Assignment 1 Report

Indian Institute of Technology, Kanpur
Department of Computer Science
CS776 Deep Learning for Computer Vision
Instructor: Priyanka Bagade

Submitted: Februrary 3, 2023 Submitted by :

Chunduri Naga Venkata Hrushikesh
22111280

Gnanendra Sri Phani Sai Channamsetty
22105032

Preprint. Under review.

Contents

1 Multilayer Perceptron that has been implemented

1.1 Model Description

Implemented MLP, shown in next subsection below, consists of

1) Input layer has 512 neurons, whose input is of dimensions (512 x 50000) in case of original data set i.e $m = 50000$ and (512 x 100000) in case of augmented data set i.e $m = 100000$ where $m =$ Number of training examples
2) Two hidden layers, each with 64 neurons .
3) Output layer has 10 neurons each, representing 10 types of image classes of CIFAR-10 Data Set .

1.2 MLP Diagram

The following figure has been made by us, using LaTeX Code

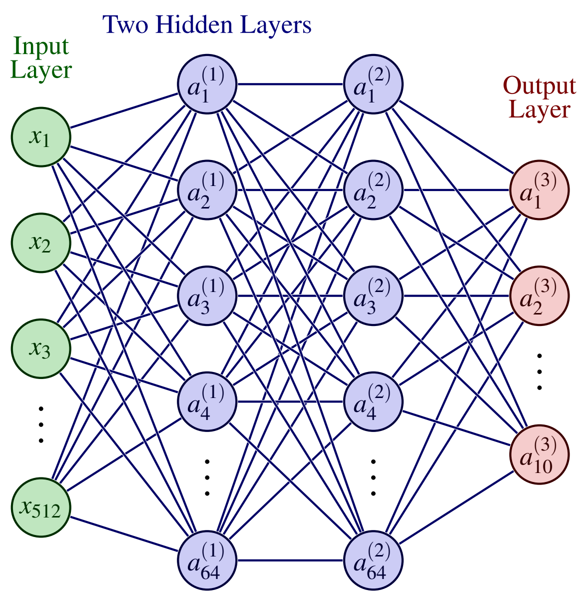

Figure 1: Implemented MLP Model

1.3 MLP Details

Table 1: MLP Details

Layer Name () represents code	**No of Neurons** $n^{[l]}$	**Weight Dims** $W^{[l]}$	**Bias Dims** $b^{[l]}$	**Activation Function(Dims)** $A^{[l]}$(m =No of Training Examples)
Input Layer (0^{th})	512	-	-	-
Hidden Layer 1(1^{st})	64	64 x 512	64 x 1	RELU(64 x m)
Hidden Layer 2(2^{nd})	64	64 x 64	64 x 1	RELU(64 x m)
Output Layer(3^{rd})	10	10 x 64	10 x 1	Softmax(10 x m)

1.4 MLP Model Procedure

It consists of 5 following steps :

1) **Initialization Step** : Weights have been initialized with small random numbers and bias with zeros. This step has been represented in code using "" Function

2) **Forward Propagation Step** : Calculate the linear part of a layer's forward propagation step (resulting in $Z^{[l]}$ in code) . Then apply Activation Function RELU on the previous resultant for first two layers and then apply Activation Function Softmax for the last output layer (resulting in $A^{[l]}$ in code) . At every step of forward function, some values are stored in a cache called "". These stored values are useful for computing gradients in step 4 .This step has been represented in code using "" Function representing the following equations.

$$z_{j,i}^{[l]} = \sum_k w_{j,k}^{[l]} a_{k,i}^{[l-1]} + b_j^{[l]}, \tag{1}$$

$$a_{j,i}^{[l]} = g_j^{[l]}(z_{1,i}^{[l]}, \ldots, z_{j,i}^{[l]}, \ldots, z_{n^{[l]},i}^{[l]}). \tag{2}$$

Using numpy,Above two equations can be vectorized as following

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

where $A^{[0]} = X$.

$$A^{[l]} = g(Z^{[l]}) = g(W^{[l]} A^{[l-1]} + b^{[l]})$$

where $g(Z^{[l]})$ could be RELU or Softmax depending on layer number.

3) **Estimating Overall Cost Function Step** : Cross Entropy has been selected as Cost Function to find deviation of prediction from actual label. This step has been represented in code using "" using the following formula:

$$J = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log\left(a^{[L](i)}\right) + (1 - y^{(i)}) \log\left(1 - a^{[L](i)}\right))$$

4) **Backward Propagation Step** : The following equations have been used in code. Their derivations are in next section.

$$dW^{[l]} = \frac{\partial J}{\partial W^{[l]}} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{\partial J}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)}$$

$$dA^{[l-1]} = \frac{\partial J}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]}$$

$$dZ^{[l]} = dA^{[l]} * g'(Z^{[l]})$$

$$dAL = \frac{\partial J}{\partial a^{[L]}_{j,i}} = \frac{1}{m} \Big(\frac{1 - y_{j,i}}{1 - a^{[L]}_{j,i}} - \frac{y_{j,i}}{a^{[L]}_{j,i}} \Big),$$

5) **Update Expressions** : Gradient Descent algorithm is being used to update expressions using following equations :

$$W^{[l]} = W^{[l]} - \alpha \, dW^{[l]}$$
$$b^{[l]} = b^{[l]} - \alpha \, db^{[l]}$$

where α is the learning rate.

2 Derivations

Chain Rule for derivatives can be given by :

$$\frac{\partial y_k}{\partial x_j} = \sum_i \frac{\partial y_k}{\partial u_i} \frac{\partial u_i}{\partial x_j}. \tag{3}$$

Additional Notation :

$$\vec{A}^{[0]} = X_{input} \tag{4}$$
$$\vec{A}^{[L]} = Y_{predicted} \tag{5}$$

Writing equations (1) and (2) in matrix form :

$$\begin{bmatrix} z^{[l]}_{1,i} \\ \vdots \\ z^{[l]}_{j,i} \\ \vdots \\ z^{[l]}_{n^{[l]},i} \end{bmatrix} = \begin{bmatrix} w^{[l]}_{1,1} & \cdots & w^{[l]}_{1,k} & \cdots & w^{[l]}_{1,n^{[l-1]}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w^{[l]}_{j,1} & \cdots & w^{[l]}_{j,k} & \cdots & w^{[l]}_{j,n^{[l-1]}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w^{[l]}_{n^{[l]},1} & \cdots & w^{[l]}_{n^{[l]},k} & \cdots & w^{[l]}_{n^{[l]},n^{[l-1]}} \end{bmatrix} \begin{bmatrix} a^{[l-1]}_{1,i} \\ \vdots \\ a^{[l-1]}_{k,i} \\ \vdots \\ a^{[l-1]}_{n^{[l-1]},i} \end{bmatrix} + \begin{bmatrix} b^{[l]}_{1} \\ \vdots \\ b^{[l]}_{j} \\ \vdots \\ b^{[l]}_{n^{[l]}} \end{bmatrix},$$

$$\begin{bmatrix} a^{[l]}_{1,i} \\ \vdots \\ a^{[l]}_{j,i} \\ \vdots \\ a^{[l]}_{n^{[l]},i} \end{bmatrix} = \begin{bmatrix} g^{[l]}_{1}(z^{[l]}_{1,i}, \ldots, z^{[l]}_{j,i}, \ldots, z^{[l]}_{n^{[l]},i}) \\ \vdots \\ g^{[l]}_{j}(z^{[l]}_{1,i}, \ldots, z^{[l]}_{j,i}, \ldots, z^{[l]}_{n^{[l]},i}) \\ \vdots \\ g^{[l]}_{n^{[l]}}(z^{[l]}_{1,i}, \ldots, z^{[l]}_{j,i}, \ldots, z^{[l]}_{n^{[l]},i}) \end{bmatrix},$$

2.1 Activation Functions Derivations

2.1.1 ReLU Function

$$
\begin{aligned}
a_{j,i}^{[l]} &= g_j^{[l]}(z_{1,i}^{[l]},\ldots,z_{j,i}^{[l]},\ldots,z_{n^{[l]},i}^{[l]}) \\
&= \max(0, z_{j,i}^{[l]}) \\
&= \begin{cases} z_{j,i}^{[l]} & \text{if } z_{j,i}^{[l]} > 0, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

After vectorization with numpy arrays, RELU function becomes

$$\vec{A}^{[l]} = \max(0, \vec{Z}^{[l]}). \tag{6}$$

2.1.2 Derivative of ReLU Function

$$
\begin{aligned}
\frac{\partial a_{j,i}^{[l]}}{\partial z_{j,i}^{[l]}} &= \begin{cases} 1 & \text{if } z_{j,i}^{[l]} > 0, \\ 0 & \text{otherwise,} \end{cases} \\
&= I(z_{j,i}^{[l]} > 0), \\
\frac{\partial a_{p,i}^{[l]}}{\partial z_{j,i}^{[l]}} &= 0, \quad \forall p \neq j.
\end{aligned}
$$

Using above, we can differentiate Cost J w.r.t $z_{j,i}^{[l]}$

$$
\begin{aligned}
\frac{\partial J}{\partial z_{j,i}^{[l]}} &= \sum_p \frac{\partial J}{\partial a_{p,i}^{[l]}} \frac{\partial a_{p,i}^{[l]}}{\partial z_{j,i}^{[l]}} \\
&= \frac{\partial J}{\partial a_{j,i}^{[l]}} \frac{\partial a_{j,i}^{[l]}}{\partial z_{j,i}^{[l]}} + \sum_{p \neq j} \frac{\partial J}{\partial a_{p,i}^{[l]}} \frac{\partial a_{p,i}^{[l]}}{\partial z_{j,i}^{[l]}} \\
&= \frac{\partial J}{\partial a_{j,i}^{[l]}} I(z_{j,i}^{[l]} > 0),
\end{aligned}
$$

2.1.3 Softmax Function

$$
\begin{aligned}
a_{j,i}^{[l]} &= g_j^{[l]}(z_{1,i}^{[l]},\ldots,z_{j,i}^{[l]},\ldots,z_{n^{[l]},i}^{[l]}) \\
&= \frac{\exp(z_{j,i}^{[l]})}{\sum_p \exp(z_{p,i}^{[l]})}.
\end{aligned}
$$

2.1.4 Derivative of Softmax Function

After constructing a computation graph, for j^{th} activation of present layer

$$
\begin{aligned}
u_{-1} &= z^{[l]}_{j,i}, \\
u_{0,p} &= z^{[l]}_{p,i}, && \forall p \neq j, \\
u_1 &= \exp(u_{-1}), \\
u_{2,p} &= \exp(u_{0,p}), && \forall p \neq j, \\
u_3 &= u_1 + \sum_{p \neq j} u_{2,p}, \\
u_4 &= \frac{1}{u_3}, \\
u_5 &= u_1 u_4 = a^{[l]}_{j,i}.
\end{aligned}
$$

After applying chain rule,

$$
\begin{aligned}
\frac{\partial a^{[l]}_{j,i}}{\partial u_5} &= 1, \\
\frac{\partial a^{[l]}_{j,i}}{\partial u_4} &= \frac{\partial a^{[l]}_{j,i}}{\partial u_5}\frac{\partial u_5}{\partial u_4} = u_1 = \exp(z^{[l]}_{j,i}), \\
\frac{\partial a^{[l]}_{j,i}}{\partial u_3} &= \frac{\partial a^{[l]}_{j,i}}{\partial u_4}\frac{\partial u_4}{\partial u_3} = -u_1 \frac{1}{u_3^2} = -\frac{\exp(z^{[l]}_{j,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2}, \\
\frac{\partial a^{[l]}_{j,i}}{\partial u_1} &= \frac{\partial a^{[l]}_{j,i}}{\partial u_3}\frac{\partial u_3}{\partial u_1} + \frac{\partial a^{[l]}_{j,i}}{\partial u_5}\frac{\partial u_5}{\partial u_1} \\
&= -u_1 \frac{1}{u_3^2} + u_4 \\
&= -\frac{\exp(z^{[l]}_{j,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2} + \frac{1}{\sum_p \exp(z^{[l]}_{p,i})}, \\
\frac{\partial a^{[l]}_{j,i}}{\partial u_{-1}} &= \frac{\partial a^{[l]}_{j,i}}{\partial u_1}\frac{\partial u_1}{\partial u_{-1}} \\
&= \left(-u_1 \frac{1}{u_3^2} + u_4\right) \exp(u_{-1}) \\
&= -\frac{\exp(z^{[l]}_{j,i})^2}{(\sum_p \exp(z^{[l]}_{p,i}))^2} + \frac{\exp(z^{[l]}_{j,i})}{\sum_p \exp(z^{[l]}_{p,i})}.
\end{aligned}
$$

But $z^{[l]}_{j,i}$ also affects other activations in the same layer. So, taking derivative for that :

$$
\begin{aligned}
u_{-1} &= z^{[l]}_{j,i}, \\
u_{0,p} &= z^{[l]}_{p,i}, && \forall p \neq j, \\
u_1 &= \exp(u_{-1}), \\
u_{2,p} &= \exp(u_{0,p}), && \forall p \neq j, \\
u_3 &= u_1 + \sum_{p \neq j} u_{2,p}, \\
u_4 &= \frac{1}{u_3}, \\
u_5 &= u_{2,p} u_4 = a^{[l]}_{p,i}, && \forall p \neq j.
\end{aligned}
$$

Applying backward propagation :

$$\begin{aligned}
\frac{\partial a^{[l]}_{p,i}}{\partial u_5} &= 1,\\
\frac{\partial a^{[l]}_{p,i}}{\partial u_4} &= \frac{\partial a^{[l]}_{p,i}}{\partial u_5}\frac{\partial u_5}{\partial u_4} = u_{2,p} = \exp(z^{[l]}_{p,i}),\\
\frac{\partial a^{[l]}_{p,i}}{\partial u_3} &= \frac{\partial a^{[l]}_{p,i}}{\partial u_4}\frac{\partial u_4}{\partial u_3} = -u_{2,p}\frac{1}{u_3^2} = -\frac{\exp(z^{[l]}_{p,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2},\\
\frac{\partial a^{[l]}_{p,i}}{\partial u_1} &= \frac{\partial a^{[l]}_{p,i}}{\partial u_3}\frac{\partial u_3}{\partial u_1} = -u_{2,p}\frac{1}{u_3^2} = -\frac{\exp(z^{[l]}_{p,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2},\\
\frac{\partial a^{[l]}_{p,i}}{\partial u_{-1}} &= \frac{\partial a^{[l]}_{p,i}}{\partial u_1}\frac{\partial u_1}{\partial u_{-1}} = -u_{2,p}\frac{1}{u_3^2}\exp(u_{-1}) = -\frac{\exp(z^{[l]}_{p,i})\exp(z^{[l]}_{j,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2}.
\end{aligned}$$

From above, we know

$$\begin{aligned}
\frac{\partial a^{[l]}_{j,i}}{\partial z^{[l]}_{j,i}} &= -\frac{\exp(z^{[l]}_{j,i})^2}{(\sum_p \exp(z^{[l]}_{p,i}))^2} + \frac{\exp(z^{[l]}_{j,i})}{\sum_p \exp(z^{[l]}_{p,i})}\\
&= a^{[l]}_{j,i}(1-a^{[l]}_{j,i}),\\
\frac{\partial a^{[l]}_{p,i}}{\partial z^{[l]}_{j,i}} &= -\frac{\exp(z^{[l]}_{p,i})\exp(z^{[l]}_{j,i})}{(\sum_p \exp(z^{[l]}_{p,i}))^2}\\
&= -a^{[l]}_{p,i}a^{[l]}_{j,i}, \quad \forall p \neq j.
\end{aligned}$$

Using above, we can differentiate Cost J w.r.t $z^{[l]}_{j,i}$

$$\begin{aligned}
\frac{\partial J}{\partial z^{[l]}_{j,i}} &= \sum_p \frac{\partial J}{\partial a^{[l]}_{p,i}}\frac{\partial a^{[l]}_{p,i}}{\partial z^{[l]}_{j,i}}\\
&= \frac{\partial J}{\partial a^{[l]}_{j,i}}\frac{\partial a^{[l]}_{j,i}}{\partial z^{[l]}_{j,i}} + \sum_{p\neq j}\frac{\partial J}{\partial a^{[l]}_{p,i}}\frac{\partial a^{[l]}_{p,i}}{\partial z^{[l]}_{j,i}}\\
&= \frac{\partial J}{\partial a^{[l]}_{j,i}}a^{[l]}_{j,i}(1-a^{[l]}_{j,i}) - \sum_{p\neq j}\frac{\partial J}{\partial a^{[l]}_{p,i}}a^{[l]}_{p,i}a^{[l]}_{j,i}\\
&= a^{[l]}_{j,i}\Big(\frac{\partial J}{\partial a^{[l]}_{j,i}}(1-a^{[l]}_{j,i}) - \sum_{p\neq j}\frac{\partial J}{\partial a^{[l]}_{p,i}}a^{[l]}_{p,i}\Big)\\
&= a^{[l]}_{j,i}\Big(\frac{\partial J}{\partial a^{[l]}_{j,i}}(1-a^{[l]}_{j,i}) - \sum_{p}\frac{\partial J}{\partial a^{[l]}_{p,i}}a^{[l]}_{p,i} + \frac{\partial J}{\partial a^{[l]}_{j,i}}a^{[l]}_{j,i}\Big)\\
&= a^{[l]}_{j,i}\Big(\frac{\partial J}{\partial a^{[l]}_{j,i}} - \sum_{p}\frac{\partial J}{\partial a^{[l]}_{p,i}}a^{[l]}_{p,i}\Big),
\end{aligned}$$

We can use numpy to vectorize above as

$$\frac{\partial J}{\partial \vec{z}^{[l]}_{:,i}} = \vec{a}^{[l]}_{:,i} \odot \Big(\frac{\partial J}{\partial \vec{a}^{[l]}_{:,i}} - \underbrace{\vec{a}^{[l]}_{:,i}{}^{\frac{\partial J}{\partial \vec{a}^{[l]}_{:,i}}}}_{\text{scalar}}\Big).$$

where $\odot$ = Element Wise Multiplication

2.2 Cost Function Derivative for initial step of Backpropagation

Cross Entropy Cost Function "J" is given by

$$\begin{aligned}
J &= f(\vec{\hat{Y}}, \vec{Y}) = f(\vec{A}^{[L]}, \vec{Y}) \\
&= \sum_j \Big(-\frac{1}{m} \sum_i (y_{j,i} \log(\hat{y}_{j,i}) + (1-y_{j,i}) \log(1-\hat{y}_{j,i})) \Big) \\
&= \sum_j \Big(-\frac{1}{m} \sum_i (y_{j,i} \log(a^{[L]}_{j,i}) + (1-y_{j,i}) \log(1-a^{[L]}_{j,i})) \Big),
\end{aligned}$$

where $j = 1, \ldots, n^{[L]}$

After making computation graph,

$$\begin{aligned}
u_{0,j,i} &= a^{[L]}_{j,i}, \\
u_{1,j,i} &= 1 - u_{0,j,i}, \\
u_{2,j,i} &= \log(u_{0,j,i}), \\
u_{3,j,i} &= \log(u_{1,j,i}), \\
u_{4,j,i} &= y_{j,i} u_{2,j,i} + (1-y_{j,i}) u_{3,j,i}, \\
u_{5,j} &= -\frac{1}{m} \sum_i u_{4,j,i}, \\
u_6 &= \sum_j u_{5,j} = J.
\end{aligned}$$

Using above, finding partial derivatives :

$$\begin{aligned}
\frac{\partial J}{\partial u_6} &= 1, \\
\frac{\partial J}{\partial u_{5,j}} &= \frac{\partial J}{\partial u_6} \frac{\partial u_6}{\partial u_{5,j}} = 1, \\
\frac{\partial J}{\partial u_{4,j,i}} &= \frac{\partial J}{\partial u_{5,j}} \frac{\partial u_{5,j}}{\partial u_{4,j,i}} = -\frac{1}{m}, \\
\frac{\partial J}{\partial u_{3,j,i}} &= \frac{\partial J}{\partial u_{4,j,i}} \frac{\partial u_{4,j,i}}{\partial u_{3,j,i}} = -\frac{1}{m}(1-y_{j,i}), \\
\frac{\partial J}{\partial u_{2,j,i}} &= \frac{\partial J}{\partial u_{4,j,i}} \frac{\partial u_{4,j,i}}{\partial u_{2,j,i}} = -\frac{1}{m} y_{j,i}, \\
\frac{\partial J}{\partial u_{1,j,i}} &= \frac{\partial J}{\partial u_{3,j,i}} \frac{\partial u_{3,j,i}}{\partial u_{1,j,i}} = -\frac{1}{m}(1-y_{j,i}) \frac{1}{u_{1,j,i}} = -\frac{1}{m} \frac{1-y_{j,i}}{1-a^{[L]}_{j,i}}, \\
\frac{\partial J}{\partial u_{0,j,i}} &= \frac{\partial J}{\partial u_{1,j,i}} \frac{\partial u_{1,j,i}}{\partial u_{0,j,i}} + \frac{\partial J}{\partial u_{2,j,i}} \frac{\partial u_{2,j,i}}{\partial u_{0,j,i}} \\
&= \frac{1}{m}(1-y_{j,i}) \frac{1}{u_{1,j,i}} - \frac{1}{m} y_{j,i} \frac{1}{u_{0,j,i}} \\
&= \frac{1}{m} \Big(\frac{1-y_{j,i}}{1-a^{[L]}_{j,i}} - \frac{y_{j,i}}{a^{[L]}_{j,i}} \Big).
\end{aligned}$$

Summarising above as :

$$\frac{\partial J}{\partial a^{[L]}_{j,i}} = \frac{1}{m} \Big(\frac{1-y_{j,i}}{1-a^{[L]}_{j,i}} - \frac{y_{j,i}}{a^{[L]}_{j,i}} \Big),$$

Using numpy, Vectorising previous equations as :

$$\frac{\partial J}{\partial \vec{A}^{[L]}} = \frac{1}{m} \Big(\frac{1}{1-\vec{A}^{[L]}} \odot (1-\vec{Y}) - \frac{1}{\vec{A}^{[L]}} \odot \vec{Y} \Big). \tag{7}$$

Now finding derivative of cost function w.r.t $a^{[L]}_{j,i}$

$$
\begin{aligned}
\frac{\partial J}{\partial z^{[L]}_{j,i}} &= \frac{\partial J}{\partial a^{[L]}_{j,i}} a^{[L]}_{j,i}(1-a^{[L]}_{j,i}) \\
&= \frac{1}{m}\Big(\frac{1-y_{j,i}}{1-a^{[L]}_{j,i}} - \frac{y_{j,i}}{a^{[L]}_{j,i}}\Big) a^{[L]}_{j,i}(1-a^{[L]}_{j,i}) \\
&= \frac{1}{m}((1-y_{j,i})a^{[L]}_{j,i} - y_{j,i}(1-a^{[L]}_{j,i})) \\
&= \frac{1}{m}(a^{[L]}_{j,i} - y_{j,i}),
\end{aligned}
$$

Vectorizing above equation with numpy as

$$
\frac{\partial J}{\partial \vec{Z}^{[L]}} = \frac{1}{m}(\vec{A}^{[L]} - \vec{Y}). \tag{8}
$$

2.3 Back Propagation Derivation

Applying Chain Rule (3)

$$
\frac{\partial J}{\partial w^{[l]}_{j,k}} = \sum_i \frac{\partial J}{\partial z^{[l]}_{j,i}} \frac{\partial z^{[l]}_{j,i}}{\partial w^{[l]}_{j,k}} = \sum_i \frac{\partial J}{\partial z^{[l]}_{j,i}} a^{[l-1]}_{k,i}, \tag{9}
$$

$$
\frac{\partial J}{\partial b^{[l]}_{j}} = \sum_i \frac{\partial J}{\partial z^{[l]}_{j,i}} \frac{\partial z^{[l]}_{j,i}}{\partial b^{[l]}_{j}} = \sum_i \frac{\partial J}{\partial z^{[l]}_{j,i}}. \tag{10}
$$

Vectorization of above two equations results in :

$$
\begin{bmatrix}
Jw^{[l]}_{1,1} & \dots & Jw^{[l]}_{1,k} & \dots & Jw^{[l]}_{1,n^{[l-1]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
Jw^{[l]}_{j,1} & \dots & Jw^{[l]}_{j,k} & \dots & Jw^{[l]}_{j,n^{[l-1]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
Jw^{[l]}_{n^{[l]},1} & \dots & Jw^{[l]}_{n^{[l]},k} & \dots & Jw^{[l]}_{n^{[l]},n^{[l-1]}}
\end{bmatrix}
=
\begin{bmatrix}
Jz^{[l]}_{1,1} & \dots & Jz^{[l]}_{1,i} & \dots & Jz^{[l]}_{1,m} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
Jz^{[l]}_{j,1} & \dots & Jz^{[l]}_{j,i} & \dots & Jz^{[l]}_{j,m} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
Jz^{[l]}_{n^{[l]},1} & \dots & Jz^{[l]}_{n^{[l]},i} & \dots & Jz^{[l]}_{n^{[l]},m}
\end{bmatrix}
\cdot
\begin{bmatrix}
a^{[l-1]}_{1,1} & \dots & a^{[l-1]}_{k,1} & \dots & a^{[l-1]}_{n^{[l-1]},1} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
a^{[l-1]}_{1,i} & \dots & a^{[l-1]}_{k,i} & \dots & a^{[l-1]}_{n^{[l-1]},i} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
a^{[l-1]}_{1,m} & \dots & a^{[l-1]}_{k,m} & \dots & a^{[l-1]}_{n^{[l-1]},m}
\end{bmatrix},
$$

$$
\begin{bmatrix}
Jb^{[l]}_{1} \\ \vdots \\ Jb^{[l]}_{j} \\ \vdots \\ Jb^{[l]}_{n^{[l]}}
\end{bmatrix}
=
\begin{bmatrix}
Jz^{[l]}_{1,1} \\ \vdots \\ Jz^{[l]}_{j,1} \\ \vdots \\ Jz^{[l]}_{n^{[l]},1}
\end{bmatrix}
+\dots+
\begin{bmatrix}
Jz^{[l]}_{1,i} \\ \vdots \\ Jz^{[l]}_{j,i} \\ \vdots \\ Jz^{[l]}_{n^{[l]},i}
\end{bmatrix}
+\dots+
\begin{bmatrix}
Jz^{[l]}_{1,m} \\ \vdots \\ Jz^{[l]}_{j,m} \\ \vdots \\ Jz^{[l]}_{n^{[l]},m}
\end{bmatrix},
$$

Preceding matrix equations can be compressed into :

$$
\frac{\partial J}{\partial \vec{W}^{[l]}} = \sum_i \frac{\partial J}{\partial \vec{z}^{[l]}_{:,i}} \vec{a}^{[l-1]}_{:,i} = \frac{\partial J}{\partial \vec{Z}^{[l]}} \vec{A}^{[l-1]}, \tag{11}
$$

$$
\frac{\partial J}{\partial \vec{b}^{[l]}} = \sum_i \frac{\partial J}{\partial \vec{z}^{[l]}_{:,i}} = \underbrace{\sum_{\text{axis}=1} \frac{\partial J}{\partial \vec{Z}^{[l]}}}_{\text{column vector}}, \tag{12}
$$

where dimensions are :

$\partial J \frac{}{\partial \vec{z}^{[l]}_{:,i}} \in^{n^{[l]}}, \frac{\partial J}{\partial \vec{Z}^{[l]}} \in^{n^{[l]} \times m}, \frac{\partial J}{\partial \vec{W}^{[l]}} \in^{n^{[l]} \times n^{[l-1]}}$ $and \frac{\partial J}{\partial \vec{b}^{[l]}} \in R^{n^{[l]}}$

The only unknown partial derivative can be found through Chain Rule (3) :

$$
\frac{\partial J}{\partial z^{[l]}_{j,i}} = \sum_p \frac{\partial J}{\partial a^{[l]}_{p,i}} \frac{\partial a^{[l]}_{p,i}}{\partial z^{[l]}_{j,i}}, \tag{13}
$$

where $p = 1, \dots, n^{[l]}$

After vectorising previous equation :

$$\begin{bmatrix} Jz^{[l]}_{1,i} \\ \vdots \\ Jz^{[l]}_{j,i} \\ \vdots \\ Jz^{[l]}_{n^{[l]},i} \end{bmatrix} = \begin{bmatrix} a^{[l]}_{1,i}z^{[l]}_{1,i} & \dots & a^{[l]}_{j,i}z^{[l]}_{1,i} & \dots & a^{[l]}_{n^{[l]},i}z^{[l]}_{1,i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a^{[l]}_{1,i}z^{[l]}_{j,i} & \dots & a^{[l]}_{j,i}z^{[l]}_{j,i} & \dots & a^{[l]}_{n^{[l]},i}z^{[l]}_{j,i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a^{[l]}_{1,i}z^{[l]}_{n^{[l]},i} & \dots & a^{[l]}_{j,i}z^{[l]}_{n^{[l]},i} & \dots & a^{[l]}_{n^{[l]},i}z^{[l]}_{n^{[l]},i} \end{bmatrix} \begin{bmatrix} Ja^{[l]}_{1,i} \\ \vdots \\ Ja^{[l]}_{j,i} \\ \vdots \\ Ja^{[l]}_{n^{[l]},i} \end{bmatrix},$$

which compresses into

$$\frac{\partial J}{\partial \vec{z}^{[l]}_{:,i}} = \frac{\partial \vec{a}^{[l]}_{:,i}}{\partial \vec{z}^{[l]}_{:,i}} \frac{\partial J}{\partial \vec{a}^{[l]}_{:,i}}, \tag{14}$$

where $\frac{\partial J}{\partial \vec{a}^{[l]}_{:,i}} \in R^{n^{[l]}}$ and $\frac{\partial \vec{a}^{[l]}_{:,i}}{\partial z^{[l]}_{:,i}} \in R^{n^{[l]} \times n^{[l]}}$

We already know from previous equations that :

$$\frac{\partial J}{\partial \vec{Z}^{[l]}} = \begin{bmatrix} J\vec{z}^{[l]}_{:,1} & \dots & J\vec{z}^{[l]}_{:,i} & \dots & J\vec{z}^{[l]}_{:,m} \end{bmatrix}, \tag{15}$$

$$\frac{\partial J}{\partial \vec{A}^{[l]}} = \begin{bmatrix} J\vec{a}^{[l]}_{:,1} & \dots & J\vec{a}^{[l]}_{:,i} & \dots & J\vec{a}^{[l]}_{:,m} \end{bmatrix}, \tag{16}$$

where : $\frac{\partial J}{\partial \vec{A}^{[l]}} \in R^{n^{[l]} \times m}$

$\frac{\partial a^{[l]}_{j,i}}{\partial z^{[l]}_{j,i}}$ has already been derived previously in page 8.

Remaining $\frac{\partial J}{\partial z^{[l]}_{j,i}}$ is dependant on $\frac{\partial J}{\partial a^{[l]}_{j,i}}$ which is computed as follows :

$$\frac{\partial J}{\partial a^{[l-1]}_{k,i}} = \sum_j \frac{\partial J}{\partial z^{[l]}_{j,i}} \frac{\partial z^{[l]}_{j,i}}{\partial a^{[l-1]}_{k,i}} = \sum_j \frac{\partial J}{\partial z^{[l]}_{j,i}} w^{[l]}_{j,k}. \tag{17}$$

Vectorizing equation (17) as :

$$\begin{bmatrix} Ja_{1,1}^{[l-1]} & \dots & Ja_{1,i}^{[l-1]} & \dots & Ja_{1,m}^{[l-1]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ja_{k,1}^{[l-1]} & \dots & Ja_{k,i}^{[l-1]} & \dots & Ja_{k,m}^{[l-1]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ja_{n^{[l-1]},1}^{[l-1]} & \dots & Ja_{n^{[l-1]},i}^{[l-1]} & \dots & Ja_{n^{[l-1]},m}^{[l-1]} \end{bmatrix}$$

$$= \begin{bmatrix} w_{1,1}^{[l]} & \dots & w_{j,1}^{[l]} & \dots & w_{n^{[l]},1}^{[l]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{1,k}^{[l]} & \dots & w_{j,k}^{[l]} & \dots & w_{n^{[l]},k}^{[l]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{1,n^{[l-1]}}^{[l]} & \dots & w_{j,n^{[l-1]}}^{[l]} & \dots & w_{n^{[l]},n^{[l-1]}}^{[l]} \end{bmatrix}$$

$$\cdot \begin{bmatrix} Jz_{1,1}^{[l]} & \dots & Jz_{1,i}^{[l]} & \dots & Jz_{1,m}^{[l]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Jz_{j,1}^{[l]} & \dots & Jz_{j,i}^{[l]} & \dots & Jz_{j,m}^{[l]} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Jz_{n^{[l]},1}^{[l]} & \dots & Jz_{n^{[l]},i}^{[l]} & \dots & Jz_{n^{[l]},m}^{[l]} \end{bmatrix},$$

Preceding matrix equations can be re written as :

dA$^{[l-1]} = \frac{\partial J}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]}$(18)
where $\frac{\partial J}{\partial \vec{A}^{[l-1]}} \in R^{n^{[l-1]} \times m}$

2.4 Update Expressions

Gradient Descent algorithm is being used to update expressions using following equations :

$$W^{[l]} = W^{[l]} - \alpha \, dW^{[l]}$$
$$b^{[l]} = b^{[l]} - \alpha \, db^{[l]}$$

where α is the learning rate.

3 Hyperparameter Tuning

Selected learning Rate of $\alpha = 0.01$ and $epochs = 20$ has been found through linear search. Also the the reason to choose 20 epochs is loss is not converging beyond it.

3.1 Loss vs Epoch Graph for Unaugmented Data Set

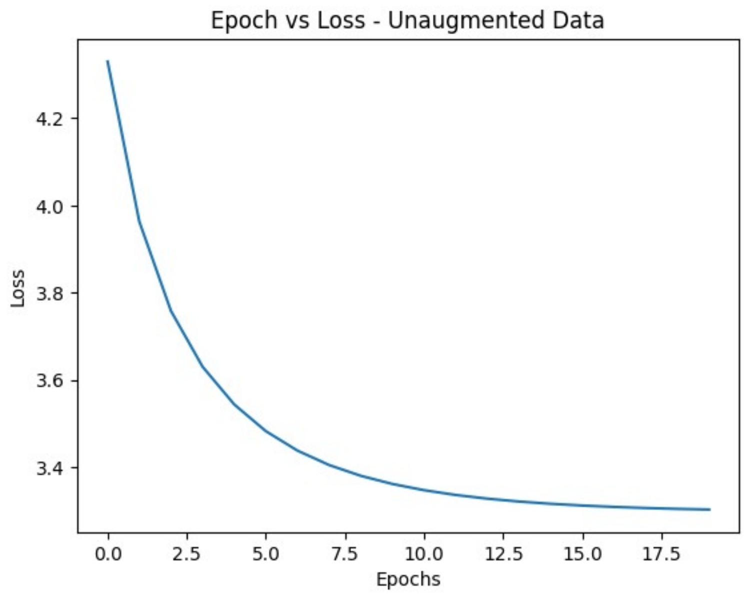

Figure 2: Loss vs Epoch Graph for Original Data Set

3.2 Loss vs Epoch Graph for Augmented Data Set

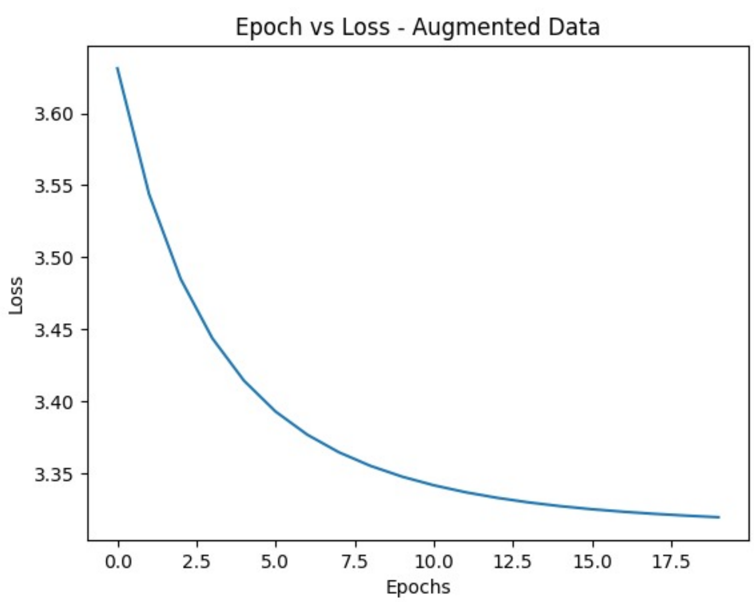

Figure 3: Loss vs Epoch Graph for Augmented Data Set

4 Evaluation Metrics

Implemented MLP Model achieved 90 % Train accuracies for both augmented and unaugmented data set , The test accuracies for unaugmented and augmented test data is 82.1 % and 82 % respectively.

4.1 Accuracy

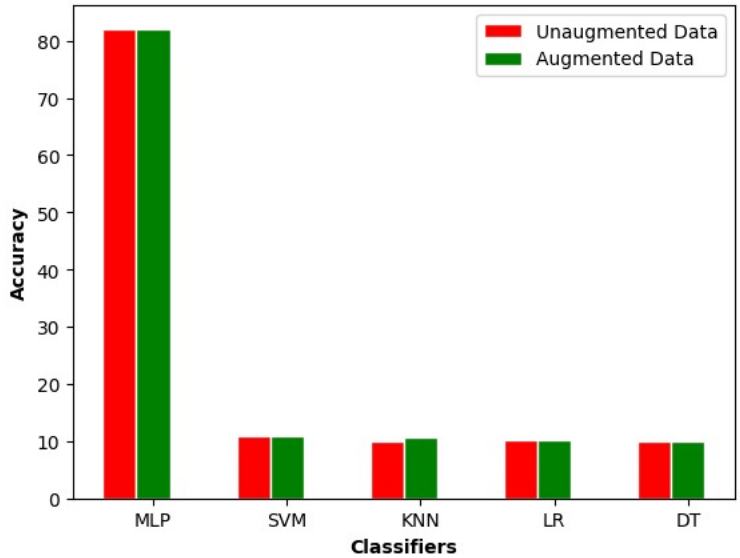

Figure 4: Accuracy Graph for both Original and Augmented Data set

4.2 Time taken for prediction

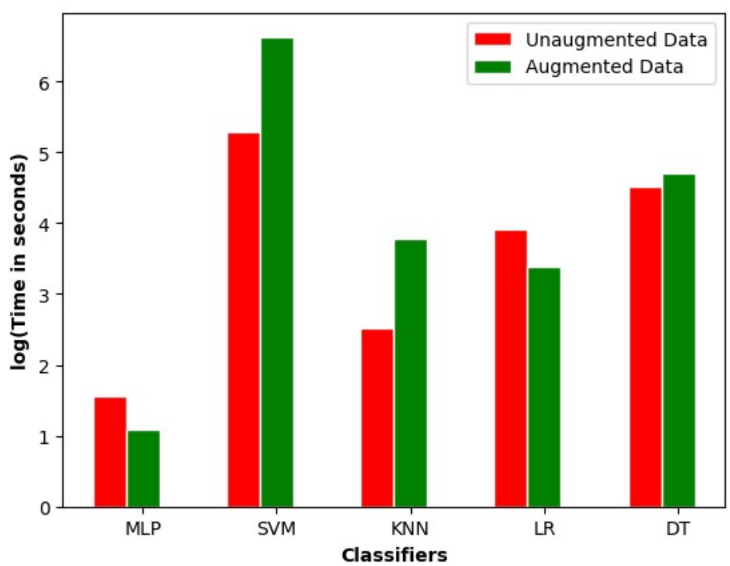

Figure 5: Prediction Time Graph for both Original and Augmented Data set

4.3 Justifications for difference in accuracy and time taken

- SVM, LR , KNN and DT Classifiers perform relatively poorly on non-linear data sets as compared to MLP Model.
- Computation time is les for MLP Model due to vectorization of forward computation process using numpy and parallel computing.

5 References

[1] https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide
[2] https://gautamnagrawal.medium.com/rotating-image-by-any-angle-shear-transformation-using-only-numpy-d2
[3] https://jonaslalin.com/2021/12/10/feedforward-neural-networks-part-1/#mjx-eqn%3Aeq%3Achain_rule
[4] https://jonaslalin.com/2021/12/22/feedforward-neural-networks-part-3//
[5] https://jonaslalin.com/2021/12/21/feedforward-neural-networks-part-2/
[6] https://www.cs.umd.edu/class/fall2018/cmsc470/slides/slides_8.pdf
[7] https://www.youtube.com/watch?v=7HPwo4wnJeA
[8] https://www.binarystudy.com/2021/09/how-to-load-preprocess-visualize-CIFAR-10-and-CIFAR-100.html#cifar10-load
[9] https://www.geeksforgeeks.org/rotate-image-without-cutting-off-sides-using-python-opencv/
[10] https://towardsdatascience.com/data-augmentation-compilation-with-python-and-opencv-b76b1cd500e0
[11] https://towardsdatascience.com/a-beginners-guide-to-image-augmentations-in-machine-learning-22c48a2fbc
[12] https://stackoverflow.com/questions/49643907/clipping-input-data-to-the-valid-range-for-imshow-with-rg
[13] https://towardsdatascience.com/a-beginners-guide-to-image-augmentations-in-machine-learning-22c48a2fbc
[14] https://stackoverflow.com/questions/43391205/add-padding-to-images-to-get-them-into-the-same-shape
[15] https://towardsdatascience.com/a-beginners-guide-to-image-augmentations-in-machine-learning-22c48a2fbc
[16] https://www.youtube.com/watch?v=YP6APKLRf58&ab_channel=AdrianDolinay
[17] https://towardsdatascience.com/how-to-cluster-images-based-on-visual-similarity-cd6e7209fe34
[18] https://www.kaggle.com/mtax687/single-layer-neural-network-using-numpy
[19] Various forums such as StackOverflow and StackExchange for debugging the code