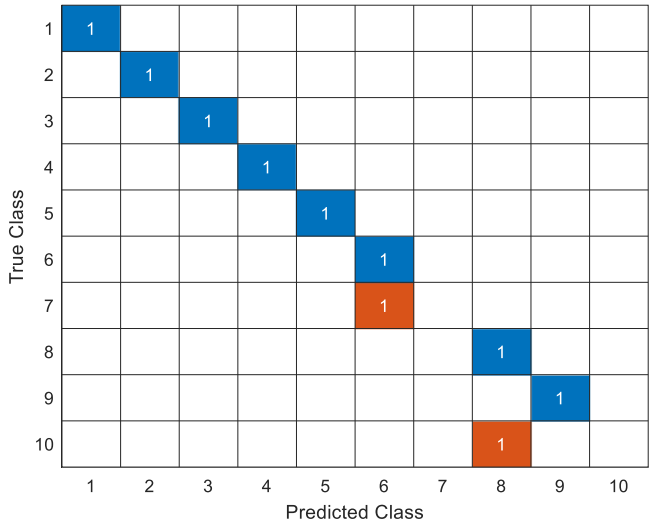
Predicted label:

2 3 4 5 6 6 8 9 8 1

The total number of digits: 10

The number of correct predicted label: 8

The prediction accuracy for the current image: 0.8000

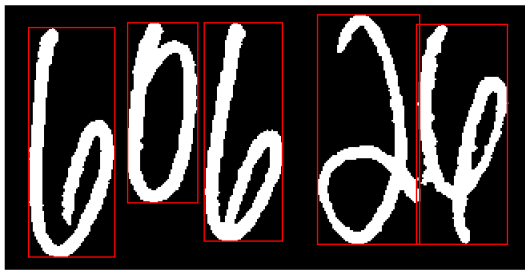


Got 80% correct prediction for this image.

(actual class:7, predicted class: 6) means the digit 6 is predicted as digit 5.
(actual class:10, predicted class: 8) means the digit 9 is predicted as digit 7.

It may because the digits in this image are little scrawl for this network.

Image 3



The result:

image3.png

Actual label:

7 1 7 3 5

Predicted label:

7 1 7 3 5

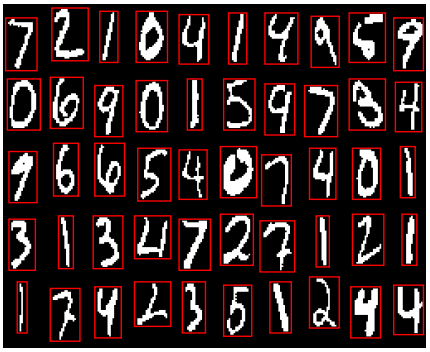
The total number of digits: 5

The number of correct predicted label: 5

The prediction accuracy for the current image: 1

Got 100% correct prediction for this image.

Image 4



The result:

image4.jpg

Actual label:

8	7	4	5	5	1	8	8	6	10
1	8	10	3	5	3	8	5	4	5
10	3	7	1	8	6	5	3	5	5
4	7	5	1	4	6	10	10	1	2
2	2	2	6	2	2	2	2	3	2

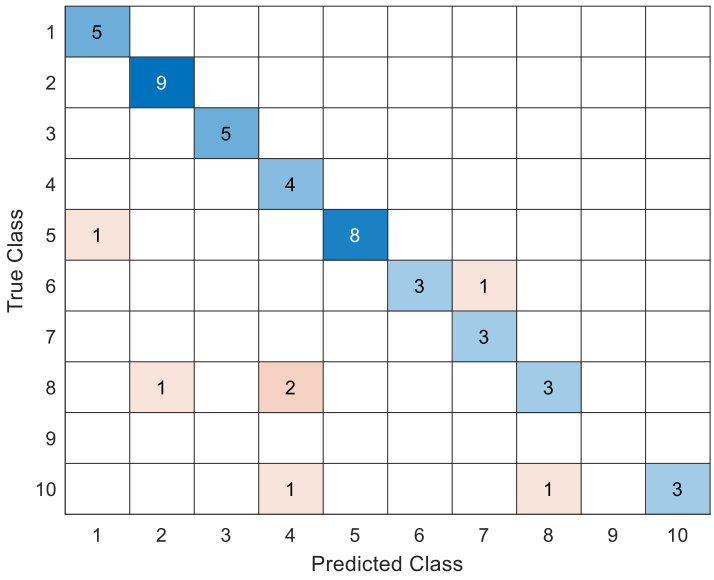
Predicted label:

8	7	4	1	5	1	8	8	7	10
1	4	10	3	5	3	4	5	4	5
8	3	7	1	2	6	5	3	5	5
4	7	5	1	4	6	10	4	1	2
2	2	2	6	2	2	2	2	3	2

The total number of digits: 50

The number of correct predicted label: 43

The prediction accuracy for the current image: 0.8600



Got 86% correct prediction for this image.

There are 2 times that the actual class is 8 but incorrectly predicted as class 4, which means digit 7 is predicted as digit 3. And it seems that digit 7 and digit 9 in this image are easy to be classified incorrectly as the other digits. It may because the digits in this image are little scrawl for this network.