Practical no.8

**Aim:** Implement MLP for classification of handwritten digits (MNIST Dataset)

# Theory

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses back propagation for training the network. MLP is a deep learning method.

**Material**

* Tensflow

# Program

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.datasets import mnist from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten from tensorflow.keras.optimizers import RMSprop

# Load the MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data() # Normalize the input images

x\_train = x\_train.reshape(x\_train.shape[0], 784).astype('float32') / 255.0

x\_test = x\_test.reshape(x\_test.shape[0], 784).astype('float32') / 255.0 # Convert the labels to one-hot encoding

y\_train = keras.utils.to\_categorical(y\_train, 10) y\_test = keras.utils.to\_categorical(y\_test, 10) # Define the MLP model

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,))) model.add(Dropout(0.2))

model.add(Dense(512, activation='relu')) model.add(Dropout(0.2)) model.add(Dense(10, activation='softmax')) # Compile the model

model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(),

metrics=['accuracy']) # Train the model

history = model.fit(x\_train, y\_train, batch\_size=128,

epochs=20, verbose=1,

validation\_data=(x\_test, y\_test))

# Evaluate the model on the test set

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0]) print('Test accuracy:', score[1])

# Output:

