# Ex No: 4 HANDWRITTEN DIGITS RECOGNITION WITH MNIST

**AIM:**

To build a handwritten digit’s recognition with MNIST dataset.

# PROCEDURE:

1. Download and load the MNIST dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

# PROGRAM:

import numpy as np

from tensorflow.keras.models import load\_model from tkinter import \*

import tkinter as tk #import win32gui

from PIL import ImageGrab, Image

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.layers import Conv2D, MaxPooling2D from tensorflow.keras import backend as K

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

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

y\_train = keras.utils.to\_categorical(y\_train, 10) y\_test = keras.utils.to\_categorical(y\_test, 10) x\_train = x\_train.astype('float32')

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

x\_test /= 255

print('x\_train shape:', x\_train.shape) print(x\_train.shape[0], 'train samples') print(x\_test.shape[0], 'test samples') batch\_size = 128

num\_classes = 10

epochs = 15

model = Sequential()

model.add(Conv2D(32, kernel\_size=(5, 5),activation='relu',input\_shape=input\_shape)) model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))AA

model.add(MaxPooling2D(pool\_size=(2, 2))) model.add(Flatten())

model.add(Dense(128, activation='relu')) model.add(Dropout(0.3)) model.add(Dense(64, activation='relu')) model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),met rics=['accuracy'])

hist = model.fit(x\_train, y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test))

print("The model has successfully trained")

score = model.evaluate(x\_test, y\_test, verbose=0) print('Test loss:', score[0])

print('Test accuracy:', score[1]) model.save('mnist.h5')

print("Saving the model as mnist.h5") model = load\_model('mnist.h5')

def predict\_digit(img):

#resize image to 28x28 pixels img = img.resize((28,28)) #convert rgb to grayscale

img = img.convert('L') img = np.array(img)

img = img.reshape(1,28,28,1) img = img/255.0

img = 1 - img #predicting

res = model.predict([img])[0] return np.argmax(res), max(res)

import matplotlib.pyplot as plt

# Use an image from the MNIST test dataset

test\_image\_array = x\_test[0] # Change the index to use different images from the test set test\_image\_pil = Image.fromarray((test\_image\_array.squeeze() \* 255).astype(np.uint8))

# Predict the digit in the image

predicted\_digit, confidence = predict\_digit(test\_image\_pil)

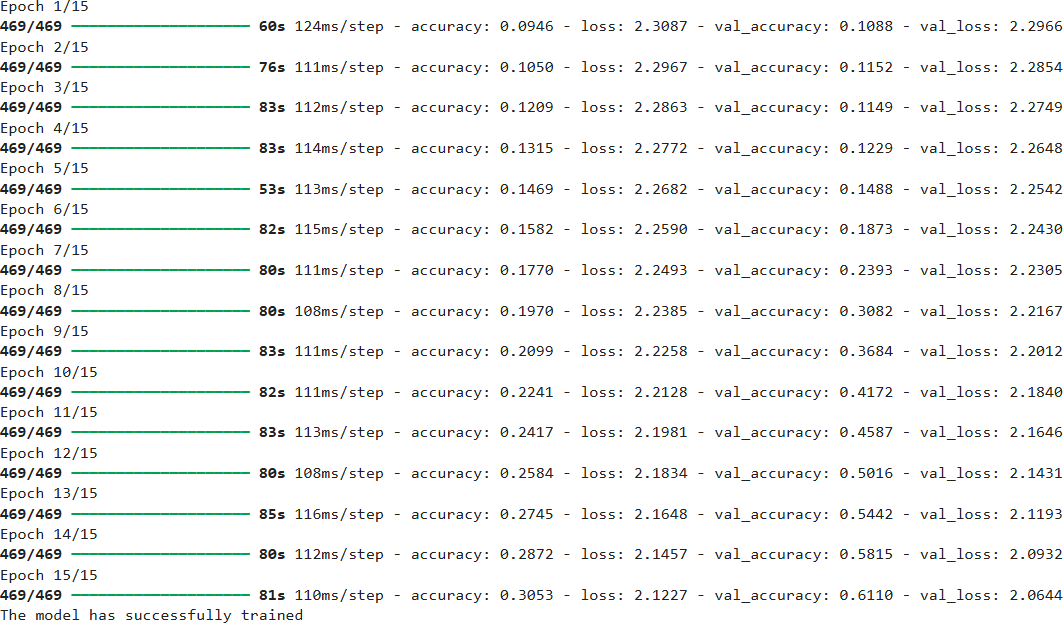
# Print the results

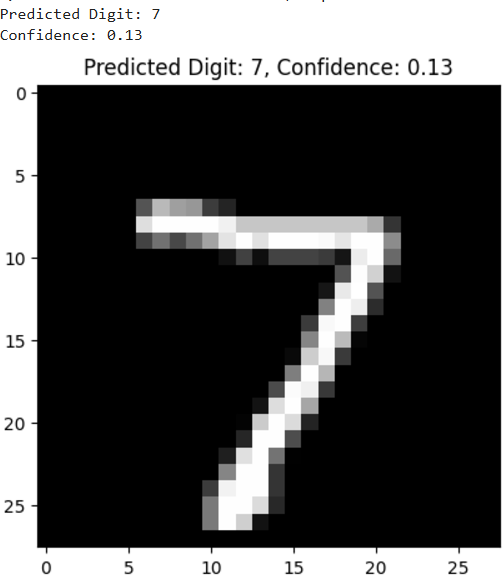
print(f"Predicted Digit: {predicted\_digit}") print(f"Confidence: {confidence:.2f}")

# Show the test image plt.imshow(test\_image\_array.squeeze(), cmap='gray')

plt.title(f"Predicted Digit: {predicted\_digit}, Confidence: {confidence:.2f}") plt.show()

# OUTPUT:





**RESULT:**

Thus a handwritten digit’s recognition with MNIST dataset is built.