<u>Assignment -2</u>

Explanation MNIST Dataset

NAME : ABISHEK CA

DEPT: CSBS

The MNIST dataset -Modified National Institute of Standards and Technology dataset is one of the most popular datasets in Machine Learning , particularly in Computer vision tasks. It is often used as a beginner's dataset to test image classification models.

Step 1- The code uses Keras's Sequential API to build a linear stack of neural network layers. The Dense layer represents a fully connected layer where each neuron is connected to all previous neurons, and the flattened layer converts 2D inputs (like images) into 1D arrays for compatibility with Dense layers. The to categorical function converts class labels into a one-hot encoded format for multi-class classification. It uses the MNIST dataset, which contains images of handwritten digits and their labels. Additionally, the matplotlib.pyplot library is used to visualize the data.

Step 2-

- **x_train / 255.0**:
  Divides each pixel value (range 0–255) by 255 to scale it to a range of 0–1.
  Normalization
  helps the model train faster and perform better.

  Similarly applied to x_test.

- **One-Hot Encoding Labels**:

  - **to_categorical(y_train, 10)**: Converts the digit labels (e.g., 3) into one-hot encoded vectors (e.g., [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]) for the 10 classes (digits 0–9).
  - Similarly applied to y_test.

Step 3-

- **Sequential**: Creates a sequential stack of layers.
- **Flatten**:
    - Input shape (28, 28) is flattened into a 1D array of size 784 (28×28) to feed into the dense layers.
- **First Dense Layer**:
    - 128 neurons with **ReLU activation**. ReLU (Rectified Linear Unit) allows the model to learn non-linear relationships.
- **Second Dense Layer**:
    - 10 neurons (one for each digit) with **softmax activation**. Softmax converts outputs to probabilities, ensuring the sum is 1.

Step 4-

- **optimizer='adam'**: Adam optimizer adjusts model weights to minimize the loss during training.
- **loss='categorical_crossentropy'**: Loss function for multi-class classification tasks, comparing predicted probabilities with actual one-hot encoded labels.
- **metrics=['accuracy']**: Tracks the accuracy of the model during training and evaluation.

Step 5-

- **fit**: Trains the model on the training data.
    - **epochs=5**: Trains the model for 5 iterations over the entire dataset.
    - **batch_size=32**: Splits data into batches of 32 samples for training. Smaller batches reduce memory usage.

Step 6-

evaluate: Tests the model on unseen data (x_test and y_test).

- Returns loss (categorical cross-entropy) and accuracy (percentage of correct predictions).

Softmax using Code :

Step 1: Exponentiate each raw score to make them positive.

Step 2: Compute the sum of these exponentials.

Step 3: Normalize each exponentiated value by dividing it by the sum.

Result: A probability distribution where all values sum to 1, useful for multi-class classification task

Confusion Matrix :

Make Predictions

- The model predicts probabilities for each digit in the test data.
- We pick the digit with the highest probability as the predicted label.
- True labels are extracted from the test data.

2. Generate Confusion Matrix

- The confusion matrix compares true labels with predicted labels.
- **Rows**: True digit classes.
- **Columns**: Predicted digit classes.

3. Plot the Confusion Matrix

- A heatmap is created to show the confusion matrix.
- **annot=True**: Displays numbers in each box.

- **fmt="d"**: Ensures the numbers are shown as whole numbers.

4. Interpreting the Confusion Matrix

1. **Diagonal Boxes**: Correct predictions. Higher numbers = better performance.
2. **Other Boxes**: Misclassifications. Higher numbers = model struggles with these digits.
3. **Takeaway**: See which digits the model gets right and which ones it confuses.