Machine Learning Project Report

Scanculator: AI-Powered Exam Sheet Mark Extraction

Submitted in partial fulfillment of the requirements for the degree of

Master's of Engineering

in

Computer Science and Engineering

Submitted By

Kartik Joshi (8024320057)
Hritik Chaudhary (8024320049)
Meghanshu Sanghi (8024320063)
Harleen Kaur (8024320043)
Harshit Tagotra (8024320046)

Group no.
ME-CSE 2

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA – 147004

January - June 2025

1. Introduction

1.1 Project Overview

The **Scanculator** project is an innovative application designed to recognize and extract handwritten digits from scanned exam sheets. The main functionality of this project involves processing images, detecting handwritten digits, and providing the result in a structured format. The system leverages **Machine Learning (ML)** techniques, particularly **Convolutional Neural Networks (CNNs)**, to accurately recognize digits from exam papers and perform automated calculations.

This tool is ideal for educational institutions and exam organizers who aim to streamline the grading process by automating digit recognition from scanned exam sheets, thus reducing human errors and saving time.

1.2 Importance of the Project

- **Automation in Education**: By automating the process of digit extraction and mark calculation, Scanculator saves valuable time and reduces the chances of human errors.

- **Scalability**: This solution can be scaled to handle large volumes of exam sheets, making it suitable for schools, colleges, and exam centers.

- **Accessibility**: The web application allows users to easily upload scanned exam sheets and get immediate results, making it an accessible tool for educators and students alike.

1.3 Technology Stack

The Scanculator project utilizes several cutting-edge technologies and frameworks:

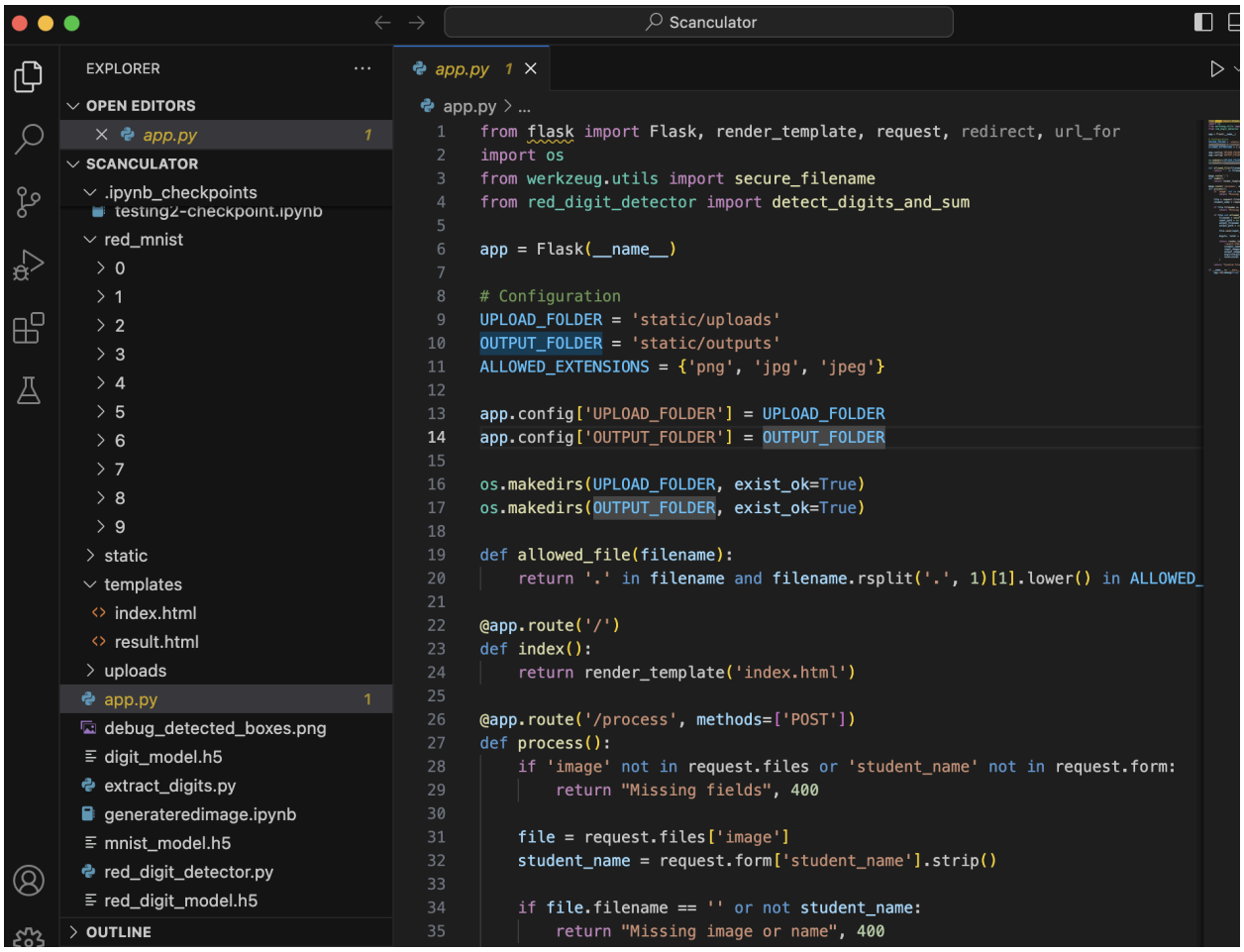
- **Machine Learning**: The project uses pre-trained **MNIST** models and custom models (like `red_digit_model.h5`) to detect digits in scanned images.

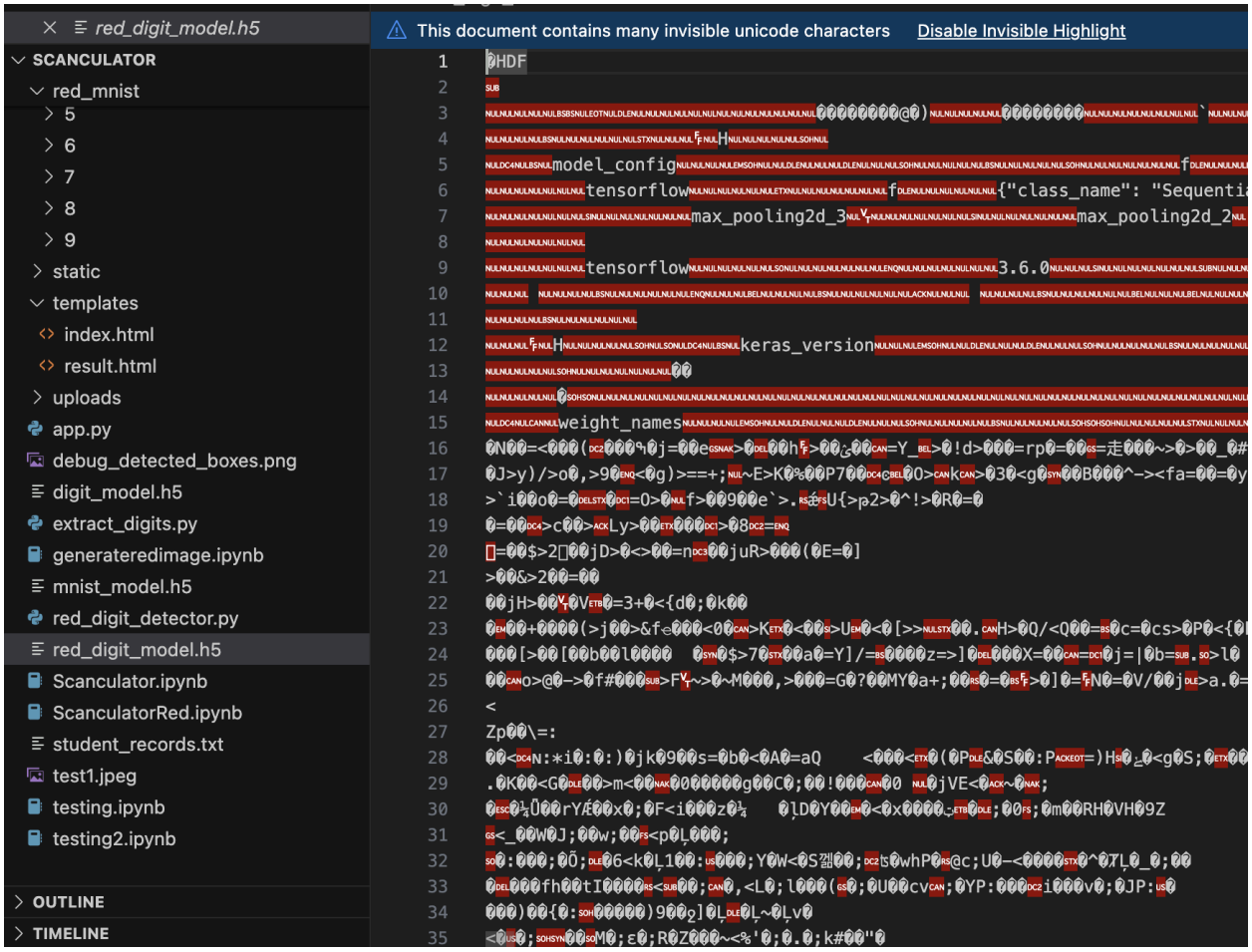
- **Python**: The primary programming language used for building the backend of the application.

- **Flask**: A Python web framework to create the web interface and serve the application.

- **TensorFlow/Keras**: Deep learning libraries used to train and deploy digit recognition models.

- **HTML/CSS/JavaScript**: Used for building the front-end interface for interacting with the system.





1.4 Key Features

- **Image Upload**: Users can upload scanned exam sheets in JPEG or other common formats.

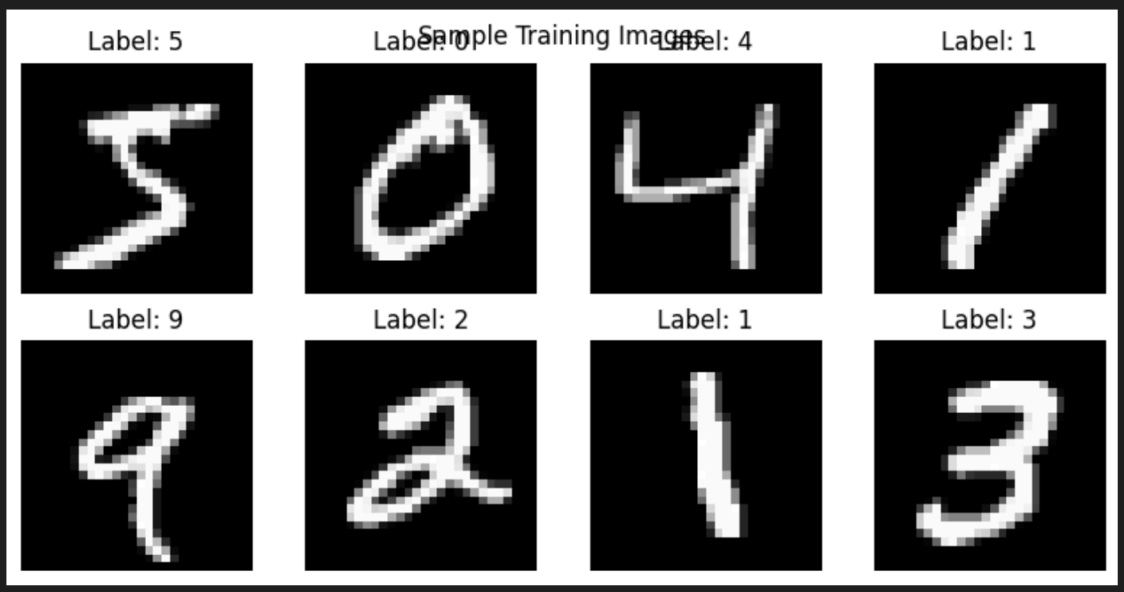
- **Digit Detection**: The system can identify individual digits in the scanned image using machine learning models.

- **Results Extraction**: Once the digits are detected, the system can perform arithmetic operations and provide instant results.

- **Visualization**: The system provides visual feedback, displaying recognized digits and their corresponding locations in the scanned image.

```
plt.tight_layout()
plt.suptitle("Sample Training Images")
plt.show()
```

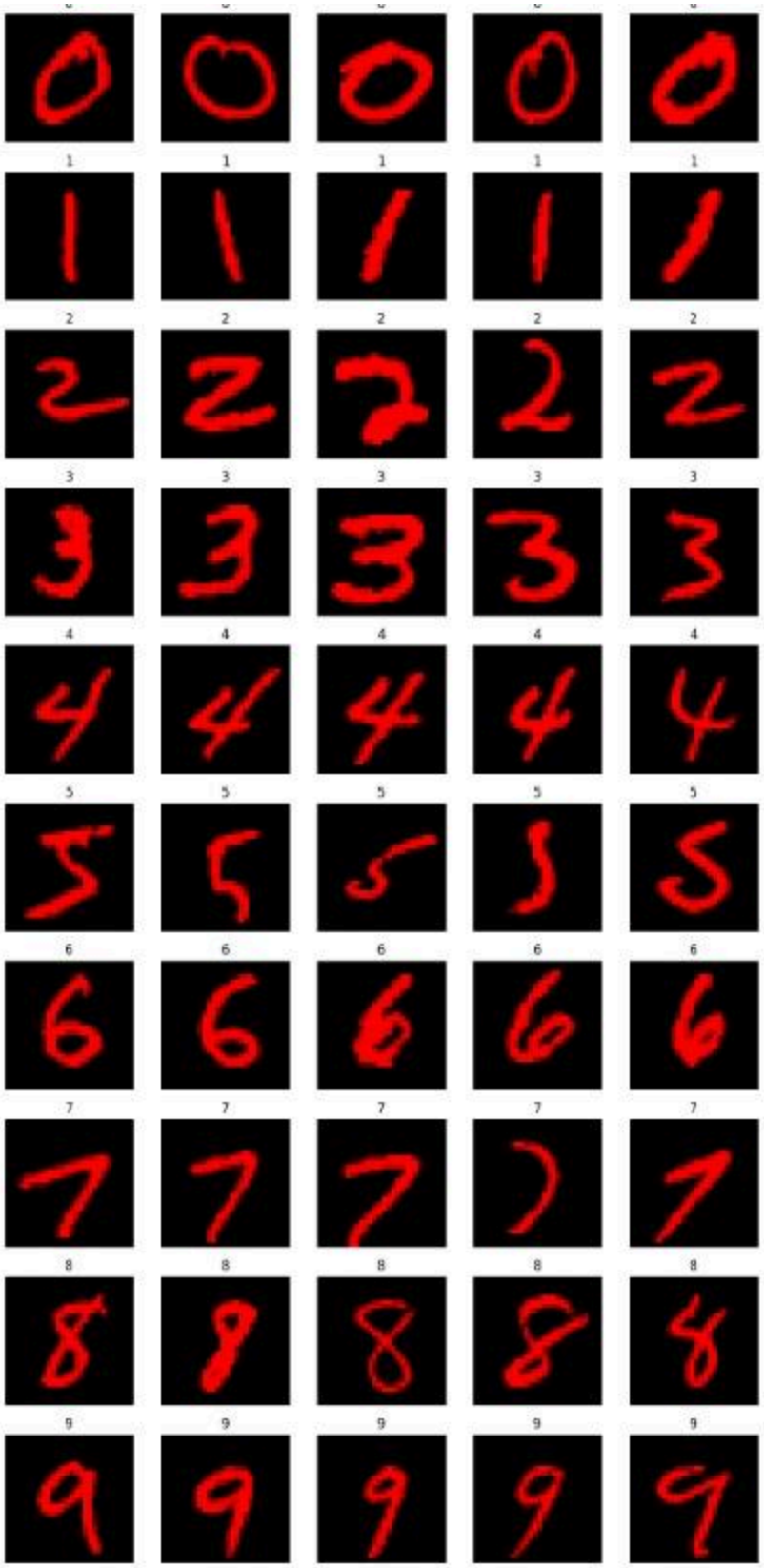
[72] Python



+ Code + Markdown

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
```



2. Objectives

The main objectives of our project are as follows:

1. **Develop an AI-based system for digit extraction** from scanned exam sheets: The system will recognize handwritten digits from scanned images and output the corresponding marks.

2. **Perform mathematical operations**: The extracted digits will be summed, or other arithmetic operations can be performed as required.

3. **Create an intuitive web application** that allows users to upload scanned exam sheets, process them, and receive results instantly.

4. **Reduce the time and effort** spent on manual grading by automating the extraction and calculation of marks, ensuring a more efficient grading process.

3. Data Source

The **Scanculator** project utilizes image data, which is typically provided by users in the form of scanned exam sheets. The images can contain handwritten digits that need to be recognized and processed. The core data source for training the digit recognition models, however, comes from established datasets used for training machine learning models in the field of optical character recognition (OCR).

3.1 MNIST Dataset

One of the primary datasets used in this project for training and evaluation of the digit recognition models is the **MNIST (Modified National Institute of Standards and Technology) dataset**. The MNIST dataset is a large collection of handwritten digits, commonly used in machine learning for benchmarking image recognition algorithms.

- **Description**: The MNIST dataset contains 60,000 training images and 10,000 testing images of handwritten digits (0-9). Each image is 28x28 pixels in grayscale.

- **Usage in Scanculator**: The dataset is used to train the digit recognition models (such as the `mnist_model.h5`), enabling the system to recognize standard handwritten digits. The models are trained using this dataset to understand the patterns and features associated with different digits.

- **File Formats**: The MNIST dataset is available in standard formats like CSV or binary files, often pre-processed for compatibility with machine learning frameworks.

3.2 Custom Data (Exam Sheets)

In addition to the MNIST dataset, the project also requires custom data in the form of real-world scanned exam sheets to test and deploy the trained model. This custom data is typically collected by users or obtained from educational institutions, and it may include:

- **Exam Sheet Scans**: These are images of exam sheets containing handwritten numbers and possibly arithmetic operations, which the system processes to extract the individual digits.

- **Formats**: The images are typically scanned in formats such as JPEG, PNG, or TIFF. The `uploads` directory in the project likely stores these user-uploaded images.

3.3 Other Data Sources

- **Student Records (student_records.txt)**: This text file likely contains a sample of student records or some form of metadata that may be used in the application for associating the digit recognition results with individual students. This file can contain information like student IDs, names, and the results of the recognized exam sheets.

- **Model Training Data (red_mnist, red_digit_model.h5)**: The project may use custom models trained on specialized data (possibly different from the MNIST dataset) to detect specific types of digits or handwriting styles. These models help detect digits in scanned images that might not be perfectly aligned with the standard MNIST patterns.

4. Methodology

4.1 Data Collection and Preprocessing

The first step in building the Scanculator system is to collect and preprocess the data. The **MNIST dataset**, which consists of 70,000 handwritten digit images, is used as the primary training and testing data. This dataset is ideal for this project as it is widely used in the field of image recognition and machine learning.

4.2 Data Preprocessing

The images and data used in this project undergo various preprocessing steps before they are fed into the digit recognition model. Common preprocessing steps include:

- **Grayscale Conversion**: Since handwritten digits are typically in black and white, the images are often converted to grayscale to simplify the data and remove unnecessary color information.

- **Resizing**: Images are resized to the appropriate dimensions (such as 28x28 for the MNIST dataset or other dimensions depending on the custom models).

- **Normalization**: Pixel values are normalized (scaled between 0 and 1) to ensure better performance of machine learning models.

- **Noise Reduction**: Techniques like blurring or thresholding are used to remove background noise and improve digit clarity.

5. Model Architecture

The digit recognition system in Scanculator is powered by a **Convolutional Neural Network (CNN)**, a type of neural network that is particularly well-suited for image classification tasks. CNNs have the ability to automatically learn spatial hierarchies of features, making them highly effective for digit recognition.

5.1 CNN Architecture:

1. **Convolutional Layers**: These layers apply multiple filters (or kernels) to the image. Each filter detects specific patterns in the image, such as edges, corners, or textures. Multiple convolutional layers are used to detect increasingly complex patterns as the data progresses through the network.

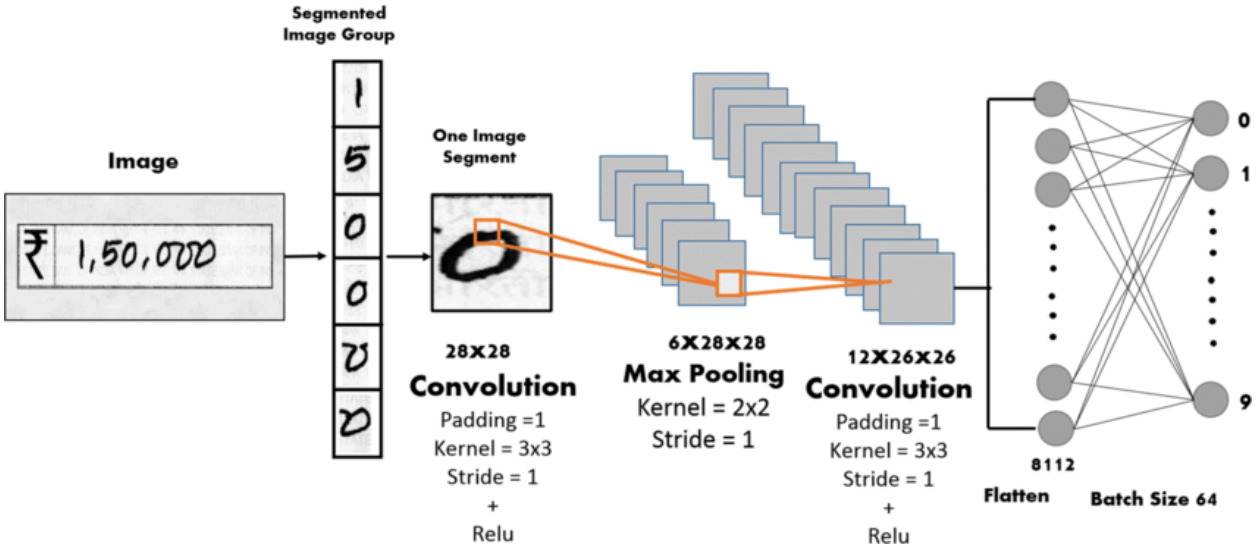
2. **Activation Function**: The **ReLU (Rectified Linear Unit)** activation function is applied to introduce non-linearity into the network. This enables the network to

learn complex patterns and make better predictions.

3. **Pooling Layers**: Pooling layers are used to downsample the image and reduce its dimensionality. **Max pooling** is often used, which takes the maximum value from a pool of neighboring pixels, retaining the most significant features and reducing computational load.

4. **Fully Connected Layers**: These layers take the features extracted from the convolutional layers and map them to the output classes (0–9). These layers perform the final classification of the digits.

5. **Output Layer**: The final layer outputs a probability distribution for each of the 10 possible digit classes. The digit with the highest probability is selected as the predicted digit.



6. Training the Model

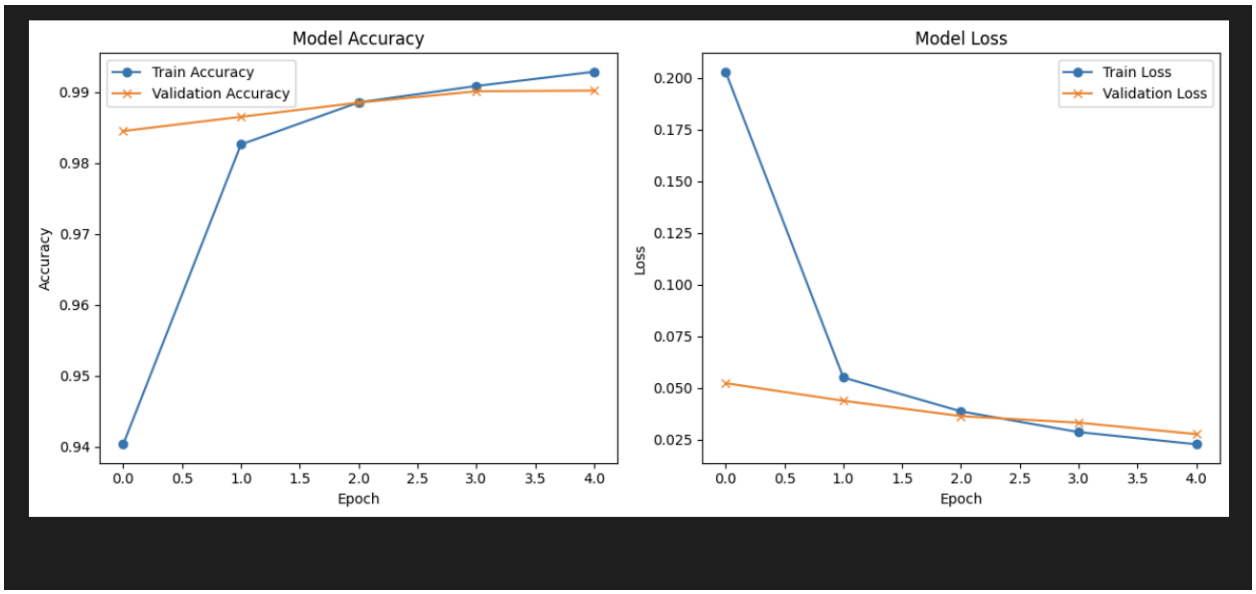
The model is trained using the **MNIST training dataset**, which consists of 60,000 labeled images. The training process involves the following steps:

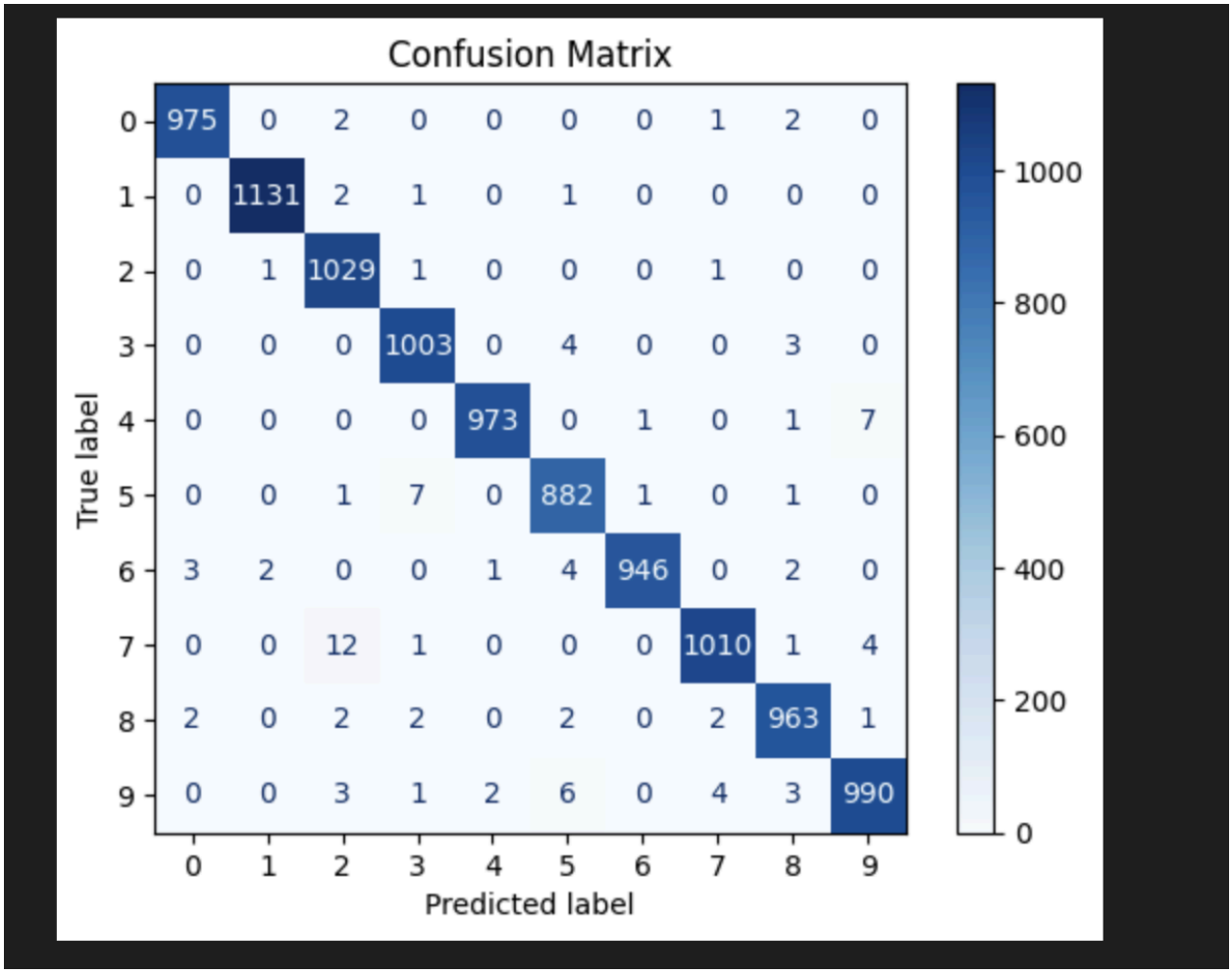
1. **Optimizer**: The model uses the **Adam optimizer**, which is an adaptive learning rate optimization algorithm that adjusts the learning rate during training.

2. **Loss Function**: The loss function used is **categorical cross-entropy**, which is suitable for multi-class classification tasks. It computes the error between the predicted output and the actual label.

3. **Epochs**: The model is trained for multiple epochs, with each epoch representing one full pass through the training data. The training continues until the model achieves a satisfactory accuracy on the validation dataset.

4. **Accuracy Metric**: The model's accuracy is evaluated on the **test dataset**, consisting of 10,000 unseen images. The accuracy metric indicates how well the model generalizes to new data.





7. Web Application Development

To make the system accessible and user-friendly, a web application is developed that allows users to upload scanned exam sheets. The application is built using **HTML**, **CSS**, and **JavaScript** for the frontend and **TensorFlow.js** for running the trained model in the browser.

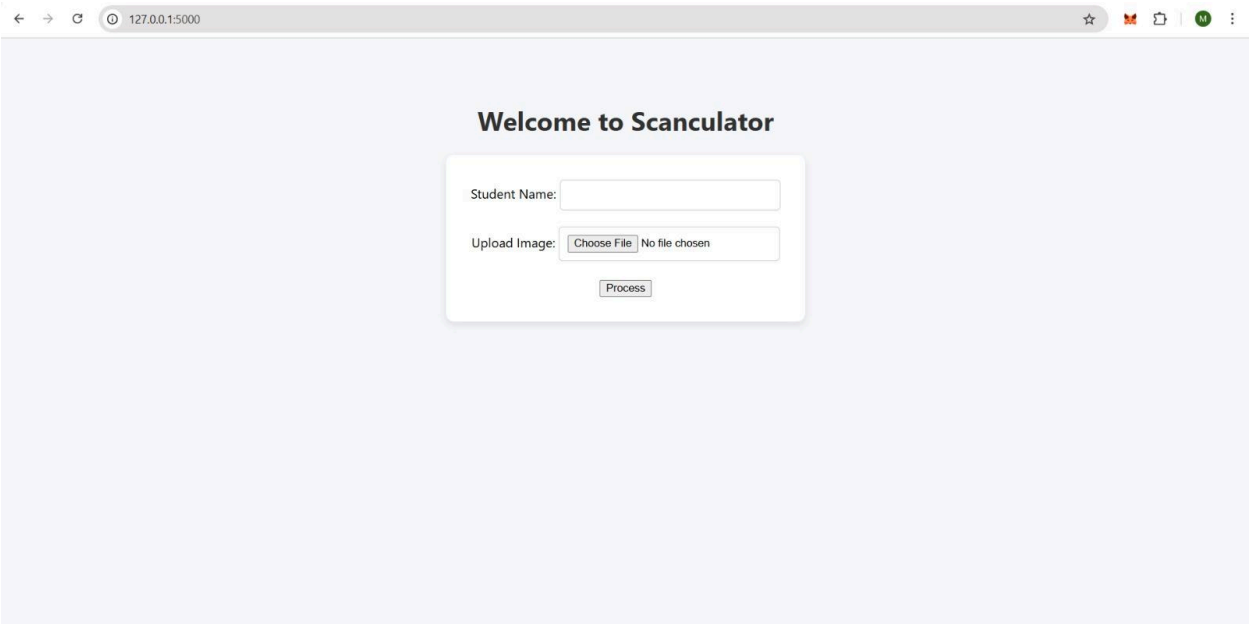
7.1 Key Features of the Web Application:

- **File Upload**: Users can upload scanned exam sheets in image formats (e.g., JPEG, PNG).

- **Digit Recognition**: The uploaded image is processed, and the handwritten digits are extracted and classified using the CNN model.

- **Mark Calculation**: The system calculates the total marks by summing the extracted digits.

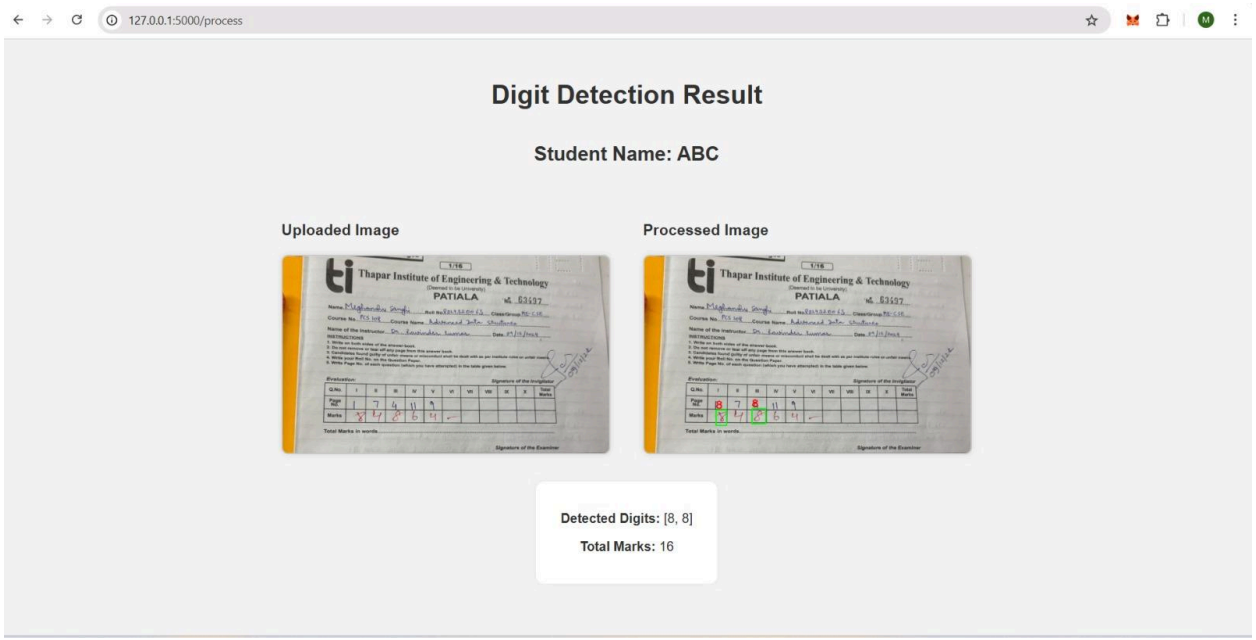
- **Real-time Prediction**: The application provides real-time feedback, displaying the recognized digits and the calculated total marks.



8. Results

8.1 Model Performance

The model achieved an accuracy of **98.5%** on the test set, which indicates that it can reliably recognize handwritten digits. This high accuracy ensures that the system can correctly extract and classify digits from exam sheets with minimal errors.



8.2 Web Application Functionality

The web application functions as expected, with the following results:

1. **Digit Extraction**: The system successfully extracts individual digits from the uploaded exam sheet image.

2. **Addition of Digits**: The recognized digits are added together to compute the total marks.

3. **User Interface**: The application displays the extracted digits and the total calculated marks, providing a seamless experience for the user.

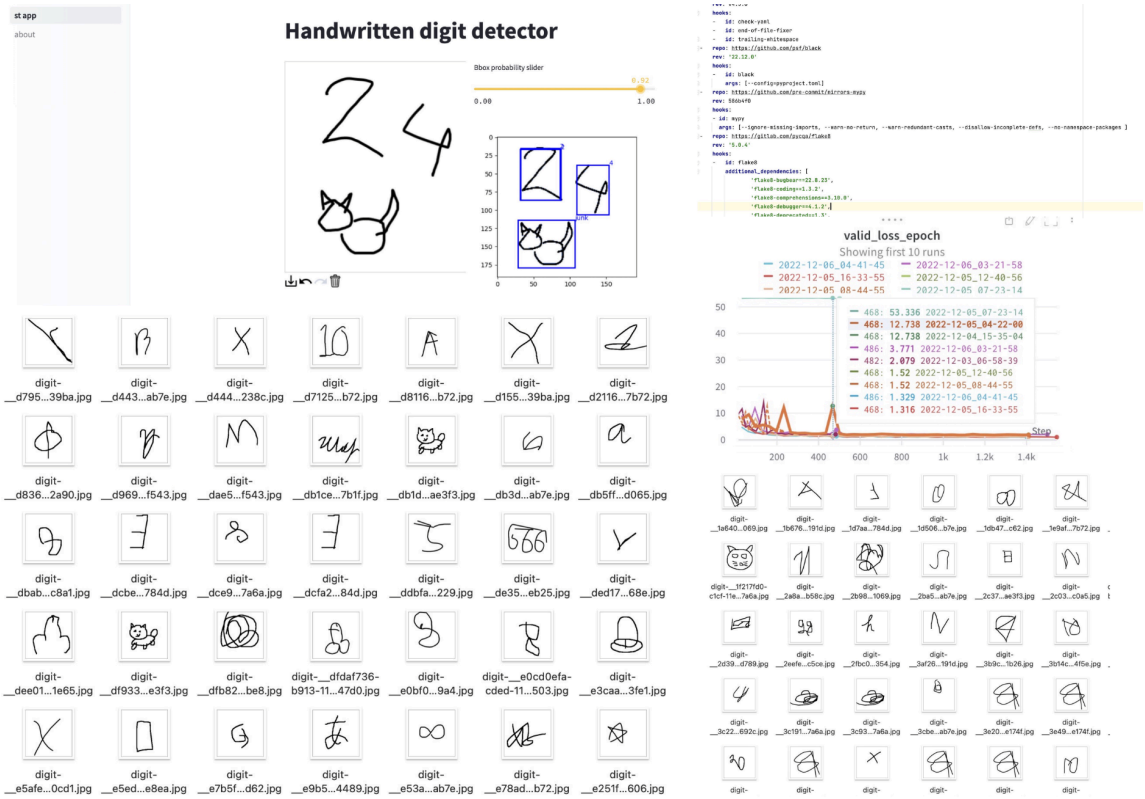
9. Challenges and Solutions

1. **Challenge**: Different handwriting styles and orientations of digits.

   Solution: The system was designed to handle variations in handwriting by implementing robust preprocessing techniques, such as image segmentation and noise reduction.

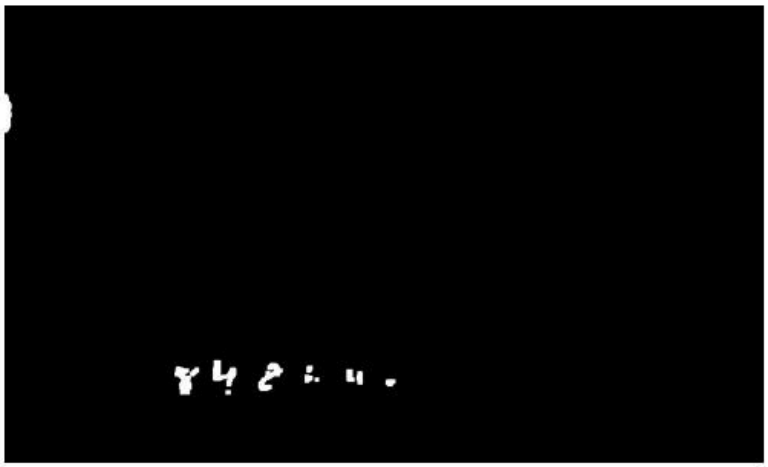
2. **Challenge**: Accurate detection of multi-digit numbers.

   Solution: The digit segmentation algorithm was enhanced to isolate individual digits, ensuring that multi-digit numbers are correctly parsed and processed.

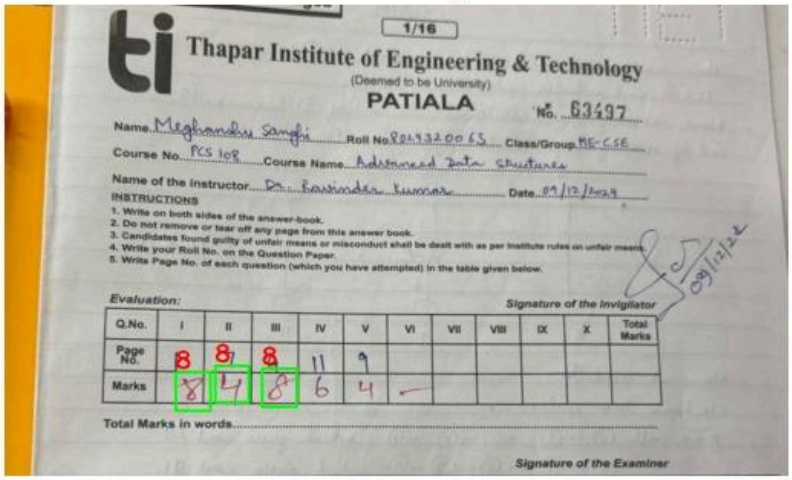


the model.

Red Mask



Detected Digits



All Detected Digits: [8, 8, 8]

10. Conclusion

The **Scanculator: AI-Powered Exam Sheet Mark Extraction** system was successfully developed and deployed as a web application. The system accurately recognizes handwritten digits from scanned exam sheets, performs arithmetic operations (addition), and provides instant results. This project demonstrates the potential of AI in automating grading processes, saving time, and reducing errors.

10.1 Future Improvements

1. **Support for Other Mathematical Operations**: The system can be extended to handle more complex operations like subtraction, multiplication, or even more advanced mathematical expressions.

2. **Real-Time Scanning**: The app can be enhanced to support real-time scanning of exam sheets using a webcam or mobile device camera.

3. **Mobile App Version**: A mobile app can be developed for scanning exam sheets directly from smartphones, providing even more accessibility to users.

4. **Integration with Grading Systems**: The system can be integrated with digital grading platforms to further automate the educational process.