

Digit Recognition with Convolutional Neural Networks

Course: CMPT 412

Name: Jad Alriyabi

Student #: 301360301

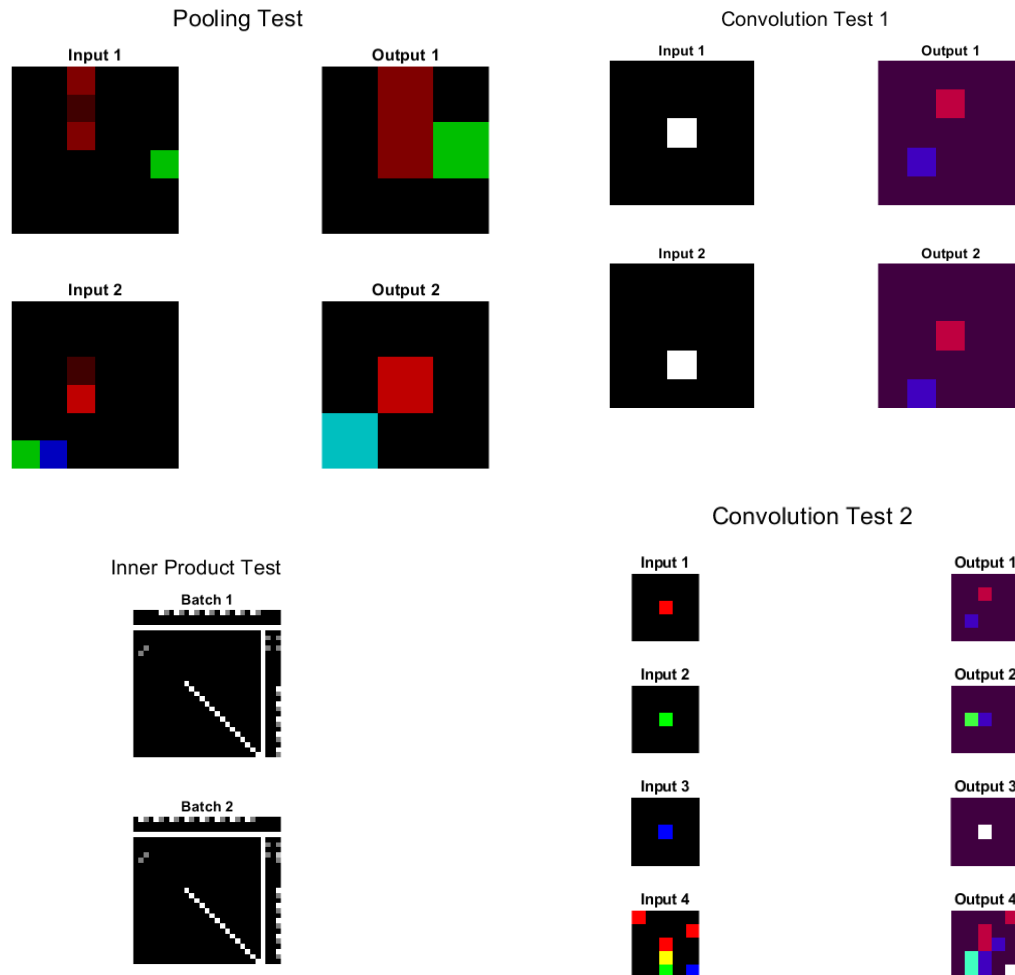
Computer ID: aalriyab@sfu.ca

The "Computer Vision" addon for MATLAB is required to run the program

Part 1: Forward Pass

The program includes an Inner Product component, which is achieved through the transposition of weights and their multiplication with each image in the batch, followed by the addition of a bias term. Additionally, the program utilizes a Max Pooling technique, in which a kernel passes through the pixels with a specified stride. The Convolution Layer can be implemented in two ways: aggregating the results from all channels to create an output channel for each filter number, or using the `im2col` function to create a vector of pixels. The latter approach, which involves the multiplication of these vectors with the weights, results in the convolution of the image, and is significantly faster due to the optimization of matrix multiplication in Matlab. Lastly, the program includes a Relu layer, which applies the element-wise operation $\max(\text{element}, 0)$ to each element in the input matrix.

Test components results:



Part 2: Back Propagation

Assuming that our output is the following:

$$h_i = f_i(w_i, h_{i-1})$$

The derivative of the loss function with respect to the activation function can be represented as follows:

$$\frac{\partial l}{\partial w_i} = \frac{\partial l}{\partial h_i} \frac{\partial h_i}{\partial w_i}$$

$$\frac{\partial l}{\partial h_{i-1}} = \frac{\partial l}{\partial h_i} \frac{\partial h_i}{\partial h_{i-1}}$$

For the forward pass of the Relu layer, the following can be applied:

$$\text{relu}(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$
$$\frac{dl}{dh_i} = \text{input.data if } x > 0, \text{ else } 0$$

As a result, the derivative of the relu function is 1 for values of x greater than 0 and 0 for values less than or equal to 0.

For the Inner Product Layer, the following can be applied

$$h_i = wx + b$$
$$\frac{dl}{dw} = \frac{dl}{dh_i} \frac{dh_i}{dw_i} = \text{output.diff} \times \text{input.data}$$
$$\frac{dl}{dh_{i-1}} = \frac{dl}{dh_i} \frac{dh_i}{dh_{i-1}} = \text{output.diff} \times \text{weights}$$
$$\frac{dl}{db} = \frac{dl}{dh_i} \frac{dh_i}{db} = \text{output.diff} \times 1$$

Part 3.1:

The model was trained for over 3000 iterations. The results for the final iterations of the model are shown below.

```
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
```

For a comprehensive report of the training set train_lenet, you may refer to result.txt in the main directory of the project.

Part 3.2: Test the network

Confusion Matrix:

True Class	0	1	2	3	4	5	6	7	8	9
	50	0	0	0	0	0	0	0	0	0
	0	65	0	1	0	0	0	0	1	0
	0	0	50	0	0	0	0	0	2	0
	0	0	0	44	0	0	0	0	0	0
	0	0	0	0	49	0	1	0	0	0
	0	0	0	0	0	44	1	0	0	0
	1	0	0	0	0	0	38	0	0	0
	0	1	1	0	0	0	0	54	0	0
	0	0	1	0	0	0	0	0	52	1
	0	0	0	1	0	1	0	0	0	41
Predicted Class										

The present study found that the neural network demonstrated difficulty in accurately identifying specific inputs, specifically numbers 6, 8 and 2. This difficulty was attributed to the circular nature of the representation of the numbers 6 and 8, which led to confusion when an additional line was present outside of the circle. Additionally, the number 2 was found to be particularly challenging for the network due to instances in which the number was represented with horizontal diagonal strokes, which likely contributed to the network's inability to correctly identify the number as a 2.

Part 3.3:

In order to evaluate the effectiveness of the neural network in a real-world scenario, a script (test_samples.m) was developed to test the network on a set of handwritten samples. These samples were collected and subsequently utilized to test the network's performance.



The outcome of executing the script test_samples yielded a score of 6/6, indicating a 100% success rate in the classification of the handwritten samples.

Part 4: Visualization

In this step, the original input image is shown below, along with the CONV layer on the left and the ReLU layer on the right.



Convolution layer:



Affected pixels in Relu Layer::



In order to visualize the functioning of the rectified linear unit (ReLU) layer, modifications were made to the standard method of visualization. The imshow function in Matlab typically displays negative pixel values as black, which would have resulted in an identical visual representation to the convolutional layer. To circumvent this issue, the values of all pixels that would not have been affected by the ReLU function (pixel values greater than 0) were set to 0 and pixel values less than 0 were set to 1. This resulted in the third figure, which illustrates the pixels affected by the ReLU function.

Upon examination of the convolutional layer, it is apparent that the network has learned to identify edges and the outline of the digits through the activation of specific filters. Some filters, however, are not activated for certain digits. For example, the third filter from the top and left appears to be completely black, and upon examination of the modified ReLU figure, it is clear that the output image is mostly black. It is important to note that black pixels in the modified ReLU figure represent untouched pixels and white pixels represent those that have been set to 0.

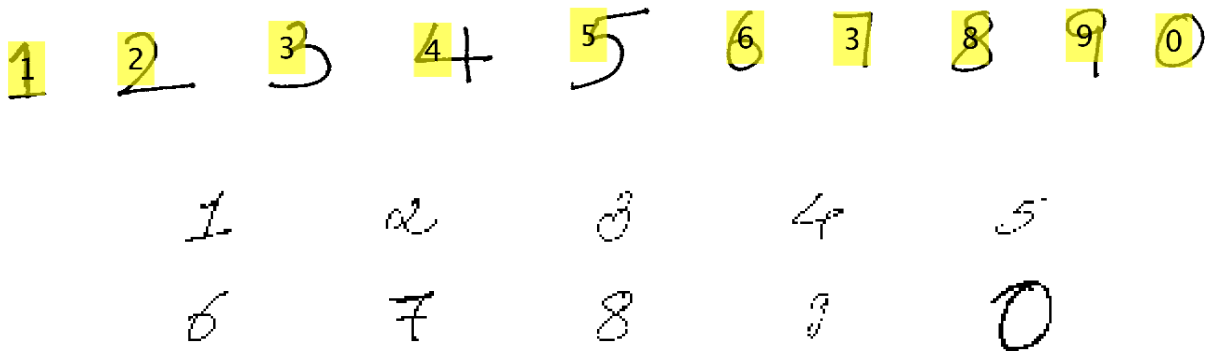
Part 5: Image Classification

Image 1:




In this particular example, the system is able to correctly divide each of the numbers into individual components. However, it incorrectly identifies a mysterious black square as its own separate component. Despite this error, the system is still able to correctly identify each divided number with the exception of the previously mentioned incorrect split.

Image 2:



For image 2, the network demonstrated an ability to accurately segment each number; however, it encountered difficulties in identification, resulting in the incorrect labeling of the 7 numbers. This difficulty is likely due to the thinly split nature of the components in the left image. Specifically, the digit 9 may be challenging to differentiate even for the human eye in such conditions. This issue could potentially be addressed by slightly dilating the pixels.

Image 3:

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

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

The network has shown to have a satisfactory level of accuracy in identifying the numbers, however, it has incorrectly segmented one number, which is reflected as an extraneous horizontal mark. This misidentification is highlighted in the left image. In total, out of the 50 numbers, 7 were mislabeled as depicted in the bottom image.