CMPT 822 ASSIGNMENT 5 Digit recognition with convolutional neural networks

Part 1: Forward Pass

1.1) Inner Product Layer

- The inner product layer (i.e.) forward propagation of the fully connected layer has been implemented with the layers forward function taking input, layer and param as argument.
- The input stores the shape and size of the input.
- The k,s,p has been calculated for the layers.
- The weights have been calculated for the params.

1.2) Pooling Layer

- Padding has been done to the input layer and stride has been taken for the input image.
- The kernel is a square kernel. Output has been calculated.

1.3) Convolution Layer: -

- A forward convolutional layer has been created with the parameters as input and layers as that of the original.
- The initial weights that are calculated has been forwarded to this convolutional layer.

1.2.2) ReLU: -

- The activation function used here is ReLU and the function of this is to convert all into 0's and 1's.
- It has been implemented in relu_forward function with the input and layer that has been calculated previously as parameters.

Part 2 Back propagation

- Using the chain rule, we are now implementing the back propagation.
- We need to first find the loss and update.

2.1) ReLU

- The activation function used here is ReLU and the function of this is to convert all into 0's and 1's.
- It has been implemented in relu_backward function with the input and layer that has been calculated previously as parameters.

2.2) Inner Product layer

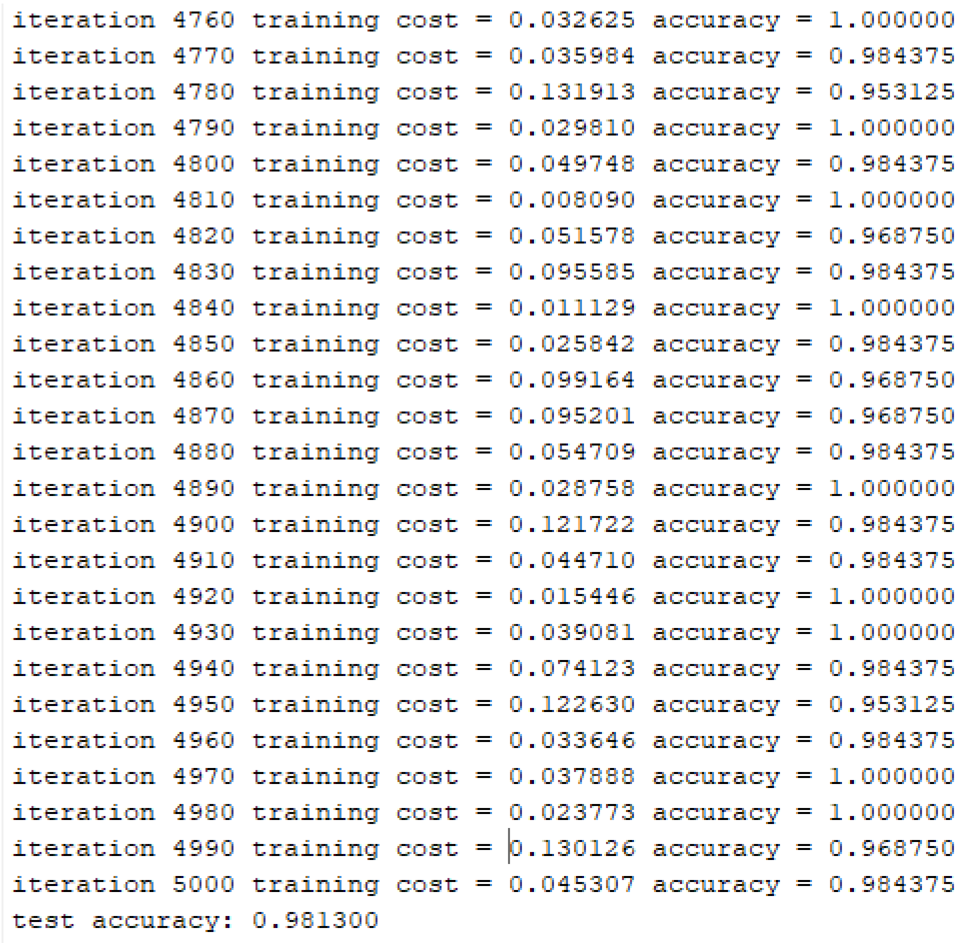
- The inner product layer (i.e.) backward propagation of the fully connected layer has been implemented with the layers forward function taking input, layer and param as argument.
- The input stores the shape and size of the input.
- The k, s, p has been calculated for the layers.

- The weights have been calculated for the params.

Part 3 Training: -

3.1 Training

- The network is updated and ran for 2000 + 3000 iterations.
- Testing interval is set for every 500 iterations and the accuracy is calculated.
- The test accuracy is provided as below: -



```
iteration 4760 training cost = 0.032625 accuracy = 1.000000
iteration 4770 training cost = 0.035984 accuracy = 0.984375
iteration 4780 training cost = 0.131913 accuracy = 0.953125
iteration 4790 training cost = 0.029810 accuracy = 1.000000
iteration 4800 training cost = 0.049748 accuracy = 0.984375
iteration 4810 training cost = 0.008090 accuracy = 1.000000
iteration 4820 training cost = 0.051578 accuracy = 0.968750
iteration 4830 training cost = 0.095585 accuracy = 0.984375
iteration 4840 training cost = 0.011129 accuracy = 1.000000
iteration 4850 training cost = 0.025842 accuracy = 0.984375
iteration 4860 training cost = 0.099164 accuracy = 0.968750
iteration 4870 training cost = 0.095201 accuracy = 0.968750
iteration 4880 training cost = 0.054709 accuracy = 0.984375
iteration 4890 training cost = 0.028758 accuracy = 1.000000
iteration 4900 training cost = 0.121722 accuracy = 0.984375
iteration 4910 training cost = 0.044710 accuracy = 0.984375
iteration 4920 training cost = 0.015446 accuracy = 1.000000
iteration 4930 training cost = 0.039081 accuracy = 1.000000
iteration 4940 training cost = 0.074123 accuracy = 0.984375
iteration 4950 training cost = 0.122630 accuracy = 0.953125
iteration 4960 training cost = 0.033646 accuracy = 0.984375
iteration 4970 training cost = 0.037888 accuracy = 1.000000
iteration 4980 training cost = 0.023773 accuracy = 1.000000
iteration 4990 training cost = 0.130126 accuracy = 0.968750
iteration 5000 training cost = 0.045307 accuracy = 0.984375
test accuracy: 0.981300
```

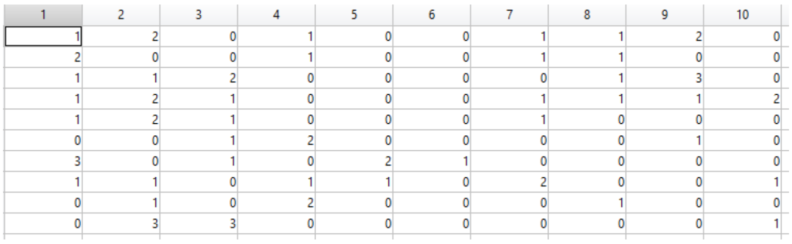
- The iteration ran for another 3000 more iterations and accuracy is provided as below: -

test_accuracy.txt

- Also provided the same result under /results folder.
- The refined network weights are updated and saved in lenet.mat file.

3.2 Test the network.

- The 10*10 confusion matrix (used the test to view the sample) has been calculated and provided as below: -
- Only a rough confusion matrix is created during this process.

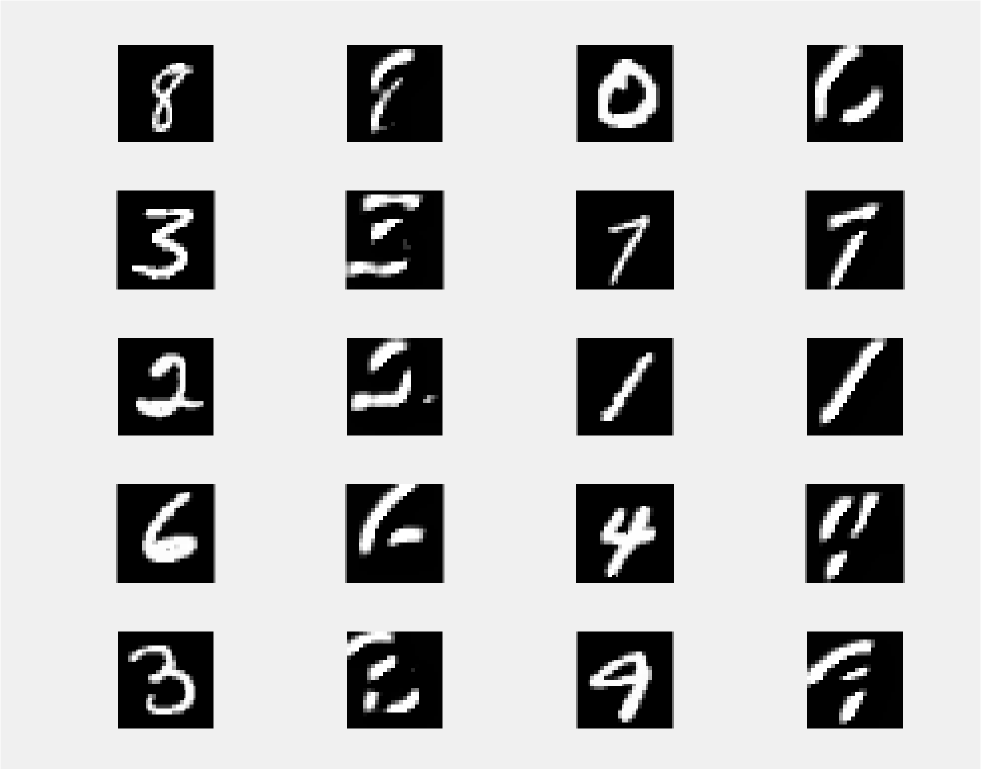


1	2	3	4	5	6	7	8	9	10
1	2	0	1	0	0	1	1	2	0
2	0	0	1	0	0	1	1	0	0
1	1	2	0	0	0	0	1	3	0
1	2	1	0	0	0	1	1	1	2
1	2	1	0	0	0	1	0	0	0
0	0	1	2	0	0	0	0	1	0
3	0	1	0	2	1	0	0	0	0
1	1	0	1	1	0	2	0	0	1
0	1	0	2	0	0	0	1	0	0
0	3	3	0	0	0	0	0	0	1

Part 4 Visualization

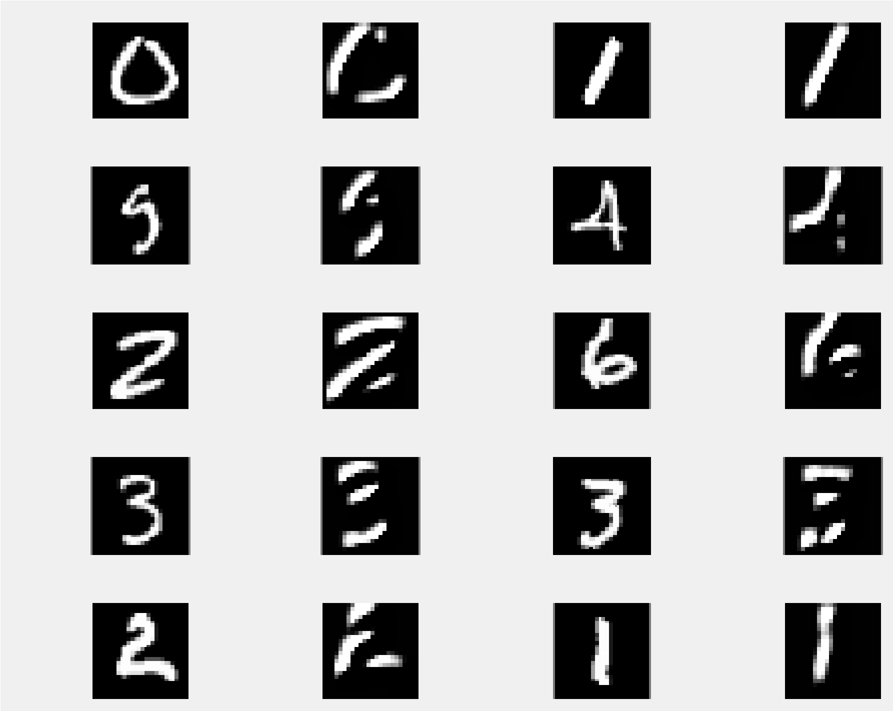
4.1 vis_data.m

- All the images are loaded and visualized for both CONV and ReLU layers respectively.
- Have visualized the images in 5*4 for better viewing perspective.



4.2 Compare the feature maps to the original image

- While comparing the original image to the convolution, we could see that during the first convolution it takes the original image, the weights are calculated and first image received is a convoluted images and moves forward to the next layer and so on.
- While comparing the original image to the relu, we could see that during the first relu it tries to convert into the 0's and 1's format.

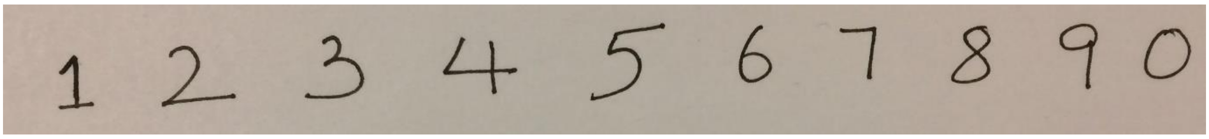


Extra credit

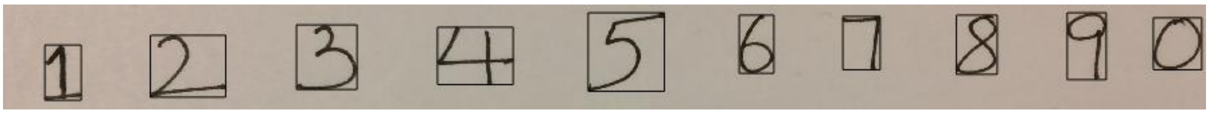
Part 5 Image Classification

- Task has been performed for Optical character recognition for the images provided in the images folder.
- Bounding box has been predicted for the numbers in the image and the predicted bounding box will depict as below: -

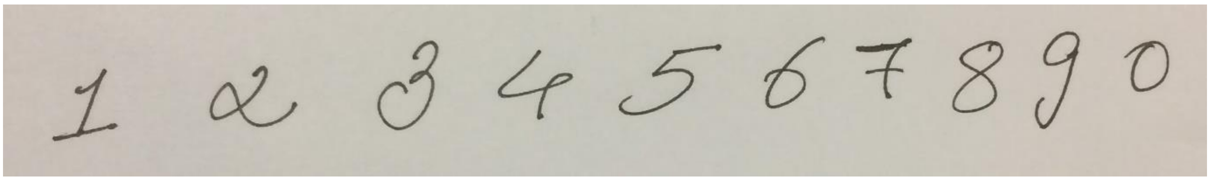
Sample image 1



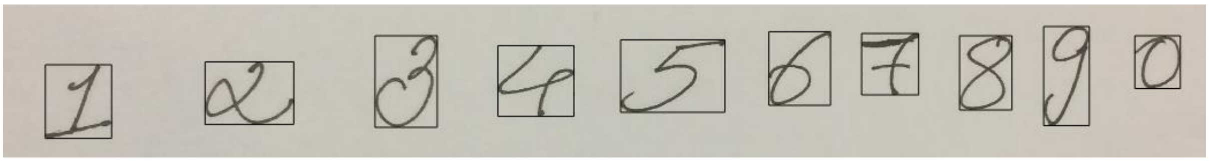
Predicted bounding box is perfect for the sample image 1



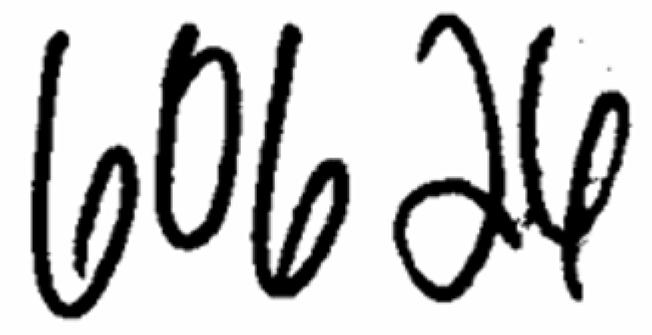
Sample image 2



Predicted bounding box is perfect for the sample image 2

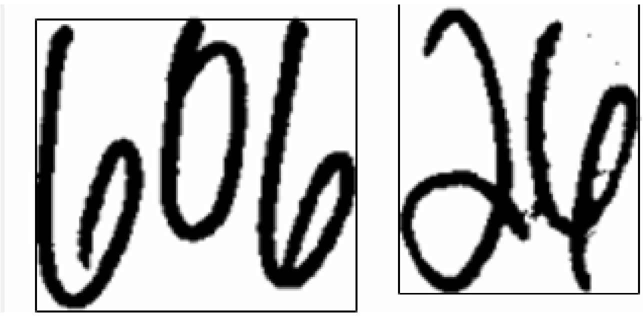


Sample image 3

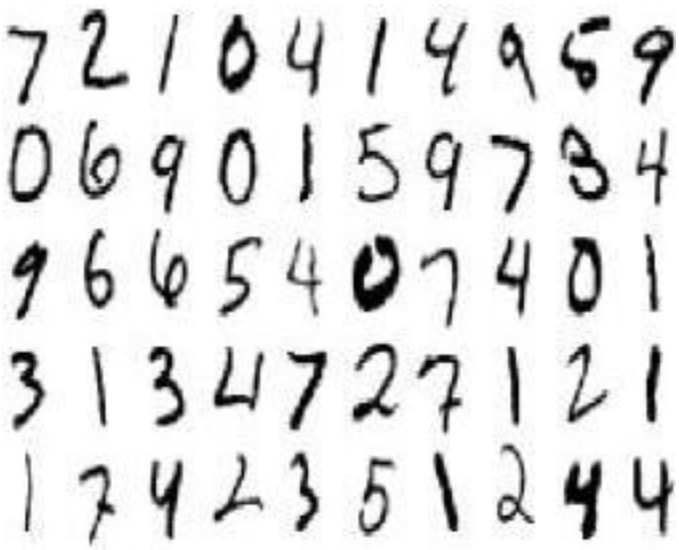


Predicted bounding box for the sample image 3.

There is a redundancy in predicting the bounding box for each of the numbers.

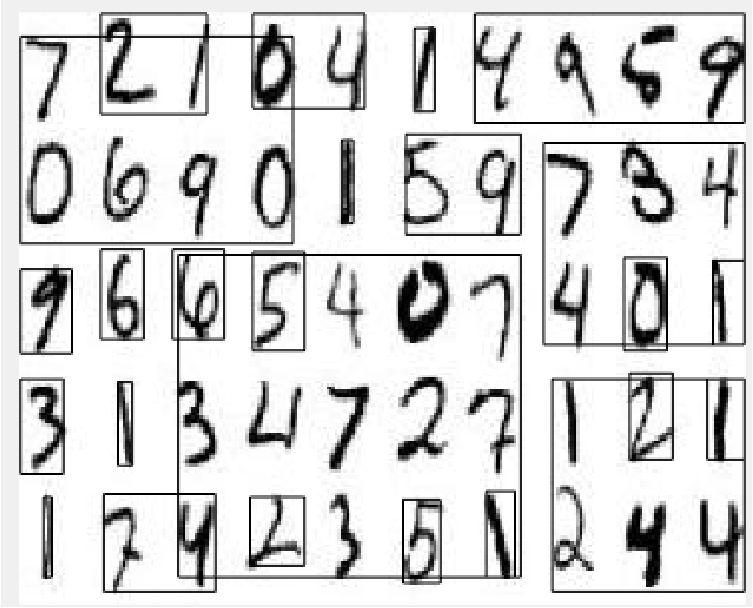


Sample image 4



Predicted bounding box for the sample image 4.

There is a redundancy in predicting the bounding when the images are very close, and the numbers overlap one with the other.


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