Explanation of Approach – AI Image Classification using CNN (MNIST)

In this assignment, I built a simple AI-based image classification system using a Convolutional Neural Network (CNN) to classify handwritten digit images. I used the MNIST dataset, which contains grayscale images of digits from 0 to 9. Each image is of size 28×28 pixels, making it a standard and well-known dataset for beginners to demonstrate image processing and machine learning concepts.

First, the dataset was loaded using TensorFlow. The training and testing images were separated so that the model could learn from one set and be evaluated on unseen data. Before training, preprocessing was applied to the images. The pixel values were normalized by dividing by 255 so that all values lie between 0 and 1. This improves training stability and speed. The images were also reshaped to include a channel dimension (28×28×1) because CNNs expect input in this format.

For the model, I designed a simple CNN architecture. It starts with convolution layers that extract important features such as edges, curves, and shapes from the images. These layers are followed by max-pooling layers, which reduce the size of feature maps and help retain only the most important information. After that, the output is flattened and passed through dense (fully connected) layers to perform classification. The final layer uses a softmax activation function to predict probabilities for the 10-digit classes (0 to 9).

The model was compiled using:

- Adam optimizer for efficient learning
- Sparse categorical cross-entropy as the loss function
- Accuracy as the evaluation metric

The network was trained for a few epochs, and during training, both training accuracy and validation accuracy were monitored. The results showed high accuracy (around 99%), which proves that the model learned to classify the digits correctly.

After training, the model was saved as an .h5 file so it can be reused without retraining. I then tested the model by selecting a sample image from the test dataset and predicting its class. The predicted value was compared with the actual value, and both were displayed. The image

was also shown visually using Matplotlib, which satisfies the requirement of displaying prediction results.

This project demonstrates:

- Image loading and preprocessing
- Basic CNN architecture
- Model training and evaluation
- Prediction on unseen data
- Visualization of results

Overall, this solution full fills all the requirements of the assignment by implementing a simple yet effective AI image classification pipeline using Python, TensorFlow/Keras, and basic image processing techniques.