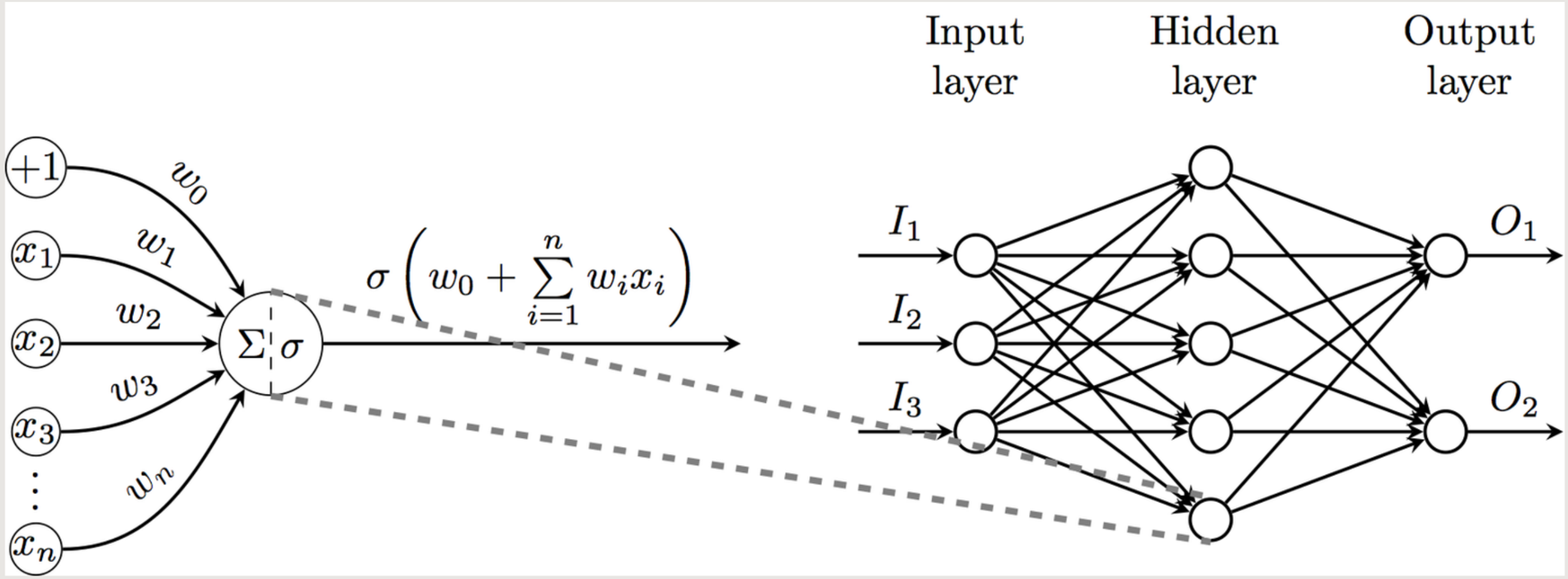
MULTI-LAYER PERCEPTRON IMPLEMENTATION FOR OPTICAL CHARACTER RECOGNIATION

Presentation for Intelligent Computing
under Dr. Pratik Chattopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology Varanasi

Presented by:
Amitesh Sahu (21074006)
Rituraj Barai (21074024)
Soustab Haldar (21074029)

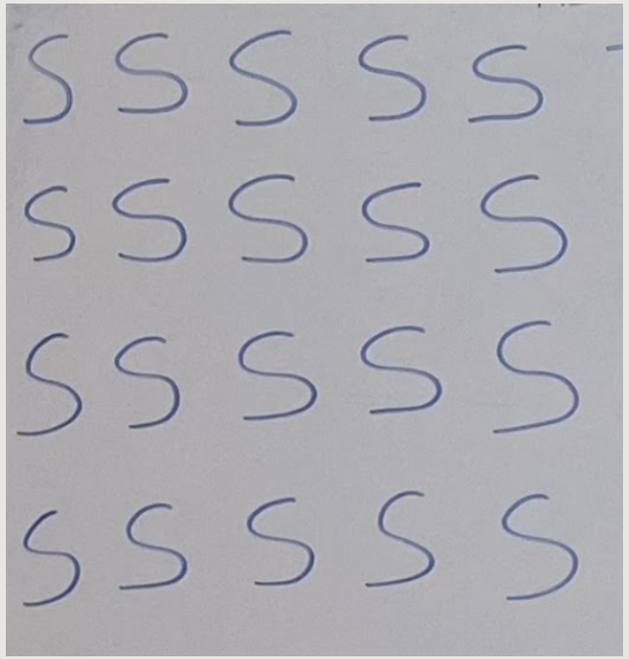
Introduction to Multi Layer Perceptron:

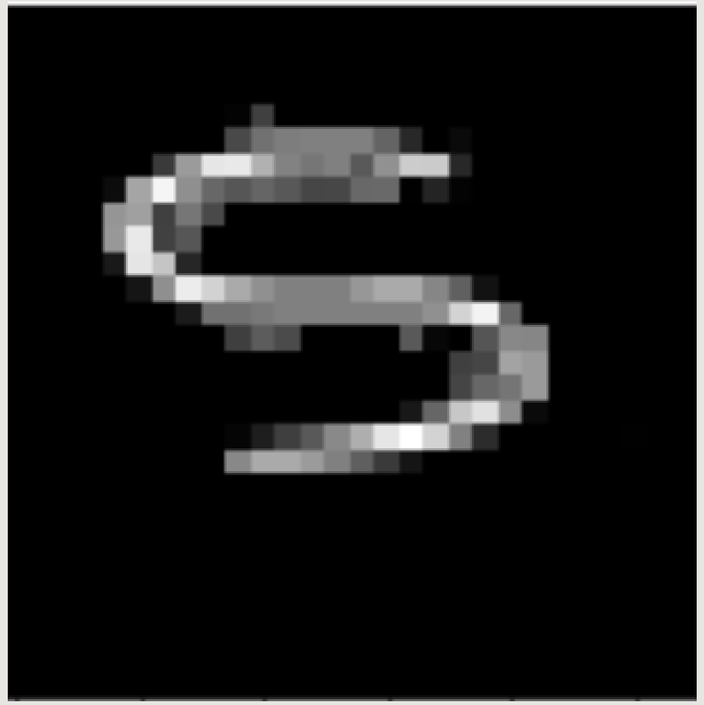
A Multi-Layer Perceptron (MLP) is a type of artificial neural network designed for supervised learning. It is a feedforward neural network that consists of multiple layers of interconnected artificial neurons (or nodes). The MLP is a fundamental and widely used neural network architecture in machine learning and deep learning



Dataset for Optical Character Recognition

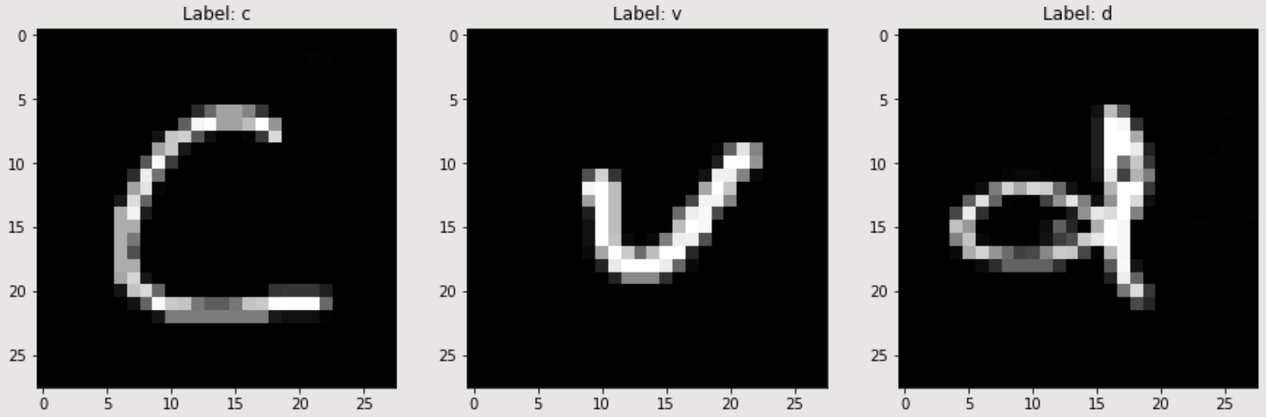
The dataset comprises of images of characters which are then cropped and preprocessed to convert them into machine readable black & white (1 channel) format of 28X28 pixels which is similar to the standard MNIST dataset.





Dataset for Optical Character Recognition

- The preprocessed images are stored in their specific labelled folders. But for them to be processed by the neural network, they need to be converted to array of pixels. These arrays will contain the pixel values for all the 784 pixels across an image and also the label of the image in integer format (1 for ‘a’, 2 for ‘b’ and so on).
- These list of arrays for all the images are stored in a CSV file for further training without all the necessary image processing data.



How does Multilayer Perceptron work?

- Randomly generate the weights of the connections using the random float generator function
- Calculate the weighted sum using the inputs and weights of the connections
- Apply the activation function (here the *tanh* function) to the weighted sum to find the value in the hidden layer and softmax function on the output layer
- Now calculate the loss by comparing the predicted value with the actual value
- Apply back propagation – alter the weights using gradient and derivative of activation function to reduce the loss

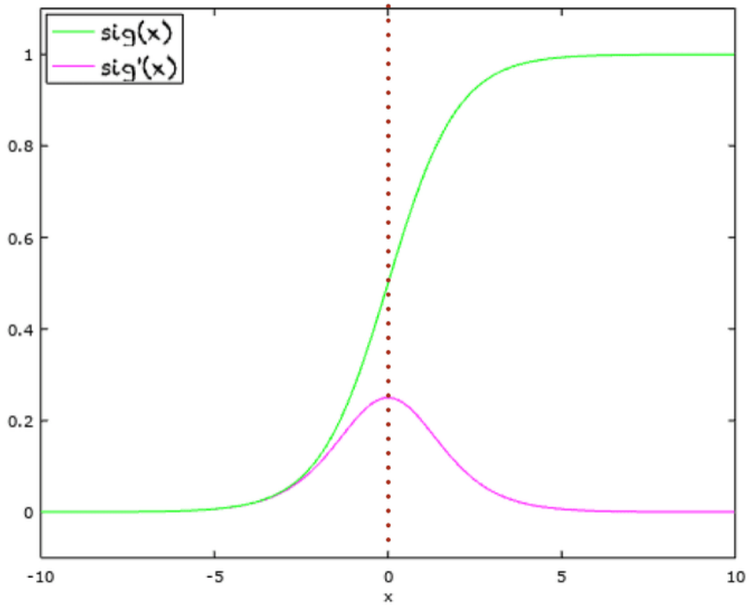
Repeat again and again till 100 epochs.

Activation Functions

Sigmoid Activation Function :

This function takes any real value as input and outputs values in the range of 0 to 1.

- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient. This is represented by an S-shape of the sigmoid activation function.



Plot of σ(x) and its derivate σ'(x)

Domain: (−∞, +∞)

Range: (0, +1)

σ(0) = 0.5

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

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

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

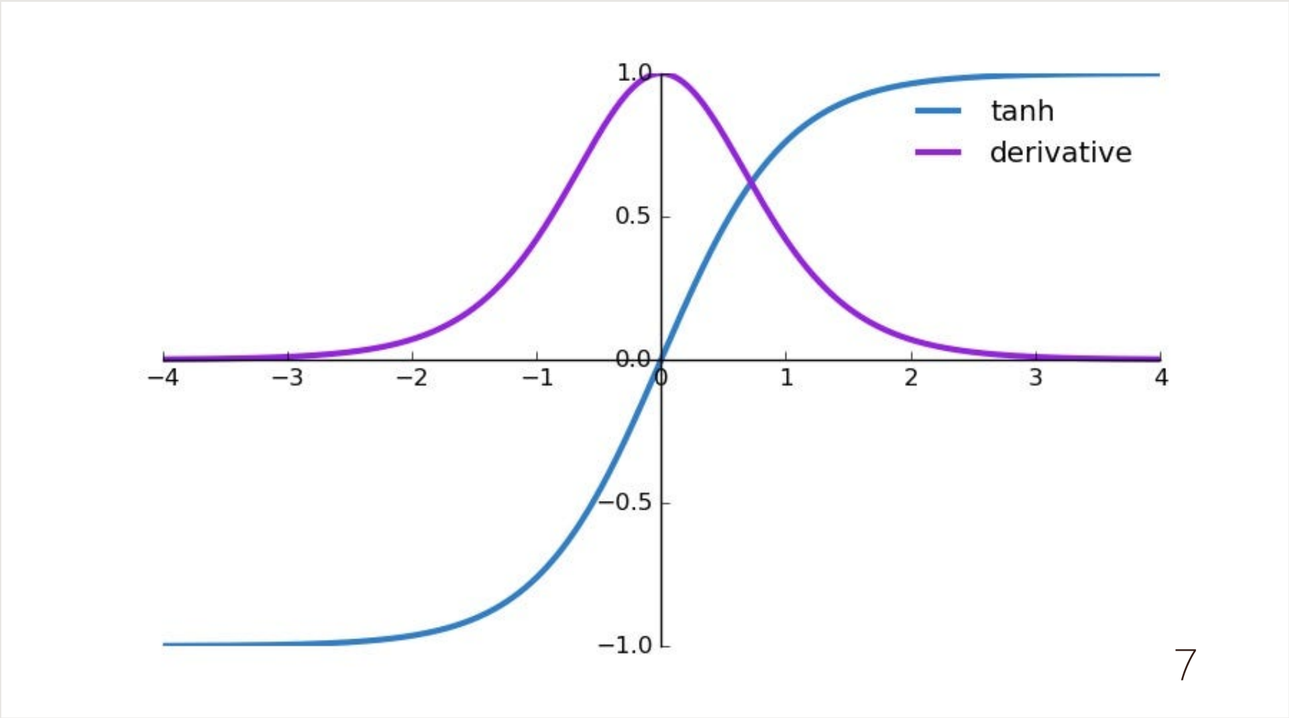
Activation Functions

Tanh Activation Function :

- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between -1 to 1; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 = \tanh(x)$$

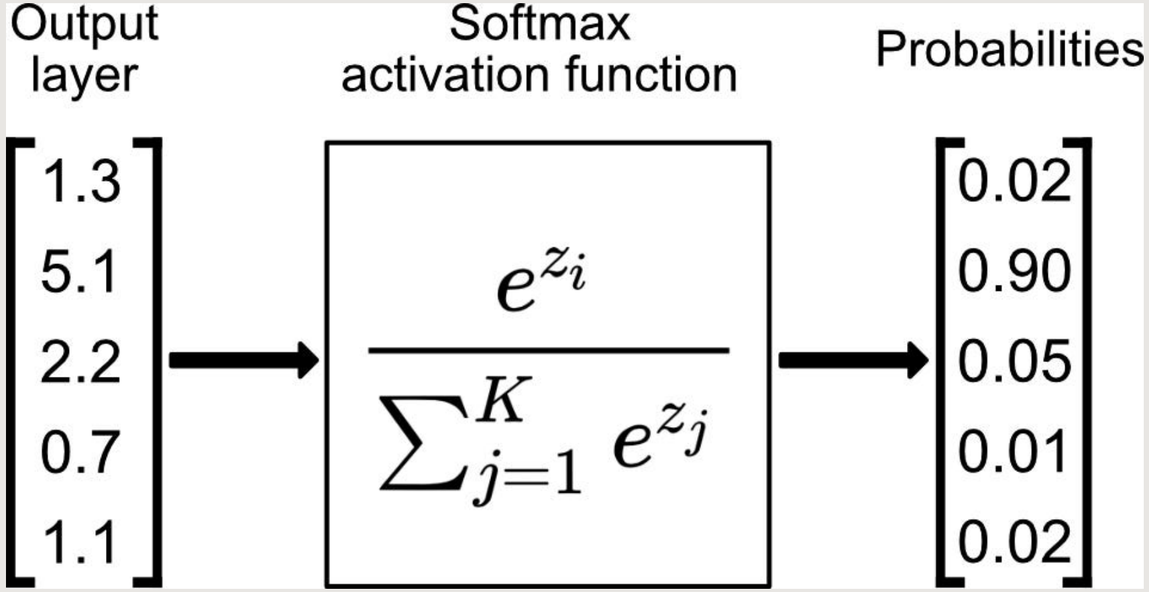
$$f'(x) = 1 - \tanh^2(x)$$



Activation Functions

Softmax Function: (for Output Layer)

- On using other activation functions on the output layer, when the sum of all neuron is calculated it can be greater than 1. Due to this, we cannot the predict the output using probabilty properly.
- Softmax activation function which effectively converts the output neurons into probabilities for each label. Thus we can calculate the accuracy of the model directly.



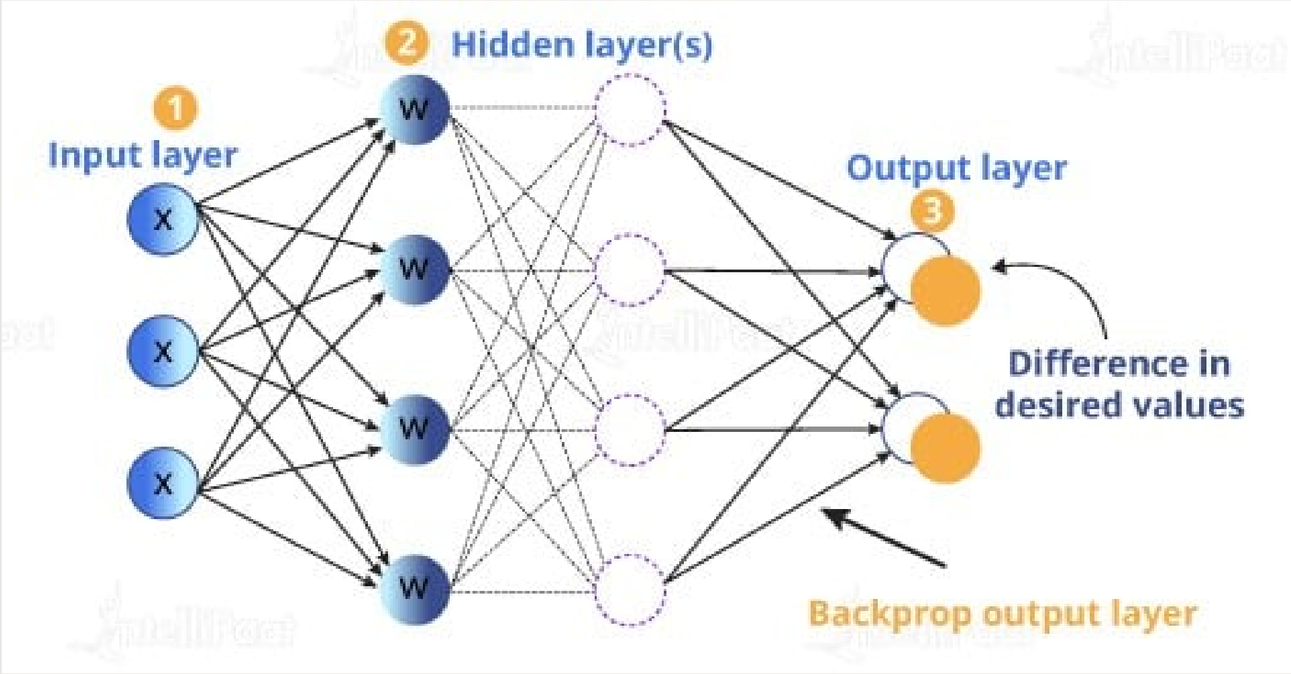
Feed Forward

- In a multilayer perceptron (MLP), "feedforward" refers to the process of propagating input data through the neural network to compute an output

  - This involves passing the input data through the layers of the network, applying weighted sums and activation functions in the hidden layers, and finally, using the softmax function to generate class probabilities for classification tasks.

Back Propagation

- Backpropagation is an optimization algorithm that involves taking the error rate of a forward propagation and feeding this loss backward through the neural network layers to fine-tune the weights.
- It uses gradient methods such as gradient descent or stochastic gradient descent to train multi-layer networks and update weights to minimize loss.



Code Structure

Parameters for Training MLP (after parameter optimization):

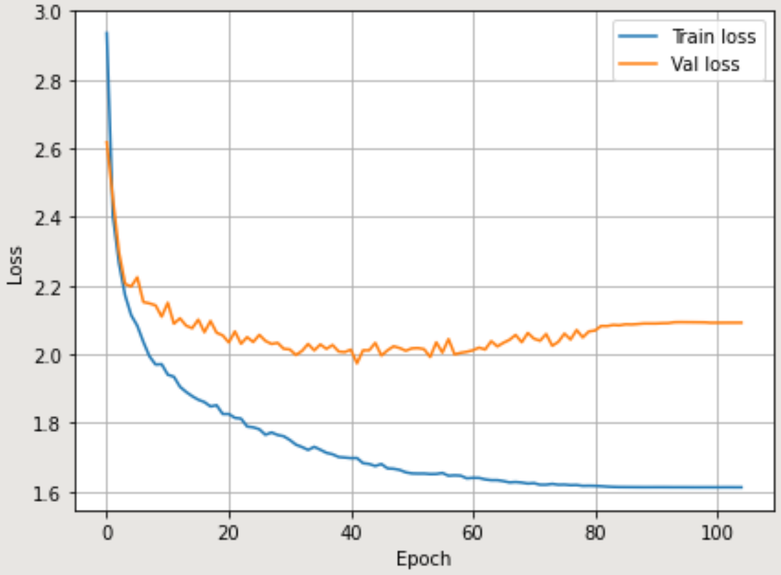
- **Number of nodes in hidden layer = 256**
- **Batch Size = 50 (number of training samples in one epoch)**
- **Epochs = 100 (number of iterations for training the model)**
- **Learning Rate = 0.80 (rate at which the weights are updated)**

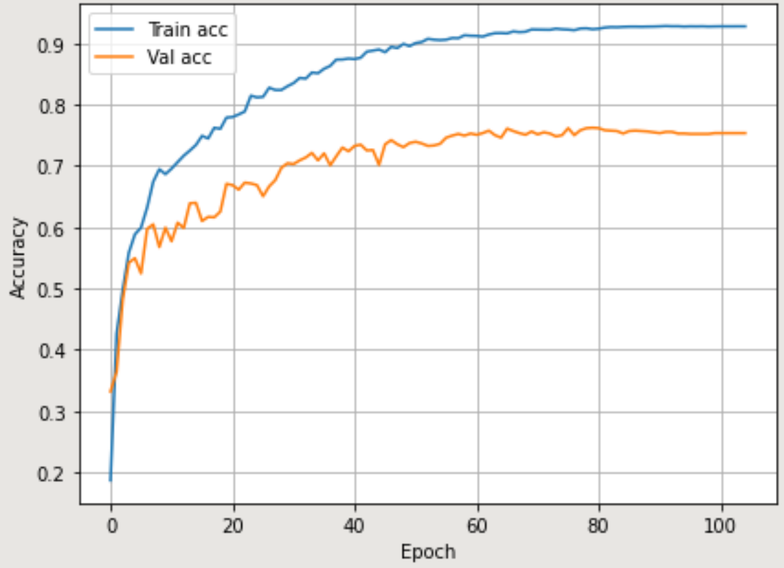
```
model = MLP(X_train,Y_train,X_test,Y_test,L=1,N_l=256)
model.train(batch_size=8,epochs=100,lr=0.8)
```

Results

The loss and accuracy measures on the training and validation data and their graphs v/s number of epochs is displayed as follows.

```
Epoch 97: loss = 1.612 | acc = 0.928 | val_loss = 2.093 | val_acc = 0.752 | train_time = 1.36 | tot_time = 1.372
Epoch 98: loss = 1.612 | acc = 0.928 | val_loss = 2.093 | val_acc = 0.752 | train_time = 1.314 | tot_time = 1.326
Epoch 99: loss = 1.612 | acc = 0.927 | val_loss = 2.093 | val_acc = 0.752 | train_time = 1.359 | tot_time = 1.37
Epoch 100: loss = 1.612 | acc = 0.928 | val_loss = 2.091 | val_acc = 0.753 | train_time = 1.306 | tot_time = 1.318
```





Results

For testing our MLP we have used various metrics:

- Accuracy: It is simply the percentage of images predicted correctly by our model on the training data.
- Precision: It is calculated as the weighted sum of the percentages truly classified values.
- Recall: It is the weighted sum of true positives from the sum of true positives and false negatives for all classes.
- F1 score: This is the best metric for testing multiclass classifications. It is calculated using the precision and recall values.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 1\text{-FNR}$$

$$\text{F1 score} = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

```
Accuracy:  0.7533692722371967
F1 score: 0.779817083279674
Recall: 0.7870619946091644
Precision: 0.7768761232291218
```

November 2023

THANK YOU

Rituraj Soustab Amitesh