CS671-Deep Learning and Applications

Assignment 1 - Task 3 Report

Layer API - A Simple Neural Network

March 2019

Vishal Anand B16040
Aman Jain B16044
Yash Agrawal B16120

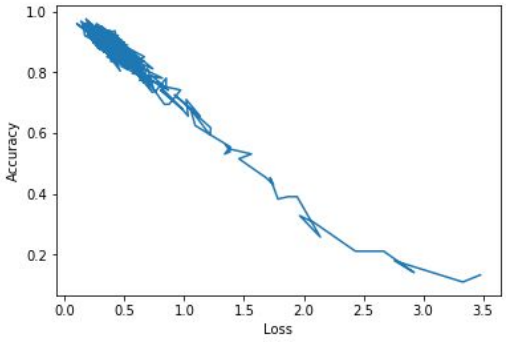
Task: To build a simple fully connected network to classify the MNIST dataset and the dataset in part 1.

On MNIST Dataset

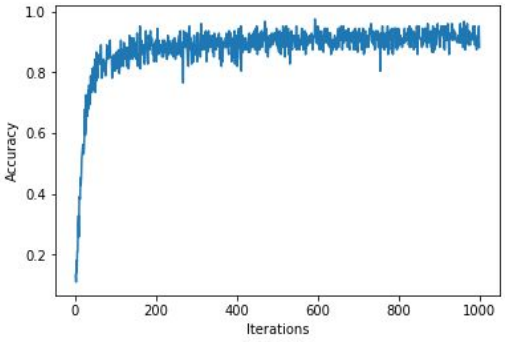
The code for the model can be found here:
https://github.com/Vishal1541/DeepLearning/tree/master/Assingment1/Task3
filename = task3_mnist.ipynb

Variation 1
The result on training and testing the mnist data with the following parameter are:

- input layer (28x28 pixels)
- 512 - 1st hidden layer
- 256 - 2nd hidden layer
- 128 - 3rd hidden layer
- 10 - output layer (0-9 digits)
- learning_rate = 0.0001
- n_iterations = 1000
- batch_size = 128
- The weights are randomly initialized using the gaussian distribution with stand. dev = 0.1
- Biases were initialised with 0.1
- Cost function: SoftMax Cross Entropy



Loss vs Accuracy graph



Iterations vs Accuracy graph

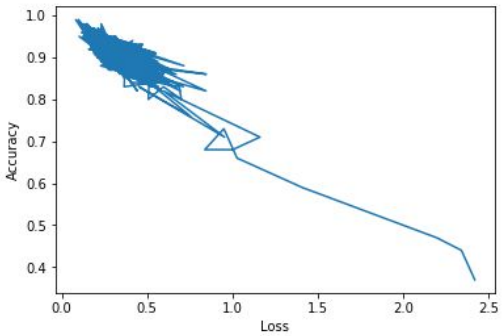
The model was tested on 10,000 images and the following results were recorded:

- Accuracy = 0.9183
- F Score = 0.9951664427098585
- Confusion Matrix
  957 0 1 1 0 4 10 2 4 1
  0 1105 2 2 0 1 4 2 19 0
  17 10 909 12 7 0 12 14 45 6
  4 2 24 905 0 24 2 15 29 5
  1 4 4 2 899 1 14 4 13 40
  12 3 4 32 8 755 17 12 42 7
  12 3 6 1 7 7 913 3 6 0
  3 7 19 8 5 1 0 957 4 24
  6 8 8 20 6 21 11 11 880 3
  9 6 1 10 21 8 0 36 15 903

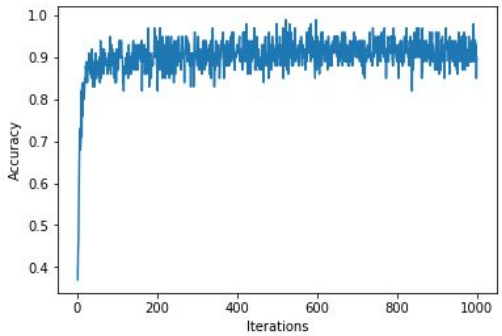
Variation 2

The result on training and testing the mnist data with the following parameter are:

- input layer (28x28 pixels)
- 500 - 1st hidden layer
- 250 - 2nd hidden layer
- 100 - 3rd hidden layer
- 10 - output layer (0-9 digits)
- learning_rate = 0.001
- n_iterations = 1000
- batch_size = 100
- The weights are randomly initialized using the gaussian distribution with stand. dev = 0.1
- Biases were initialised with 0.1
- Cost function: SoftMax Cross Entropy



Loss vs Accuracy graph



Iterations vs Accuracy graph

The model was tested on 10,000 images and the following results were recorded:

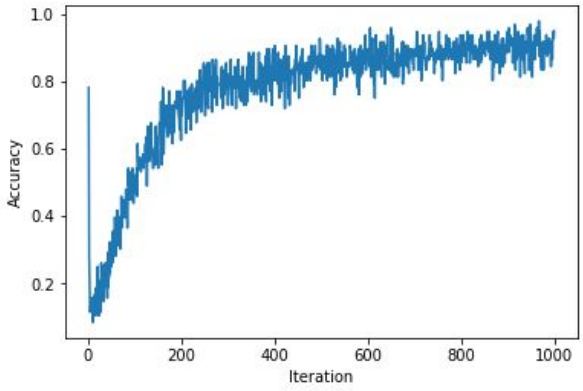
- Accuracy = 0.9158
- F Score = 0.9953405619333099
- Confusion Matrix
  944 0 4 3 1 12 10 4 1 1
  0 1088 7 2 1 4 3 2 28 0
  6 4 949 6 6 5 8 9 36 3
  1 1 33 866 1 59 2 15 22 10
  1 3 13 1 891 1 13 3 18 38
  7 3 10 21 7 790 11 13 25 5
  11 2 20 1 3 15 904 1 1 0
  1 5 17 7 7 2 0 956 3 30
  10 3 10 22 7 36 5 15 861 5
  11 4 1 12 24 8 0 33 7 909

On image dataset from part1

Variation 1

The result on training and testing the data with the following parameter are:

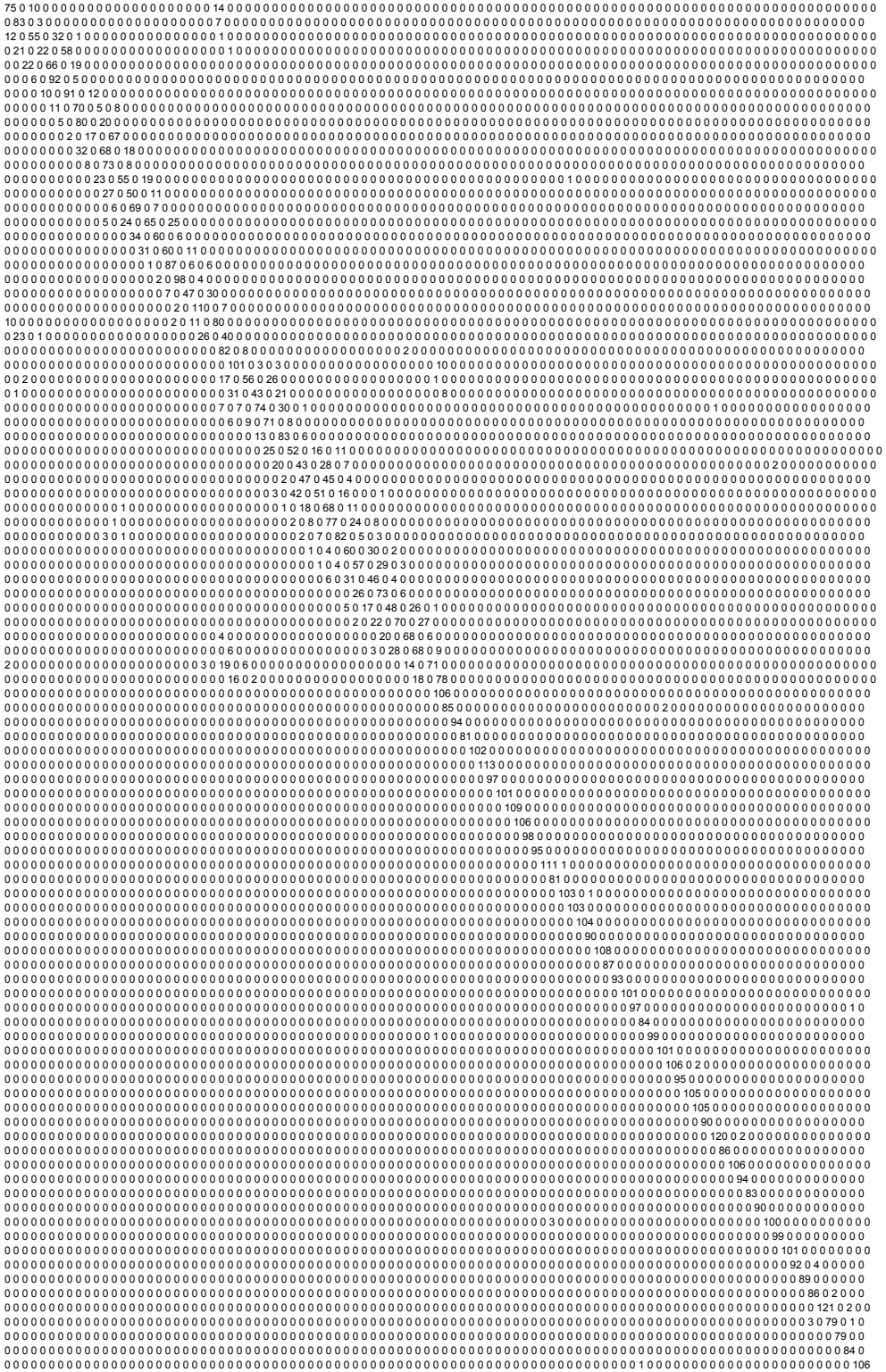
- input layer (28x28x3 pixels)
- 800 - 1st hidden layer
- 96 - output layer
- learning_rate = 0.01
- n_iterations = 1000
- batch_size = 96
- The weights are randomly initialized using the gaussian distribution with stand. dev = 0.1
- Biases were initialised with 0.2
- Activation Function: ReLU
- Cost function: SoftMax Cross Entropy



Iterations vs Accuracy graph

The model was tested on 9600 images and the following results were recorded:

- Accuracy = 0.8234375
- F Score = 0.9975274904463891
- Confusion Matrix (96 x 96)



Inferences

The training phase of the model greatly depends on the hyperparameters we chose, eg. the initialization of the weights and biases and the activation functions determines to great extend how slow or fast our model will converge. How good the model trains also depends on how dense our network is and how many neurons in each layers are. If we increase this too much, then it'd lead to overfitting. And if we decrease it too much, it'd lead to underfitting. Therefore, we should train the model on different networks and choose the one which gives high accuracy on training and testing data.