EXPT NO : 5 **A python program to implement Multi Layer**

DATE: 20/09/2024 **Perceptron With Backpropagation**

AIM:

To write a python program to implement Multilayer perceptron with backpropagation .

PROCEDURE:

Implementing Multilayer perceptron with backpropagation using the Keras dataset involve the following steps:

Step 1: Import Necessary Libraries

First, import the libraries that are essential for data manipulation, visualization, and model building.

```
# importing modules

import tensorflow as tf

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation

import matplotlib.pyplot as plt
```

Step 2: Load the Keras Dataset

The Keras dataset can be loaded.

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

OUTPUT :

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Step 3: Data Preprocessing

Ensure the data is clean and ready for modeling. Since the Iris dataset is clean, minimal preprocessing is needed.

```
# Cast the records into float values

x_train = x_train.astype('float32')

x_test = x_test.astype('float32')



# normalize image pixel values by dividing

# by 255

gray_scale = 255

x_train /= gray_scale

x_test /= gray_scale



print("Feature matrix:", x_train.shape)

print("Target matrix:", x_test.shape)

print("Feature matrix:", y_train.shape)

print("Target matrix:", y_test.shape)
```

OUTPUT :

```
Feature matrix: (60000, 28, 28)
Target matrix: (10000, 28, 28)
Feature matrix: (60000,)
Target matrix: (10000,)
```

Step 4 : Train a Model

```
model = Sequential([


  # reshape 28 row * 28 column data to 28*28 rows

  Flatten(input_shape=(28, 28)),


  # dense layer 1

  Dense(256, activation='sigmoid'),


  # dense layer 2

  Dense(128, activation='sigmoid'),


  # output layer

  Dense(10, activation='sigmoid'),

])
```

OUTPUT:

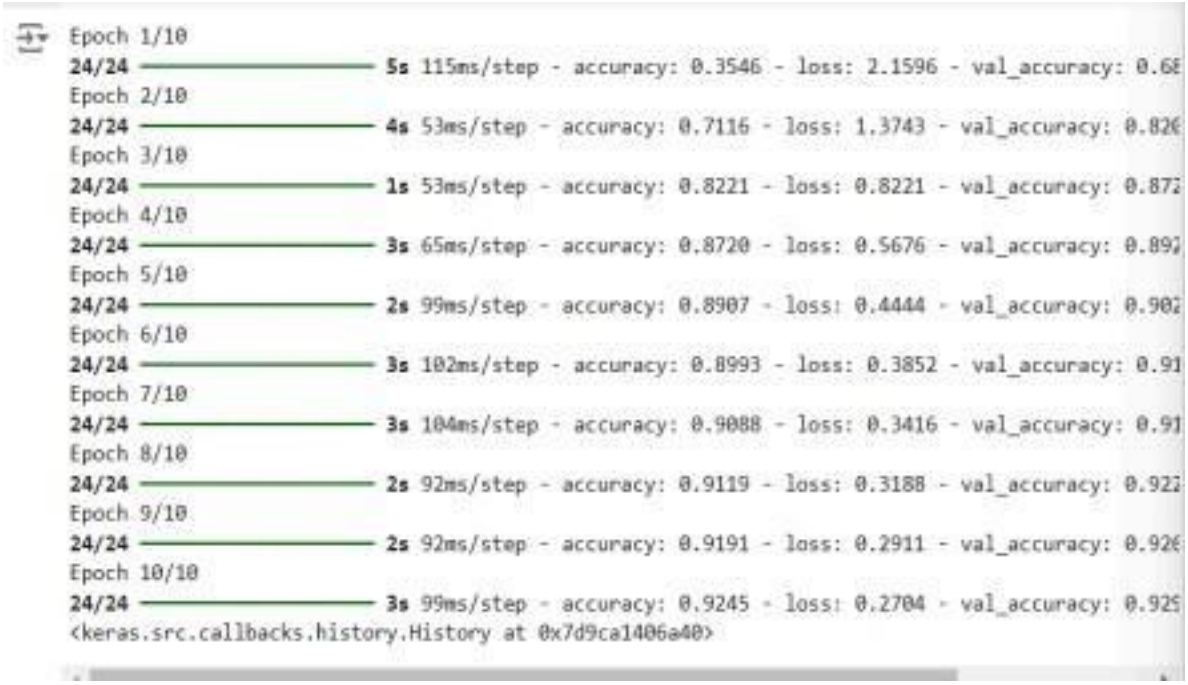
Step 5 : Make Predictions

Use the model to make predictions based on the independent variable.

```
model.compile(optimizer='adam',

      loss='sparse_categorical_crossentropy',
      metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=10,
    batch_size=2000,
    validation_split=0.2)
```

OUTPUT:



Step 6 : Evaluate the Model

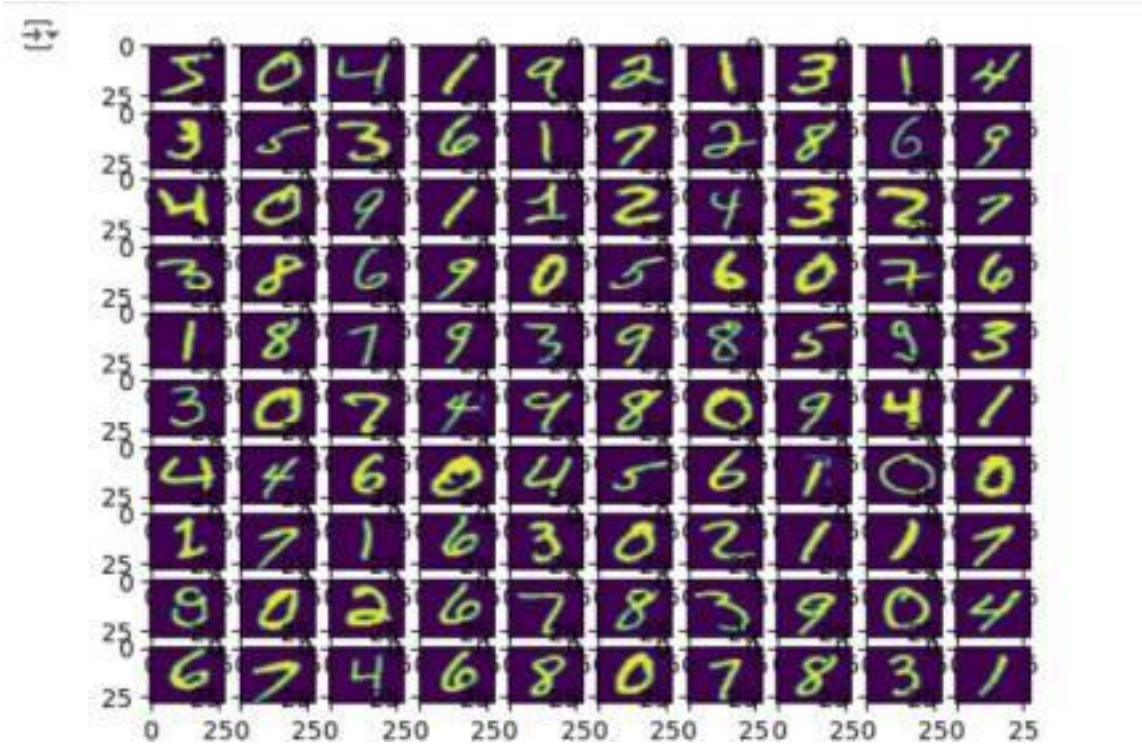
Evaluate the model performance.

```
results = model.evaluate(x_test, y_test, verbose = 0)

print('test loss, test acc:', results)


fig, ax = plt.subplots(10, 10)

k = 0

for i in range(10):
  for j in range(10):
    ax[i][j].imshow(x_train[k].reshape(28, 28),
                    aspect='auto')

    k += 1
```

```
plt.show()
```

OUTPUT :

```
test loss, test acc: [0.2589016258716583, 0.9277999997138977]
```



RESULT:

This step-by-step process will help us to implement MultiLayer Perceptron with Backpropagation models using the Keras dataset and analyze their performance.