Kudupudi Mohan Kumar
Roll No: 22105037

Sai Charan Mekala
Roll No: 22105069

Assignment 1

(CS776A) Deep learning for Computer vision

2023-02-05

1 Mutli Layer Perceptron

1.1 Introduction

These following diagram shows multi layer perceptron with 2 hidden layer. Input size is 512, h1 hidden layer has 64 neurons, h2 hidden layer has 64 neurons. Output layer is used for multi-class classification with 10 neurons. The activation function used in hidden layers is Relu and in output layer is Softmax. The loss function used is cross entropy loss.

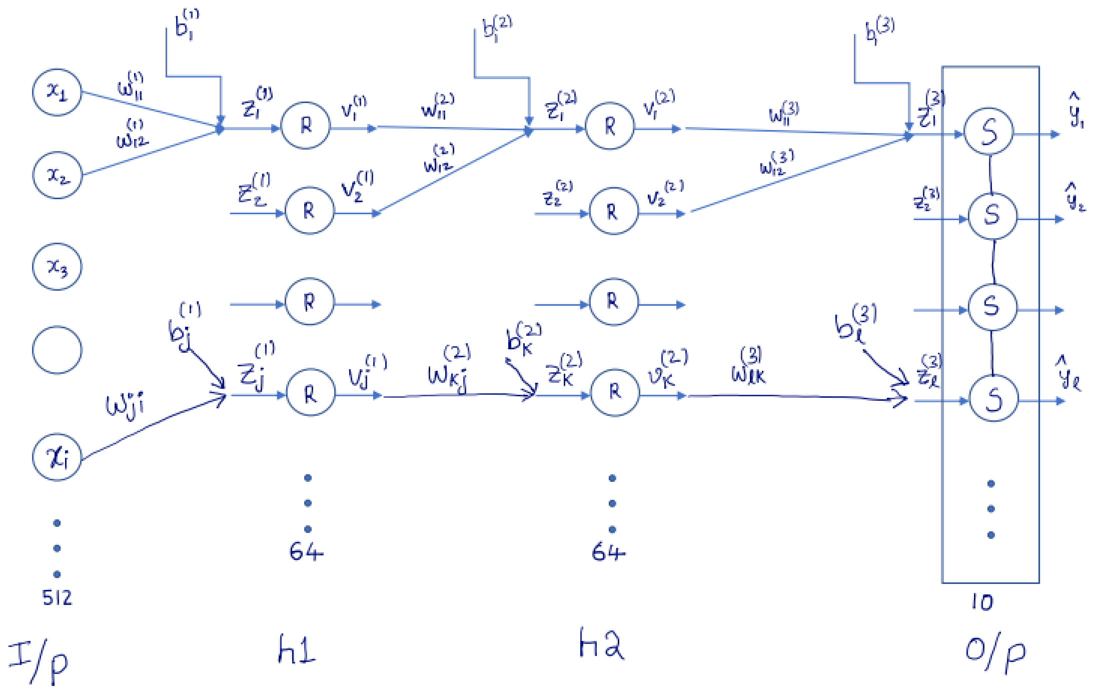

Figure 1: Multi Layer Perceptron with 2 hidden layers

$$\mathbf{X} = [x_i]_{512\text{x}1} \qquad \mathbf{b_1} = [b_j]_{64\text{x}1} \qquad \mathbf{R_j} = \text{Relu Activation}$$

$$\mathbf{W_1} = \left[w_{ij}^{(1)}\right]_{64\text{x}512} \qquad \mathbf{b_2} = [b_k]_{64\text{x}1} \qquad \mathbf{S} = \text{Softmax Activation}$$

$$\mathbf{W_2} = \left[w_{jk}^{(2)}\right]_{64\text{x}64} \qquad \mathbf{b_3} = [b_l]_{10\text{x}1}$$

$$\mathbf{W_3} = \left[w_{kl}^{(3)}\right]_{10\text{x}64}$$

where x_i is the i^{th} input, $w_{ij}^{(1)}$ is the weight from the i^{th} input to the j^{th} neuron in the first layer, b_j is the bias of the j^{th} neuron in the first layer, $w_{jk}^{(2)}$ is the weight from the j^{th} neuron in the first layer to the k^{th} neuron in the second layer, b_k is the bias of the k^{th} neuron in the second layer, $w_{kl}^{(3)}$ is the weight from the k^{th} neuron in the second layer to the l^{th} neuron in the third layer, b_l is the bias of the l^{th} neuron in the third layer, y_l is the l^{th} output, z_j is the j^{th} neuron in the first layer, a_k is the k^{th} neuron in the second layer, h_l is the l^{th} neuron in the third layer, R_j is the Relu activation function, S is the Softmax activation function.

1.2 Activation Functions

R indicates the Relu activation function in the hidden layers h_1 & h_2, which is given by

$$\text{Relu}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$$

S indicates the Softmax activation function in O/P layer, which is given by

$$\text{Softmax}(\mathbf{X}) = \begin{bmatrix} \frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}} \\ \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}} \\ \vdots \\ \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}} \end{bmatrix}_{n\text{x}1}$$

1.3 Forward Propagation

In farword propagation, the output of previous layer will be the input to the next layer. The final output of the layer is calculated using the activation function of that layer. The output of the last layer is the final output of the network. The forward propagation is given by (ref to fig 1)

$$\mathbf{Z_1} = \left[Z_j^{(1)}\right]_{64\text{x}1} \qquad \mathbf{V_1} = \left[v_j^{(1)}\right]_{64\text{x}1}$$

$$\mathbf{Z_2} = \left[Z_k^{(2)}\right]_{64\text{x}1} \qquad \mathbf{V_2} = \left[v_k^{(2)}\right]_{64\text{x}1}$$

$$\mathbf{Z_3} = \left[Z_l^{(3)}\right]_{10\text{x}1} \qquad \hat{\mathbf{y}} = \left[\hat{y}_l^{(3)}\right]_{10\text{x}1}$$

$$\mathbf{Z_1} = \mathbf{W_1 X} + \mathbf{b_1} \qquad \mathbf{V_1} = \text{Relu}(\mathbf{Z_1})$$
$$\mathbf{Z_2} = \mathbf{W_2 A_1} + \mathbf{b_2} \qquad \mathbf{V_2} = \text{Relu}(\mathbf{Z_2})$$
$$\mathbf{Z_3} = \mathbf{W_3 A_2} + \mathbf{b_3}$$

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{Z_3})$$

1.4 Backpropagation

The backpropagation algorithm is used to calculate the gradient of the error function with respect to the weights and biases of the neural network. The algorithm is given below

1. Calculate the output of the neural network for a given input using forward propagation
2. Calculate the error of the output
3. Calculate the gradient of the error function with respect to the weights and biases of the neural network
4. Update the weights and biases of the neural network

Derivative of Relu activation function

$$\frac{\partial \mathbf{R}(\mathbf{x})}{\partial \mathbf{x}} = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$$

Derivative of Softmax activation function

Since softmax is vector function which takes in vector and returns vector, we need to calculate the derivative of each element of the vector with respect to each element of the vector. Hence during backpropagation we have to use Jacobian matrix of Softmax function.

$$\left[\frac{\partial S(\mathbf{z})}{\partial \mathbf{z}}\right]_{ij} = \begin{cases} (1 - \text{Softmax}(z_i))\,\text{Softmax}(z_j) & \text{if} \quad i \neq j \\ -\text{Softmax}(z_i)\text{Softmax}(z_j) & \text{if} \quad i = j \end{cases}$$

The Jacobian matrix of Softmax function is given by

$$J_{\text{softmax}} = \begin{bmatrix} \frac{\partial S_1}{\partial z_1} & \frac{\partial S_1}{\partial z_2} & \cdots & \frac{\partial S_1}{\partial z_n} \\ \frac{\partial S_2}{\partial z_1} & \frac{\partial S_2}{\partial z_2} & \cdots & \frac{\partial S_2}{\partial z_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial S_n}{\partial z_1} & \frac{\partial S_n}{\partial z_2} & \cdots & \frac{\partial S_n}{\partial z_n} \end{bmatrix}$$

Proof:

S_i is the i^{th} element of the Softmax vector $\mathbf{S}$, z_j is the j^{th} element of the input vector $\mathbf{z}$.

$$\frac{\partial}{\partial z_j}\log(S_i) = \frac{1}{S_i}\frac{\partial S_i}{\partial z_j}$$

$$\frac{\partial S_i}{\partial z_j} = S_i \frac{\partial}{\partial z_j}\log(S_i)$$

$$\frac{\partial}{\partial z_j}\log(S_i) = \frac{\partial z_i}{\partial z_j} - \frac{\partial}{\partial z_i}\log\left(\sum_{l=1}^{n} e^{z_l}\right)$$

$$\text{where} \qquad \frac{\partial z_i}{\partial z_j} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

$$\text{here, Kronecker delta} \qquad \delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

$$\frac{\partial}{\partial z_j}\log(S_i) = \delta_{ij} - \frac{\partial}{\partial z_i}\frac{1}{\sum_{l=1}^{n} e^{z_l}}\left(\sum_{l=1}^{n} e^{z_l}\right)$$

$$\frac{\partial S_i}{\partial z_i} = S_i \frac{\partial}{\partial z_j}\log(S_i) = S_i\left(\delta_{ij} - S_j\right)$$

$$S'_{ij} = \begin{cases} -S_i S_j & \text{if} \quad i \neq j \\ S_i\left(1 - S_j\right) & \text{if} \quad i = j \end{cases}$$

Error function

Error function used is cross entropy error and is given by

$$E(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{l=1}^{m} y_l \log(\hat{y}_l) \qquad m \text{ is no of classes}$$

differentiating the error function with respect to $\hat{y}_i$

$$\frac{\partial E}{\partial \hat{y}_i} = -\frac{\partial}{\partial \hat{y}_i}\sum (y_l)\log(\hat{y}_l)$$

$$\left[\frac{\partial E}{\partial \hat{y}_i}\right]_{mx1} = -\frac{y_i}{\hat{y}_i}$$

$$\text{Entropy}'(\mathbf{y}, \hat{\mathbf{y}}) = \begin{bmatrix} \frac{y_1}{\hat{y}_1} \\ \\ \frac{y_2}{\hat{y}_2} \\ \vdots \\ \frac{y_m}{\hat{y}_m} \end{bmatrix}_{mx1}$$

Single layer backpropagation

For a single layer, backpropagation is given by fig 2.

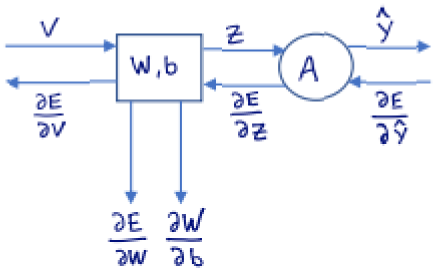

Figure 2: Single layer backpropagation

The derivative of the error function with respect to weights is given by

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial z_1}\frac{\partial z_1}{\partial w_{ij}} + \frac{\partial E}{\partial z_2}\frac{\partial z_2}{\partial w_{ij}} + \cdots + \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} \\ &= \frac{\partial E}{\partial z_j} v_i \end{aligned}$$

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial E}{\partial \mathbf{Z}} \cdot \mathbf{V}^T$$

The derivative of the error function with respect to biases is given by

$$\begin{aligned} \frac{\partial E}{\partial b_j} &= \frac{\partial E}{\partial z_1}\frac{\partial z_1}{\partial b_j} + \frac{\partial E}{\partial z_2}\frac{\partial z_2}{\partial b_j} + \cdots + \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial b_j} \\ &= \frac{\partial E}{\partial z_j} \end{aligned}$$

$$\frac{\partial E}{\partial \mathbf{b}} = \frac{\partial E}{\partial \mathbf{Z}}$$

The derivative of the error function with respect to inputs is given by

$$\begin{aligned}\frac{\partial E}{\partial v_i} &= \frac{\partial E}{\partial z_1}\frac{\partial z_1}{\partial v_i} + \frac{\partial E}{\partial z_2}\frac{\partial z_2}{\partial v_i} + \cdots + \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial v_i} \\ &= \frac{\partial E}{\partial z_1} w_{i1} + \frac{\partial E}{\partial z_2} w_{i2} + \cdots + \frac{\partial E}{\partial z_j} w_{ij}\end{aligned}$$

$$\frac{\partial E}{\partial \mathbf{V}} = \mathbf{W}^T \cdot \frac{\partial E}{\partial \mathbf{Z}}$$

The derivative of the error function with respect to the input for the activation function is given by

Incase of ReLU

$$\frac{\partial E}{\partial \mathbf{Z}} = \begin{bmatrix} \frac{\partial E}{\partial z_1}\frac{\partial z_1}{\partial v_1} & \frac{\partial E}{\partial z_2}\frac{\partial z_2}{\partial v_1} & \cdots & \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial v_1} \end{bmatrix}_{1\text{x}n}$$

$$= \begin{bmatrix} \frac{\partial E}{\partial z_1} R'(v_1) & \frac{\partial E}{\partial z_2} R'(v_1) & \cdots & \frac{\partial E}{\partial z_j} R'(v_1) \end{bmatrix}_{1\text{x}n}$$

$$= \frac{\partial E}{\partial \hat{\mathbf{y}}} \odot R'(\mathbf{Z})$$

Incase of Softmax

$$\begin{aligned}\frac{\partial E}{\partial z_i} &= \frac{\partial E}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial z_i} + \frac{\partial E}{\partial \hat{y}_2}\frac{\partial \hat{y}_2}{\partial z_i} + \cdots + \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_i} \\ &= \frac{\partial E}{\partial \hat{y}_1}(-S_1 S_i) + \frac{\partial E}{\partial \hat{y}_2}(-S_2 S_i) + \cdots + \frac{\partial E}{\partial \hat{y}_i}(S_i)(1 - S_i)\end{aligned}$$

$$\frac{\partial E}{\partial \mathbf{Z}} = S'(\mathbf{Z}) \cdot \frac{\partial E}{\partial \hat{\mathbf{y}}}$$

Summary:

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial E}{\partial \mathbf{Z}} \cdot \mathbf{V}^T$$

$$\frac{\partial E}{\partial \mathbf{b}} = \frac{\partial E}{\partial \mathbf{Z}}$$

$$\frac{\partial E}{\partial \mathbf{V}} = \mathbf{W}^T \cdot \frac{\partial E}{\partial \mathbf{Z}}$$

$$\frac{\partial E}{\partial \mathbf{Z}} = \frac{\partial E}{\partial \hat{\mathbf{y}}} \odot R'(\mathbf{Z})$$ - In case of Relu

$$\frac{\partial E}{\partial \mathbf{Z}} = S'(\mathbf{Z}) \cdot \frac{\partial E}{\partial \hat{\mathbf{y}}}$$ - In case of Softmax

Now the result of backpropagation of a layer is

1. used to update the weights and biases of the current layer.
2. used as input to backpropagate the previous layer(s).

Multi-layer backpropagation

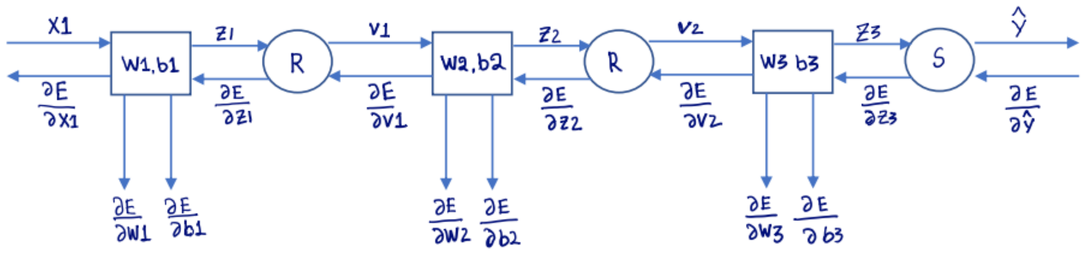

Figure 3: Backpropagation algorithm for multi-layer perceptron with 2 hidden layers

The backpropagation algorithm for a multi-layer perceptron with 2 hidden layers is given by fig 3.

$$\frac{\partial E}{\partial \hat{\mathbf{y}}} = \text{Entropy}'\left(\mathbf{y}, \hat{\mathbf{y}}\right)$$

$$\frac{\partial E}{\partial \mathbf{Z}_3} = S'\left(\mathbf{Z_3}\right) \frac{\partial E}{\partial \hat{\mathbf{y}}}$$

$$\frac{\partial E}{\partial \mathbf{W}_3} = \frac{\partial E}{\partial \mathbf{Z}_3} \cdot \mathbf{V}_2^T$$

$$\frac{\partial E}{\partial \mathbf{b}_3} = \frac{\partial E}{\partial \mathbf{Z}_3}$$

$$\frac{\partial E}{\partial \mathbf{V}_2} = \mathbf{W}_3^T \cdot \frac{\partial E}{\partial \mathbf{Z}_3} \qquad \frac{\partial E}{\partial \mathbf{V}_1} = \mathbf{W}_2^T \cdot \frac{\partial E}{\partial \mathbf{Z}_2}$$

$$\frac{\partial E}{\partial \mathbf{Z}_2} = \frac{\partial E}{\partial \mathbf{V}_2} \odot R'\left(\mathbf{Z}_2\right) \qquad \frac{\partial E}{\partial \mathbf{Z}_1} = \frac{\partial E}{\partial \mathbf{V}_1} \odot R'\left(\mathbf{Z}_1\right)$$

$$\frac{\partial E}{\partial \mathbf{W}_2} = \frac{\partial E}{\partial \mathbf{Z}_2} \cdot \mathbf{V}_1^T \qquad \frac{\partial E}{\partial \mathbf{W}_1} = \frac{\partial E}{\partial \mathbf{Z}_1} \cdot \mathbf{X}^T$$

$$\frac{\partial E}{\partial \mathbf{b}_2} = \frac{\partial E}{\partial \mathbf{Z}_2} \qquad \frac{\partial E}{\partial \mathbf{b}_1} = \frac{\partial E}{\partial \mathbf{Z}_1}$$

$$\mathbf{W}_1 = \mathbf{W}_1 - \eta \frac{\partial E}{\partial \mathbf{W}_1} \qquad \mathbf{b}_1 = \mathbf{b}_1 - \eta \frac{\partial E}{\partial \mathbf{b}_1}$$

$$\mathbf{W}_2 = \mathbf{W}_2 - \eta \frac{\partial E}{\partial \mathbf{W}_2} \qquad \mathbf{b}_2 = \mathbf{b}_2 - \eta \frac{\partial E}{\partial \mathbf{b}_2}$$

$$\mathbf{W}_3 = \mathbf{W}_3 - \eta \frac{\partial E}{\partial \mathbf{W}_3} \qquad \mathbf{b}_3 = \mathbf{b}_3 - \eta \frac{\partial E}{\partial \mathbf{b}_3}$$

where
S' is the derivative of the softmax function (Jacobian of Softmax function)
R' is the derivative of the ReLU function
$\odot$ is the element-wise multiplication and
η is the learning rate.

In Back propagation, derivatives of error funciton are calculate with respect to weights and baises. Weights and baises are updated using gradient discent. The MLP model is trained with multiple learning rates.

2 Results and Discussion

2.1 Image trasformation

We have taken following image as an example to demonstrate the image trasformation.

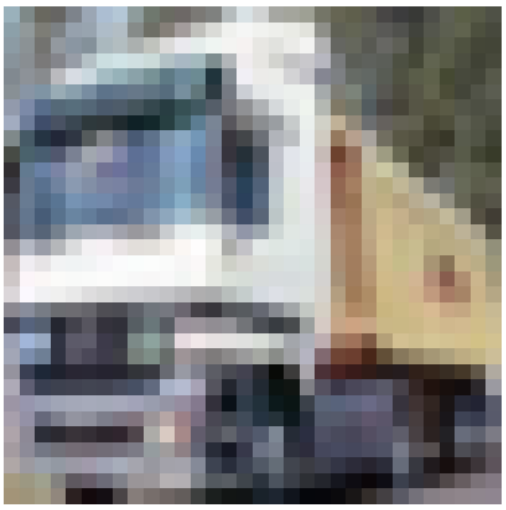

Figure 4: Sample image

Image transformation done on the sample image are shown below.

1. Enhancing the Image

2. Posterizing the image

3. Random Rotation of the image

4. Contrast and Horizontal Flip

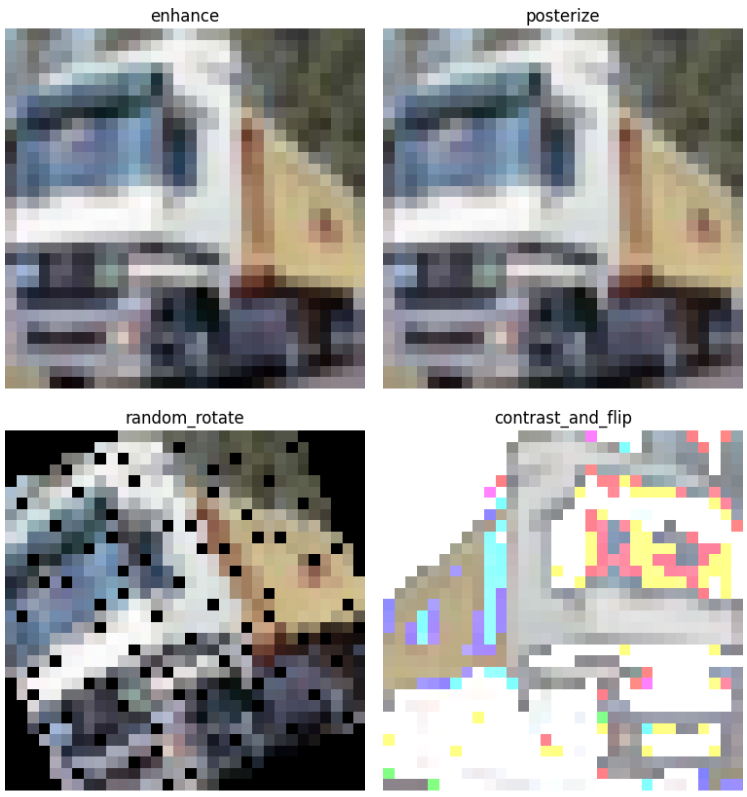

Figure 5: Image transformations on sample image

2.2 Model Training

Hyperparameters in MLP

Epochs and Learning rate are some of the hyper parameter that needs to be choosen for training the model. Initially, with learning rate = 0.001 the model is trained with 25 epochs and 50 epochs. It is observed that the error is continously decreasing. Hence, epochs is increased to 100. Now, error graph is showing small negative slope (almost constant) at 100th epoch. As epochs increases training time increase, hence epoch is limited to 100 and learning is changed.

With 100 epochs, the model is trained with learning rates = $\{0.001, 0.01, 0.1\}$ and error trends are observed.

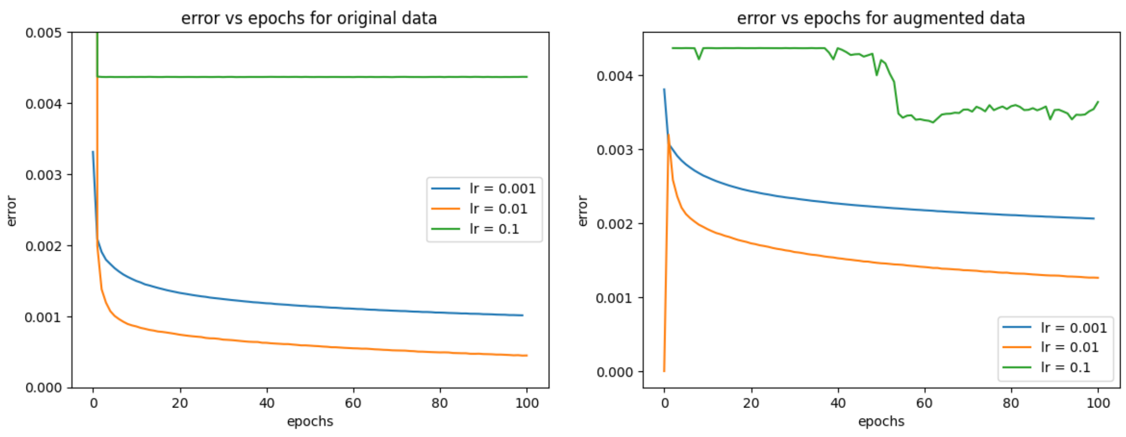

Figure 6: Training with different learning rates

With epochs = 100, it can be observed that with learning rate = 0.01 minimum error is obtained. If learning rate is taken more than 0.01 the error is increasing. Hence, Learning rate is choosen to be 0.01.

Performance Evaluation

The acuracy performance of all machine laearning models on train and test data are shown in the figure below.

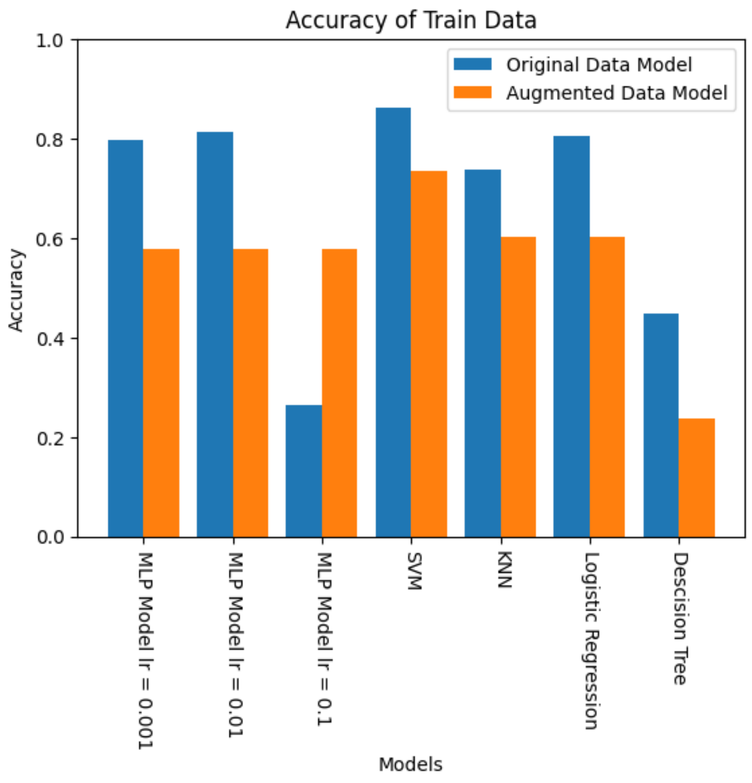

Figure 7: Perfoemance evaluation on train data

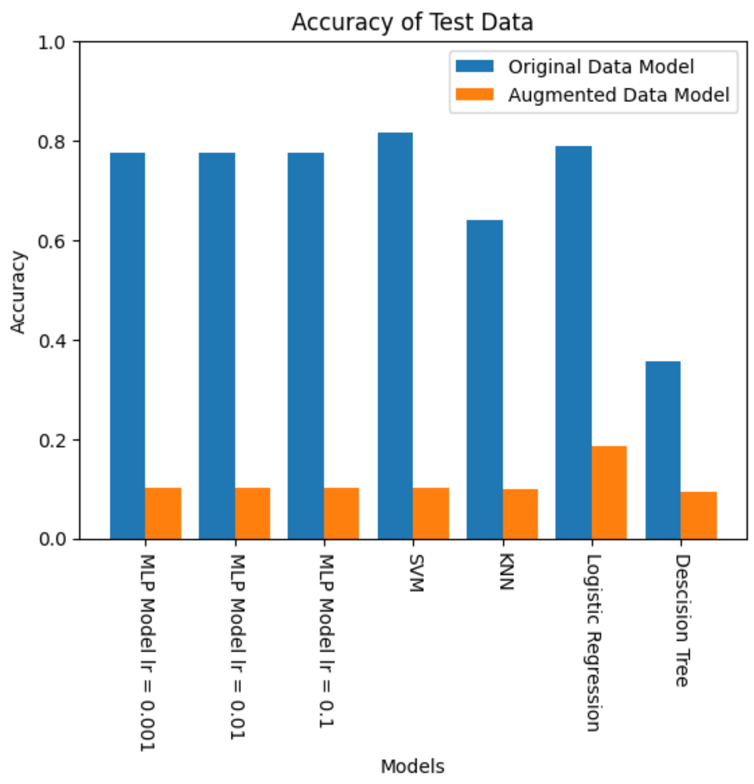

Figure 8: Perfoemance evaluation on test data

The accuracies of all the models are shown in the table 1.

	Train Data		Test Data	
	Original	Augmented	Original	Augmented
MLP lr=0.001	0.7972	0.5780	0.7761	0.1016
MLP lr=0.01	0.8125	0.5780	0.7761	0.1016
MLP lr=0.1	0.2653	0.5780	0.7761	0.1016
SVM	0.8631	0.7336	0.8149	0.1024
KNN	0.7369	0.6032	0.6414	0.1000
Logistic Regression	0.8065	0.6022	0.7888	0.0187
Decision Tree	0.4486	0.2364	0.3554	0.0940

Table 1: Accuracy of different models on train and test data

Conclusion

SVM model has the highest accuracy on train and test data. MLP model has the second highest accuracy on train data. Decision tree model has the lowest accuracy on test data. The models trained on augmented data has lower accuracy than the models trained on original data. This may be because, some of the relations between the features may be lost in the image augmentation process. the technique used for image augmentation may not be sufficient to increase the accuracy of the model.

References

[1] Derivative of the Softmax Function and the Categorical Cross-Entropy Loss

[2] Neural Network from scratch in Python