Task 3 : Number Recognition

Yes, handwritten digit recognition systems can be used to detect scanned images of handwritten digits. In fact, this is one of the most common applications of handwritten digit recognition.

Here is a brief overview of how a handwritten digit recognition system works:

1. The system first scans the handwritten digit image and converts it to a digital format.
2. The system then preprocesses the image to normalize the size and brightness of the digit.
3. The system then extracts features from the image, such as the shape of the digit, the distribution of pixels, and the curvature of the lines.
4. The system then uses a machine learning algorithm to classify the extracted features and predict the digit.

Neural networks are a type of machine learning algorithm that is particularly well-suited for handwritten digit recognition. Neural networks are able to learn complex patterns in the data, which allows them to accurately classify handwritten digits.

The MNIST dataset is a popular dataset of handwritten digits that is often used to train and evaluate handwritten digit recognition systems. The MNIST dataset contains 60,000 training images and 10,000 test images of handwritten digits from 0 to 9.

To build a handwritten digit recognition system using the MNIST dataset, we can use the following steps:

1. Load the MNIST dataset.
2. Preprocess the training and test images.
3. Train a neural network to classify the handwritten digits.
4. Evaluate the performance of the neural network on the test set.

Once the neural network is trained, we can use it to predict the digits in scanned images of handwritten digits. This can be done by extracting features from the scanned image and then using the neural network to classify the features.

Handwritten digit recognition systems have a wide range of applications. For example, they can be used to:

- Digitize handwritten documents, such as bank checks and postal forms.
- Process handwritten input in mobile devices and tablets.
- Develop educational applications that teach children how to write numbers.
- Develop security applications that verify users' identities by asking them to write a specific number.

Handwritten digit recognition is a powerful technology that has the potential to revolutionize the way we interact with computers.

PROGRAM :

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
```

```
X_test /= 255

y_train = tf.keras.utils.to_categorical(y_train)

y_test = tf.keras.utils.to_categorical(y_test)

model = Sequential()

model.add(Dense(512, activation='relu', input_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer=Adam(learning_rate=0.001),

        loss='categorical_crossentropy',

        metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=128)

test_loss, test_acc = model.evaluate(X_test, y_test)

print('Test accuracy:', test_acc)
```

OUTPUT :

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/10
469/469 [==============================] - 11s 21ms/step - loss: 0.2678 - accuracy:
0.9212
Epoch 2/10
469/469 [==============================] - 13s 27ms/step - loss: 0.1061 - accuracy:
0.9671
Epoch 3/10
469/469 [==============================] - 10s 21ms/step - loss: 0.0773 - accuracy:
0.9764
Epoch 4/10
469/469 [==============================] - 7s 14ms/step - loss: 0.0594 - accuracy:
0.9812
Epoch 5/10
```

```
469/469 [==============================] - 8s 16ms/step - loss: 0.0487 - accuracy:
0.9842
Epoch 6/10
469/469 [==============================] - 7s 14ms/step - loss: 0.0392 - accuracy:
0.9874
Epoch 7/10
469/469 [==============================] - 8s 16ms/step - loss: 0.0334 - accuracy:
0.9893
Epoch 8/10
469/469 [==============================] - 7s 15ms/step - loss: 0.0331 - accuracy:
0.9890
Epoch 9/10
469/469 [==============================] - 7s 16ms/step - loss: 0.0264 - accuracy:
0.9913
Epoch 10/10
469/469 [==============================] - 8s 16ms/step - loss: 0.0252 - accuracy:
0.9914
313/313 [==============================] - 1s 3ms/step - loss: 0.0623 - accuracy:
0.9831
Test accuracy: 0.9830999970436096
```
