Project-2: NEURAL NETWORK AND CONVOLUTION NEURAL NETWORK

Shubhangi Mane
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
UB Id:50295705
smane2@buffalo.edu

ABSTRACT

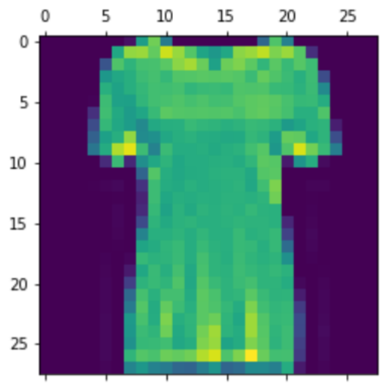
Understanding images is one of the crucial activity in modern world. In this project, the classification of images is complemented by using neural networks and a model is created for recognizing an image and identify it as one of ten classes. These results can be used to make a proper judgment as to the classify fashion images and categories.

INTRODUCTION:

In this project we are building Neural Networks and Convolutional Neural Network on Fashion MNIST data, which is collected from Zalando's article images and has 70000 sample inputs with 784 attributes. Data set is pre-processed first and fed to Neural network and Convolutional neural network. The results obtained are evaluated on various parameters like Accuracy. The target values are scalars that can take values from 1 of 10 classes. Although the training target values are discrete we use Neural network to obtain real values which is more useful for classification.
The classification table from 60000 samples shows the occurrence from prediction and observation samples, producing percentage of correct classification for results is 80%. The accuracy is compared with validated samples which are 10000 samples and the percentage of correct classification is 81.2%

DATASET:

Fashion MNIST dataset contains 70000 instances with 784 attributes. Each image is stored in computers as matrix of pixels, where each pixel is one or more number. The values are in the range 0 to 255, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. Each example is a 28X28 grayscale image, associated with each is a label from 10 classes. Fashion MNIST reader notebook is used to load the dataset. We have used neural networks to train the model. This analysis aims to observe which features are most helpful in predicting label of images and to see general trends that may aid us in model selection and hyper parameter selection.



The labels associated with the images are as follows:

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

SPLITING THE DATASET:

We have divided the data set into two parts namely Training set, Test set. Training input set contains 60000 instances with 784 attributes and Training output set contains 60000 instances with 1 attribute. Test input set each contain 10000 instances with 784 attributes and Test output set contains 10000 instances with 1 attribute.

The model is initially fit on a Training dataset. The model is trained on the training dataset using a supervised learning method. In practice, the training dataset often consist of pairs of an input vector and the corresponding output vector which is commonly denoted as the target, either predicts the label.

INPUT DATASET:

DATASET	**INSTANCES**	**ATTRIBUTES**
TRAINING	60000	784
TEST	10000	784

OUTPUT DATASET

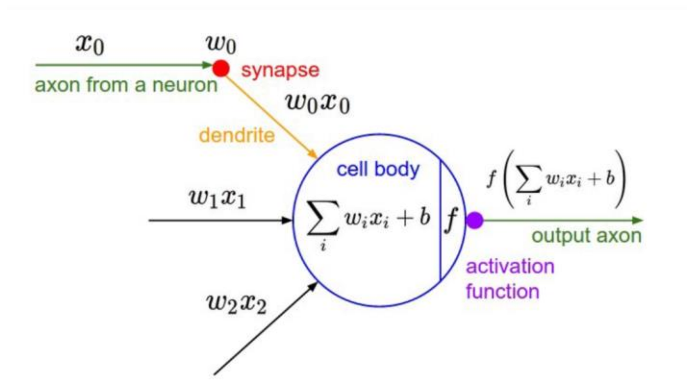
DATASET	**INSTANCES**	**ATTRIBUTES**
TRAINING	60000	1
TEST	10000	1

PREPROCESSING:

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn Most of the times, your dataset will contain features highly varying in magnitudes, units and range. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. We have scaled these values to a range of 0 to 1 before feeding them to the neural network model. To do so, we have divided the values by 255.Normalization is applied for both the inputs and output datasets.

ARCHITECTURE:

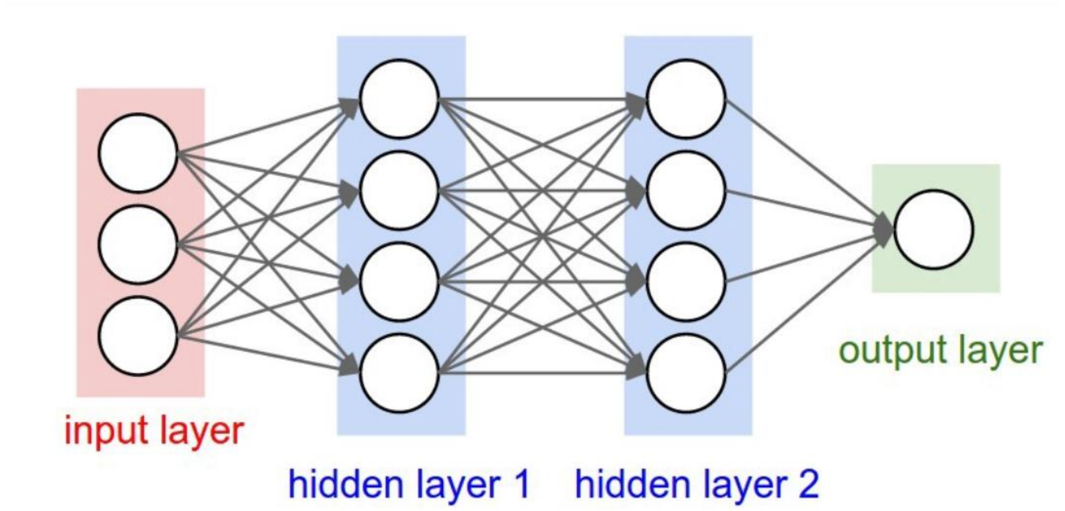
Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. Neural networks are multi-layer networks of neurons that we use to classify labels. A neural network contains layers of interconnected nodes. The idea of artificial neuron is that we have a data vector X, a vector of parameters W and a bias vector b.



The output of the neurons is given by the below equation and f here is an activation function that controls the output signal.

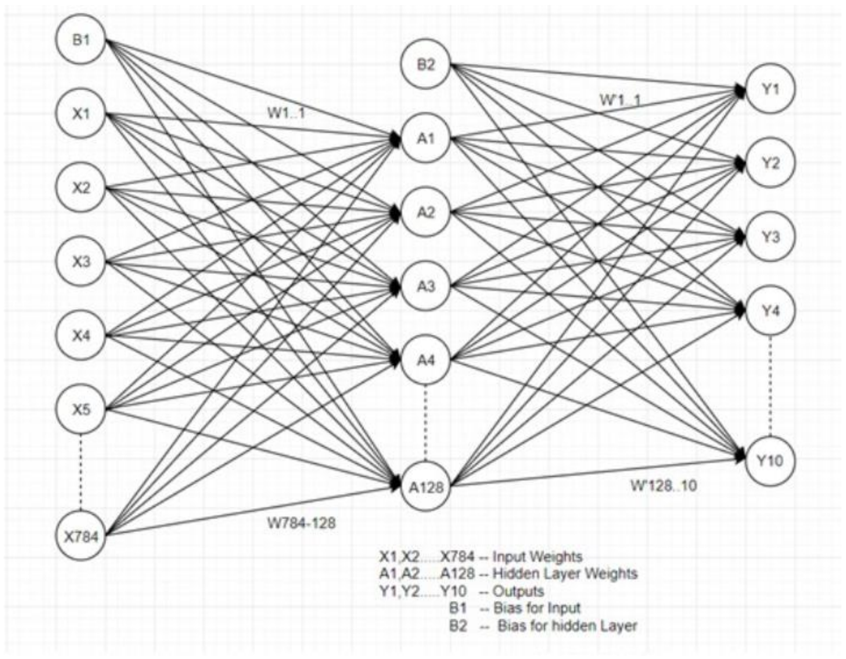
$$Y = f\left(\sum_i w_i x_i + b\right)$$

In a multi layered Neural network, input layer collects the input patterns. The output layer has classifications or output signals. Hidden layers fine-tune the input weightings until the neural network's error is minimal



We have developed a neural network where we have 784 units in Input layer and 128 units in hidden layer with 10 units in output layer for each label.

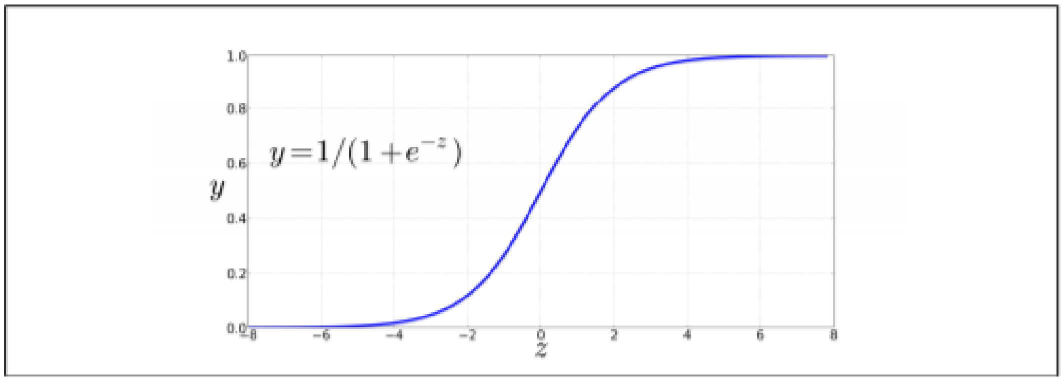
We have used sigmoid as activation function from input to hidden layer and softmax as activation function from hidden layer to output



Sigmoid Function:

The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

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



Softmax Function:

Softmax function calculates the probabilities distribution of the event over ‘n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_K^{k=1} e^{z_k}} \text{for } j = 1, \ldots, k$$

Hyper-parameter: A hyper parameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training. In this project we have one hyper parameter learning rate. We can tune this learning rate until we get minimum cost function.

We can get updated weights and bias by subtracting the delta_weight and delta_bias from weight and bias. We continue this till we get minimum cost.
Epoch: One epoch is defined as one-time execution of all the steps beginning from sigmoid function to updating weights. After every epoch the weights and bias values will get updated with the below discussed equations:

Neural networks consists of three tasks:

1. Forward Propagation
2. Calculate the error
3. Backward Propagation

1. Forward Propagation:

The information travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes. Initially we set the values of the input node. Weight vector is initialized with random values and the bias vector is initialized with zeros. We are using the sigmoid activation function for hidden layer and then we calculate the hidden node activation values. Next calculate the output node values. We are using softmax activation function on the hidden node values to get the final output.

z_input = train_input* weight_input + bias_input
a_input = sigmoid(z_input)

z_hidden = (a_input * weight_hidden) + bias_hidden
a_hidden = softmax(z_hidden)

2. Error Calculation:

Error is computed based on the prediction and the provided labels. We are using Log loss function to evaluate the error

$$\text{Cross-Entropy} = -\sum_{c=1}^{C} y_{o,c} \log(p_{o,c})$$

Where C is number labels, y is the binary indicator if the label is correctly predicted or not, and p is the predicted probability that o is of label C

3. Back Propagation

We use the calculated error to adjust the weights. We move the error backwards through our model starting from output layer to the hidden layers, all the way to the input layer of our model. So, basically in back propagation we calculate the error attribute to each unit and that in turn allows us to calculate the partial derivatives and the gradients.

Backward propagation from Output layer to Hidden layer:

delta_weight_hidden= a_input.T * (a_hidden - train_output)
weight_hidden = weight_hidden - (learning_rate*delta_weight_hidden)

delta_bias_hidden= (a_hidden - train_ output)
bias_hidden = bias_hidden - (learning_rate*delta_bias_hidden)

Backward propagation from Hidden layer to Input layer:

```
delta_weight_input=train_X (1-a_input) a_input a_hidden, train_Y1 weight_hidden
weight_input=weight_input-(learning_rate*delta_weight_input)

delta_bias_input = (1-a_input)a_input a_hidden, train_Y1 weight_hidden
bias_input  = bias_input - (learning_rate * delta_bias_input)
```

MULTI LAYER NEURAL NETWORK USING KERAS AND TENSORFLOW:

In task 2 of the project we have implemented multi-layer neural networks using keras and TensorFlow.
Step 1: We can access the fashion MNIST directly from TensorFlow. After the data is loaded four arrays are returned which represent train_images, train_labels, test_images and test_labels. These images are 28X28 arrays, with pixel values ranging from 0 to 255.
Step 2: This step corresponds to preprocessing the data, where in images are scaled between 0 to 1 values, which is obtained by dividing the values by 255.
Step 3: This step corresponds to building the model. In order to build the model, we need to configure the layers and then compile the model. We have configured three hidden layers.
Step 4: The first hidden layer of the network we use Flatten, which transforms the image from 2D to 1D array. The second hidden layers is configured using Dense where we have 128 nodes and the final layer is configured using Dense with 10 nodes one for each class.
Step 5: The compile step measures how accurate the model is based on loss function. Based on the values obtained in loss, model is optimized and then accuracy is calculated.
Step 6: Next, we train the model, using fit method and the model learns to associate images and labels.
Step 7: Later, we can use the model to predict the labels for test dataset.
Step 8: Based on the labels predicted we evaluate the accuracy of the model.

CONVOLUTIONAL NEURAL NETWORK:

Convolutional neural networks is a class of deep, forward (not recurrent) artificial neural networks that are applied to analyzing visual imagery. For visual image processing convolutional neural network is the best choice when compared with traditional neural networks. It is composed of convolutional layers and pooling layers.

Convolutional Layer: In this layer all images can be encoded in terms of 0's and 1's. Here feature detector is simply a matrix, which contains values corresponding to a feature image. The matrix overlays a section of the image and performs a bitwise multiplication with all the values of that location. The result of the bitwise multiplications are summed up and put in corresponding location of the feature map. It then shift to another section of the image and repeat the process until it has traversed the entire image.

Pooling Layer: We do not make use of feature detector in this like in convolutional layer. Instead we use max pooling. The process of max pooling consist in taking a highest value within the area of feature map laid by the window and putting in the corresponding location of the pooled feature map. Pooling is useful as it reduces the size of the image making it easier to compute and detect patterns despite differences in spatial orientation.

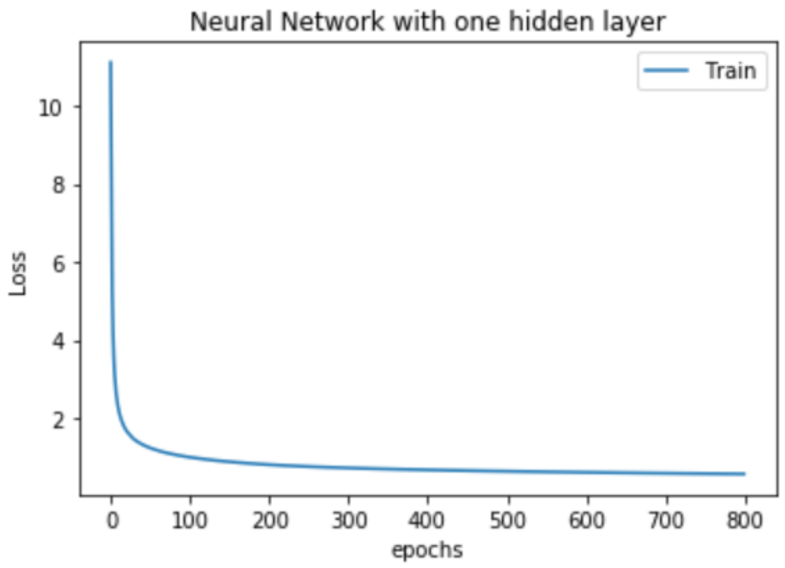
Below are the steps we performed while implementing the convolutional neural network with keras: We imported the required libraries for this project. Then we loaded the dataset of Fashion MNIST. Once it is done, we divide the dataset into training data and testing data. Then we modified the value of each pixel such that they are in the range of 0 and 1 so that we can improve the rate at which the model runs. This model can not work with categorical data directly so we must use one hot encoding. In hot encoding, the digits 0 through 9 is represented by set of 9 zeros and a single one.

Once preprocessing is done, we will train our system with the function fit. After this we calculate the loss and accuracy with the function evaluate.

RESULTS AND OBSERVATIONS:

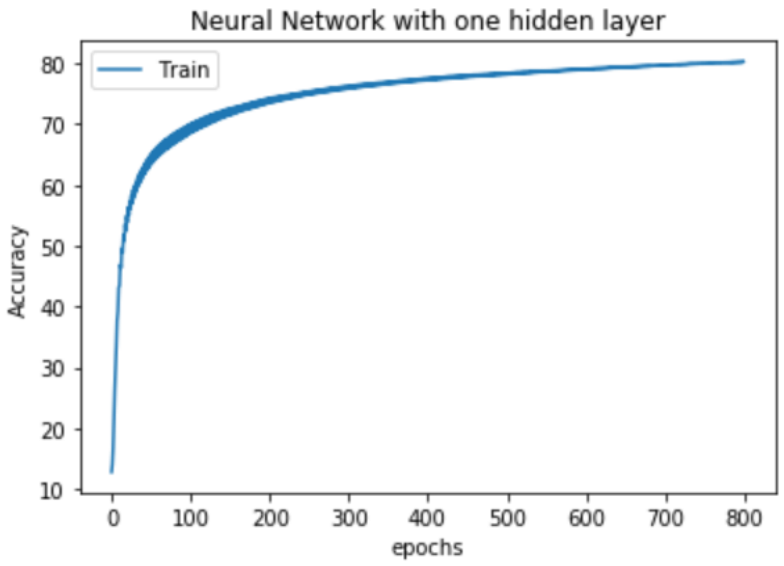
1. RESULTS AND ANALYSIS OF TRAINING SET FOR NEURAL NETWORK WITH ONE HIDDEN LAYER:

The below graph depicts Cost versus Number of iterations at **learning rate=0.5**



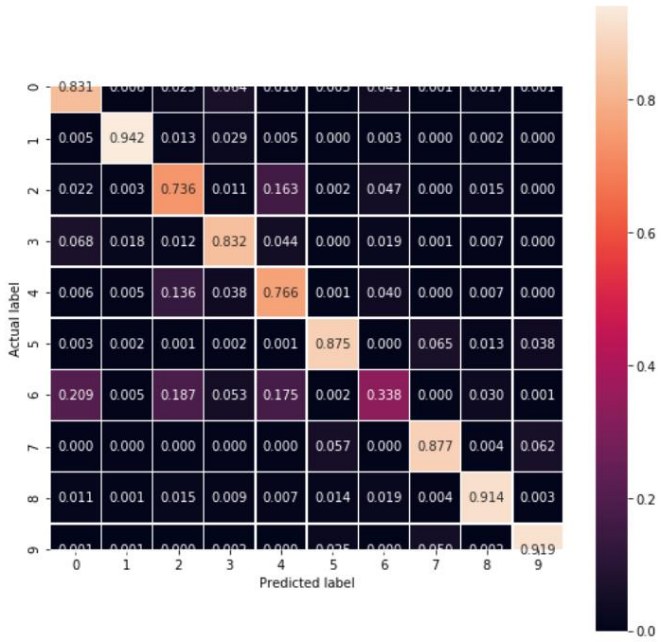
The above graph predicts as the number of iterations increases, the model is trained better and loss incurred is less for every successive iterations.

Accuracy graph for Training set:



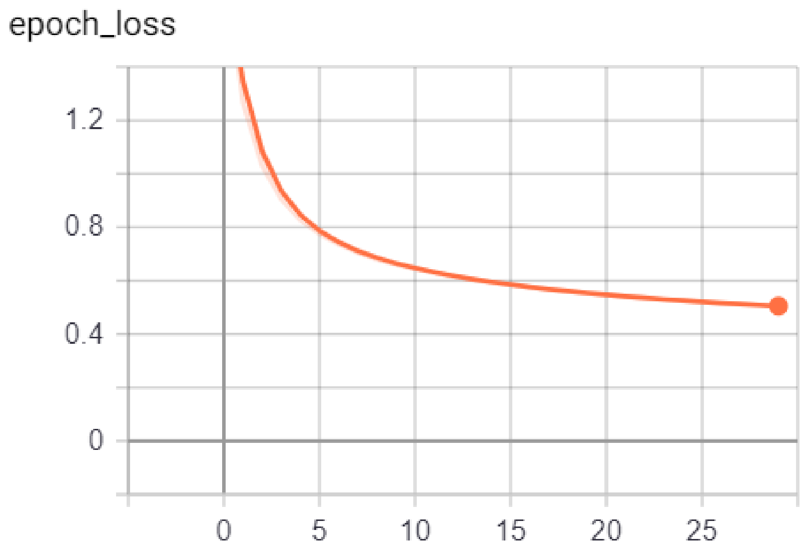
Accuracy for training set is 80.29%

CONFUSION MATRIX:



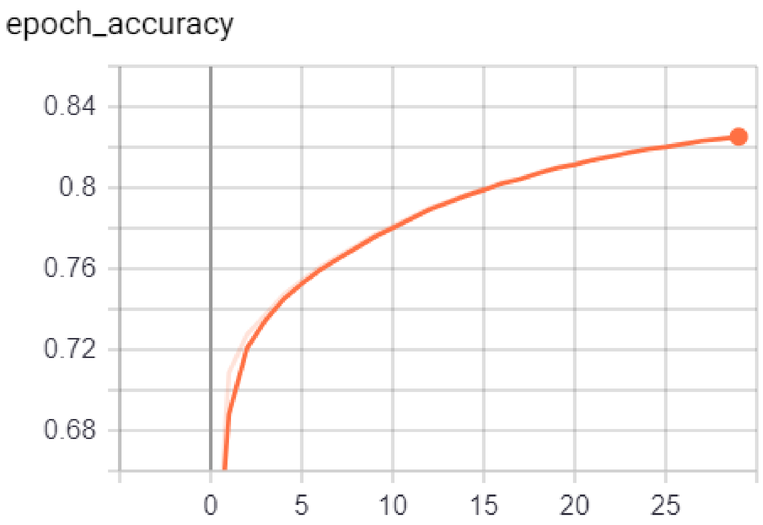
2. RESULTS AND ANALYSIS FOR MULTI LAYER NEURAL NETWORK

The below graph depicts Cost versus Number of iterations



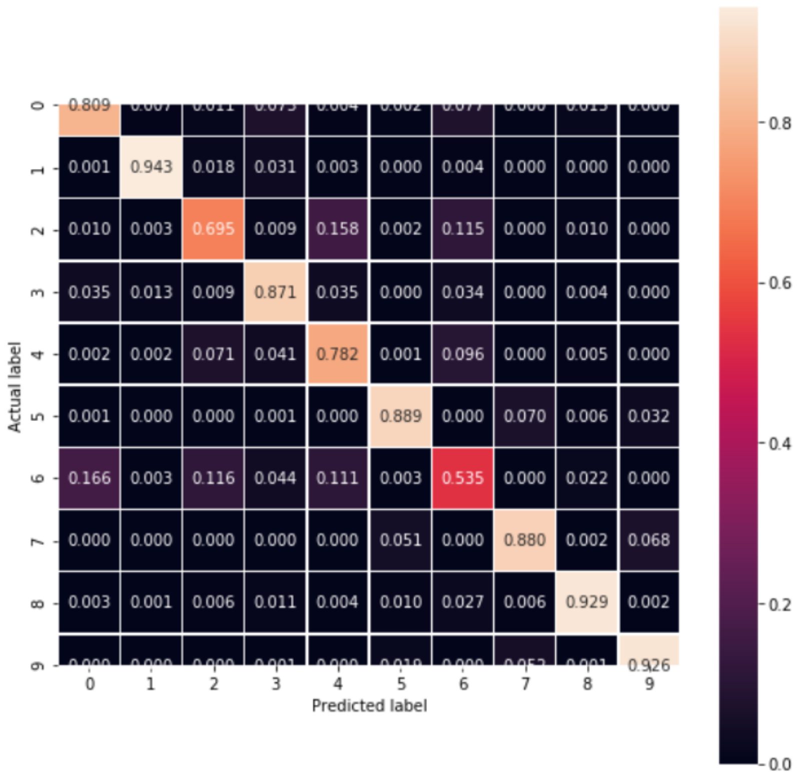
The above graph predicts as the number of iterations increases, the model is trained better and loss incurred is less for every successive iterations.

Accuracy graph for Training set:



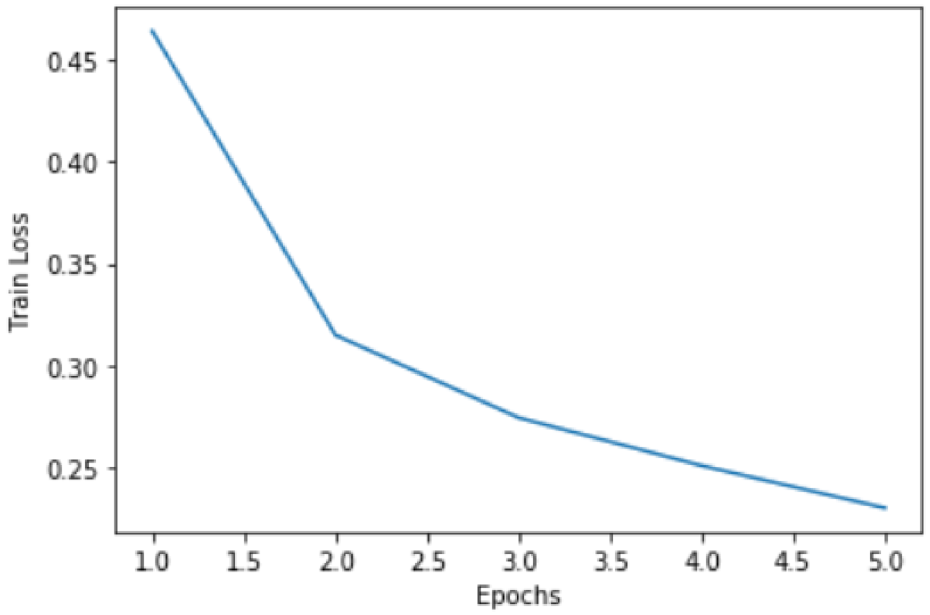
Accuracy for training set is 81.19%

CONFUSION MATRIX:



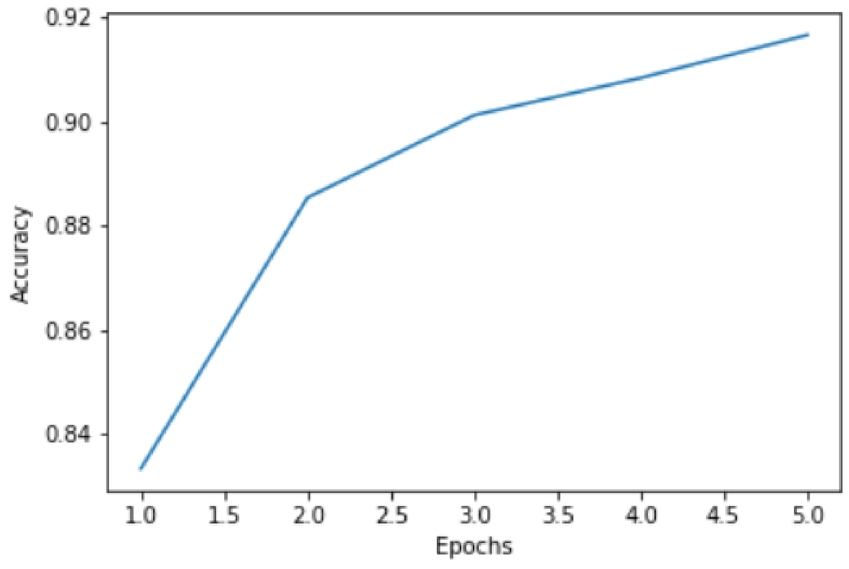
3. RESULTS AND ANALYSIS FOR CONVOLUTIONAL NEURAL NETWORK

The below graph depicts Cost versus Number of iterations



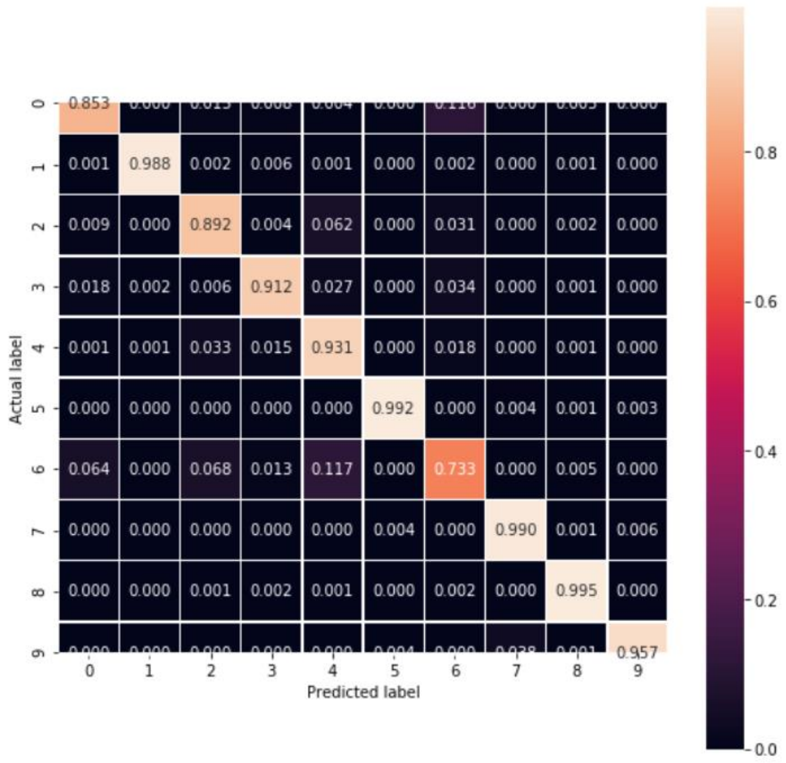
The above graph predicts as the number of iterations increases, the model is trained better and loss incurred is less for every successive iteration.

Accuracy graph for Training set:



Accuracy for training set is 91.57%

CONFUSION MATRIX FOR TRAINING DATA:



CONCLUSION:s

Initially we designed a model with a single hidden layer and trained the model tuning the hyper parameters and we have observed that the best **learning rate is at 0.5**, for single hidden layer Neural network, where accuracy is 80%, also labels of the images were predicted correctly for test data. In task 2, we have designed a multilayer neural network with Keras and TensorFlow where we have three hidden layers, and accuracy is 81% which depicts, as we have more hidden layers the model is trained better. In task 3, Convolutional Neural network is built and has accuracy of 89%.

Following are the observations on Test dataset:

Accuracy for Neural network with one hidden layer is: 80%
Accuracy for Neural network with multi hidden layer is: 81%
Accuracy for Convolution Neural Network is: 89%

REFERENCES:

1. https://blog.statsbot.co/neural-networks-for-beginners-d99f2235efca
2. https://medium.com/tensorflow/hello-deep-learning-fashion-mnist-with-keras-50fcff8cd74a
3. https://towardsdatascience.com/fashion-product-image-classification-using-neural-networks-machine-learning-from-scratch-part-e9fda9e47661
4. https://www.tensorflow.org/tutorials/keras/classification
5. https://www.investopedia.com/terms/n/neuralnetwork.asp
6. https://skymind.ai/wiki/neural-network
7. https://towardsdatascience.com/understanding-neural-networks-19020b758230
8. https://dzone.com/articles/the-very-basic-introduction-to-feed-forward-neural
9. https://www.tensorflow.org/tensorboard/scalars_and_keras