**Module 12**

**Name: Upadhyay Sachin Naresh**

**Batch id: AIWDEOB 300821**

**Topic: Functional APIs**

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

#Functional API for the MNIST data set using CNN model

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import numpy as np

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 tensorflow.keras.utils import to\_categorical

# loading MNIST dataset

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

# Converting from sparse label to categorical

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

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

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

# reshaping and normalizing 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

# Defining network parameters for the model

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

batch\_size = 128

kernel\_size = 3

filters = 64

dropout = 0.3

# Using The functional API to Build the CNN model

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)

# Converting image to vector before connecting to dense layer

y = Flatten()(y)

# dropout regularization

y = Dropout(dropout)(y)

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

# building the model by supplying inputs/outputs

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

# network model in text

model.summary()

# classifier loss, Adam optimizer, classifier accuracy

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

# train the model with input images and labels

history = model.fit(x\_train,y\_train,validation\_data=(x\_test, y\_test),epochs=20,batch\_size=batch\_size,validation\_split=0.2)

# model accuracy on test dataset

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

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

import matplotlib.pyplot as plt

acc = history.history['acc']

val\_acc = history.history['val\_acc']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation acc')

plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.legend()

plt.show()

