MNIST CNN Classifier 🧠🔢

Code created by:R.Sindhu

This repository contains a Convolutional Neural Network (CNN) implemented in PyTorch to classify handwritten digits from the MNIST dataset. The project includes an interactive Tkinter-based GUI to train, test, and visualize predictions.

**Features 🚀**

* **CNN Architecture**: Uses two convolutional layers with ReLU activation, dropout layers for regularization, and fully connected layers for classification.
* **Dataset Handling**: Automatically downloads and preprocesses the MNIST dataset using torchvision.datasets.
* **Training & Evaluation**: Implements a training loop with Adam optimizer and negative log-likelihood loss.
* **GUI Interface**: Allows users to train, test, and view sample predictions in a user-friendly Tkinter application.
* **Visualization**: Displays real-time training loss curves and sample predictions.

**Main Components**

1. **CNN Model** (CNN class)
   * A simple CNN architecture with two convolutional layers, ReLU activation, max pooling, dropout layers, and fully connected layers.
   * Outputs class probabilities using log\_softmax.
2. **GUI Application** (MNISTApp class)
   * Provides an interactive interface to train and test the model.
   * Displays training progress and evaluation metrics (accuracy, loss).
   * Shows sample predictions using Matplotlib.
3. **Training and Testing Functions**
   * Uses the **Adam optimizer** and **negative log-likelihood loss**.
   * Includes a training loop with real-time progress updates.
   * Tests model performance on the MNIST test set.
4. **Data Loading**
   * Downloads and processes the MNIST dataset using torchvision.datasets.
   * Normalizes images for optimal CNN performance.
5. **Visualization Features**
   * Plots training loss over iterations.
   * Displays predicted labels alongside actual images.

**How It Works**

* The user runs the script, launching a Tkinter-based interface.
* The GUI allows training the model with a click of a button.
* Once trained, the model can be tested, and its accuracy is displayed.
* Users can also view sample predictions.

**Libraries Used**

* torch, torchvision (Deep learning framework)
* matplotlib (Visualization)
* numpy (Numerical computations)
* tkinter (GUI development)
* PIL (Image processing)