PROJECT TITLE : RECOGNIZING HANDWRITTEN DIGITS USING MACHINE LEARNING

PROGRAM:

```
# Import required libraries
import numPy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Enable memory growth for GPU (optional)
physical_devices = tf.config.list_physical_devices('GPU')
if physical_devices:
  try:
    tf.config.set_memory_growth(physical_devices[0], True)  # Updated API
  except RuntimeError as e:
    print(f"GPU memory growth setting failed: {e}")

# Load the MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape and normalize the input data
X_train = X_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
X_test = X_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0

# Convert class labels to one-hot encoding
```

```
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
# Build the CNN model
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  # 10 classes for digits 0–9
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Print model summary
model.summary()

# Train the model
model.fit(X_train, y_train, batch_size=128, epochs=5, validation_split=0.1)
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print("\nTest Accuracy: {:.4f}".format(accuracy))
```

output:

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 128)	204,928
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1,290

```
 Total params: 225,034 (879.04 KB)
 Trainable params: 225,034 (879.04 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 41s 93ms/step - accuracy: 0.7987 - loss: 0.6566 - val_accuracy: 0.9815 - val_loss: 0.0670
Epoch 2/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 39s 94ms/step - accuracy: 0.9674 - loss: 0.1141 - val_accuracy: 0.9858 - val_loss: 0.0521
Epoch 3/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 42s 95ms/step - accuracy: 0.9762 - loss: 0.0794 - val_accuracy: 0.9875 - val_loss: 0.0428
Epoch 4/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 39s 92ms/step - accuracy: 0.9811 - loss: 0.0616 - val_accuracy: 0.9897 - val_loss: 0.0386
Epoch 5/5
401/422 ━━━━━━━━━━━━━━━━━━━━ 1s 88ms/step - accuracy: 0.9833 - loss: 0.0551
```