**Module 12**

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

**Code:**

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

# Load MNIST dataset

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

# Preprocess the data

train\_images = train\_images.reshape((60000, 28, 28, 1)).astype('float32') / 255

test\_images = test\_images.reshape((10000, 28, 28, 1)).astype('float32') / 255

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

# Define the model using Functional API

inputs = layers.Input(shape=(28, 28, 1))

x = layers.Conv2D(32, (3, 3), activation='relu')(inputs)

x = layers.MaxPooling2D((2, 2))(x)

x = layers.Conv2D(64, (3, 3), activation='relu')(x)

x = layers.MaxPooling2D((2, 2))(x)

x = layers.Conv2D(64, (3, 3), activation='relu')(x)

x = layers.Flatten()(x)

x = layers.Dense(64, activation='relu')(x)

outputs = layers.Dense(10, activation='softmax')(x)

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

# Compile the model

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(train\_images, train\_labels, epochs=5, batch\_size=6400, validation\_split=0.2)

# Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

**Output:**

history = model.fit(train\_images, train\_labels, epochs=5, batch\_size=6400, validation\_split=0.2)

Epoch 1/5

8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.9958 - loss: 0.0133 - val\_accuracy: 0.9897 - val\_loss: 0.0408

Epoch 2/5

8/8 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9961 - loss: 0.0129 - val\_accuracy: 0.9897 - val\_loss: 0.0382

Epoch 3/5

8/8 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9966 - loss: 0.0119 - val\_accuracy: 0.9905 - val\_loss: 0.0355

Epoch 4/5

8/8 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9972 - loss: 0.0097 - val\_accuracy: 0.9908 - val\_loss: 0.0342

Epoch 5/5

8/8 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9973 - loss: 0.0092 - val\_accuracy: 0.9912 - val\_loss: 0.0336

2024-05-16 14:37:07.687822: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF\_ENABLE\_ONEDNN\_OPTS=0`.

2024-05-16 14:37:11.280581: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF\_ENABLE\_ONEDNN\_OPTS=0`.

2024-05-16 14:37:18.395383: I tensorflow/core/platform/cpu\_feature\_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.

To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9898 - loss: 0.0300

Test accuracy: 0.9919000267982483