Name : Yash Sarang	Class/Roll No. : D16AD / 47	Grade :

Title of Experiment :
Design and implement a CNN model for digit recognition

Objective of Experiment :
To design deep learning models for supervised, unsupervised and sequence learning.

Outcome of Experiment :
Build and train deep learning models such as Auto encoders, CNNs, RNN, LSTM etc.

Problem Statement :
The objective is to design and implement a Convolutional Neural Network (CNN) model to accurately recognize handwritten digits from the MNIST dataset. The CNN will be trained to classify digits into the appropriate categories (0-9).

Description / Theory :

Convolutional Neural Network (CNN):

A CNN is a type of deep neural network specifically designed for processing grid-like data, such as images. CNNs consist of convolutional layers that apply filters to extract local patterns from the input data. They are highly effective in image recognition tasks.

Digit Recognition with CNN:

In the context of digit recognition, a CNN is designed to take an input image of a handwritten digit and classify it into one of the ten digits (0-9). The model learns to recognize features at different levels of abstraction, enabling accurate classification.

Flowchart :
1. Load and Preprocess Data:
    - Load the MNIST dataset and preprocess it, preparing the data for training and testing.
2. Design CNN Architecture:
    - Create a CNN architecture comprising convolutional layers, pooling layers, fully connected layers, and an output layer.
3. Compile CNN Model:
    - Compile the CNN model by specifying an appropriate loss function (e.g., categorical cross-entropy) and an optimizer (e.g., Adam).
4. Train CNN Model:
    - Train the CNN model using the preprocessed training data, specifying the number of epochs and batch size.

5. Evaluate CNN Model:
   - Evaluate the trained CNN model on the test dataset to assess its accuracy and performance.

Program:

```
[1]  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 Dense, Flatten, Conv2D, MaxPooling2D
     from tensorflow.keras.utils import to_categorical

     # Load the MNIST dataset and preprocess
     (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

     # Preprocess the data
     train_images = train_images.astype('float32') / 255.0
     test_images = test_images.astype('float32') / 255.0
     train_images = np.expand_dims(train_images, axis=-1)  # Add a channel dimension
     test_images = np.expand_dims(test_images, axis=-1)
     train_labels = to_categorical(train_labels, 10)
     test_labels = to_categorical(test_labels, 10)

     # Define the CNN architecture
     model = Sequential([
         Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
         MaxPooling2D((2, 2)),
         Conv2D(64, (3, 3), activation='relu'),
         MaxPooling2D((2, 2)),
         Flatten(),
         Dense(64, activation='relu'),
         Dense(10, activation='softmax')
     ])

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
     11490434/11490434 [==============================] - 0s 0us/step
```

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

     # Train the model
     model.fit(train_images, train_labels, epochs=5, batch_size=64, validation_data=(test_images, test_labels))

     # Evaluate the model
     test_loss, test_acc = model.evaluate(test_images, test_labels)
     print("Test accuracy:", test_acc)

     Epoch 1/5
     938/938 [==============================] - 53s 55ms/step - loss: 0.1674 - accuracy: 0.9501 - val_loss: 0.0509 - val_accuracy: 0.9836
     Epoch 2/5
     938/938 [==============================] - 46s 49ms/step - loss: 0.0495 - accuracy: 0.9842 - val_loss: 0.0424 - val_accuracy: 0.9853
     Epoch 3/5
     938/938 [==============================] - 50s 53ms/step - loss: 0.0346 - accuracy: 0.9894 - val_loss: 0.0380 - val_accuracy: 0.9879
     Epoch 4/5
     938/938 [==============================] - 47s 50ms/step - loss: 0.0258 - accuracy: 0.9921 - val_loss: 0.0313 - val_accuracy: 0.9892
     Epoch 5/5
     938/938 [==============================] - 46s 49ms/step - loss: 0.0199 - accuracy: 0.9941 - val_loss: 0.0337 - val_accuracy: 0.9893
     313/313 [==============================] - 3s 11ms/step - loss: 0.0337 - accuracy: 0.9893
     Test accuracy: 0.989300012588501
```

Results and Discussions :

The CNN model designed for digit recognition achieved an outstanding accuracy of approximately 98.5% on the MNIST test dataset. The model effectively learned and recognized intricate patterns and features in the handwritten digits, showcasing the power of CNNs in image classification tasks. The training process revealed a steady decrease in the loss function, indicating successful convergence and learning. The model's performance was further bolstered by the use of appropriate activation functions, convolutional layers, and pooling layers, allowing it to capture the hierarchical features present in the digit images.

Conclusion :

In conclusion, the designed CNN model proved highly effective in recognizing handwritten digits, showcasing the potential of CNNs in image classification tasks. The achieved high accuracy emphasizes the importance and relevance of CNNs in real-world applications, including digit recognition for various domains like finance, automation, and digitized document processing. The success of this CNN model lays the foundation for further exploration and application of deep learning in image recognition and classification, opening doors to more advanced CNN architectures and expanding the horizons of digit recognition technology.

******