Ex No: 4 **Date: 16-08-2024**

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:

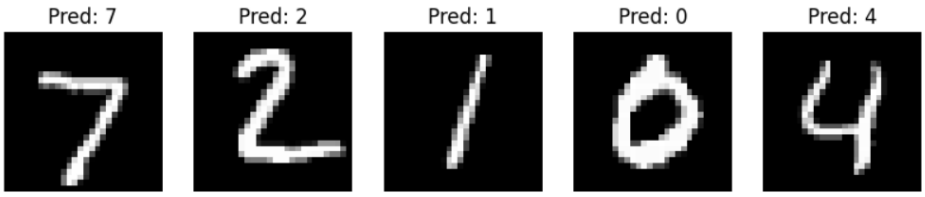
```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt


# Step 1: Download and load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()


# Step 2: Perform analysis and preprocessing of the dataset
# Normalize the images to the range 0-1
```

```
x_train = x_train / 255.0
x_test = x_test / 255.0

# Convert labels to one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Step 3: Build a simple neural network model using Keras/TensorFlow
model = Sequential([
    Flatten(input_shape=(28, 28)),     # Flatten the 28x28 images to 1D
    Dense(128, activation='relu'),     # Hidden layer with 128 neurons and ReLU activation
    Dense(10, activation='softmax')    # Output layer with 10 neurons for 10 classes (digits 0-9)
])

# Step 4: Compile and fit the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, validation_split=0.2)  # Train the model with 5 epochs

# Step 5: Perform prediction with the test dataset
predictions = model.predict(x_test)

# Step 6: Calculate performance metrics
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc:.4f}')
```

```
# Display a few predictions
plt.figure(figsize=(10, 5))
for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(x_test[i], cmap='gray')
    plt.title(f'Pred: {predictions[i].argmax()}')
    plt.axis('off')
plt.show()
```

OUTPUT:

```
1500/1500 [==============================] - 11s 5ms/step - loss: 0.2895 - accuracy: 0.9178 - val_loss: 0.1594 - val_accuracy: 0.9538
Epoch 2/5
1500/1500 [==============================] - 7s 4ms/step - loss: 0.1307 - accuracy: 0.9606 - val_loss: 0.1228 - val_accuracy: 0.9628
Epoch 3/5
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0899 - accuracy: 0.9731 - val_loss: 0.1023 - val_accuracy: 0.9693
Epoch 4/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0675 - accuracy: 0.9795 - val_loss: 0.0949 - val_accuracy: 0.9707
Epoch 5/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0519 - accuracy: 0.9841 - val_loss: 0.0883 - val_accuracy: 0.9747
313/313 [==============================] - 1s 2ms/step
313/313 [==============================] - 2s 4ms/step - loss: 0.0743 - accuracy: 0.9772
Test accuracy: 0.9772
```



RESULT:

Thus, the handwritten digit's recognition with MNIST dataset was successfully implemented.