NEURAL NETWORK IMPLEMENTATION WITH FASHION MNIST DATASET

ROHITH KUMAR GADALAY
UNIVERSITY AT BUFFALO
rohithku@buffalo.edu

Abstract

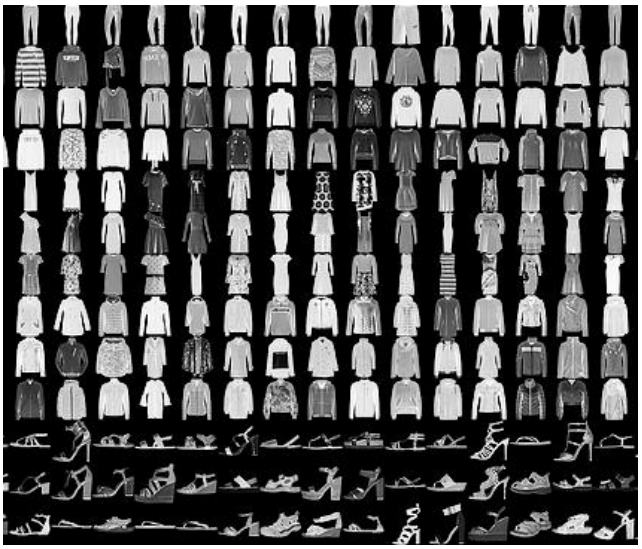
In this paper, the primary task is to implement neural network in three different mechanisms. The first one is neural network implementation from scratch with one hidden layer, multi-layer neural network implementation with an open source neural network library, KERAS and convolutional neural network implementation with an open source neural network library, KERAS. The dataset used for this implementation is Fashion MNIST dataset.

1 INTRODUCTION

A neural network is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output usually in another form Neural networks are just one of many tools and approaches used in machine learning algorithms. Neural network is a highly efficient and robust algorithm used widely today. It emulates the human brain in terms of how we learn, and process data. Neural networks can represent complex non-linear functions and have applications in complex learning problems such as computer vision. They are flexible and can also be configured to give better results depending on the problem requirements.

2 DATASET

For training and testing of our classifiers, we will use the Fashion-MNIST dataset. The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.



Each training and test example is assigned to one of the labels as shown below table.

1	T-shirt/top
2	Trouser
3	Pullover
4	Dress
5	Coat
6	Sandal
7	Shirt
8	Sneaker
9	Bag
10	Ankle Boot

Labels for Fashion-MNIST dataset

3 PRE-PROCESSING

3.1 Neural Network implementation with one hidden layer

In this experiment we need to follow some steps in order to get the required solution:
1.Loading the dataset and mapping the variables appropriately to train and test values.
2.Normalizing the data and reshaping it.
3.Initialize the hidden nodes, epochs and learning rate.

3.2 Multi-Layer Neural Network implementation with KERAS

In this experiment we need to follow some steps in order to get the required solution:
1. Loading the dataset and mapping the variables appropriately to train and test values.
2.Normalizing the data and reshaping it.

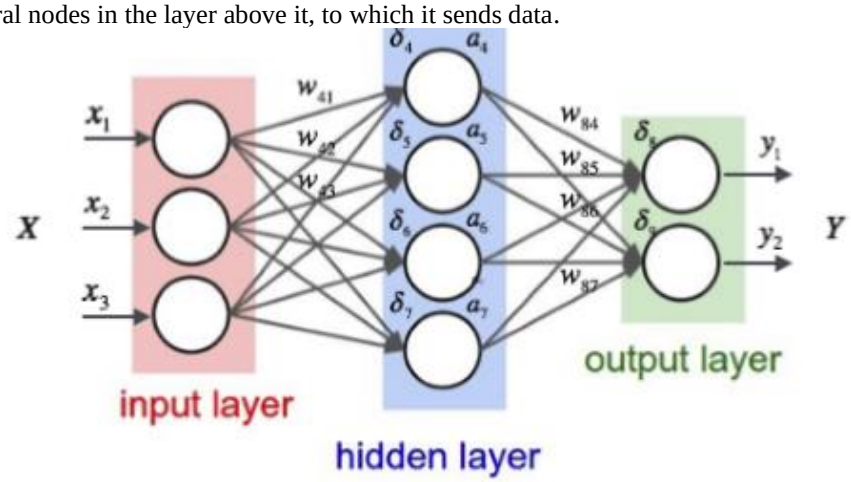
3.Initializing the number of epochs.

3.3 Convolutional Neural Network implementation with KERAS

In this experiment we need to follow some steps in order to get the required solution:
1. Loading the dataset and mapping the variables appropriately to train and test values.
2.Initializing the number of epochs.
3.Reshaping and normalizing the retrieved data.

4 ARCHITECTURE

Neural Network is loosely modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales. A neural net consists of thousands or even millions of simple processing nodes(neurons) that are densely interconnected. Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data.



The nodes or neurons of the input layer is passive. All they do is pass the data received from input to the next layer. Nodes in Hidden layer and output layer are active. The values entering a hidden node are multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number. Neural networks can have any number of layers, and any number of nodes per layer. Most applications use the three-layer structure with a maximum of a few hundred input nodes.

4.1 Neural Network implementation with one hidden layer

The neural network solution is implemented by mapping the input x towards a particular category by using the sigmoid function.

The forward pass on the input x executes the following computation.

$$\hat{y} = \sigma(w^T x + b).$$

Here σ is the sigmoid function where its value is as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

We will use cross entropy for calculating the cost function. The formula for a single training example is:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}).$$

Averaging over a training set of m examples we have:

$$L(Y, \hat{Y}) = -\frac{1}{m} \sum_{j=0}^{m} \sum_{i=0}^{n} y_i^{(j)} \log(\hat{y}_i^{(j)}).$$

For back propagation, we need to know how the loss function L changes with respect to each component *wj* of w. That is we must compute each δL/δ*wj*

The formulas for these steps are:

$$\begin{aligned} z &= w^T x + b, \\ \hat{y} &= \sigma(z), \\ L(y, \hat{y}) &= -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}). \end{aligned}$$

Computing δL/δ*wj* we get,

$$\frac{\partial L}{\partial z_i} = \hat{y}_i - y_i.$$

4.2 Multi-Layer Neural Network implementation with KERAS

- For building the model, we have used sequential method of KERAS and have given input with Flatten method with shape as 28x28**.**
- I have used a single hidden layer and given the activation function as sigmoid.
- I have used soft-max function for the output layer.
- Then the model is compiled with an optimizer and loss attributes.
- The model is fit against the training data, epochs and validation data.
- The graph is plotted against the loss and the number of epochs.
- The confusion matrix is calculated, and the accuracy is also retrieved.

4.3 Convolutional Neural Network implementation with KERAS

- For building the model, we have used sequential method of KERAS.
- Two convolution layers are added with activation function as RELU.
- The best features are chosen via max- pooling.
- The soft-max activation function is used to get the required output.
- Then the model is compiled with an optimizer and loss attributes.
- The model is fit against the training data, epochs and validation data.
- The graph is plotted against the loss and the number of epochs.
- The confusion matrix is calculated, and the accuracy is also retrieved.

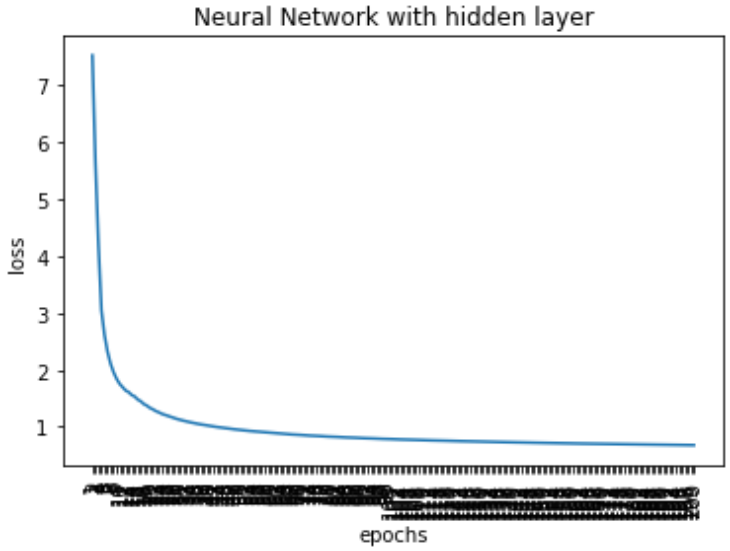
5 RESULTS

5.1 Neural Network implementation with a hidden layer

In this experiment, the learning rate is 1, number of hidden nodes is 64

The graph between loss and epochs=200 is as follows:

Accuracy on the data is: **74.85000000000001 %**

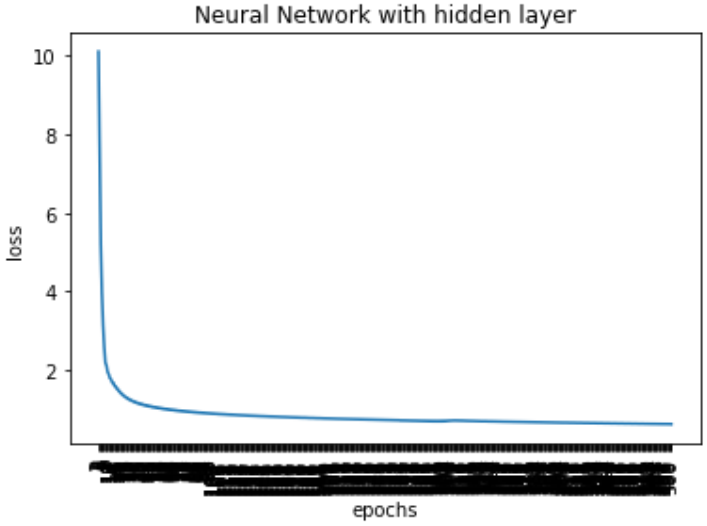


Confusion matrix is:

```
[[747  22  37  62   4   1 176   0   8   0]
 [  7 902   0  40   7   0   2   0   1   0]
 [ 16  16 527   6 148   4 127   0  20   0]
 [ 74  41  11 765  39   2  42   0  10   4]
 [ 17   4 249  55 664   1 149   0   4   1]
 [  6   0   7   1   2 805  10  70  18  32]
 [108  12 146  59 122   4 450   0  42   0]
 [  0   0   0   0   1  86   0 840  13  57]
 [ 25   3  21  11  13  21  43   6 882   3]
 [  0   0   2   1   0  76   1  84   2 903]]
```

The graph between loss and epochs=500 is as follows:

Accuracy on the data is: **76.94%**



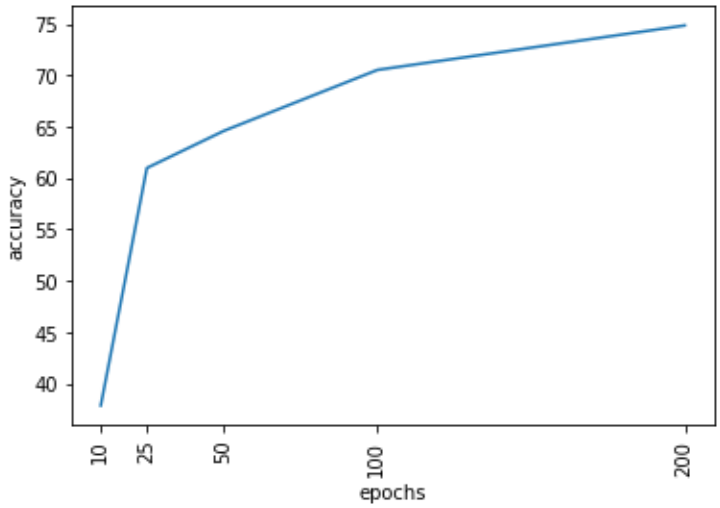
Confusion matrix is:

```
[[834  14  27  75   3   2 253   0   5   2]
 [  7 913   3  22   2   0   4   0   3   0]
 [ 37  19 740  27 293   0 237   0  43   3]
 [ 66  46  15 817  68   3  41   0   8   1]
 [ 10   6 165  35 585   0 144   0   6   0]
 [  2   0   4   1   2 849   3  53   9  25]
 [ 24   1  37  16  40   1 286   0  13   0]
 [  0   0   0   0   0  73   0 856   7  53]
 [ 19   1   8   7   6  19  32   8 902   4]
 [  1   0   1   0   1  53   0  83   4 912]]
```

The accuracy varies with different epochs. The table is as follows:

Epochs	Accuracy
10	37.84%
25	60.97 %
50	64.58 %
100	70.52 %
200	74.85%

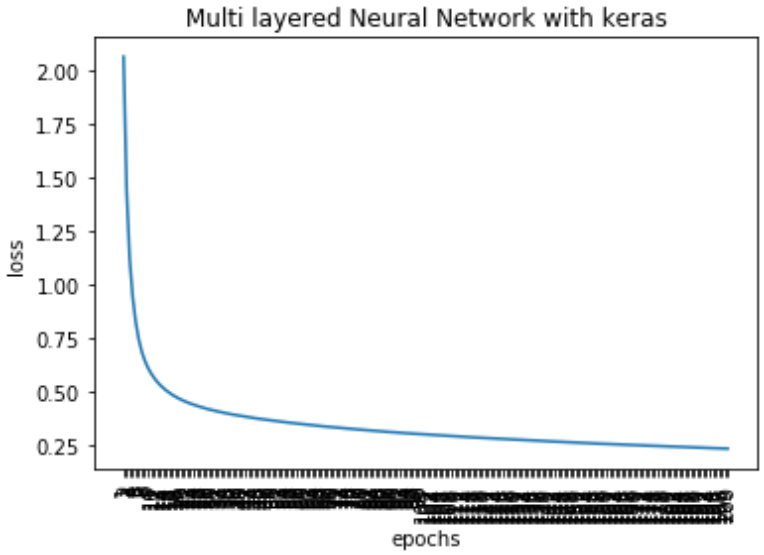
The graph is as follows:



5.2 Multi-Layer Neural Network implementation with KERAS

The graph between loss and epochs=200 is as follows:

Accuracy on the data is: **88.09 %**



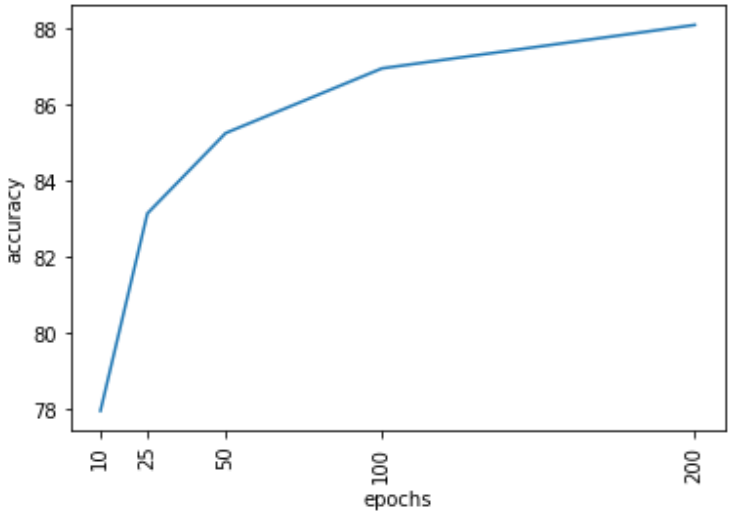
Confusion matrix is:

```
[[844   1  15  33   4   2  94   0   7   0]
 [  2 967   1  24   3   0   2   0   1   0]
 [ 17   0 806  18  98   0  60   0   1   0]
 [ 21   8  10 908  27   0  22   0   4   0]
 [  1   1 101  37 803   0  55   0   2   0]
 [  0   0   0   1   0 944   0  32   2  21]
 [141   1  81  38  66   0 659   0  14   0]
 [  0   0   0   0   0  22   0 956   0  22]
 [  4   0   4   7   5   2   6   3 969   0]
 [  0   0   0   0   0   7   1  39   0 953]]
```

The accuracy varies with different epochs. The table is as follows:

Epochs	Accuracy
10	77.96%
25	83.14 %
50	85.25 %
100	86.95 %
200	88.09 %

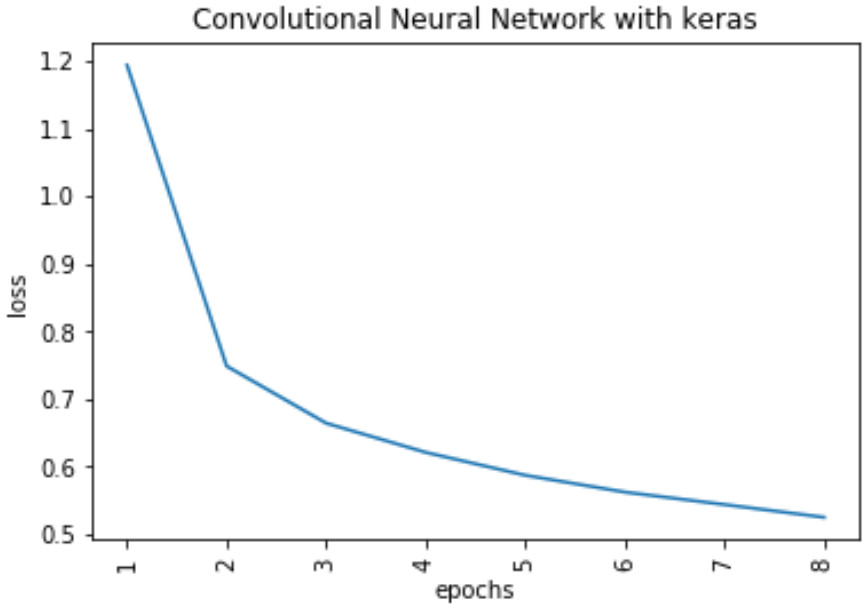
The graph is as follows:



5.3 Convolutional Neural Network implementation with KERAS

The graph between loss and epochs=8 is as follows:

Accuracy on the data is: **83.22%**



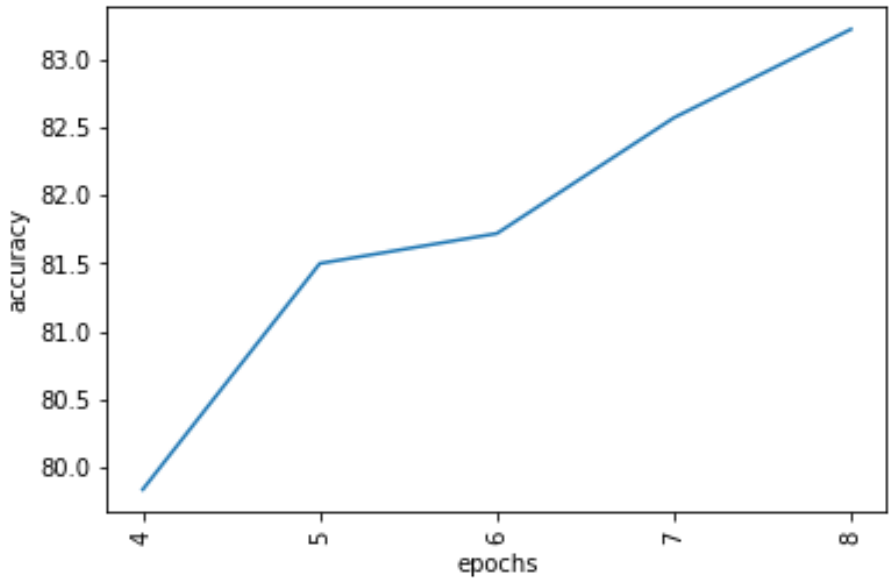
```
Confusion matrix is:

[[807   3  23  60   7   2  82   0  16   0]
 [  3 952   4  28   9   0   2   0   2   0]
 [ 13   2 678  13 234   1  46   0  13   0]
 [ 22  14  15 880  36   1  27   0   5   0]
 [  0   1  67  38 838   1  48   0   7   0]
 [  0   0   0   1   0 924   0  49   3  23]
 [194   2 130  39 199   0 408   0  28   0]
 [  0   0   0   0   0  29   0 940   0  31]
 [  2   1   8   8   6   2   7   6 960   0]
 [  0   0   0   0   0   8   0  56   1 935]]
```

The accuracy varies with different epochs. The table is as follows:

Epochs	Accuracy
4	79.84%
5	81.50%
6	81.72%
7	82.57%
8	83.22%

The graph is as follows:



6 CONCLUSION

With this experiment, the Neural Network with hidden layer, Multi-Layer Neural Network with KERAS and Convolutional Neural Network with KERAS has been implemented. The model is trained with different epochs. The graph for loss versus epochs and accuracy versus epochs are plotted. It is observed that the accuracy and loss increase as the epoch value increases. When the epochs are increased to a very high value the accuracy value increase dramatically which is not ideal.

7 REFERENCES

* Used the docx file in the sit https://nips.cc/Conferences/2015/PaperInformation/StyleFiles
* https://jonathanweisberg.org/post/A%20Neural%20Network%20from%20Scratch%20-%20Part%201/
* https://towardsdatascience.com/a-simple-2d-cnn-for-mnist-digit-recognition-a998dbc1e79a
* https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d