Exp 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:

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 = 50 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')) 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(),metrics

=['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]) import matplotlib.pyplot as plt

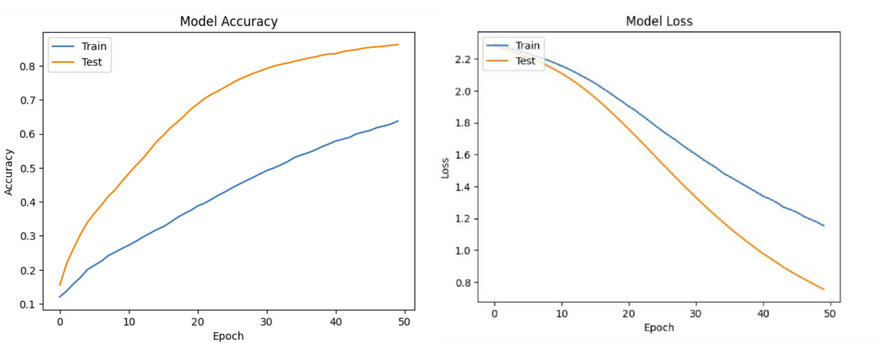
plt.plot(hist.history['accuracy']) plt.plot(hist.history['val\_accuracy']) plt.title('Model Accuracy') plt.ylabel('Accuracy') plt.xlabel('Epoch') plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

# Plot training & validation loss values plt.plot(hist.history['loss']) plt.plot(hist.history['val\_loss']) plt.title('Model Loss') plt.ylabel('Loss') plt.xlabel('Epoch') plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

Output



Result

Handwritten digit recognition with MNIST has been successfully created.