Date of Performance:

Date of Submission :

EXPERIMENT NUMBER: 4

Aim: Implement a Backpropagation algorithm to train a DNN with at least 2 hidden layers.

Objective:

To implement various training algorithms for feedforward neural networks.

Software Used :Python

Theory:

(a) Backpropagation algorithm

Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.
Backpropagation in neural network is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.

Two Types of Backpropagation Networks are:

- Static Back-propagation
- Recurrent Backpropagation

Static back-propagation:

It is one kind of Backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.
The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

Algorithm:

Step 1: Inputs X, arrive through the preconnected path.

Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.

Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer. Step

4: Calculate the error in the output

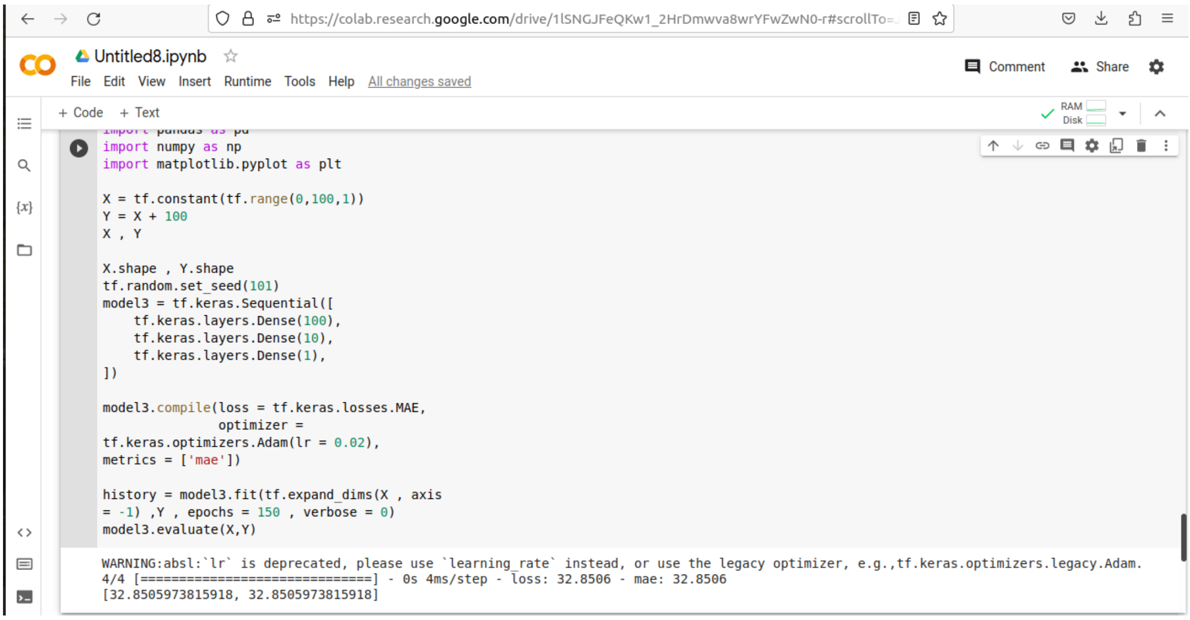
Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error. Step

6: Repeat the process until the desired output is achieve

Program:

```
import tensorflow as tf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

X = tf.constant(tf.range(0,100,1))

Y = X + 100

X , Y

X.shape , Y.shape
 tf.random.set_seed(101)
 model3 = tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1),
 ])

 model3.compile(loss = tf.keras.losses.MAE,
          optimizer =
 tf.keras.optimizers.Adam(lr = 0.02),
          metrics = ['mae'])

 history = model3.fit(tf.expand_dims(X , axis
 = -1) ,Y , epochs = 150          , verbose = 0)


 model3.evaluate(X,Y
```

Output:



Conclusion/Outcome:

Thus we have implemented a Backpropagation algorithm to train a DNN with at least 2 hidden layers.

Marks & Signature:

R1 (4 Marks)	R2 (4 Marks)	R3 (4 Marks)	R4 (3 Mark)	Total (15 Marks)	Signature