Computational Neuroscience Assignment-3
Convolutional Neural Network
BT6270

Sweta Kumari
BT17D019

[1] Download the MNIST dataset from the link: http://yann.lecun.com/exdb/mnist/ to the distributed folder where code is present. Report the dataset details.
Ans.

MNIST is handwritten digits dataset which is a large subset of NIST's Special Database 3 and Special Database 1. NIST's Special Database 3 was collected from highly educated persons (Census Bureau employees) and NIST's Special Database 1 was collected from high-school students. Therefore, NIST's SD3 is much cleaner than NIST's SD1. This is the reason that NIST's SD3 is much easier to recognize than NIST's SD1.

The training dataset in MNIST is composed of 30,000 patterns from NIST's SD3 (much cleaner) and 30,000 patterns from NIST's SD1(less cleaner).

The testing dataset in MNIST is composed of 5,000 patterns from NIST's SD3 (much cleaner) and 5,000 patterns from NIST's SD1(less cleaner).

The digit in MNIST dataset is normalized in size and centred in a fixed-size image. Because of this the error rate improves and which is proven by using some classification methods such as Support Vector Machine (SVM) and K-nearest neighbours.

Detail in MNIST dataset:

- MNIST training dataset contains 60,000 (But in CNN code, only 10,000 is used)
- MNIST testing dataset contains 10,000
- Image dimension : 28 x 28
- No. of channels: 1 (black and white image)
- No. of classes: 10 (handwritten digits: 0,1,2,3,4,5,6,7,8,9)

[2] Distributed code has one set of 'conv-pool' layer followed by fully connected layer.

(i). Run the code ('Traincnn') and report the performance and training time (without any changes in the code).

Ans.

Training time: 106.09 seconds
Accuracy is: 94.270000%

Visualize the weights and report the weight patterns.

Ans.

I have analysed the weights pattern for all digits (0-9) but here some interesting patterns has been observed and clearly can be seen in digits (3, 6, 7, 8,). I have shown the pictures of weights and their corresponding feature maps for the digits (3, 6, 7, 8) to interpret the pattern of weights. This gives the understanding of how the weights are classifying the different digits after detecting some horizontal lines, vertical lines, edges and small curves based on the mapped features.

S No.	Weight	Close Patterns
1	Weight 1	Background detector
2	Weight 2	Unclear
3	Weight 3	Background detector
4	Weight 4	Unclear
5	Weight 5	Background detector
6	Weight 6	Edge detector(Unclear)
7	Weight 7	Background detector
8	Weight 8	Line detector
9	Weight 9	Background detector
10	Weight 10	Background detector
11	Weight 11	Sloppy line detector
12	Weight 12	Background detector
13	Weight 13	Line detector
14	Weight 14	Background detector
15	Weight 15	Background detector
16	Weight 16	Line detector
17	Weight 17	Vertical line detector
18	Weight 18	Background detector
19	Weight 19	Curve detector
20	Weight 20	Horizontal detector

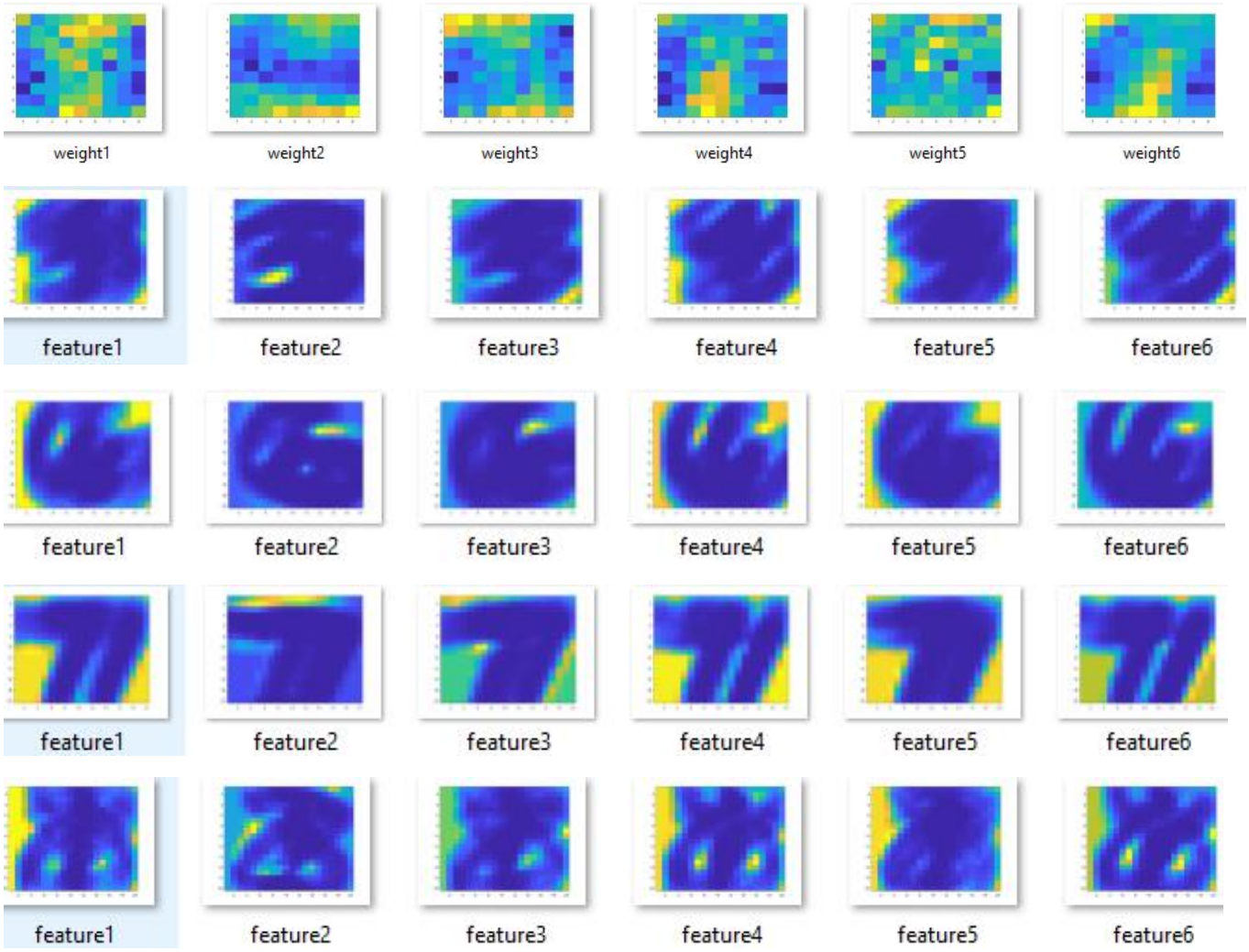
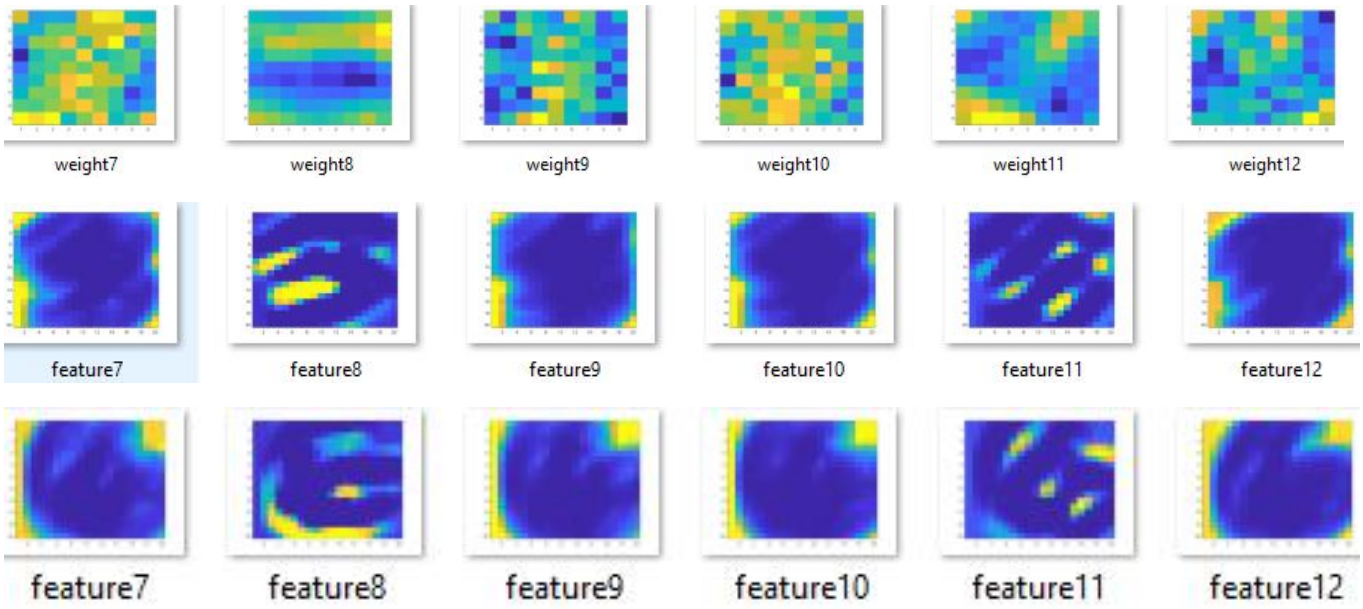


Fig: weights from 1-6 and their corresponing feature maps for digits 3, 6, 7 and 8.



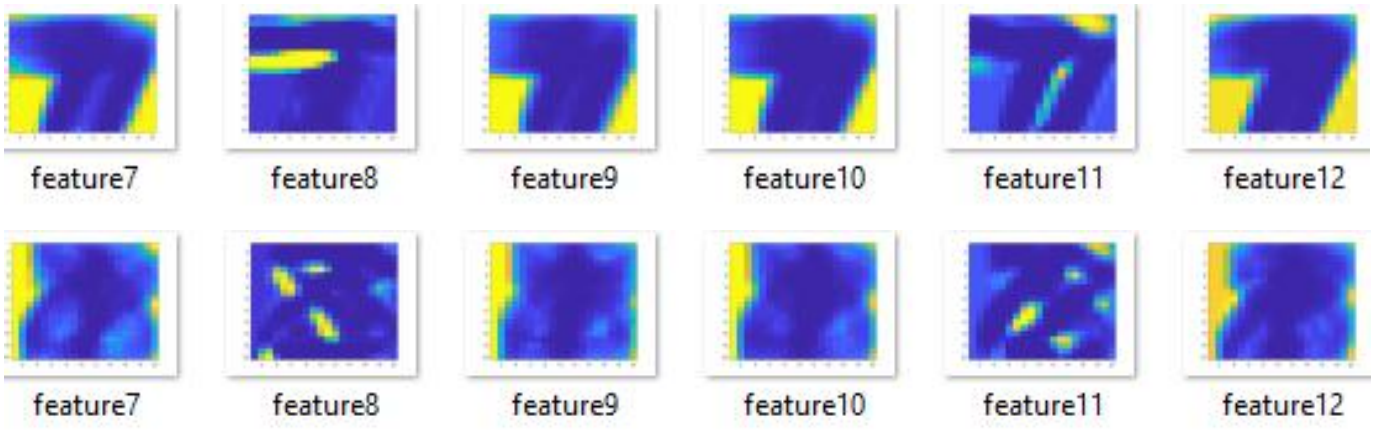


Fig: weights from 7-12 and their corresponing feature maps for digits 3, 6, 7 and 8.

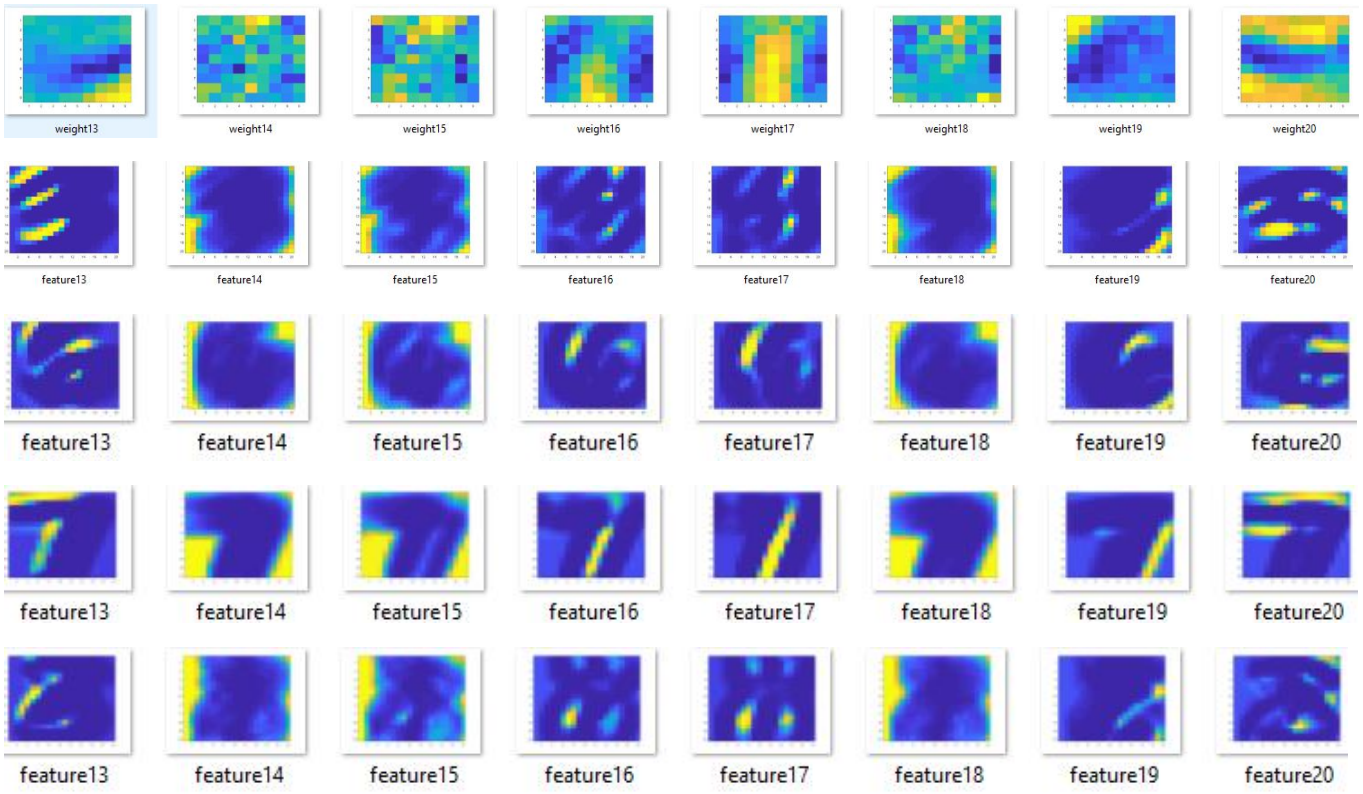


Fig: weights from 13-20 and their corresponing feature maps for digits 3, 6, 7 and 8.

(ii). Visualize the feature maps and identify which of them respond to edges or lines.

Ans.

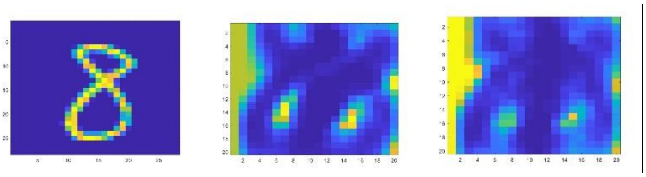
This table is showing the feature maps which are responding to edges or lines. But for some of the features it is very difficult to say that it is responding to completely edges or lines. After observation from the above plots of pictures of feature maps with their corresponding weights it might be the case that it is not

necessary for every feature map that it will respond to edges or lines, it can respond to some small grooves or curves. Therefore the below table is the approximate identified patterns of the above feature maps.

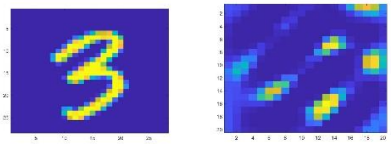
S No.	Weight	Close Patterns
1	feature 1	Edge
2	feature 2	Edge(Unclear)
3	feature 3	Edge
4	feature 4	Edge
5	feature 5	Edge
6	feature 6	Edge(Unclear)
7	feature 7	Edge
8	feature 8	Line
9	feature 9	Edge
10	feature 10	Edge
11	feature 11	Line
12	feature 12	Edge
13	feature 13	Line
14	feature 14	Edge
15	feature 15	Edge
16	feature 16	Line
17	feature 17	Line
18	feature 18	Edge
19	feature 19	Line
20	feature 20	Line

Observation:

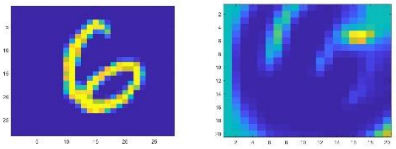
The feature maps seems to have some interesting pattern. It can be seen that for multiple weight patterns the corresponding feature map is making two copies of the images and placing those in nearby and extracting information from the activity shown in the resulting grooves.



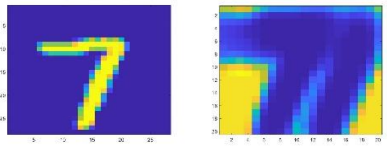
Input **feature-6** **feature-1**



Input **feature-11**



Input **feature-6**



Input **feature-1**

iii. Restrict your training set to first 1,000 images. Make changes to the following, one after other and report the trained network performance in form of a table. (3 variations each)

Ans. Set the training data points from 10, 000 to 1,000 images and checked the accuracy with 1,000 test data.

a. No. of epochs ('e' in Traincnn)

Note:-As the number of epochs are increasing, the accuracy and training time also increasing.

Reason:-It might be the case that the optimal minima of the total loss was not reached and the weights are getting updated with increase in the number of epochs. Once optimal minima is reached the accuracy of model will stop to increase by increasing the number of epochs.

Epochs	Accuracy (%)	Training Time (sec.)
2	79.90	9.87
3	**83.50**	**9.97**
4	84.00	10.11
5	84.40	10.13

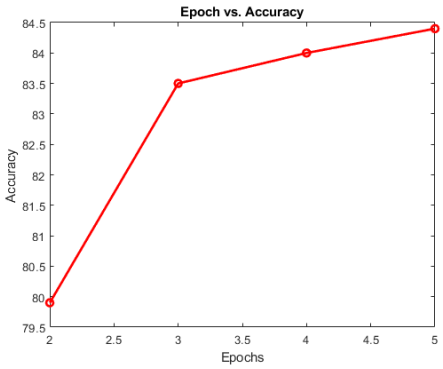


Fig: Plot of epoch vs. accuracy

b. No. of feature maps in the convolution layer ('numFilters' in Traincnn)

Note:-If we increase the number of filters the performance of the training network is improves till a certain extent after that the performance starts to reduce.

Reason:-Almost all filters detect the same feature. So the filters cannot extract all sufficient features to classify the data points into their actual class.

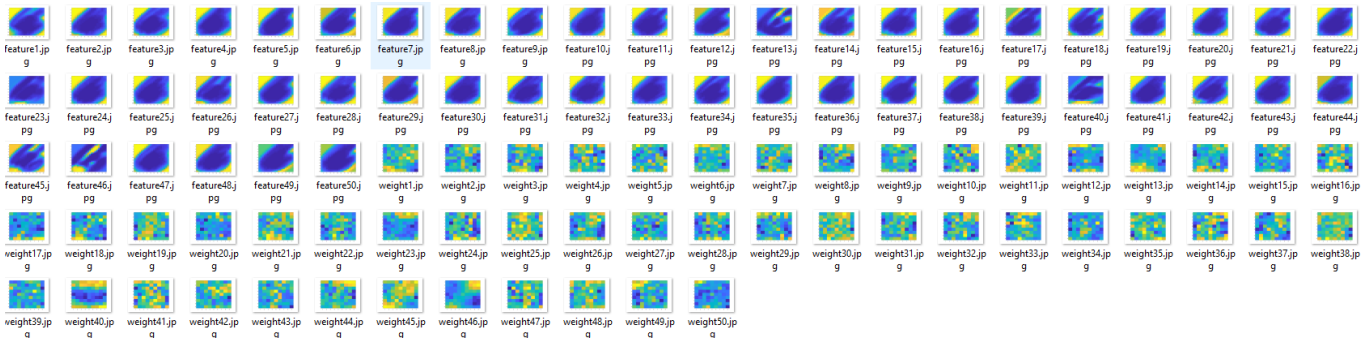


Fig: Screenshot of all the features with their corresponding weights (numFilters = 50 and input image = 6).

Training time increases due to increase in number of parameters.

numFilters	Accuracy (%)	Training Time (sec.)
10	81.80	5.25
20	**83.50**	**9.96**
30	81.10	14.78
40	80.20	19.57
50	78.70	25.69
60	78.30	28.92

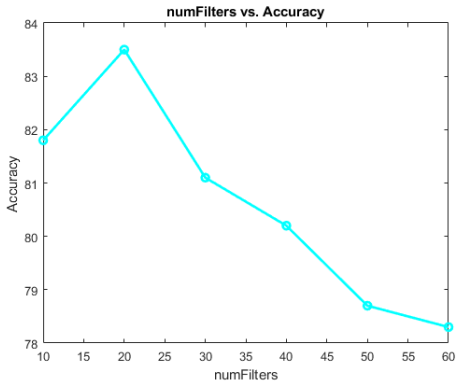


Fig: Plot of numFilters vs. accuracy

c. Convolution layer filter dimension ('FilterDim' in Traincnn).

Note:-If we increase the number of filter size the performance of the training network is improving but after a certain point it starts to decrease.

Reason:-when the filter is sufficiently able to scan the complete images then it extracts the important feature and improve the accuracy but after a certain extent when the filter is not sufficiently able to move and scan on the complete input image due to very large and close filter dimension to the input image dimension, then it skips the some portion of the image and cannot extract the features from the complete image.

Training time decreases because the filter dimension goes to very close to the input image dimension if we increase it. It scans the image with a very less number of times by striding the filter.

filterDim	Accuracy (%)	Training Time (sec.)
5	73.80	13.50
9	**81.20**	**9.92**
13	88.10	7.28
17	87.90	4.25
21	86.9	2.42
25	85.20	1.12
27	83.90	0.77

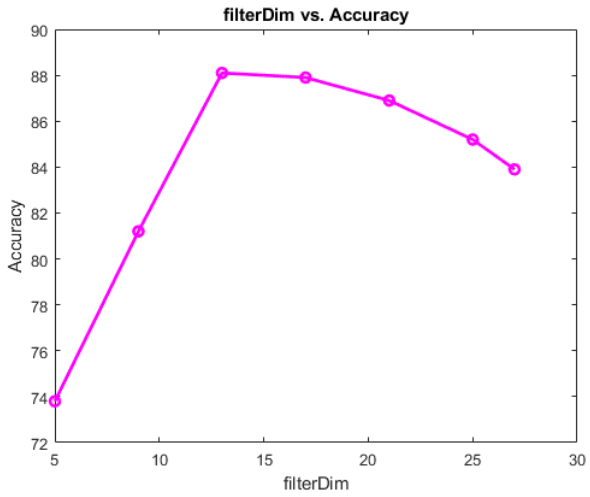


Fig: Plot of filterDim vs. accuracy