
Deep Learning and its Applications

CS671

Course Instructor : Dr. Aditya Nigam

Assignment 2 : Question 1

CNN to Classify MNIST and Line Dataset

Group 13

Aman Khandelwal	B16007
Amrendra Singh	B16010
Bharat Lodhi	B16015

<https://github.com/ashking13th/deepLearning>

https://students.iitmandi.ac.in/~b16010/CS671_grp_13/

1 Problem Statement

The goal of this assignment is to learn the basic principles of designing deep convolutions neural networks for image classification.

Making a CNN model from scratch to classify the images of the line dataset into the respective 96 classes and MNIST dataset into 10 classes.

1. Variations

We tried multiple variations of the models. We started with the one given in the problem sheet and then tried the suggested variations given there. Then we went on to try various variation of our own.

1 MNIST Dataset

Model 1

1. 7x7 Convolutional Layer with 36 filters and stride of 1.
2. ReLU Activation Layer.
3. 7x7 Convolutional Layer with 36 filters and stride of 1.
4. ReLU Activation Layer.
5. 7x7 Convolutional Layer with 36 filters and stride of 1.
6. ReLU Activation Layer.
7. fully connected layer with 1024 output units.
8. ReLU Activation Layer.

Achieved Accuracy: 99.26%

Class	1	2	3	4	5	6	7	8	9	10
Class 1	977	0	0	0	0	0	1	1	1	0
Class 2	0	1128	2	1	1	0	1	0	2	0
Class 3	1	1	1022	1	1	0	0	4	2	0
Class 4	0	0	1	1007	0	1	0	0	1	0
Class 5	0	0	0	0	973	0	0	0	0	9
Class 6	1	0	0	4	0	885	1	0	1	0
Class 7	5	2	0	0	2	1	946	0	2	0
Class 8	0	1	1	0	0	0	0	1023	1	2
Class 9	0	0	2	1	0	0	0	1	969	1
Class 10	0	0	0	1	6	3	0	2	1	996

Table 1.1. Confusion Matrix for MNIST Dataset

Class	precision	recall	F1-score
0	0.99	1.00	0.99
1	1.00	0.99	1.00
2	0.99	0.99	0.99
3	0.99	1.00	0.99
4	0.99	0.99	0.99
5	0.99	0.99	0.99
6	1.00	0.99	0.99
7	0.99	1.00	0.99
8	0.99	0.99	0.99
9	0.99	0.99	0.99

Table 1..2. Classification Summary

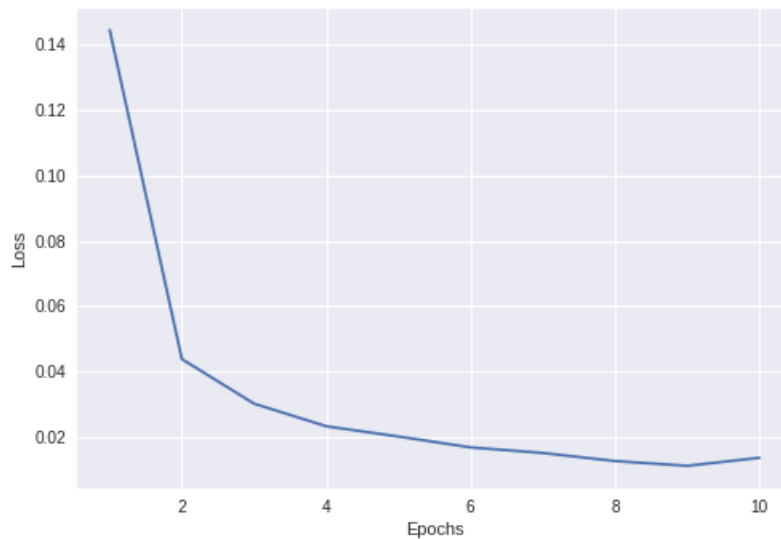


Figure 1..1. Loss vs Epoch Learning Curve for MNIST Dataset

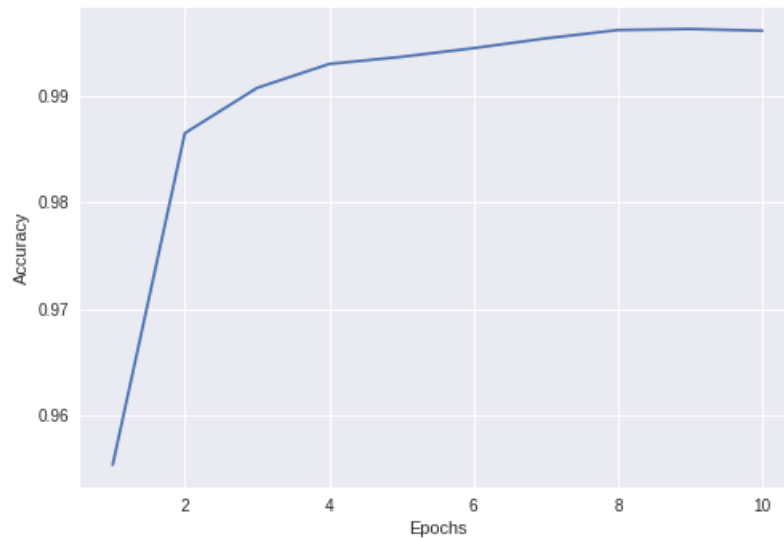


Figure 1..2. Training data Acc vs Epoch Learning Curve for MNIST Dataset

Model 2

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Achieved Accuracy: 97.34%

Model 3

1. 3x3 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Achieved Accuracy: 68.37%

Model 4

1. 3x3 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. 2 fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Achieved Accuracy: 67.54%

Model 5

1. 7x7 Convolutional Layer with 64 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Achieved Accuracy: 97.67%

Model 6

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. 7x7 Convolutional Layer with 32 filters and stride of 1.
3. ReLU Activation Layer.
4. Batch Normalization Layer
5. 2x2 Max Pooling layer with a stride of 2
6. fully connected layer with 1024 output units.
7. ReLU Activation Layer.

Achieved Accuracy: 98.54%

Model 7

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. 7x7 Convolutional Layer with 64 filters and stride of 1.
3. ReLU Activation Layer.
4. Batch Normalization Layer
5. 2x2 Max Pooling layer with a stride of 2
6. fully connected layer with 1024 output units.
7. ReLU Activation Layer.

Achieved Accuracy: 98.54%

2 Line Dataset

Model 1

1. 7x7 Convolutional layers with 36 filters an stride of 1
2. relu activation
3. 2x2 Max Pooling layer with a stride of 2
4. Batch Normalization
5. 7x7 Convolutional layers with 36 filters an stride of 1
6. Dense layer with 1024 outputs
7. Relu activation

Achieved Accuracy: 99.97%

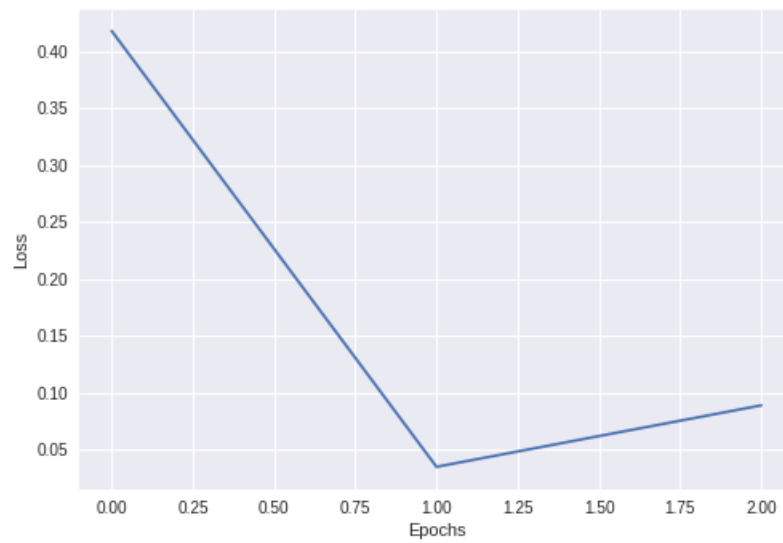


Figure 1..3. Loss vs Epoch Learning Curve for Line Dataset

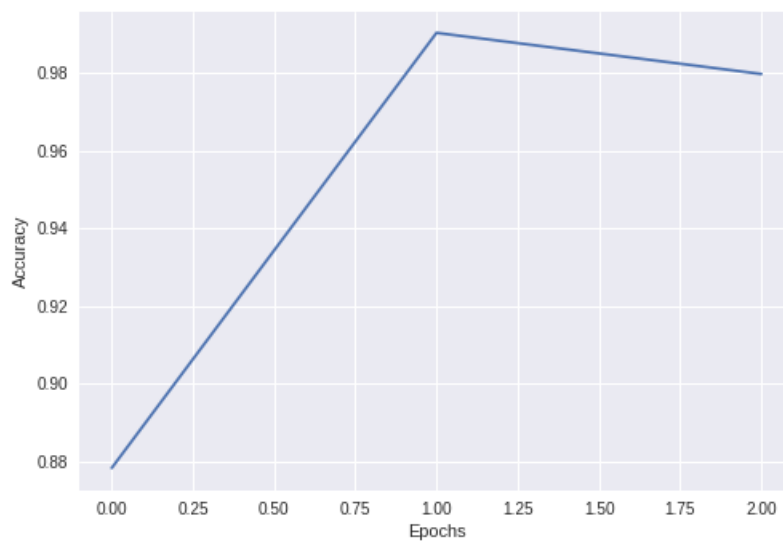


Figure 1..4. Training data Acc vs Epoch Learning Curve for Line Dataset

Model 2

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Accuracy Achieved: 99.5%

Model 3

1. 7x7 Convolutional layers with 36 filters an stride of 1
2. relu activation
3. 7x7 Convolutional layers with 36 filters an stride of 1
4. relu activation
5. 2x2 Max Pooling layer with a stride of 2
6. Batch Normalization
7. 7x7 Convolutional layers with 36 filters an stride of 1
8. Dense layer with 1024 outputs
9. Relu activtion

Accuracy Achieved: 95.95%

3 Inferences

- Increasing number of filters was only marginally helpful.
- Increasing the number of convolution layers increases accuracy.
- Adding more fully connected layers did not increase accuracy by any significant number.
- Accuracy is also affected by the number of training epochs. Although train accuracies increased with the increase in the number of epochs, but the test accuracies started reducing after a certain point. This showcased that too many iterations of training over the same data leads to over-fitting. The network starts memorizing the train data instead of learning from it. The optimum number of epochs was found to be 3.

Bibliography

- [1] Stack Overflow
<https://stackoverflow.com>
- [2] Tensorflow Tutorials
<https://www.tensorflow.org/tutorials>
- [3] Data Science Blog
[https://medium.com/datadriveninvestor/my-take-at-the-mnist-dataset\
-97304dff2057](https://medium.com/datadriveninvestor/my-take-at-the-mnist-dataset\-97304dff2057)
- [4] Analytics Vidha
[https://www.analyticsvidhya.com/blog/2016/10/
an-introduction-to-implementing-neural-networks-using-tensorflow/](https://www.analyticsvidhya.com/blog/2016/10/an-introduction-to-implementing-neural-networks-using-tensorflow/)