REAL TIME HAND GESTURE RECOGNITION FOR MATHEMATICAL CALCULATIONS

A PROJECT REPORT

Submitted by

Sajal Korde - 221B319

Shruti Bhargava - 221B374

Snehil Sharma - 221B387

Under the guidance of: Prof. Sanjay Garg

MAY - 2025

Submitted in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY, AB ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA

DECLARATION BY THE STUDENTS

We hereby declare that the work reported in the B. Tech. project entitled as "**Real Time Hand Gesture Recognition for Mathematical Calculations**", in partial fulfillment for the award of degree of Bachelor of Technology submitted at **Jaypee University of Engineering and Technology, Guna**, as per best of our knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation, we will solely be responsible.

Sajal Korde (221B319)

Shruti Bhargava (221B374)

Snehil Sharma (221B387)

Department of Computer Science and Engineering

Jaypee University of Engineering and Technology

Guna, M.P., India

Date: 05/05/2025

JAYPEE UNIVERSITY OF ENGINEERING &TECHNOLOGY
Grade 'A+' Accredited with by NAAC & Approved U/S 2(f) of the UGC Act, 1956
A.B. Road, Raghogarh, Dist.: Guna (M.P.) India, Pin-473226
Phone: 07544 267051, 267310-14, Fax: 07544 267011
Website: www.juet.ac.in

CERTIFICATE

This is to certify that the work titled "**Real Time Hand Gesture Recognition for Mathematical Calculations**" submitted by "**Sajal Korde (221B319), Shruti Bhargava (221B374), Snehil Sharma (221B387)**" in partial fulfillment for the award of degree of B.Tech. of **Jaypee University of Engineering & Technology, Guna** has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.

Signature of Supervisor

Prof. Sanjay Garg

Head of Department (CSE)

Date: 05/05/2025

ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to Jaypee University of Engineering and Technology, our college, for giving us this chance to share our collective talents. We would also like to extend our sincere gratitude to Prof. Sanjay Garg, our project mentor and supervisor, who has supported us during all the challenges and issues we encountered enroute to finishing our project. We also want to express our gratitude to our mentor, who spent valuable time proofreading and fixing our mistakes in addition to offering us solutions and pointing us in the right direction. We also acknowledge the invaluable assistance provided by our parents and friends in helping us to complete this project on time. To cap it off, we extend our heartfelt thanks to the team members of "REAL TIME HAND GESTURE RECOGNITIION FOR MATHEMATICAL CALCULATIONS."

Sajal Korde (221B319)

Shruti Bhargava (221B374)

Snehil Sharma (221B387)

Date: 05/05/2025

EXECUTIVE SUMMARY

As technology advances, new methods for interacting with machines are emerging beyond traditional keyboards and touchscreens. Recognizing the challenges in expressing complex mathematical symbols quickly and efficiently, our project focuses on developing an AI-powered air-writing calculator that enables users to perform mathematical operations by writing in the air. By utilizing hand tracking and Convolutional Neural Networks (CNNs), we aim to create an intuitive tool that simplifies mathematical input. The motivation for this project arises from the difficulty of typing or verbally describing complex mathematical expressions such as integration symbols, differentiation operations, and matrices. These symbols are often not easily accessible on standard keyboards, and explaining them in words can be time-consuming. Our system addresses this challenge by allowing users to directly write mathematical symbols in the air, providing a faster and more natural means of input. Currently, the system recognizes basic operations such as addition, subtraction, multiplication, division, factorial, and square root. However, it is designed to be scalable, with the potential for future expansions to more advanced operations. To achieve this, we use hand tracking to monitor finger movements and create a virtual writing canvas. The captured gestures are processed through a CNN model trained to recognize mathematical symbols and numbers. Efficient hand landmark detection is implemented using MediaPipe, ensuring smooth gesture tracking and high accuracy. Our Streamlit-based interface integrates real-time camera feed, gesture visualization, expression building, and result display, offering a seamless user experience. The project has significant potential applications in educational environments, remote learning, technical presentations, and assistive technologies. It also emphasizes the growing importance of natural and contactless user interfaces. In summary, our AI-powered air-writing calculator provides an innovative, scalable solution to enhance mathematical input, making human-computer interaction more natural, efficient, and accessible.

List of Figures

Table of Contents

Chapter-1 Introduction

1.1 Problem Statement

Traditional input methods for complex mathematical expressions, such as keyboards or voice commands, are inefficient and cumbersome. Many mathematical symbols, like integration, differentiation, and matrices, are either difficult to access or require time-consuming searches. This creates barriers for students, professionals, and anyone working with advanced math, slowing down calculations and problem-solving.

Existing solutions, such as digital calculators and equation editors, still require users to navigate complex interfaces or input symbols manually, which can be tedious and error-prone. Additionally, these methods may not be accessible for individuals with disabilities or those who struggle with physical input devices.

There is a clear need for a more intuitive, efficient, and accessible way to input mathematical expressions. This project addresses that gap by using air-writing technology to recognize and process mathematical symbols in real-time, offering a faster and more natural alternative for mathematical input.

1.2 Motivation for Work

The motivation behind developing the AI-powered air-writing calculator arises from the need to improve the efficiency and accessibility of mathematical input. Traditional methods for entering complex mathematical symbols—whether through typing on a keyboard or using voice commands—are often inefficient, cumbersome, and time-consuming. This project is driven by the following core motivations:

- **Simplifying Mathematical Input**: Existing input methods for complex mathematical expressions are slow and prone to error. Many symbols and operations, such as integrals, derivatives, or matrices, are either hard to access or require tedious navigation through menus. Our aim is to create an intuitive, user-friendly system that allows individuals to input mathematical equations simply by writing in the air. This contactless method of inputting data makes mathematical problem-solving significantly faster, smoother, and more efficient, eliminating the need for manual searching or explaining of symbols.
- **Enhancing Productivity**: Students, professionals, researchers, and educators often waste valuable time searching for or explaining mathematical symbols, which can be frustrating and disruptive to their workflow. By enabling users to write directly in the air, our system

facilitates an uninterrupted flow of work. This enhancement in productivity allows for more time spent on solving problems rather than managing inputs, ultimately increasing efficiency and streamlining the learning or work process.

- **Accessibility and Inclusivity**: Many traditional mathematical input systems are not designed with accessibility in mind, leaving individuals with physical disabilities at a disadvantage. Typing or using voice commands is not always feasible for everyone. Our project focuses on inclusivity, providing a natural, gesture-based input method that is contactless, easy to use, and accessible to people with diverse physical abilities. This ensures that more individuals, regardless of their physical limitations, can engage with and solve mathematical problems effectively.
- **Harnessing Technological Advancements**: In today's world of rapid technological progress, we have an unparalleled opportunity to leverage cutting-edge tools such as artificial intelligence (AI), hand tracking, and Convolutional Neural Networks (CNNs) to revolutionize the way we interact with machines. Our project takes advantage of these advanced technologies to create a system that is both highly intuitive and scalable. By incorporating AI and hand tracking, we are developing a solution that provides real-time recognition of hand gestures, making it a versatile tool for a variety of users.
- **Usability and Practicality**: For any technological solution to be truly effective, it must seamlessly integrate into the user's routine. Our air-writing calculator is designed to be both easy to use and unobtrusive. The system is intuitive, providing users with an enhanced experience without requiring additional steps or distractions. The goal is to provide an interface that feels natural and is practical for users in real-world settings, minimizing the learning curve while maximizing utility.
- **Expanding the Horizon**: While the current version of the system supports basic mathematical operations such as addition, subtraction, multiplication, division, factorials, and square roots, the true potential lies in its scalability. Our system is designed to grow with users' evolving needs. Future updates will allow it to handle more advanced operations such as integration, differentiation, and matrix calculations, ensuring it remains a powerful and relevant tool for complex mathematical problem-solving in diverse fields.

1.3 Goal and Objectives

The primary goal of this project is to develop an AI-powered air-writing calculator that allows users to perform mathematical operations through hand gestures. The system will leverage hand

detection and Convolutional Neural Networks (CNNs) to process and recognize gestures, enabling a seamless and intuitive method of input for mathematical expressions.

The main objectives of the project are:

- **Hand Detection**: Implement an efficient system for detecting hand movements and gestures in real time, creating a virtual canvas for mathematical input.
- **Numeral Extraction**: Develop the capability to extract numerals and mathematical symbols from the detected hand gestures accurately.
- **Expression Calculation**: Implement functionality to calculate the respective mathematical expressions based on the recognized gestures and symbols.

Result Display: Display the calculated result to the user in real-time, offering immediate feedback on the entered expression.

1.4 Project Overview

The AI-powered air-writing calculator aims to revolutionize how users perform mathematical operations by utilizing hand gestures. This project addresses the challenge of efficiently inputting complex mathematical symbols, which are often difficult to type or explain through conventional methods.

The system uses hand detection technology to track the user's hand movements, extracting numerals and symbols from air-written gestures. Convolutional Neural Networks (CNNs) are employed to accurately recognize these gestures and calculate the corresponding mathematical expressions. The results are then displayed in real-time, offering an intuitive and efficient way to perform calculations.

Currently, the system supports basic mathematical operations such as addition, subtraction, multiplication, division, factorials, and square roots. However, the design is scalable, allowing future expansion to include more advanced operations, such as integration and differentiation.

The project aims to enhance productivity, accessibility, and user experience by providing a contactless solution for mathematical input. By integrating advanced technologies like hand tracking and CNNs, the air-writing calculator simplifies the process of mathematical problem-solving, making it faster, more efficient, and accessible for users across various fields, including education and professional environments.

Chapter-2 Literature Survey

2.1 Overview of Existing Research

This section introduces the background and research context for real-time hand gesture recognition systems aimed at performing mathematical calculations through air-writing. The ability to recognize and interpret gestures for numeric and symbolic input has significant implications in making human-computer interaction more natural, contactless, and accessible, especially in fields like education, augmented reality, and assistive technology.

Several research efforts have explored hand gesture recognition, air-writing, and symbol interpretation, employing diverse approaches based on computer vision and machine learning. Early studies focused on static gesture recognition using traditional image processing techniques. Recent advancements have incorporated deep learning methods, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to handle dynamic gestures and sequential input more effectively. Techniques like background subtraction, hand key-point detection, and trajectory tracking have been used to capture hand movement in real-time, while recognition models are trained to classify digits, operators, and even complex mathematical expressions.

Research challenges include achieving high accuracy in varied lighting conditions, handling occlusions, differentiating between similar gestures, and ensuring low-latency performance for real-time applications. Some studies emphasize the integration of spatial-temporal modeling, attention mechanisms, and multi-modal data (e.g., RGB-D cameras) to enhance recognition reliability and robustness.

Overall, the field of real-time gesture recognition for mathematical calculations remains a promising area of study, with ongoing innovations aimed at improving system efficiency, user adaptability, and seamless integration into interactive environments.

Handwriting Recognition in Air using a Smartphone IMU (14 March 2023, Hao Li et al.)-

The system captures 3D motion trajectories using a smartphone’s inertial sensors to recognize air-written text. A transformer-based model segments and interprets strokes, achieving robust handwriting recognition without visual input devices.

Air-Writing Character Recognition Using OpenPose and Deep Neural Network (27 December 2024, Ki-Seo Chungetal.)-

The method uses OpenPose to track fingertip motion and recognize air-written characters with deep neural networks. It improves air-writing accuracy under free-form motion conditions, using sequential hand joint data as input features.

Write in Air: Handwriting Recognition Using Inertial Measurement Unit (7 September 2018, Saket Anand et al.)-

This study introduces a smartphone-based system that uses inertial sensors to recognize English letters written in the air. A recurrent neural network processes the accelerometer and gyroscope data, achieving high accuracy for isolated character recognition.

Finger Air Writing Recognition System with Wearable Inertial Sensor and Hidden Markov Model (15 October 2019, Daisuke Asano et al.)-

The system employs a wearable ring sensor to capture hand motions for air-writing, using a Hidden Markov Model to classify character sequences. It focuses on reducing noise and drift errors for stable real-time performance.

Real-time Air Writing Recognition for Digits using CNN-RNN Model (2024, A. Pradeep et al.)-

The approach combines convolutional and recurrent neural networks to recognize air-written digits from continuous hand motion. By using trajectory images and sequential modeling, it achieves high real-time accuracy with low latency.

2.2 Summary

In these studies, various systems for air-writing recognition were developed using different sensors and models. One approach uses a smartphone's IMU (accelerometer and gyroscope) to capture 3D motion trajectories, employing a Transformer-based model to recognize handwritten characters in real time. Another system tracks fingertip motion using OpenPose and classifies characters with a deep neural network. Some studies use inertial sensors or wearable devices, combined with models like RNN or HMM, to recognize motion-based handwriting. These systems aim to provide natural, contactless methods for handwriting input, offering potential applications in mobile devices, smart classrooms, and virtual environments.

Chapter-3 System Requirements

3.1 Hardware Requirements

Minimum Specifications

- Processor: Intel Core i3 (2.4 GHz or faster)

  Reason: The model inference and OpenCV processing require moderate computational power for real-time operations.
- RAM: 8 GB

  Reason: To be able to handle image processing, model inference, and running multiple software simultaneously.
- Storage: At least 10 GB free space

  Reason: To store the dataset, trained model, and necessary dependencies.
- Camera: Standard USB or built-in webcam with 720p resolution

  Reason: To record the live video feed of the driver for analysis.
- Display: Monitor of at least a 1024x768 resolution

  Reason: To show the live feed and analysis during testing.

Suggested Specifications

- CPU: Intel Core i5 or AMD Ryzen 5 (2.8 GHz above)

  Reason: It will offer faster processing for the high-accuracy real-time detection.
- RAM: 16 GB

  Reason: This is optimum and shall ensure high performance during training, inference, and multi-tasking.
- GPU: At least NVIDIA GTX 1050 or higher, CUDA-enabled

  Reason: It accelerates model training and speeds up image processing tasks.
- Storage:50 GB SSD or HDD (SSD preferable because the read/write speeds are faster)

  Reason: To have the space for dataset storage, pre-processed file storage, and extra software.
- Camera: Full HD (1080p) webcam

  Reason: Higher accuracy face and eye detection.

3.2 Software Requirements

3.2.1 Operating System

- Windows: Windows 10 or 11 (64-bit)
- Linux: Ubuntu 20.04 or higher
- macOS: Monterey or Ventura

3.2.2 Programming Language

(Python 3.6 or above)

Python is an interpreted, high-level, general-purpose programming language. Python is simple and easy to read syntax emphasizes readability and therefore reduces system maintenance costs. Python supports modules and packages, which promote system layout and code reuse. It saves space but it takes slightly higher time when its code is compiled. Indentation needs to be taken care of while coding. Python does the following:

- Python can be used on a server to create web applications.
- It can connect to database systems. It can also read and modify files.
- It can be used to handle big data and perform complex mathematics.
- It can be used for production-ready software development.

Python has many inbuilt library functions that can be used easily for working with machine learning algorithms. All the necessary python libraries must be pre- installed using “pip” command.

3.2.3 Python Libraries

This project integrates several powerful Python libraries to facilitate real-time hand gesture recognition, symbol classification, and mathematical evaluation. Each library is chosen for its specific capabilities in computer vision, machine learning, numerical computation, and hand tracking. Below is a summary of the essential libraries employed in the implementation:

- **TensorFlow & Keras**

  TensorFlow is an open-source machine learning framework developed by Google. Keras is a high-level API built on top of TensorFlow for quick model prototyping. These libraries are used to build and train the Convolutional Neural Network (CNN) for symbol classification (digits and mathematical operators).

  Key Features:

  - Simplified neural network construction with Keras Sequential API.

- Efficient GPU/CPU-accelerated training.
- Includes utilities for dataset handling and preprocessing.
- Easy integration of callbacks like Early Stopping for training optimization.

➢ **NumPy**

NumPy is the fundamental package for scientific computing in Python. It is used for numerical operations, array manipulation, and matrix transformations, which are crucial for image preprocessing and dataset management.

Key Features:

- Efficient array broadcasting and mathematical operations.
- Support for multidimensional arrays (e.g., image tensors).
- Fast performance due to underlying C implementations.
- Useful in pixel normalization and model input reshaping.

➢ **OpenCV (cv2)**

OpenCV is a widely-used computer vision library for image and video processing. In this project, it supports real-time webcam capture, image transformations, drawing on virtual canvases, and preprocessing for model inference.

Key Features:

- Real-time video frame capture and manipulation.
- Functions for grayscale conversion, thresholding, and resizing.
- Drawing utilities for gesture-based input (e.g., line drawing).
- Integration with machine learning models for visual feedback.

➢ **cvzone**

cvzone is a user-friendly computer vision wrapper built on OpenCV and MediaPipe. It simplifies hand detection and tracking, making it ideal for gesture-based interfaces.

Key Features:

- Easy integration of hand detection with landmark tracking.
- Built-in finger counting and gesture interpretation.
- Simplified drawing and annotation features.
- Supports real-time processing with minimal code.

➢ **MediaPipe (via cvzone)**

MediaPipe, developed by Google, provides fast and robust real-time hand tracking. Through cvzone, this functionality is used to detect hand landmarks and finger positions required to interpret user gestures.

Key Features:

- High-accuracy hand landmark detection.
- Fast, on-device real-time performance.
- Provides finger state detection (up/down) out of the box.
- Minimal setup and hardware requirements.

- **Scikit-learn**

  Scikit-learn is a comprehensive machine learning library. In this project, it is primarily used to split the dataset into training and testing subsets for model evaluation.

  Key Features:

  - Simple train_test_split() utility for dataset partitioning.
  - Robust support for preprocessing and evaluation workflows.
  - Lightweight and easy to integrate with other ML libraries.
  - Enhances reproducibility and structured ML pipelines.

- **math**

  The built-in Python math module is used for evaluating complex mathematical expressions. It supports functions such as factorial, square root, and constants like π (pi) that are required for expression computation.

  Key Features:

  - Accurate mathematical constants and functions.
  - Supports factorial, square root, and trigonometric operations.
  - Lightweight and requires no installation.
  - Enables safe expression evaluation via controlled eval().

- **NumPy**

  NumPy is a fundamental Python library for numerical computations. It was essential for handling image data arrays, performing normalization, reshaping inputs for the CNN model, and processing prediction outputs. Its optimized operations made dataset manipulation and preprocessing highly efficient.

  Key Features:

  - Efficient multi-dimensional array and matrix operations.

- Essential for preprocessing and reshaping image data.
- Improves performance when handling large datasets.

➢ **Matplotlib** (optional during development)

Matplotlib was used during the model development phase for visualizing training and validation accuracy and loss curves. This helped in monitoring the model's learning progress and fine-tuning training parameters.

Key Features:

- Enables visualization of training metrics over epochs.
- Helps detect overfitting or underfitting trends visually.
- Useful for diagnostic analysis during model development.

3.2.4 Model Requirements

Trained CNN Model: Type: Convolutional Neural Network (CNN); used for symbol classification in real-time hand gesture recognition.

Model File: A pre-trained CNN model file, saved in .h5 format (airwriting_model.h5), is loaded at runtime to make predictions based on user-drawn air gestures.

Model Input Size: The model takes grayscale images of size 28x28 pixels as input, which corresponds to the pre-processed bounding box region extracted from the hand-drawn symbol path.

Model Output: The CNN classifies the input into 23 classes, which include digits (0-9) and mathematical symbols such as +, -, *, /, (,), [,], {, }, pi, sqrt, and !.

Training Data: The model is trained on a combined dataset consisting of: The MNIST dataset for digits 0–9 and a custom collected dataset for mathematical symbols, augmented with transformations such as rotation, scaling, and shifting to simulate real-world gesture variance.

Model Accuracy: The CNN model achieved a classification accuracy of 99.28% on the validation dataset, demonstrating high reliability for real-time air-writing applications.

3.2.5 Real Time Detection Requirements

➢ **Video Capture and Hand Detection:**

- The video feed is captured from the webcam using the OpenCV library. A minimum resolution of 720p is recommended to ensure reliable detection and tracking of hand landmarks.
- MediaPipe Hands, integrated via the cvzone library, is used to detect and track 21 hand landmarks in real-time.

- **Frame Processing:**
  - Each frame is analyzed to detect the user's index fingertip movement, which is used to draw symbols in the air.
  - The fingertip coordinates are stored to build a continuous trail or path, simulating air-writing.
  - Once a prediction gesture is triggered, the drawn symbol is extracted as a 28x28 grayscale image, preprocessed (resized, normalized), and passed to the CNN model for classification.
- **Gesture-Based Interaction and Expression Evaluation:**
  - Specific hand gestures are used to perform functional operations:
    - Five fingers up: Clears the canvas.
    - Second and fifth fingers up: Triggers prediction of the drawn symbol.
    - All fingers except thumb up: Evaluates the complete mathematical expression.
    - Only pinky up: Removes the last predicted symbol.
  - The expression is dynamically updated based on predicted symbols and gestures.
- **User Interface:**
  - A real-time display window is shown using OpenCV's imshow, presenting the webcam feed, current predicted symbols, and the evolving mathematical expression.
  - Visual overlays like bounding boxes, trails, and text annotations are added for an interactive and intuitive user experience.

3.2.6 Performance and Optimization Requirements

Latency: The system is designed to operate with minimal latency, ensuring real-time symbol prediction and expression evaluation while capturing video input. Low delay between drawing a symbol and receiving its prediction is essential for seamless user experience.

Accuracy: The CNN model, trained on MNIST digits and a custom mathematical symbol dataset, achieves high classification accuracy (up to 99.28%). This high accuracy ensures that the predicted symbols are reliable, reducing the need for repeated attempts and minimizing errors in expression construction.

Scalability: The detection pipeline is built to perform robustly across different backgrounds, lighting conditions, and hand variations without the need for extensive reconfiguration. Minimal preprocessing and real-time landmark detection allow the system to generalize across various users and environments with consistent performance.

3.2.7 Jupyter Notebook

Jupyter Notebook is an interactive web application that allows you to create and share documents that contain live code, visualizations, and narrative text. It's widely used in data science, machine learning, and academic research because it provides a flexible environment for experimentation and documentation. With Jupyter, you can write code in a variety of programming languages, visualize data, and include explanatory text, all in one place.

3.2.8 PyCharm

For Python programming, Jupyter Notebook is a popular choice, but if you prefer a more traditional development environment, PyCharm is an excellent option. PyCharm is a well-liked integrated development environment (IDE) that offers sophisticated code analysis, debugging, and project navigation tools. It supports databases, web development frameworks, and popular version control systems. With its intuitive interface and intelligent coding assistance, PyCharm enhances the efficiency of Python development. Whether you choose Jupyter Notebook for its interactivity or PyCharm for its robust features, both tools can effectively support your programming needs.

Chapter-4 Design and Implementation

4.1 Proposed System and Advantages

4.1.1 Proposed System

The proposed system introduces a real-time AI-powered air-writing calculator that enables users to draw mathematical expressions in the air using hand gestures. The system leverages computer vision, hand tracking, and a Convolutional Neural Network (CNN) to recognize digits and mathematical symbols. The goal is to offer an intuitive and contactless method for performing calculations by combining live webcam feed, gesture recognition, and symbol classification.

The key components of the system are:

- **Webcam Integration for Live Hand Gesture Capture**: A webcam is used to continuously capture live video frames of the user's hand. These frames are processed in real-time to detect hand landmarks and monitor movement, allowing the system to track the user's air-drawn gestures seamlessly.
- **Hand Detection and Tracking using MediaPipe (via CvZone):** The system employs MediaPipe's hand tracking solution to detect and extract 21 hand landmarks. These landmarks are used to interpret gestures and record the trajectory of the index finger, which acts as a virtual pen to draw symbols in the air. Specific gestures (e.g., all five fingers up to clear the canvas, or a specific gesture to trigger prediction) control the flow of input.

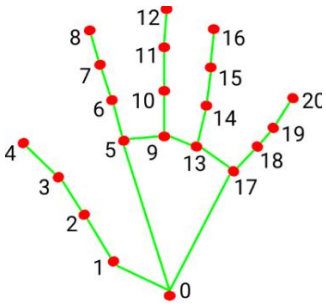


Figure 4.1.1.1: 21 Hand Landmarks

- **CNN Model for Symbol Prediction:** A custom-trained CNN model is used to classify the hand-drawn symbols. The model supports 23 classes, including digits (0–9) and mathematical symbols such as +, -, *, /, (,), sqrt, pi, and more. The CNN architecture is chosen for its robustness and high accuracy (up to 99.8%) in spatial pattern recognition.
- **Expression Evaluation and Interaction Logic:** The recognized symbols are sequentially appended to form a mathematical expression. Specific gestures allow users to evaluate (=),

delete the last symbol, or clear the entire expression. Upon evaluation, the result is computed using Python's eval function, and both the input expression and result are displayed on-screen.

- **User-Friendly Interface and Real-Time Feedback:** The system provides a live visual interface showing the hand tracking path, predicted symbols, and final evaluated results. The entire pipeline is implemented in Python, ensuring modularity and extensibility. The system runs smoothly on standard hardware and allows for future enhancements, such as voice feedback or multi-digit continuous input.

This innovative, gesture-based calculator provides a non-contact, interactive, and accessible solution for performing arithmetic operations in real time, making it especially useful in educational and assistive technology contexts.

4.1.2 Advantages

The proposed AI-powered air-writing calculator offers several advantages over traditional input methods and other gesture-recognition-based systems, as outlined below:

- **Contactless and Non-Intrusive Input:** The system uses a standard webcam and computer vision for input, eliminating the need for physical contact with a device. This makes it hygienic, especially in shared environments, and more accessible for users with physical impairments.
- **Real-Time Detection and Feedback:** Leveraging a fast and optimized CNN model along with MediaPipe hand tracking, the system processes gestures and predicts symbols in real time. This ensures a smooth and interactive user experience without noticeable lag.
- **Efficient and Lightweight Deployment:** The CNN model is trained specifically for this application, enabling it to achieve high accuracy (up to 99.8%) while remaining computationally efficient. It runs effectively on standard hardware without the need for specialized GPUs.
- **Wide Symbol Support and Expression Handling:** The calculator supports a variety of mathematical symbols beyond digits, including operators and special functions like sqrt and pi. This expands the system's utility beyond basic arithmetic to more advanced calculations.
- **Robust Hand Tracking and Gesture Recognition:** By using MediaPipe's hand landmark detection, the system offers stable and precise finger tracking even under varying lighting and background conditions, making it reliable in real-world environments.

- **Modular and Scalable Architecture:** The design is modular, allowing easy integration of new features such as handwriting recognition for letters, voice feedback, or multilingual support. This future-proof approach ensures long-term usability and extensibility.

4.2 Methodology

4.2.1 Data Collection

Handwritten digits and operator dataset consists of 23 classes which was collected from three different sources. The handwritten digits (0-9) dataset was taken from MNIST dataset which is a publicly available benchmark dataset widely used for training and testing in the field of machine learning and computer vision. The dataset for basic mathematical operators like addition, subtraction, multiplication and division was taken from Kaggle. This dataset was further extended to include symbols like '(', ')', '[', ']', '{', '}', '!' and constant pi. This dataset was also taken from Kaggle. The combined dataset has 128599 images which was divided into training and testing. The training dataset has 107628 images and test dataset has 20971 images. Few images of dataset are shown in Fig. highlighting the visual diversity within the dataset images.

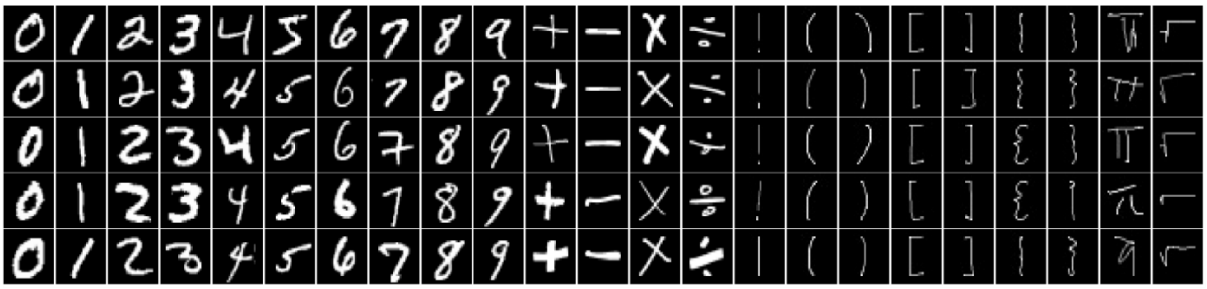


Figure 4.2.1.1: Sample instances of dataset

4.2.2 Data Preprocessing

The classification model developed in this work relies on two key sources of data: the MNIST dataset for handwritten digits and a custom-curated dataset containing mathematical operators and special symbols. To ensure uniformity and compatibility with the neural network input requirements, all images underwent systematic preprocessing.

- **MNIST Dataset:** The MNIST dataset which can be accessed via keras.datasets.mnist, contains grayscale images of handwritten digits ranging from 0 to 9. Each image is 28×28 pixels in size. The preprocessing of this dataset involved:
  - Normalization: In order to scale all pixel values from [0,255] to [0,1], they were divided by 255.0. The model performs better and converges more quickly during training because to this standardization.

- Reshaping: Convolutional neural networks require input data with a channel dimension, therefore each image was reshaped from (28, 28) to (28, 28, 1), where the final dimension denotes a single grayscale channel.

➢ **Custom Operator and Symbol Dataset:** To extend the calculator's functionality beyond numerical digits, a custom dataset of 13 operator and function symbols (+, −, *, /, !, (,), [,], {, }, pi, sqrt) was created and standardized to match the MNIST format. Each image was first converted to grayscale and processed using binary inverse thresholding to enhance symbol visibility against a dark background. The region containing the symbol was then cropped using bounding box extraction and resized proportionally to fit within a 20×20 area. To maintain consistency, the resized symbol was cantered on a 28×28 black canvas, replicating MNIST's layout. All images were normalized to a [0, 1] pixel intensity range and organized into class - labelled folders for training and testing. Invalid or unreadable images were removed during preprocessing. This uniform preprocessing pipeline ensured consistent input across all 23 symbol classes—digits and operators—resulting in a clean, balanced dataset optimized for CNN-based classification.

4.2.3 Model Selection

Selecting the right model to recognize air-written numbers was one of the most important choices we made for our project. After testing different approaches, we decided to use a Convolutional Neural Network (CNN) because it works extremely well with images and is ideal for understanding hand gestures drawn in the air.

Why CNN for Air-Writing Recognition?

- **Perfect for Visual Data:** CNNs are designed to work with images and visual patterns. Since our project relies on identifying hand-written numbers made in the air, we needed a model that can "see" and understand shapes. CNNs are great at this—they can learn the structure and motion of digits written by hand, helping the system to read them correctly.
- **Automatic Learning of Features:** One of the best things about CNNs is that they can figure out what parts of the image are important without us having to tell them. While traditional methods need hand-picked features, CNNs automatically learn shapes, strokes, and edges of the numbers, making it easier for our calculator to recognize different handwriting styles.
- **High Accuracy:** CNNs are known for being very accurate when it comes to classifying images. This helps our air-writing calculator to work well even if the lighting changes or the hand is at a different angle. It also makes the system more reliable for real-time use.

- **Easy to Improve and Adapt:** Another advantage of CNNs is that they are flexible. If we want to make the system better in the future—like adding support for more gestures or improving accuracy—we can easily change or expand the model. This means our calculator can keep improving as technology evolves.

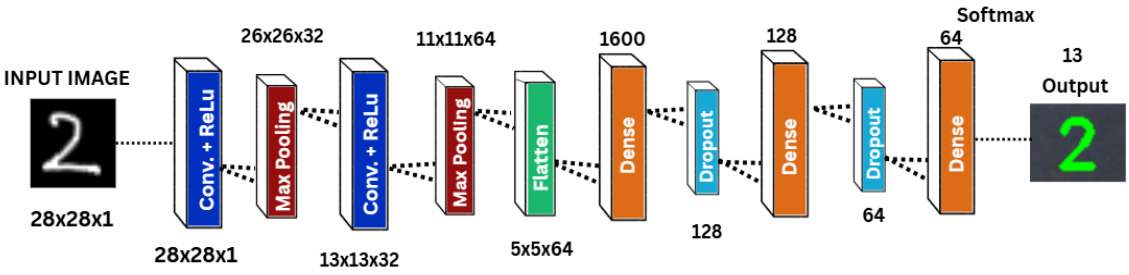


Figure 4.2.3.1: Custom CNN Model Architecture

4.3 System Design

4.3.1 System Architecture

The architecture of the real-time air-writing calculator system is designed to recognize hand-drawn mathematical symbols and compute results instantly. It processes video from the webcam, detects hand gestures, classifies the drawn symbol, builds a mathematical expression, and evaluates the result. The system is organized into the following key layers:

- **Input Layer (Camera Feed)**
  - Purpose: Capture real-time video to track hand gestures.
  - Implementation: The webcam is initialized using OpenCV and cvzone. Each frame is continuously read and flipped for a mirror view. This frame is used for gesture detection and drawing.
  - Code:

```
cap = cv2.VideoCapture(0)
detector = HandDetector(staticMode=False, maxHands=1, modelComplexity=1,
detectionCon=0.5, minTrackCon=0.5)
```

- **Hand Detection and Gesture Tracking Layer**
  - Purpose: Detect the user's hand and track finger movements.
  - Implementation: The HandDetector module from cvzone detects the hand and tracks key landmarks. Based on which fingers are raised, specific actions are triggered:
    - Index finger up → drawing gesture
    - All fingers up → clear canvas
    - Index + pinky → symbol prediction

- Four fingers up (excluding thumb) → evaluate expression
- Pinky only → delete last symbol

Figure 4.3.1.1: Hand Gestures

➢ **Drawing and Preprocessing Layer**

- Purpose: Capture air-written digits/operators and prepare them for model input.
- Implementation: The system draws white strokes on a black canvas as the user writes in the air. The canvas is then converted to grayscale, thresholded, cropped to the region of interest, resized to 28x28 pixels, normalized, and reshaped into a format suitable for CNN input.
- Code:

```
def draw(info, prev_pos, canvas):
    fingers, lmList = info
    current_pos = None
    if fingers == [0, 1, 0, 0, 0]:
        current_pos = lmList[8][0:2]
        if prev_pos is None:
            prev_pos = current_pos
        cv2.line(canvas, tuple(current_pos), tuple(prev_pos), (255, 255,
255), 10)
    return current_pos, canvas

def preprocess_for_prediction(canvas):
. . .
```

➢ **Model Inference and Symbol Prediction Layer**

- Purpose: Predict the digit or mathematical symbol from the air-drawn gesture.
- Implementation: A pre-trained CNN model (combined_model.h5) is loaded using TensorFlow. The processed image is passed to the model, which outputs a probability vector over 24 possible classes (digits, operators, parentheses, constants). The class with the highest probability is selected and mapped to its corresponding symbol using a lookup dictionary.

➢ **Expression Handling and Evaluation Layer**

- Purpose: Build and evaluate the mathematical expression.

- Implementation: Recognized symbols are appended to an ongoing expression. The expression is evaluated using Python's eval() with limited scope (only safe math functions allowed). Special symbols like !, pi, and sqrt are handled through mappings to math functions.
- Code:

```
def evaluate_expression(expression):
    expression = expression.replace("!", "math.factorial")
    try:
        result = eval(expression, {"__builtins__": None}, {"math": math,
"pi": math.pi, "sqrt": math.sqrt})
        return result
    except Exception as e:
        return f"Error: {str(e)}"
```

➢ **Real-Time Feedback and UI Layer**

- Purpose: Display predictions, expressions, and results in real time.
- Implementation: Using OpenCV, the system overlays the predicted symbol, the current expression, and the evaluated result directly onto the video feed. A second window shows the canvas with the user's air-writing.

This layered approach allows for smooth interaction, efficient symbol recognition, and quick computation, making the system an intuitive air-writing-based calculator that functions in real time.

4.3.2 Component Interaction

The system operates through a pipeline of interdependent components, each performing a critical function and passing data to the next. The interaction between components follows a real-time, sequential data flow, while some layers provide feedback to others. Below is how the components interact:

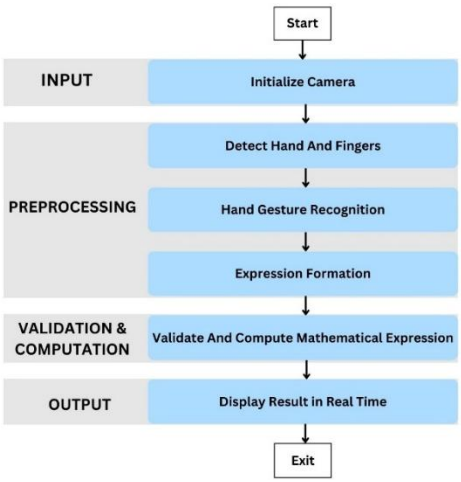


Figure 4.3.2.1: Implementation Flowchart

Chapter-5 Result and Discussion

5.1 Performance Metrics

In order to assess the model's performance, we looked at several measures, including a confusion matrix, ROC-AUC, accuracy, precision, recall, F1-score, and precision-recall curve. Together, these metrics give us a general picture of how well the model performs in relation to various classification-related factors. Accuracy is a measure for determining the overall correctness of a model. It indicates the number of correctly predicted instances in relation to the total number of instances. The higher the figure representing accuracy score, the more correct predictions are made by the model.

5.1.1 Accuracy

Accuracy is a measure for determining the overall correctness of a model. It indicates the number of correctly predicted instances in relation to the total number of instances. The higher the figure representing accuracy score, the more correct predictions are made by the model.

$$Accuracy = \frac{(True\ Positives + True\ Negatives)}{Total\ Instances}$$

While accuracy is widely used, it may not fully reflect the performance when the class distribution is imbalanced, therefore additional metrics are considered.

5.1.2 Precision

Precision is how many of the predicted positive instances are actually positive. That means reducing the false positive counts. The higher the precision, the more probably the prediction for a positive result of the model is correct.

$$Precision = \frac{(True\ Positives)}{(True\ Positives + False\ Positives)}$$

Precision is particularly useful when false positives (incorrect symbol predictions) are need to be minimized.

5.1.3 Recall

Recall or sensitivity measures how many of the actual positives the model predicted as positive. It aims to minimize false negatives. High recall means that the model captures almost all positive examples in the dataset.

$$Recall = \frac{(True\ Positives)}{(True\ Positives + False\ Negatives)}$$

A high recall means the model correctly identifies most of the actual instances of a symbol. It is important when the cost of missing a symbol is high.

5.1.4 F1 Score

F1-score is a measure that combines both precision and recall and is thus known as a harmonic mean of the two. It is especially useful when the precision and recall are to be balanced and only a single value is required to carry both notions. The F1 score gets high when the values of both precision and recall are high

$$F1\ score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

F1-score provides a balanced metric when both false positives and false negatives are important.

5.1.5 Summary

The model has 99.28% accuracy on a held-out test dataset, indicating that it is strongly capable of precisely and accurately classifying hand gestures. Such a level of accuracy is valuable in real-time tasks that require speed and accuracy in the predictive process. Furthermore, with a precision of 0.9893, nearly 99% of the gestures predicted by the model will be correct, minimizing the potential for false positives. Using the best metric of recall (0.9894), the model will recognize nearly 99% of all true gestures it should recognize, minimizing the potential for missing important symbols. The F1 score, or balance of precision and recall, is 0.9894, indicating that the model offers an appropriate trade-off of the two metrics, which is important to both high-level accuracy and form recognition.

5.2 Result Analysis

The results demonstrate that the model, trained with the Adam optimizer, achieved excellent performance across multiple metrics, including test accuracy, precision, recall, F1-score, ROC-AUC, and precision-recall curve. The high-test accuracy reflects the model's strong ability to generalize to unseen data, while the high precision and recall indicate that the model is both reliable in its predictions and effective at capturing relevant symbols. The F1 score and ROC-AUC score further confirm the model’s balanced performance, with the PR curve area suggesting that the model is well-tuned to handle any potential class imbalance. These results indicate that the model is well-suited for real-time air-writing applications where accurate gesture recognition is essential.

5.2.1 Model Accuracy vs. Epochs

The accuracy curve reflects how the model performance improves with each epoch. Accuracy starts from a low accuracy, and as it gets trained with the provided training data, the accuracy gradually improves. It is ideally a sharp increase in accuracy with stabilization, indicating that the model reached convergence quickly with good ideal conditions, preventing it from overfitting the training data.

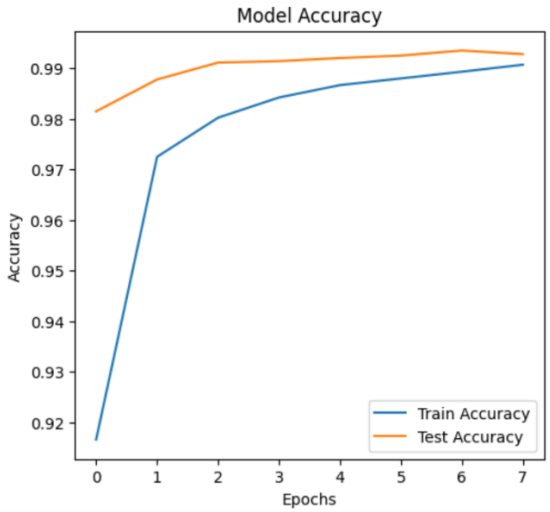


Figure 5.2.1.1: Accuracy vs. Epochs Graph

5.2.2 Model Loss vs. Epochs

The loss curve reflects the improvement on the loss function during training. With good ideal conditions, the epoch loss curve shows a steady decrease. The steep and smooth decrease ensures that the model is minimising the error, and shows that there are no unexpected changes resulting in the model not learning properly.

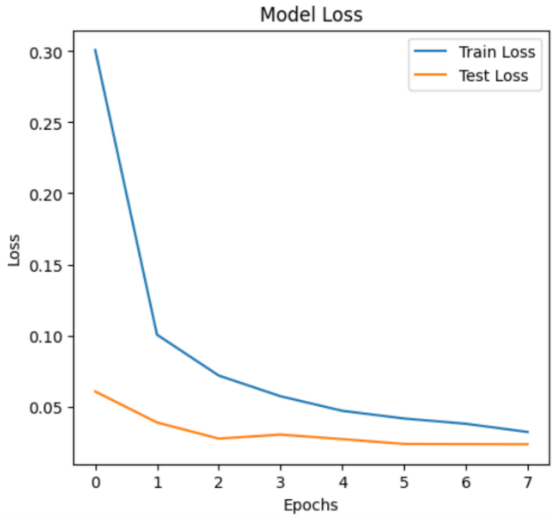


Figure 5.2.2.1: Loss vs. Epochs Graph

5.2.3 ROC-AUC Curve

The ROC curve is a visual representation of the model's ability to discriminate between classes with a curve indicating superior performance, and the area under the curve (AUC) reporting the level of performance. In this study, an AUC score of 1.0000 means that the model could discriminate between classes perfectly.

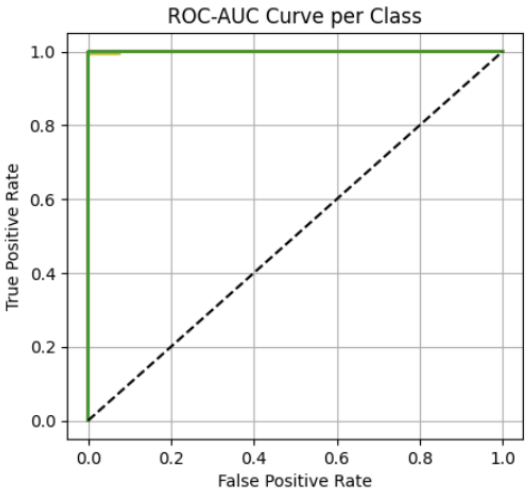


Figure 5.2.3.1: ROC Curve

5.2.4 Precision-Recall Curve

The PR curve represents the trade-off between precision and recall at different thresholds. A larger area corresponds to a model with good balance between the two, getting the relevant instances but minimizing the false positives.

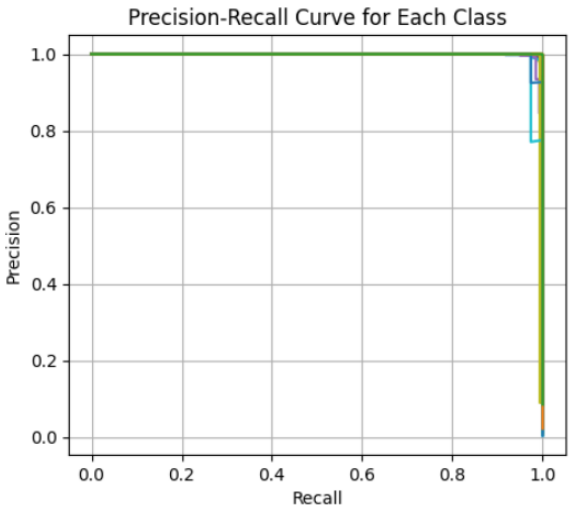


Figure 5.2.4.1: Precision-Recall Curve

5.2.5 Confusion Matrix

The confusion matrix provides another visual representation of classification performance across all 23 classes that allows further detail into how many symbols the model has truly classified correctly, as well as into the number of mis-classifications and can provide additional insights about where the model may have weaknesses in performance.

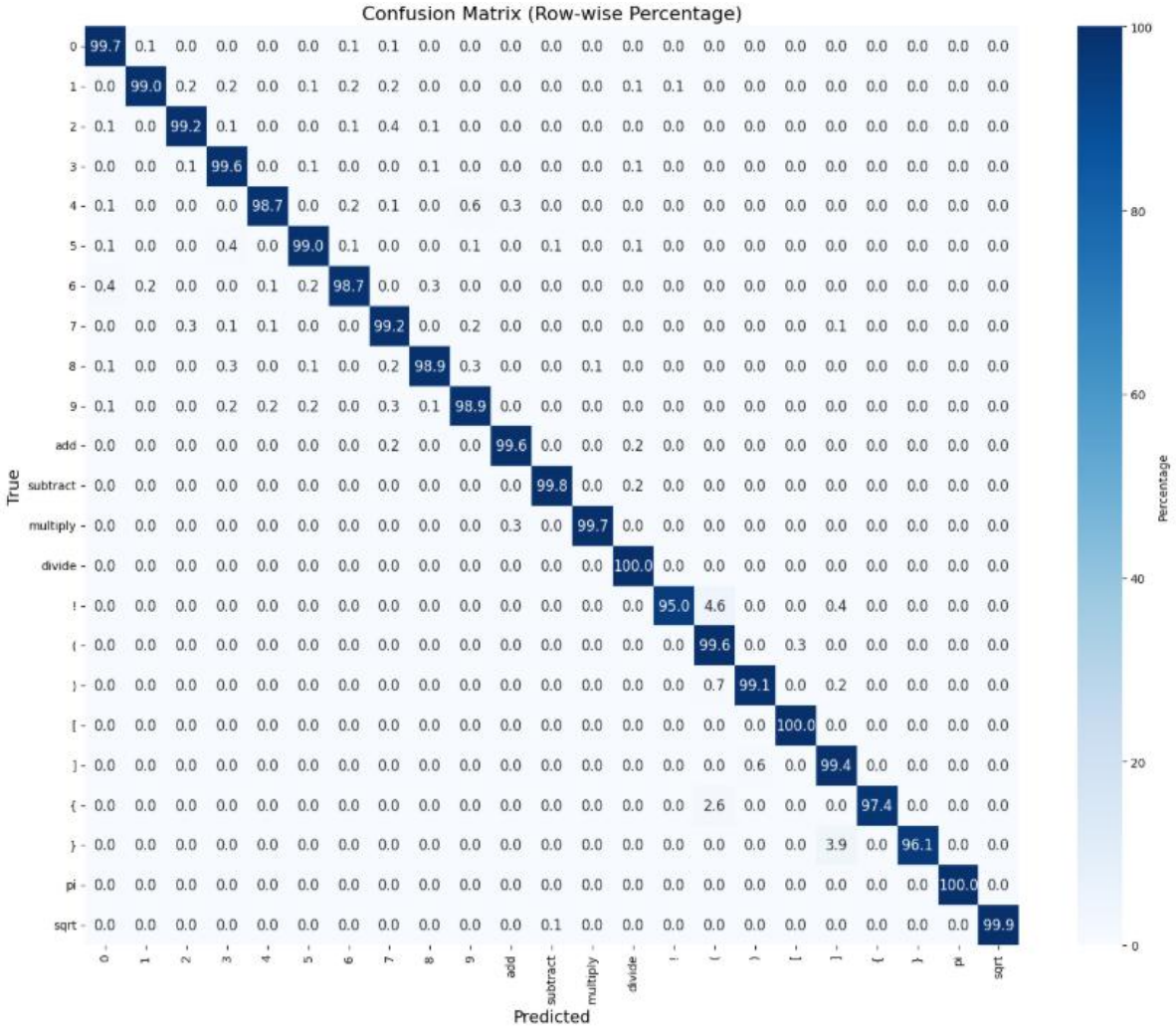
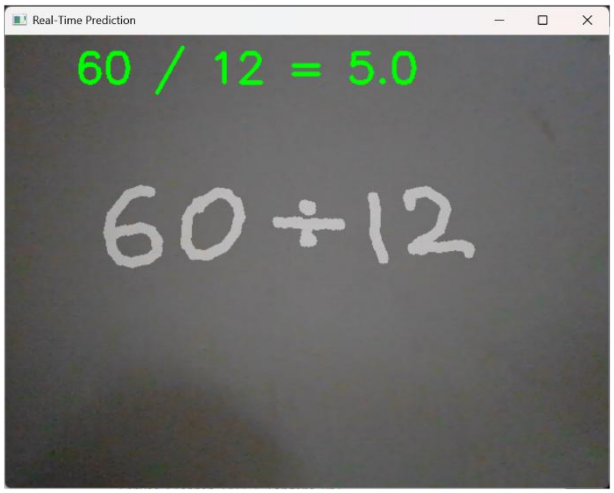
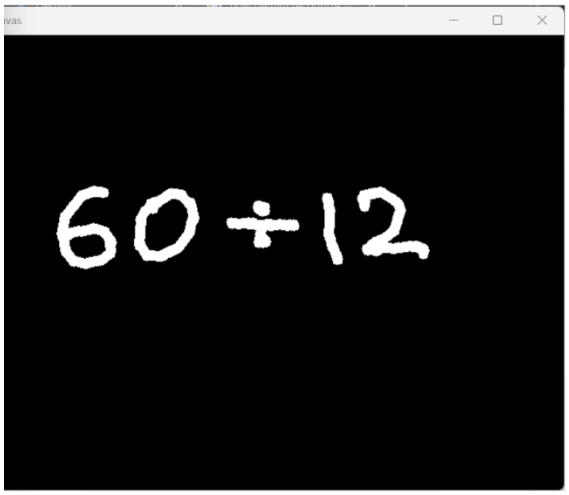
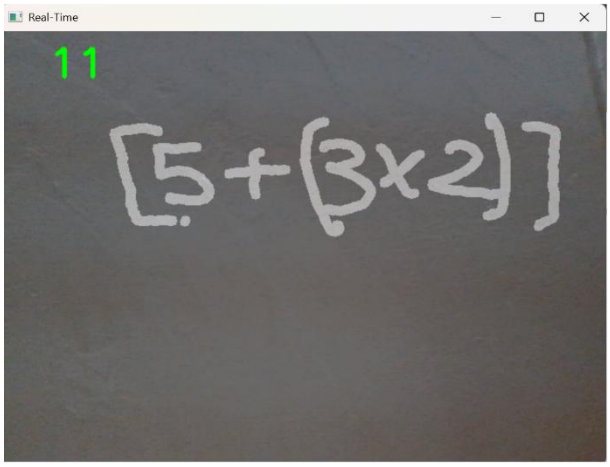


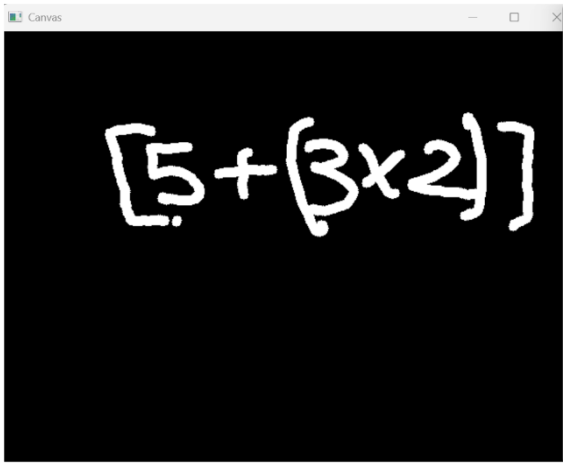
Figure 5.2.5.1: Confusion Matrix (Percentages)

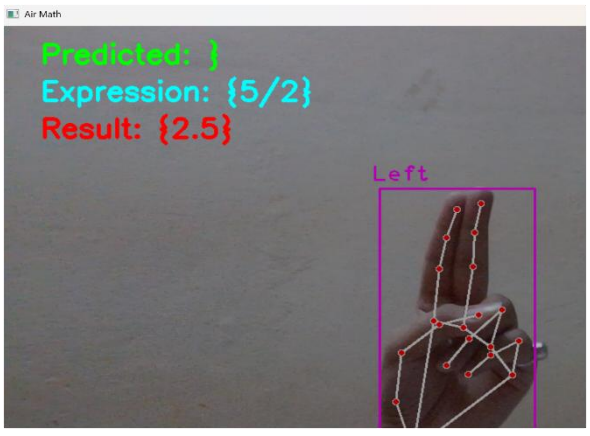
5.3 Screenshots

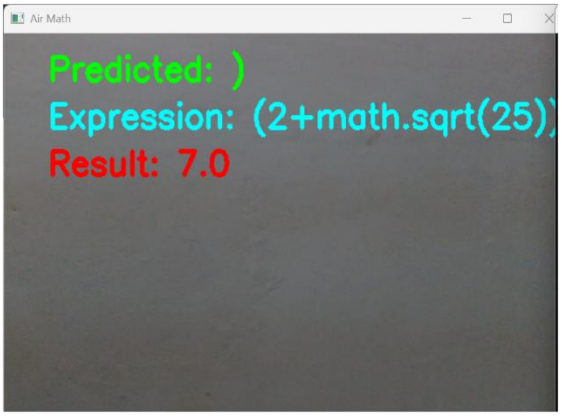


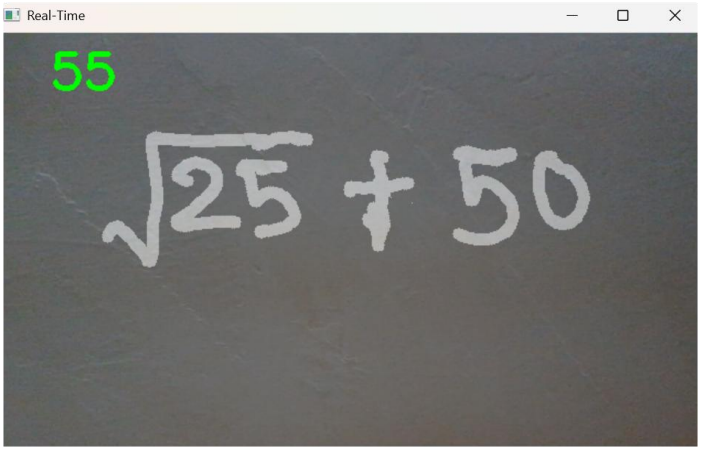




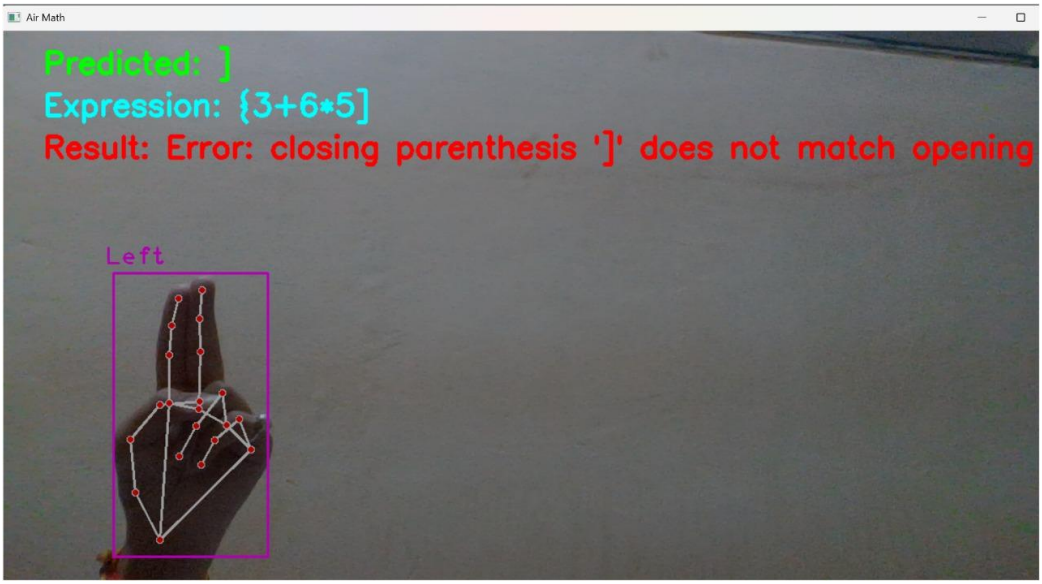







Real-Time
55
√25 + 50


Air Math
Predicted:]
Expression: [5*3+2*math.sqrt(100)]
Result: [35.0]
Left


Air Math
Predicted:]
Expression: {3+6*5]
Result: Error: closing parenthesis ']' does not match opening
Left

Conclusion

The project *"Real-Time Hand Gesture Recognition for Mathematical Calculations"*, implemented as an air-writing-based calculator, demonstrates an effective integration of computer vision, deep learning, and symbolic computation to enable mathematical interaction using hand gestures. By leveraging a webcam and real-time hand tracking via the cvzone and OpenCV libraries, the system successfully detects and interprets specific gestures corresponding to numerical digits and mathematical operators.

The core of the system uses the HandDetector module to identify the position and gesture of the user's hand, especially tracking the index finger for drawing in the air. This gesture trail is visualized on a canvas and then processed into a binary image. The processed image is fed into a trained deep learning model (combined_model.h5) which classifies the drawn symbol into one of several predefined classes, including digits (0–9), operators (+, -, *, /), brackets, mathematical constants (like pi), and functions (sqrt, !).

The expression-building mechanism is intuitive: specific finger combinations trigger symbol recognition, expression clearing, evaluation, or backspace operations. This interaction mimics touchless input, providing a clean and user-friendly interface for performing calculations in mid-air. The evaluated expressions are parsed safely using Python's eval function with restricted access, supporting complex calculations involving factorials, square roots, and constants like π.

This approach addresses the growing need for touch-free technology, especially relevant in contexts such as education, public terminals, and healthcare, where hygiene and accessibility are critical. It also shows promise as an assistive tool for individuals with limited mobility.

Despite its effectiveness, the system faces challenges such as inconsistent gesture recognition under poor lighting, false positives in complex backgrounds, and slight latency in real-time processing. Future improvements could include adding 3D depth sensing for better gesture tracking, expanding the gesture vocabulary, and optimizing model performance for mobile or embedded environments.

In conclusion, this project successfully demonstrates a novel method for performing real-time mathematical calculations using only hand gestures. It highlights the potential of combining gesture recognition with deep learning and symbolic math to create intuitive, accessible, and hygienic human-computer interfaces.

References

[1] M. Jabde, C. H. Patil, A. D. Vibhute, and S. Mali, “An Air-Written Real-Time Multilingual Numeral String Recognition System Using Deep Convolutional Neural Networks,” Journal of Computer Science, vol. 20, no. 12, pp. 1712–1722, 2024, doi: 10.3844/jcssp.2024.1712.1722.

[2] S. Mukherjee, A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, “Fingertip Detection and Tracking for Recognition of Air-Writing in Videos,” Sep. 2018, doi: 10.1016/j.eswa.2019.06.034.

[3] P. Roy, S. Ghosh, and U. Pal, “A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.07989

[4] M. Wu, K. Huang, Y. Cai, S. Hu, Y. Zhao, and W. Wang, “Finger in Camera Speaks Everything: Unconstrained Air-Writing for Real-World,” Dec. 2024, [Online]. Available: http://arxiv.org/abs/2412.19537

[5] A. Nazemi, N. Tavakolian, D. Fitzpatrick, C. a Fernando, and C. Y. Suen, “Offline handwritten mathematical symbol recognition utilising deep learning,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.07395

[6] A. Suryaperdana Agoes, “Applying Hand Gesture Recognition for User Guide Application Using MediaPipe,” 2021.

[7] L. Alzubaidi et al., “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” J Big Data, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[8] A. Fred Agarap and A. M. Fred Agarap, “Deep Learning using Rectified Linear Units (ReLU)”, doi: 10.48550/arXiv.1803.08375.

[9] M. Franke and J. Degen, “The softmax function: Properties, motivation, and interpretation *.”

[10] “MNIST Dataset,” Keras. [Online]. Available: https://keras.io/api/datasets/mnist/

[11] “Mathematics symbols data,” Kaggle. [Online]. Available: https://www.kaggle.com/datasets/amitamola/mathematics-symbols-data

[12] “Math Operators and Digits,” Kaggle. [Online]. Available: https://www.kaggle.com/datasets/patrikmesec/math-operators-and-digits

[13] A. Géron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow.”

[14] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in ACM International Conference Proceeding Series, 2006, pp. 233–240. doi: 10.1145/1143844.1143874.

[15] R. Andonie, “Hyperparameter optimization in learning systems,” Journal of Membrane Computing, vol. 1, no. 4, pp. 279–291, Dec. 2019, doi: 10.1007/s41965-019-00023-0.

Contributors

1. **SAJAL KORDE**

   Enrolment Number: 221B319

   Email: 221b319@juetguna.in

   Personal Email: sajalkorde01@gmail.com
   Address: Itarsi, Madhya Pradesh

2. **SHRUTI BHARGAVA**

   Enrolment Number: 221B374

   Email: 221b374@juetguna.in

   Personal Email: shrutibhargava2004@gmail.com
   Address: Betul, Madhya Pradesh

3. **SNEHIL SHARMA**

   Enrolment Number: 221B387

   Email: 221b387@juetguna.in

   Personal Email: ssnehil06@gmail.com
   Address: Bhilwara, Rajasthan