Convolutional Neural Network (CNN) — Classifying Handwritten Digits

Dataset: MNIST Dataset (torchvision.datasets.MNIST)

Objective: Build and train a Convolutional Neural Network (CNN) to classify handwritten digits (0-9) from the MNIST dataset.

1. **Load & Preprocess the Data**
    - Load the MNIST dataset.
    - Normalize the images so that the pixel values are between 0 and 1 (important for neural networks).
2. **Visualize the Data**
    - Pick any one digit (0–9) and display a few example images of that digit from the MNIST dataset.
    - Optional: Display a few examples of other digits.
3. **Build the CNN**
    - Implement a CNN with at least two convolutional layers, ReLU activation, and max pooling layers.
    - Experiment with the architecture (e.g., number of layers, filter sizes) and hyperparameters (e.g., learning rate, batch size) to improve performance.
    - Use Cross Entropy Loss as the loss function and Adam as the optimizer.
4. **Train the Model**
    - Train the model for 10 epochs.
    - After each epoch, calculate and print:
        - Training and test loss.
        - Training and test accuracy.
5. **Create the Following Plots**
    - Plot 1: Training and test loss vs. epochs
    - Plot 2: Training and test accuracy vs. epochs
    - Plot 3: Final confusion matrix (visualized as a heatmap with class labels)

Optional Extension: Improving Model Generalization with Data Augmentation

Objective: Explore how data augmentation can help your CNN generalize better by artificially expanding the training dataset with transformed images. This improves robustness to variations in input and reduces overfitting.

1. **Understand the Concept of Data Augmentation**
    - Read about how data augmentation works and why it is useful in image classification tasks:
        - It creates new variations of training images using transformations such as rotation and shifting.
        - It helps the model learn to recognize patterns under different orientations.
2. **Apply Data Augmentation to the Training Set**
    - Use torchvision.transforms to apply augmentations (e.g., random rotation, random translation) to the training set only.
3. **Train the Model with Augmented Data**
    - Retrain your CNN using the augmented training data.
    - Keep the rest of your model and training configuration the same (e.g., architecture, optimizer, learning rate).
4. **Compare with the Original Model**
    - Compare the final test accuracy and loss of the original model and the augmented model.