**Module 12**

Use Functional API to build a model on MNIST Dataset from keras

# import libraries

import numpy as np

import tensorflow as tf

from tensorflow.keras.layers import Dense, Dropout, Input

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten

from tensorflow.keras.models import Model

from tensorflow.keras.datasets import mnist

from keras.utils import to\_categorical

# load data

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# convert sparse label to categorical values

num\_labels = len(np.unique(y\_train))

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

# preprocess the input images

image\_size = x\_train.shape[1]

x\_train = np.reshape(x\_train,[-1, image\_size, image\_size, 1])

x\_test = np.reshape(x\_test,[-1, image\_size, image\_size, 1])

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# parameters for the network

input\_shape = (image\_size, image\_size, 1)

batch\_size = 128

kernel\_size = 3

filters = 64

dropout = 0.3

# utiliaing functional API to build cnn layers

inputs = Input(shape=input\_shape)

y = Conv2D(filters=filters,kernel\_size=kernel\_size,activation='relu')(inputs)

y = MaxPooling2D()(y)

y = Conv2D(filters=filters,kernel\_size=kernel\_size,activation='relu')(y)

y = MaxPooling2D()(y)

y = Conv2D(filters=filters,kernel\_size=kernel\_size,activation='relu')(y)

# convert image to vector

y = Flatten()(y)

# dropout regularization

y = Dropout(dropout)(y)

outputs = Dense(num\_labels, activation='softmax')(y)

# model building by supplying inputs/outputs

model = Model(inputs=inputs, outputs=outputs)

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(x\_train,y\_train,validation\_data=(x\_test, y\_test),epochs=20,batch\_size=batch\_size)

# accuracy evaluation

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

print("nTest accuracy: %.1f%%" % (100.0 \* score[1]))

################################ END ####################################