CAPSTONE PROJECT

HANDWRITTEN DIGITS RECOGNIZER APP

PRESENTED BY

STUDENT NAME: SUDEEP SWARANKAR

COLLEGE NAME: C.V. RAMAN GLOBAL UNIVERSITY, BHUBANESHWAR

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID: sudeepswarankar@gmail.com

AICTE STUDENT ID: STU657b0f13a3fbf1702563603

OUTLINE

PROBLEM STATEMENT

- The objective of this project is to develop an accurate and efficient machine learning model that can automatically recognize handwritten digits from images using the MNIST dataset. Handwritten digit recognition is a fundamental task in image classification and has wide applications in fields like postal mail sorting, bank check processing, and digitized document analysis. This project aims to implement a Convolutional Neural Network (CNN) to classify grayscale images of digits (0–9) with high accuracy, leveraging deep learning techniques for feature extraction and pattern recognition.

PROPOSED SOLUTION

- The proposed system aims to address the challenge of accurately recognizing handwritten digits using a deep learning-based approach. This involves leveraging Convolutional Neural Networks (CNNs), a class of deep neural networks highly effective in image classification tasks. The solution will consist of the following components:
- Data Collection:
  - Utilize a publicly available dataset such as the MNIST dataset, which contains 70,000 labeled images of handwritten digits (0–9) in grayscale format.
  - Collect additional handwritten digit samples to enhance the model's generalization.
- Data Preprocessing:
  - Normalize pixel values and reshape the images to match the input format required by the CNN model.
  - Apply data augmentation techniques such as rotation, zooming, and shifting to enhance model robustness and prevent overfitting.
- CNN Algorithm:
  - Build an CNN using input, hidden, and output layers with activation functions like ReLU and softmax.
  - Train the model on labeled data and validate it using a separate validation set to monitor and improve performance.
- Deployment:
  - Develop a graphical user interface (GUI) or web-based application that allows users to input handwritten digits and receive real-time classification results.
  - Deploy the solution on a scalable and reliable platform, considering factors like server infrastructure, response time, and user accessibility.
- Evaluation:
  - Assess model performance using metrics like accuracy, precision, recall, and confusion matrix, and test it on unseen data for real-world applicability.
  - Fine-tune hyperparameters and retrain the model based on validation results and performance feedback to enhance accuracy.
- Result: :
  - The system will be capable of accurately recognizing and classifying handwritten digits from input images.
  - It will automate digit recognition tasks in various real-life applications, improving speed, accuracy, and efficiency in data processing systems.

SYSTEM APPROACH

- System requirements:
  - **Hardware:**
    - A computer with at least 4 GB RAM (8 GB or more recommended)
    - GPU (optional but recommended for faster training)
    - Minimum 10 GB of free storage space
  - **Software:**
    - Operating System: Windows, macOS, or Linux
    - Python (version 3.10or above)
    - Jupyter Notebook or any Python IDE (e.g., VS Code, Jupyter Notebook)

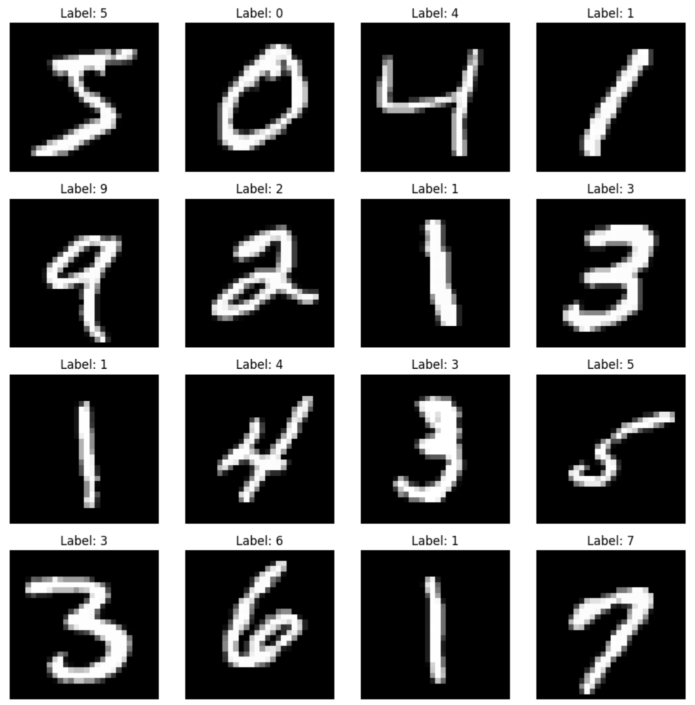
- Library required to build the model:
  - NumPy-numerical operations
  - Pandas-data handling
  - Matplotlib/Seaborn-visualization
  - TensorFlow/Keras-to build and train CNN model
  - Scikit-learn-for preprocessing and evaluation
  - OpenCV-for image handling and custom digit input preprocessing

ALGORITHM & DEPLOYMENT

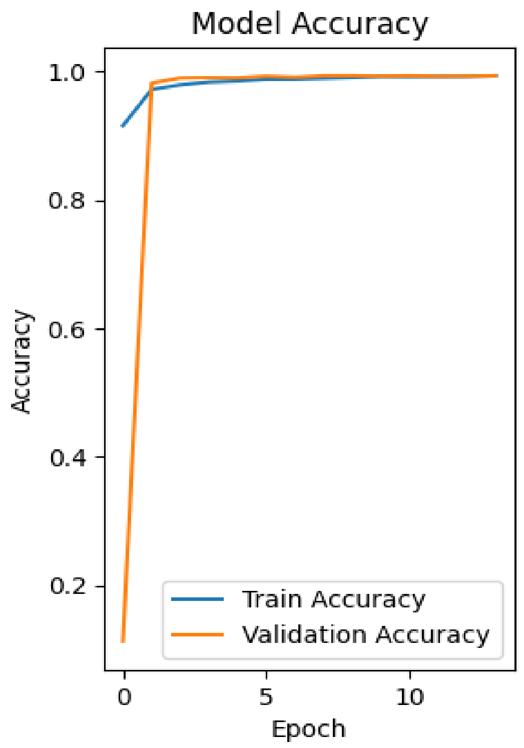
- **Algorithm Selection:**
  - A Convolutional Neural Network (CNN) was selected for the handwritten digit recognition task due to its effectiveness in capturing spatial patterns in image data. Unlike traditional Artificial Neural Networks (ANNs), CNNs are designed to recognize local features through convolutional layers, making them highly suitable for image classification tasks. The MNIST dataset, composed of 28x28 grayscale images of handwritten digits, benefits from this approach as CNNs can automatically learn features such as edges, curves, and shapes, which are essential for distinguishing between digits.
- **Data Input:**
  - The model uses images from the MNIST dataset, each a 28x28 grayscale image of a digit (0–9). Pixel values are normalized to the range [0, 1] for efficient training. Unlike ANN, the images are reshaped to (28, 28, 1) to preserve spatial information for CNN processing. Labels are one-hot encoded to support multi-class classification.
- **Training Process:**
  - The CNN model is trained using the Adam optimizer with categorical cross-entropy as the loss function. The architecture includes convolutional layers with ReLU activation, followed by max pooling layers for spatial reduction. Dropout is applied to prevent overfitting, and the final dense layer uses softmax to output class probabilities. Early stopping and model checkpointing are used to monitor validation performance and save the best model.
- **Prediction Process:**
  - After training, the CNN model predicts new digit images by first normalizing and reshaping them to (28, 28, 1), just like the training data. The model then outputs probability scores for all 10 digit classes, and the class with the highest score is chosen as the final prediction.

VISUALIZATIONS

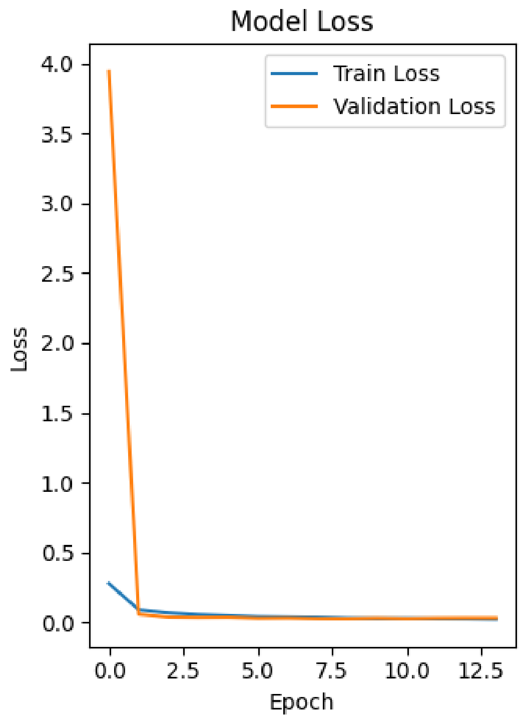
- Visualization of dataset images:



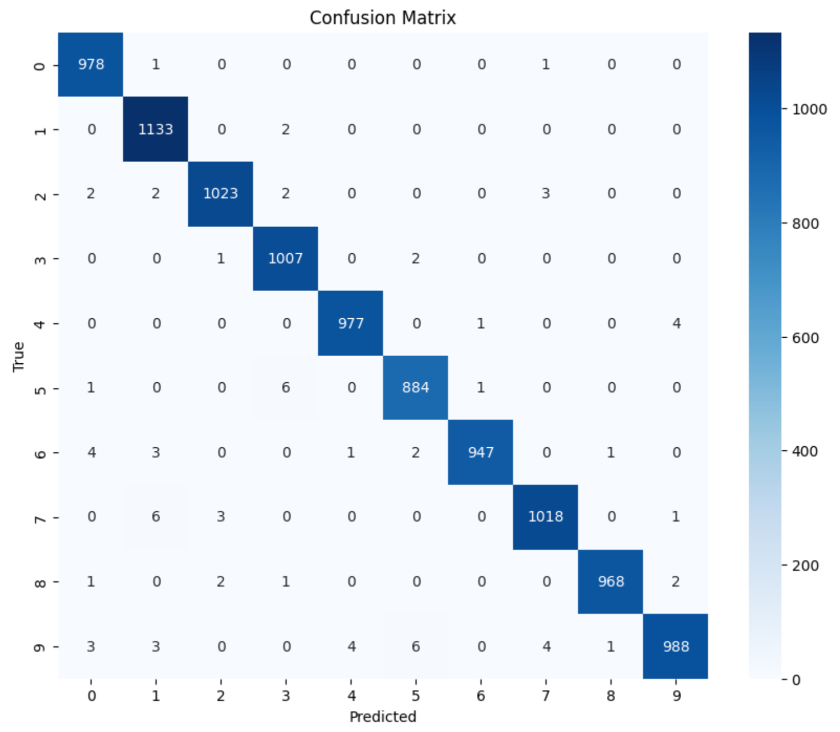
- Train and validation accuracy graph:



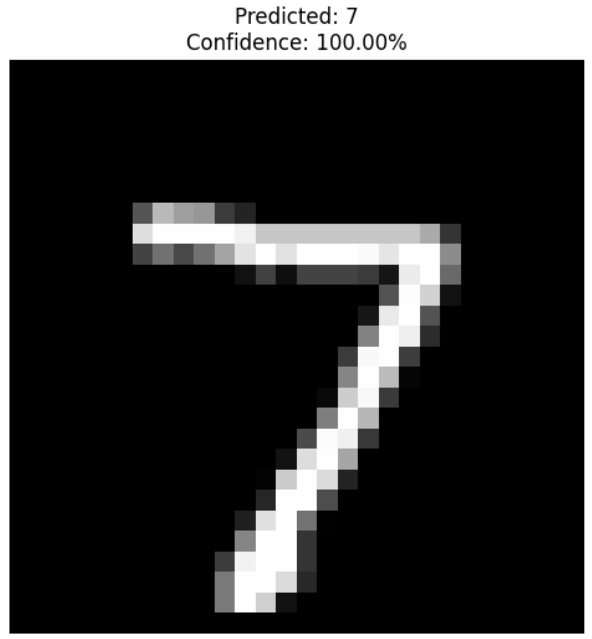
- Train and validation loss graph:

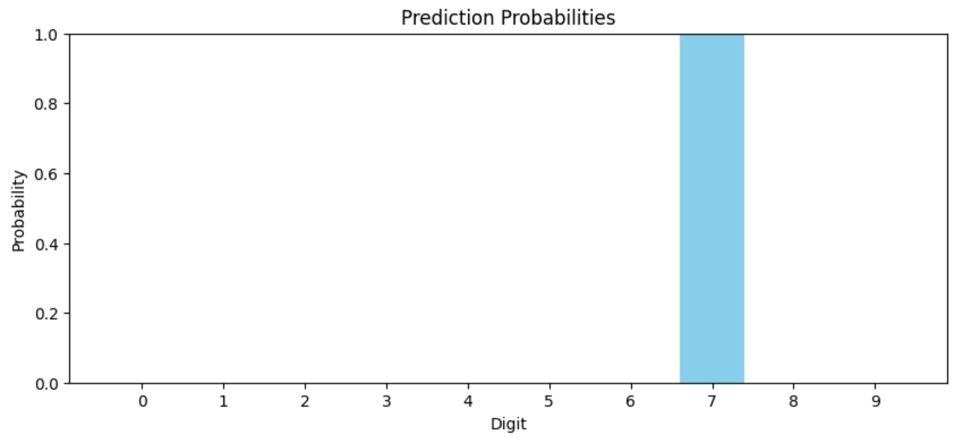


- **Confusion Matrix:**



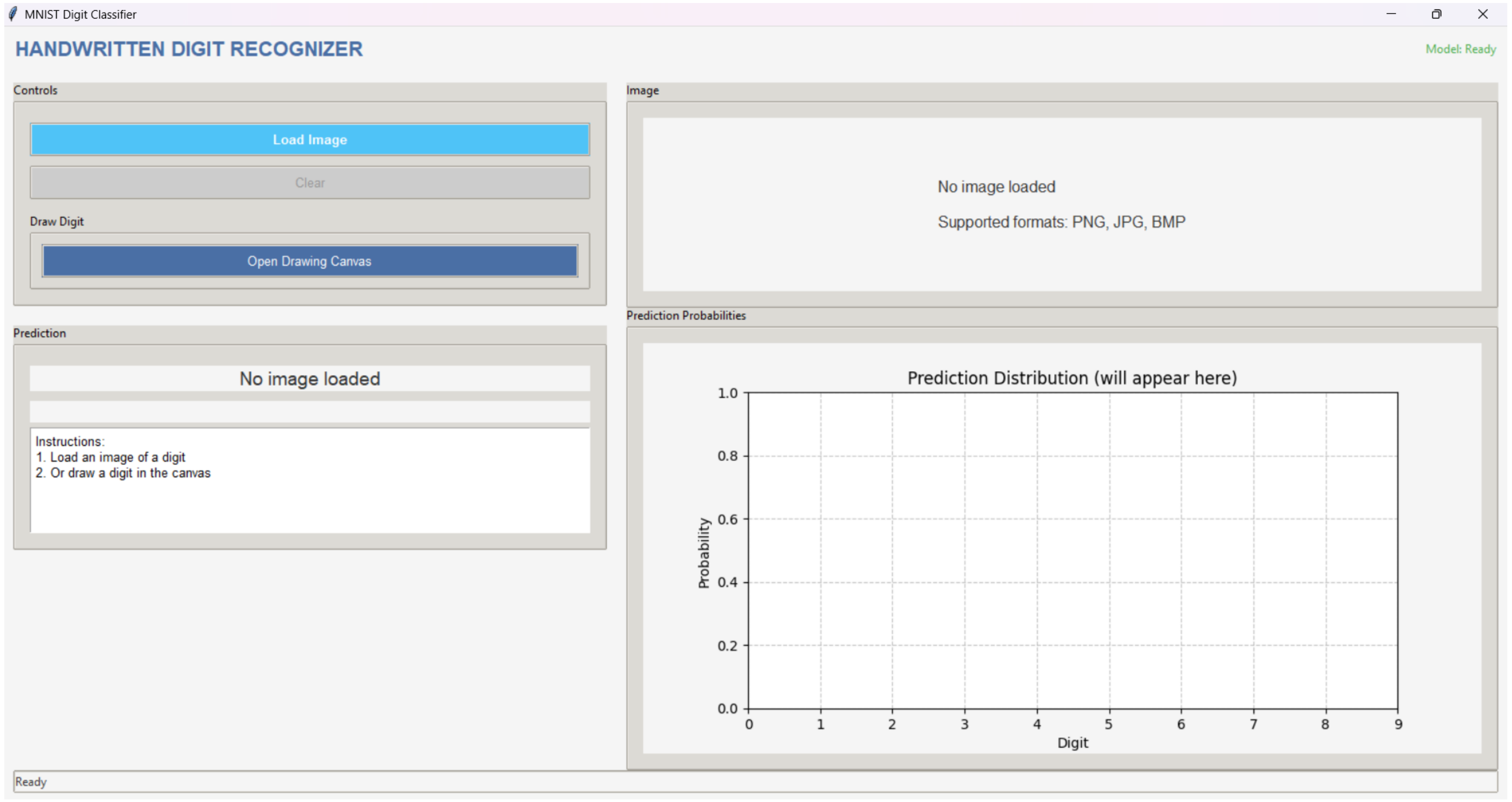
- **Digit Prediction, Confidence and Probability:**





RESULT

App Interface:



MNIST Digit Classifier

HANDWRITTEN DIGIT RECOGNIZER

Model: Ready

Controls

Load Image

Clear

Draw Digit

Open Drawing Canvas

Image

No image loaded

Supported formats: PNG, JPG, BMP

Prediction

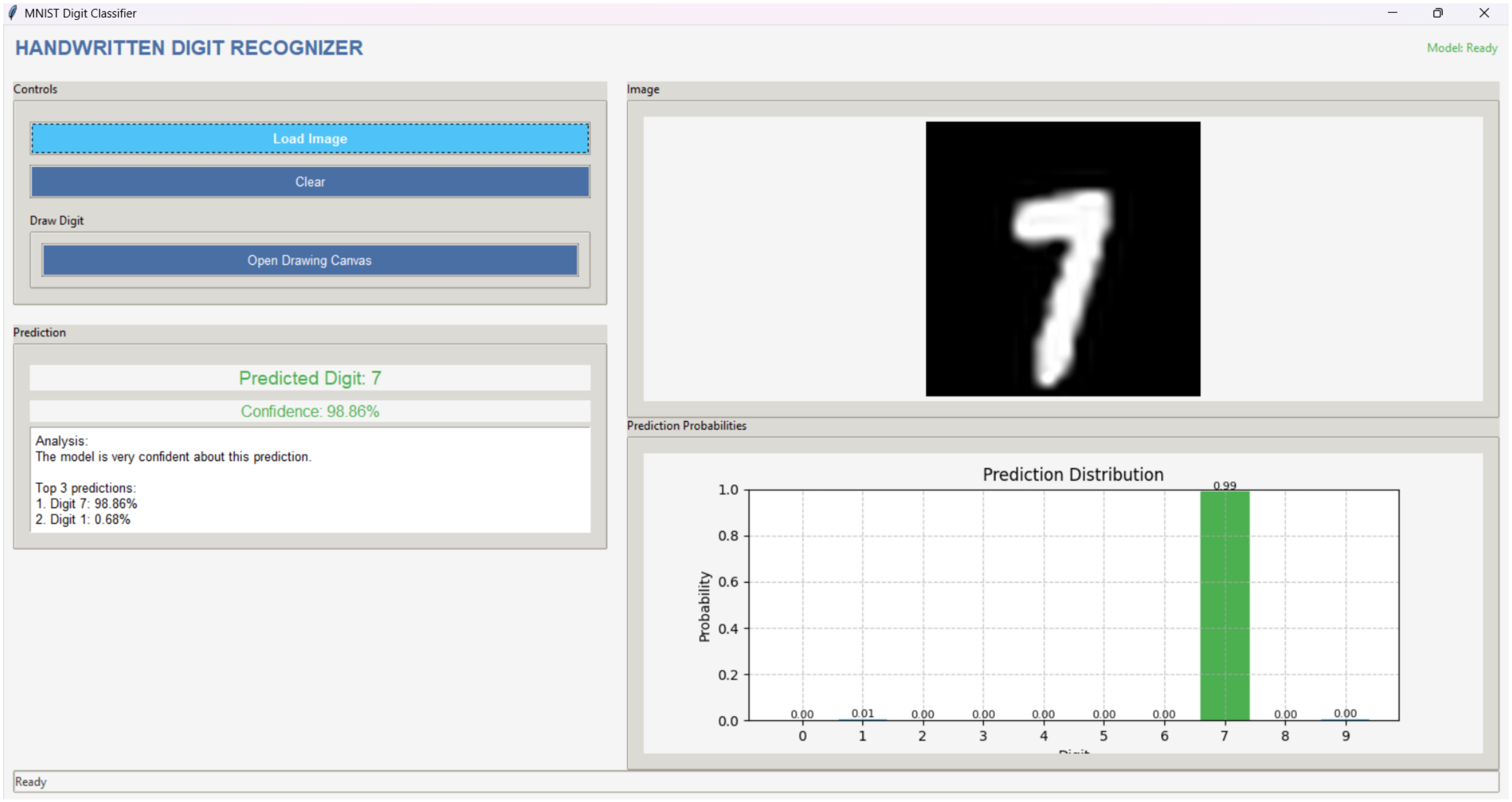
No image loaded

Instructions:
1. Load an image of a digit
2. Or draw a digit in the canvas

Prediction Probabilities

Ready

Prediction by loading an image:


MNIST Digit Classifier

HANDWRITTEN DIGIT RECOGNIZER

Model: Ready

Controls

Load Image

Clear

Draw Digit

Open Drawing Canvas

Prediction

Predicted Digit: 7

Confidence: 98.86%

Analysis:
The model is very confident about this prediction.

Top 3 predictions:
1. Digit 7: 98.86%
2. Digit 1: 0.68%

Image

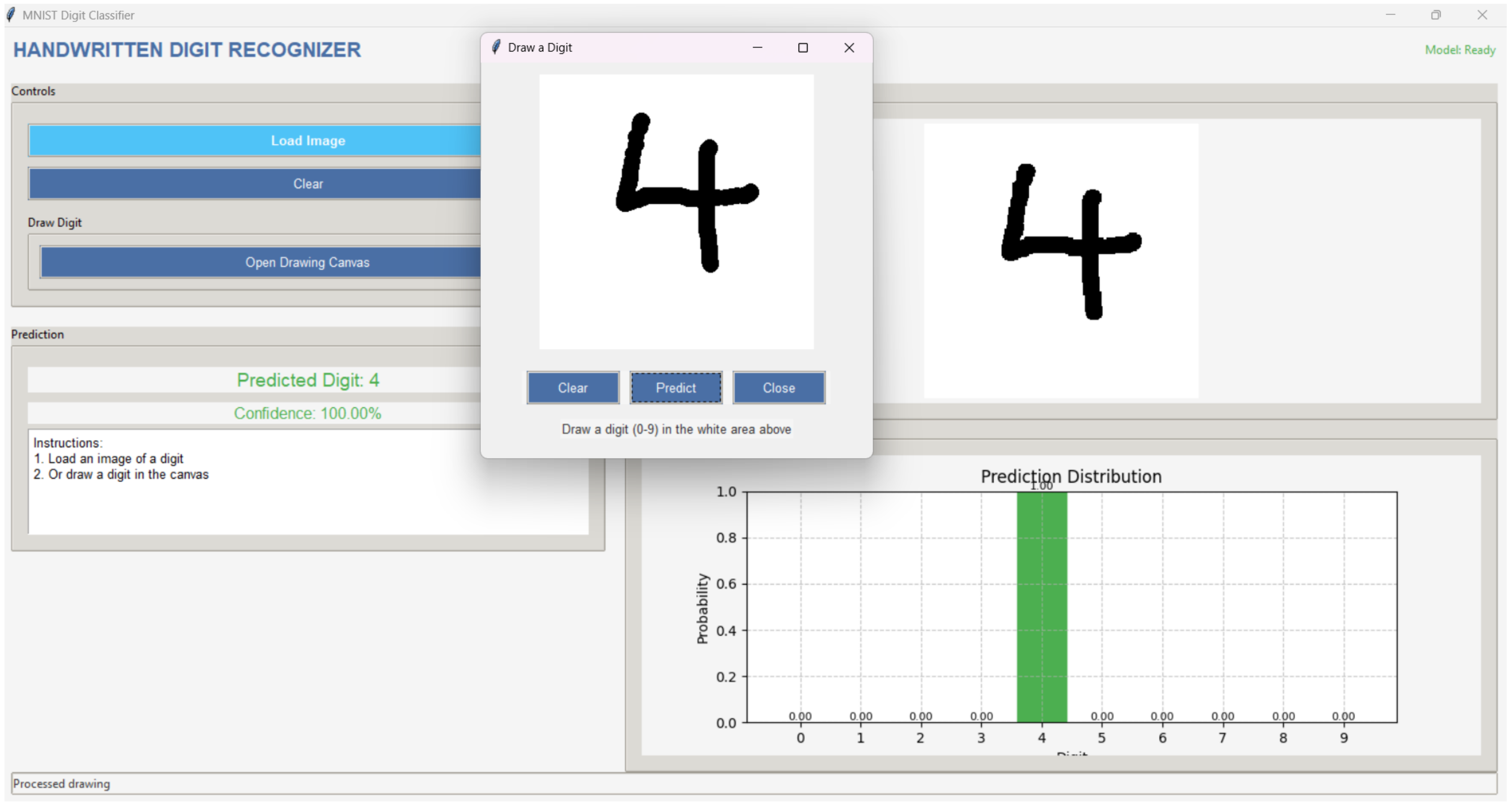
Prediction Probabilities

Prediction Distribution

Digit	0	1	2	3	4	5	6	7	8	9
Probability	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00

Ready

Prediction by drawing a digit in canvas:



Predicted Digit: 4

Confidence: 100.00%

Instructions:
1. Load an image of a digit
2. Or draw a digit in the canvas

Draw a digit (0-9) in the white area above

Digit	0	1	2	3	4	5	6	7	8	9
Probability	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00

Processed drawing

FEATURES OF THE APP

- Application Purpose:
  - The app is designed to classify handwritten digits (0–9) using a trained deep learning model.
- GUI Framework:
  - Built using **Tkinter** for the user interface. Styled with the ttk.Style configuration to enhance visual appearance.
- Image Input Options:
  - Likely supports image file uploads and freehand drawing using ImageGrab and PIL.Image.
- Image Preprocessing:
  - Prepares the input image to match the model’s expected input format (28x28 pixels).
- Prediction Display:
  - Shows the predicted digit on the GUI. Includes confidence scores or visual plots using matplotlib and seaborn.
- Visualization Features:
  - Embeds Matplotlib plots in the Tkinter window using FigureCanvasTkAgg..Used for displaying probability distributions or image visualizations.
- Interactive Feedback:
  - Uses messagebox to display user messages or warnings. Responsive buttons with hover and click effects using style mapping.
- Loading Screen:
  - Implements a loading screen while the model is being initialized.

CONCLUSION

The implemented CNN model successfully achieved high accuracy in recognizing handwritten digits from the MNIST dataset. By using convolutional layers for feature extraction and techniques like dropout and early stopping to prevent overfitting, the model demonstrated strong generalization on unseen data. This project highlights the effectiveness of CNNs in image classification tasks and provides a robust foundation for building more advanced computer vision systems. The trained model can be deployed in real-world applications, offering real-time digit recognition through both local and web-based interfaces.

FUTURE SCOPE

- **Model Enhancement:**
  - The current CNN model can be further improved by experimenting with deeper architectures or using pre-trained models through transfer learning to achieve higher accuracy and better feature extraction.
- **Custom Dataset Integration:**
  - The system can be extended to support handwritten digit recognition in other languages or on real-world documents by training on custom datasets beyond the MNIST dataset.
- **Mobile and IoT Integration:**
  - The model can be optimized for deployment on mobile devices and embedded systems like Raspberry Pi, enabling offline and portable digit recognition applications.
- **Multi-character Recognition:**
  - Future versions can be designed to recognize full handwritten strings or alphanumeric sequences, making the solution suitable for reading forms, postal codes, license plates, and more.

REFERENCES

GitHub Link: https://github.com/SudeepSwarankar/Handwritten-Digit-Classifier-APP

Ghost Script Link: https://www.ghostscript.com/

(*NOTE-Please install Ghost Script to use 'Drawing canvas' of the app properly. Please add path of Ghost Script bin folder in environment variable)

- https://scikit-learn.org/stable/
- https://docs.python.org/3/library/tkinter.html
- https://pandas.pydata.org/docs/
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*.
- James, G., et al. (2013). *An Introduction to Statistical Learning*. Springer
- Python Software Foundation. Tkinter documentation.
- Seaborn, Matplotlib, and Pandas official documentation.

Thank you