Documentation

1. Overview

The Digit Recognition System is a machine learning project designed to recognize handwritten digits drawn on a graphical user interface (GUI). It employs a Convolutional Neural Network (CNN) trained on the MNIST dataset to classify user-drawn digits with high accuracy. This documentation provides a comprehensive guide for understanding, deploying, and maintaining the system.

https://github.com/MemaroX/CV_FInalProject

---

2. System Components

2.1 Data Module

- **Objective**: Load and preprocess both training data (MNIST dataset) and user-drawn inputs to ensure compatibility with the trained model.
- **Features**:
    - Normalization of pixel values between 0 and 1.
    - Conversion of grayscale images to a consistent 28x28 pixel format.

2.2 Model Module

- **Objective**: Train a CNN capable of recognizing and classifying digits.
- **Architecture**:
    - **Input Layer**: Accepts 28x28 grayscale images.
    - **Convolutional Layers**: Extract spatial features using filters.
    - **Pooling Layers**: Reduce spatial dimensions to prevent overfitting.
    - **Dense Layers**: Perform classification using softmax activation.

2.3 GUI Module

- **Objective**: Provide an intuitive platform for users to draw digits and receive predictions in real time.
- **Tools Used**: Tkinter for GUI development, OpenCV for image preprocessing.
- **Functionalities**:
    - Canvas for drawing digits.
    - Buttons to clear the canvas and process predictions.
    - Real-time display of prediction results.

2.4 Prediction Module

- **Objective**: Process user input, predict the digit, and provide feedback.
- **Steps**:
  - Extract the user’s drawn image from the canvas.
  - Resize and preprocess the image.
  - Use the trained model to predict the digit.
  - Display the result to the user.

---

3. Technologies Used

- **Programming Language**: Python 3.10
- **Libraries/Frameworks**:
  - TensorFlow and Keras: Model development and training.
  - Tkinter: GUI implementation.
  - NumPy and OpenCV: Data handling and preprocessing.
  - Matplotlib: Visualizations and debugging.
- **Dataset**: MNIST (Modified National Institute of Standards and Technology).

---

4. Implementation Details

4.1 Data Preprocessing

- MNIST images are flattened, normalized, and augmented to improve generalization.
- User-drawn images are converted to grayscale, resized to 28x28 pixels, and normalized.

4.2 Model Training

- CNN was trained using categorical cross-entropy loss and Adam optimizer.
- Batch size: 128
- Epochs: 10
- Validation Accuracy: 98.7% on the MNIST test set.

4.3 GUI Functionality

- Canvas with mouse event listeners to capture input.
- Preprocessing pipeline linked to the "Predict" button.
- Output displayed dynamically.

4.4 Testing and Evaluation

- Benchmarked model on MNIST with accuracy of 98.7%.

- Tested GUI inputs using various handwriting styles, achieving consistent predictions.

---

5. Usage

1. Run the `main.ipynb` file.
2. Draw a digit (0–9) on the canvas.
3. Press "Predict" to classify the digit.
4. Press "Clear" to reset the canvas.

Proposal

1. Objective

To develop a real-time digit recognition application that uses a Convolutional Neural Network (CNN) to classify handwritten digits drawn on a digital canvas.

---

2. Background and Motivation

Handwritten digit recognition is a classic problem in computer vision and machine learning. It forms the foundation for various real-world applications such as postal code sorting, bank check digitization, and accessibility tools for the visually impaired.
The proposed system bridges the gap between theoretical concepts and practical applications, offering an interactive demonstration of deep learning capabilities.

---

3. Methodology

3.1 Data Handling

- Use the MNIST dataset for training and evaluation.
- Preprocess user inputs to match the format of the MNIST data.

3.2 Model Development

- Design and train a CNN with layers optimized for feature extraction and classification.
- Evaluate and fine-tune the model for improved accuracy.

3.3 Interface Design

- Develop a GUI using Tkinter that allows users to draw digits interactively.
- Integrate prediction functionality for real-time feedback.

---

4. Expected Outcomes

1. A CNN model trained to classify digits with over 98% accuracy.
2. A functional GUI that supports user interaction and provides immediate feedback.
3. An educational tool showcasing machine learning in action.

Final Report

1. Abstract

This project explores the design and implementation of a Digit Recognition System, blending deep learning with graphical user interaction. Leveraging the MNIST dataset and a Convolutional Neural Network (CNN), the system achieves high prediction accuracy on handwritten digits and serves as a practical application of AI concepts.

---

2. Introduction

The ability to recognize handwritten digits is an essential milestone in computer vision. This project develops a comprehensive solution, encompassing a CNN trained on the MNIST dataset and a Tkinter-based GUI for user interaction. The system is an educational and functional tool demonstrating deep learning's potential.

---

3. System Design

3.1 Data Preprocessing

- MNIST dataset images were resized, normalized, and augmented.
- User-drawn inputs underwent similar preprocessing for seamless model integration.

3.2 Model Development

- CNN architecture included:
  - **Conv2D layers** for spatial feature extraction.
  - **MaxPooling layers** for dimensionality reduction.
  - **Dense layers** for classification.
- Training involved a learning rate scheduler and early stopping for optimization.

3.3 GUI Functionality

- A Tkinter-based canvas provided a user-friendly interface.
- Buttons for clearing the canvas and predicting the digit enhanced interactivity.

---

4. Results and Analysis

- **Training Results**: Achieved a 98.7% accuracy on the MNIST test set.
- **Real-Time Performance**: Maintained an average accuracy of 92.3% for user-drawn inputs.
- **Observations**:
  - Model performed well on clean, well-defined digits.
  - Variability in handwriting styles occasionally reduced accuracy.

---

5. Conclusion and Future Work

The Digit Recognition System is a successful demonstration of deep learning's practical applications. Future enhancements could include:

1. Support for multi-digit recognition.
2. Integration of additional datasets to improve model robustness.
3. Deployment as a web or mobile application.