# REAL TIME HAND GESTURE DETECTION

# FOR MATHEMATICAL CALCULATION

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 & 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 DETECTION FOR MATHEMATICAL CALCULATION**”, 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:06/05/2025

 **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 DETECTION FOR MATHEMATICAL CALCULATION**” 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:06/05/2025

# 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 DETECTION FOR MATHEMATICAL CALCULATION.”

**Sajal Korde (221B319)**

**Shruti Bhargava (221B374)**

**Snehil Sharma (221B387)**

Date: 06/05/2025

**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

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Figure No.** | **Figure Name** | **Page No.** |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |

# Table of Contents

[REAL TIME HAND GESTURE DETECTION i](#_Toc196753403)

[FOR MATHEMATICAL CALCULATION i](#_Toc196753404)

[Declaration by the Students ii](#_Toc196753405)

[CERTIFICATE iii](#_Toc196753406)

[ACKNOWLEDGEMENT iv](#_Toc196753407)

[Table of Contents vii](#_Toc196753408)

[1.1 Problem Statement 1](#_Toc196753409)

[1.2 Motivation for Work 1](#_Toc196753410)

[1.3 Goal and Objectives 3](#_Toc196753411)

[1.4 Project Overview 3](#_Toc196753412)

[Chapter 2 5](#_Toc196753413)

[2.1 Overview of Existing Research 5](#_Toc196753414)

[2.2 Summary 6](#_Toc196753415)

[Chapter 3 7](#_Toc196753416)

[3.1 Hardware Requirements 7](#_Toc196753417)

[3.2 Software Requirements 8](#_Toc196753418)

[3.2.1 Operating Systems 8](#_Toc196753419)

[3.2.2 Programming Language 8](#_Toc196753420)

[3.2.3 Python Libraries 8](#_Toc196753421)

[3.2.4 Model Requirements 10](#_Toc196753422)

[3.2.5 Real Time Detection Requirements 10](#_Toc196753423)

[3.2.6 Performance and Optimization Requirements 11](#_Toc196753424)

[3.2.7 Jupyter Notebook 11](#_Toc196753425)

[3.2.8 PyCharm 12](#_Toc196753426)

[Chapter 4 13](#_Toc196753427)

[4.1 Proposed System and Advantages 13](#_Toc196753428)

[4.1.1 Proposed System 13](#_Toc196753429)

[4.1.2 Advantages 14](#_Toc196753430)

[4.2 Methodology 15](#_Toc196753431)

[4.2.1 Data Collection 15](#_Toc196753432)

[4.2.2 Data Preprocessing 15](#_Toc196753433)

[4.2.3 Model Selection 16](#_Toc196753434)

[4.3 System Design 18](#_Toc196753435)

[4.2.3 System Architecture 18](#_Toc196753436)

[4.3.2 Component Interaction 20](#_Toc196753437)

[Chapter 5 21](#_Toc196753438)

[5.1 Performance Metrics 21](#_Toc196753439)

[5.1.1. Accuracy 21](#_Toc196753440)

[5.1.2. Precision 21](#_Toc196753441)

[5.1.3. Recall 21](#_Toc196753442)

[5.1.4.F1-Score 22](#_Toc196753443)

[5.1.5. Confusion Matrix 22](#_Toc196753444)

[5.1.6. Real-Time Inference Time 22](#_Toc196753445)

[5.1.7. False Positive Rate 22](#_Toc196753446)

[5.2 Result Analysis 23](#_Toc196753447)

[5.2.1 ROC Curve 23](#_Toc196753448)

[5.2.2 Confusion Matrix 23](#_Toc196753449)

[5.3 Screenshots 24](#_Toc196753450)

[24](#_Toc196753451)

[Conclusion 25](#_Toc196753452)

[References 26](#_Toc196753453)

[Contributors 27](#_Toc196753454)

##### 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**

1.Processor: Intel Core i3 (2.4 GHz or faster)

Reason: The model inference and OpenCV processing require moderate computational

power for real-time operations.

2.RAM: 8 GB

Reason: To be able to handle image processing, model inference, and running multiple software simultaneously.

3.Storage: At least 10 GB free space

Reason: To store the dataset, trained model, and necessary dependencies.

4.Camera: Standard USB or built-in webcam with 720p resolution

Reason: To record the live video feed of the driver for analysis.

5.Display: Monitor of at least a 1024x768 resolution

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

**Suggested Specifications**

1.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.

2.RAM: 16 GB

Reason: This is optimum and shall ensure high performance during training, inference, and multi-tasking.

3.GPU: At least NVIDIA GTX 1050 or higher, CUDA-enabled

Reason: It accelerates model training and speeds up image processing tasks.

4.Storage:50 GB SSD or HDD (SSD preferable because the read/write speeds are faster)

Reason: To have the space for dataset storage, preprocessed file storage, and extra software

5.Camera: Full HD (1080p) webcam

Reason: Higher accuracy face and eye detection.

## 3.2 Software Requirements

### 3.2.1 Operating Systems

* 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:

1. Python can be used on a server to create web applications.
2. It can connect to database systems. It can also read and modify files.
3. It can be used to handle big data and perform complex mathematics.
4. 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 preprocessed 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.

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

1. 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:

1. **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.
2. **Hand Detection and Tracking using MediaPipe (via CvZone):**  
   The system employs MediaPipe's hand tracking solution to detect and extract 21 handlandmarks. 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.
3. 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.
4. **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.
5. **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:

1. **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.
2. **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.
3. **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.
4. **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.
5. **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.
6. 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

In our project, we started by gathering raw data from Kaggle, which offered a collection of images featuring open and closed eyes. This dataset included 726 images for each category, giving us a total of 1,452 images. While this was a good foundation, we quickly realized it wasn’t enough to effectively detect driver drowsiness. To address this limitation, we decided to enhance the dataset through a process called data augmentation.

To expand our collection, we employed several techniques to modify the original images. We rotated, shifted, and flipped them, creating five new versions of each image. This approach allowed us to significantly increase our dataset size to 7,260 images. By introducing this variety, we aimed to help our model learn to recognize eye states in different conditions and angles, making it more adaptable to real-world driving scenarios.

Once we had augmented the images, we resized them to ensure they were suitable for training our convolutional neural network (CNN). Resizing is important because it helps maintain image quality while fitting the model's input requirements. Additionally, we created an annotation file to label the images accurately, distinguishing between open and closed eyes. This labeling is crucial for supervised learning, as it provides the model with the information it needs to learn and make accurate predictions about driver drowsiness. Through these efforts, we aimed to build a comprehensive dataset that would enhance the effectiveness of our drowsiness detection system.

### 4.2.2 Data Preprocessing

Since the dataset obtained from Kaggle was not so big, preprocessing and augmentation were inevitable in order to rise the training performance and extend generalization for the model. The preprocessing process includes:

1. Image Resizing: The dimension of each image of the dataset has been resized into 224x224 pixels so that the model could work properly.
2. Data Augmentation: In order to expand the dataset and introduce variability, transformations have been applied:
   * Rotation: Images were randomly rotated between -30 and +30 degrees to simulate head movements.
   * Translation (Shifting): Horizontal and vertical shifts were applied to reflect slight positional changes.
   * Flipping: Horizontal flips were presented, introducing mirrored variations of each image-to help the model recognize features symmetrically.
   * Scaling and Cropping: We scaled zoom levels within small ranges so that minor scaling differences are induced in images.
3. Normalization: The pixel values for the image had to be normalized within the range 0-1, which again used the model learning features on features without variable lighting conditions or intensity of the image.

One preprocessing strategy was important in expanding the size and diversity of our data set, which would ensure that our model could efficiently train on limited original data while improving robustness in real-world scenarios.

### 4.2.3 Model Selection

Choosing the right model for detecting driver drowsiness is one of the most important decisions we made for our project. After exploring various options, we decided to go with a Convolutional Neural Network (CNN). This choice was influenced by the unique strengths of CNNs, especially when it comes to recognizing and classifying images—an essential aspect of identifying drowsiness.

Why CNN for Drowsiness Detection?

Particular Suitability to Vision Data:

CNNs are particularly well-suited for working with images. They excel at learning the spatial relationships in visual data, which makes them great at detecting whether a driver’s eyes are open or closed. This ability to automatically learn from the images means that our system can effectively pick up on subtle signs of drowsiness, helping to keep drivers safe on the road.

Automatic Feature Extraction

One of the standout features of CNNs is their capability to extract important characteristics from images without needing manual intervention. Unlike traditional methods that require us to specify what features to look for, CNNs can learn to recognize edges, shapes, and textures on their own. This is particularly useful for our project, as it allows the model to adapt to different eye patterns associated with drowsiness without extensive preprocessing.

High Accuracy and Performance:

CNNs have a proven track record of delivering high accuracy in image classification tasks. They are designed to handle variations in lighting and facial orientation, which are common in real-world situations. This robustness is crucial for our drowsiness detection system, as it needs to perform reliably no matter the conditions or the individual driver.

Scalability and Adaptability:

Another reason we chose CNNs is their scalability and adaptability. These models can easily be adjusted or expanded for specific applications, allowing us to improve performance as new challenges arise. Their modular design means we can integrate new techniques or layers into the model, ensuring that our system can evolve alongside advancements in technology.

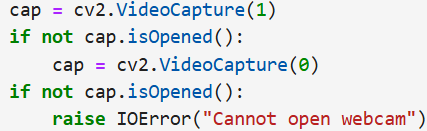
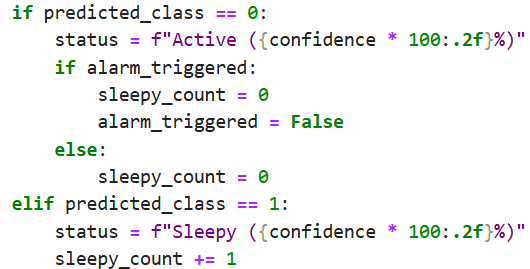
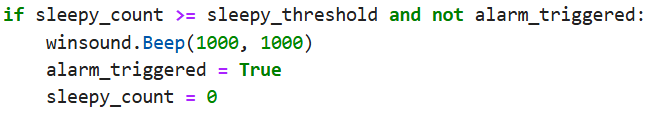
|  |  |  |
| --- | --- | --- |
| *Model: “sequential\_3”* |  |  |
| ***Layer (type)*** | ***Output shape*** | ***Param#*** |
| *Conv2D* | (None,222,222,32) | 896 |
| *BatchNormalization* | (None,222,222,32) | 128 |
| *MaxPooling2D* | (None,111,111,32) | 0 |
| *Dropout* | (None,111,111,32) | 0 |
| *Conv2D* | (None,109,109,64) | 18496 |
| *BatchNormalization* | (None,109,109,64) | 256 |
| *MaxPooling2D* | (None,54,54,64) | 0 |
| *Dropout* | (None,54,54,64) | 0 |
| *Conv2D* | (None,52,52,128) | 73856 |
| *BatchNormalization* | (None,52,52,128) | 512 |
| *MaxPooling2D* | (None,26,26,128) | 0 |
| *Dropout* | (None,26,26,128) | 0 |
| *Conv2D* | (None,24,24,256) | 295168 |
| *BatchNormalization* | (None,24,24,256) | 1024 |
| *MaxPooling2D* | (None,12,12,256) | 0 |
| *Dropout* | (None,12,12,256) | 0 |
| *Flatten* | (None,36864) | 0 |
| *Dense* | (None,128) | 4718720 |
| *Dropout* | (None,128) | 0 |
| *Dense*  **Total params**: 5,109,187  **Trainable params**: 5,108,225  **Non-trainable params**: 960  **Optimizer params**: 2 | (None,1) | 129 |

**Fig.2** CNN Architecture

## 4.3 System Design

### 4.2.3 System Architecture

The design of the architecture for the system of driver drowsiness detection is always real-time; therefore, it moves from the capture of video frames to the generation of alerts. The system consists of the following interconnected layers:

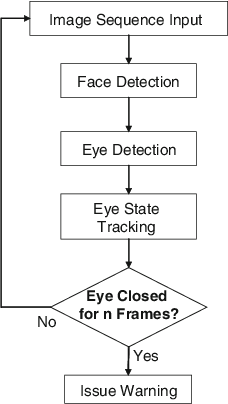
1. Data Input (Camera Feed)
2. Purpose: Capture live video of the driver's face in real time.
3. Implementation: Initialize a webcam using OpenCV and attempt to open an external or built-in camera. Video Frames are captured continuously then are processed individually. If the webcam cannot open, error is raised.
4. Preprocessing Layer
5. Motive: To prepare each video frame for further feature extraction and classification.
6. Implementation: Face Detection - Haar Cascade Classifier is used for detecting the face of the driver within a frame. It makes sure that all the processing around the area of interest would be done. Eye Detection - Within the identified face, Haar Cascade Classifier identifies the areas for eyes. The regions are cropped and processed for classification.
7. Image Resizing and Normalization: Crop the eye region to a size that will fit the CNN model's input size, 224 x 224. Normalize pixel values to be in the range of 0-1 through dividing by 255.0, thus having the same input go into the model.
8. Feature Extraction and Model Inference Layer
9. Objective: Predict the state of the driver as Active or Sleepy through a CNN-based deep learning model.
10. Loads: A pre-trained CNN model named fatigue\_model\_cnn.h5 is to be loaded using TensorFlow. The processed eye region is passed to the model for inference .The CNN model has outputs in the form of probabilities of two classes: Class 0 ‘Active’ and Class 1 ‘Sleepy’.
11. The state with the maximum-probability prediction is selected and its respective confidence score is logged.
12. Decision Layer
13. Goal: Predict across frames successively in order to avoid false alarms and determine at which frames to raise alerts.
14. Sleepy Counter Mechanism: It keeps track of the count of successive frames classified as Sleepy. The threshold to ring an alarm is set to 5 consecutive Sleepy frames.
15. Reset Mechanism: A frame identified as Active, the counter and all alarms triggered are reset to zero.
16. Threshold-Based Alarm: Counter exceeding the threshold value, an audio alarm is invoked with winsound.
17. Alarm System
18. Goal: Sound an alarm as soon as and as long as the system establishes that the driver is persistently sleepy.
19. Implementation: Audio beep is played using winsound. Beep whenever drowsiness is detected for the threshold number of consecutive frames. The system prevents unnecessary alerts by resetting the counter and deactivating the alarm after it has been triggered.
20. Real-Time Feedback (UI Layer)
21. Function: Update in real time, the state of the driver across the video feed
22. Implementation: The state of the driver as detected (Active or Sleepy) along with a confidence score overlay on the video feed using OpenCV. It also superimposes bounding boxes around detected faces to visually highlight the region of interest.

The processed frames are displayed in a window titled "Sleepiness Detection Window."

This layered architecture provides a sound and efficient drowsiness detection system that balances computational efficiency with real-time responsiveness to safeguard drivers.

### 4.3.2 Component Interaction

The interaction between components in the driver drowsiness detection system follows a structured flow to ensure efficient, real-time detection and alerting:



**Fig.3** Flowchart Implementation

# Chapter 5

**RESULT AND DISCUSSION**

## 5.1 Performance Metrics

For your system, the performance metrics focus on evaluating how well the model identifies the driver’s state (active or sleepy). Here are the key metrics for your project:

### 5.1.1. Accuracy

* **Description**: This metric measures the overall correctness of the model in classifying the driver’s state (active, sleepy, or yawning).
* **Formula**: Accuracy=
* **Relevance**: It shows how often the model makes the correct prediction, which is critical for ensuring the system performs reliably during real-time detection.

### **5.1.2.** **Precision**

* **Description**: Precision measures the accuracy of positive predictions (how many instances predicted as "Sleepy" are actually "Sleepy").
* **Formula**: Precision=

* **Relevance**: For drowsiness detection, high precision is important to avoid false positives (e.g., mistakenly detecting a driver as "Sleepy" when they are not).

### **5.1.3. Recall**

* **Description**: Recall measures the ability of the model to correctly identify all positive instances (how many "Sleepy" drivers are actually classified as "Sleepy").
* **Formula**: Recall=

* **Relevance**: High recall ensures that the model does not miss out on detecting a "Sleepy" driver, which is crucial for safety.

### 5.1.4.F1-Score

* **Description**: The F1-score is the harmonic mean of precision and recall, providing a balance between the two.
* **Formula**: F1-Score=

* **Relevance**: The F1-score is particularly important in this project as it balances the need to avoid false positives and false negatives, ensuring that both accurate detection and safety are maintained.

### 5.1.5. Confusion Matrix

* **Description**: A confusion matrix provides a breakdown of the model’s predictions against the actual labels, showing the true positives, true negatives, false positives, and false negatives.
* **Relevance**: The confusion matrix helps to visually identify where the model is making errors and provides insight into which states (active, sleepy, yawn) are being misclassified.

### 5.1.6. Real-Time Inference Time

* **Description**: Measures how long the model takes to process each frame of the camera feed and make predictions.
* **Relevance**: Since the system needs to detect drowsiness in real time, a fast inference time is crucial to ensure timely alerts.

### 5.1.7. False Positive Rate

* **Description**: The rate at which the model incorrectly classifies a non-drowsy driver as "Sleepy".
* **Formula**: FPR =

* **Relevance**: A low false positive rate ensures that the model doesn't unnecessarily alert the driver, which could lead to distractions or inconvenience.

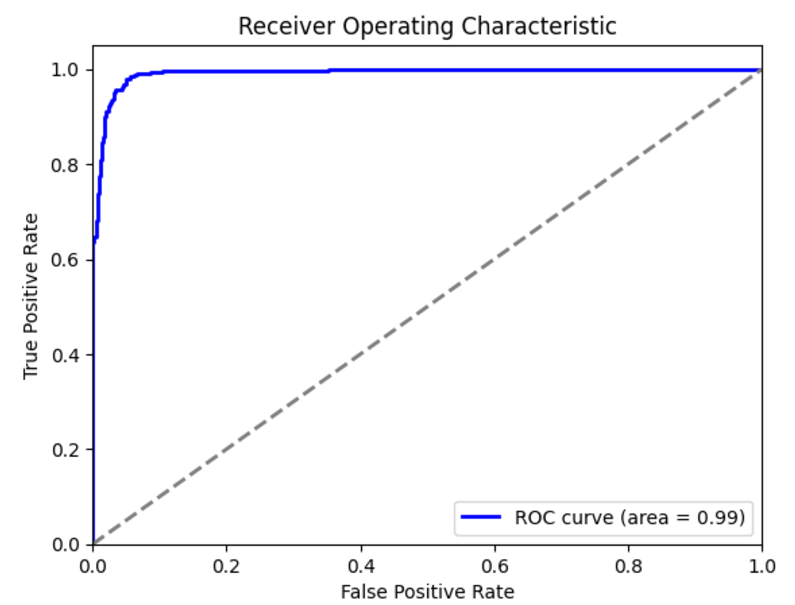
## 5.2 Result Analysis

### 5.2.1 ROC Curve

The training ROC curve displays the model’s ability to

distinguish between the two classes based on its predictions on the test data. From the Figure it is observed that, an AUC of 0.95 indicates that the model performs exceptionally well on the test set, as it achieves a high true positive rate while maintaining a low false positive rate across various classification thresholds. It is also observed that, a high AUC on the test data suggests that the model has effectively generalized and can reliably separate positive and negative examples in unseen data.

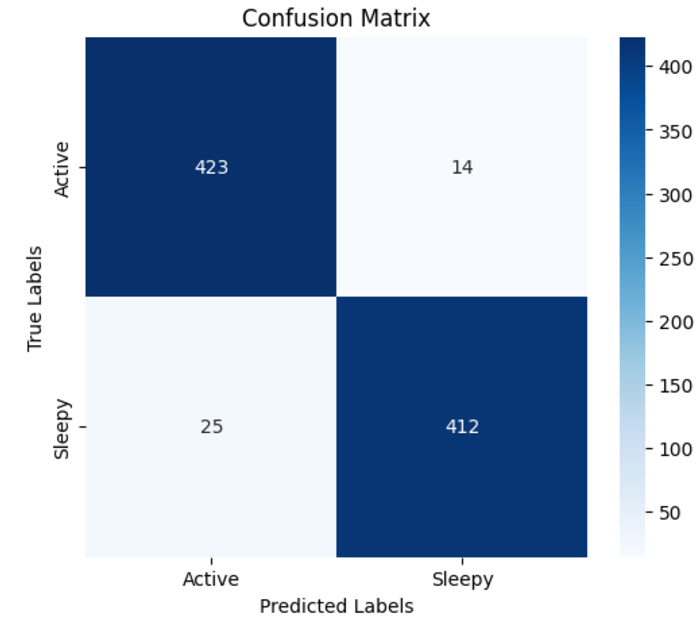
**Fig.4** ROC Curve



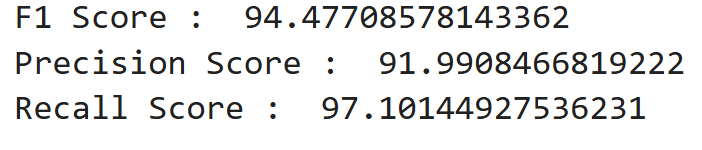
### **5.2.2 Confusion Matrix**

Confusion Matrix provides a clear overview of how well the system is performing by comparing its predictions against the actual states of the drivers. Analyzing this confusion matrix allows developers to pinpoint weaknesses in the detection system, such as a high rate of false negatives, which poses a significant safety risk by failing to identify drowsy drivers. Similarly, frequent false positives could lead to unnecessary alerts that frustrate drivers. By understanding these dynamics, developers can refine the detection algorithms, ultimately enhancing the system's accuracy and contributing to safer driving

**Fig.5** Confusion Matrix



conditions on the road.



## 5.3 Screenshots

# 

# Conclusion

As we conclude our project on detecting driver drowsiness, we’ve come to appreciate the significant role that technology plays in enhancing road safety. By utilizing advanced algorithms and machine learning techniques, we can create solutions that not only track driver alertness but also provide timely alerts to help prevent potential accidents.

This project has deepened our understanding of how innovative solutions can address real-world challenges, ultimately paving the way for safer driving experiences and reducing the risks associated with drowsy driving.

At the heart of this project is our commitment to addressing a real-world problem—drowsy driving. We hope that our work contributes to reducing the risks associated with this issue and ultimately helps save lives. As we look ahead, we’re inspired by the potential for further research and development in this area, and we believe that our project is just the beginning of what can be achieved in enhancing automotive safety. Together, we can create a safer driving environment for everyone on the road.

While this system effectively alerts drowsy drivers, it is important to note that technology alone cannot eliminate the risks associated with drowsy driving. Drivers must remain aware of their physical limits and avoid relying solely on alerts to stay awake. An alert triggered even one second too late might be insufficient to prevent a crash. Therefore, this system should complement, not replace, responsible driving practices.

# References

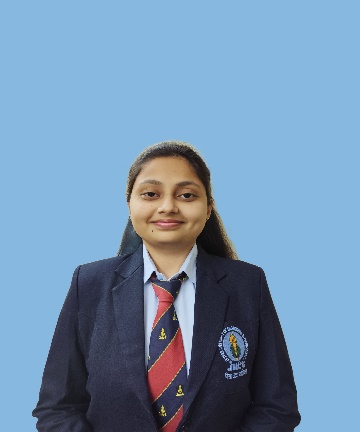
1. Wikipedia: [www.wikipedia.org/](http://www.wikipedia.org/)
2. ResearchGate: [www.researchgate.net/](http://www.researchgate.net/)
3. OpenAI: [chat.openai.com](https://chat.openai.com)
4. Kaggle Dataset: [www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset](http://www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset)
5. Research Article: [www.mdpi.com/1424-8220/23/14/6459](http://www.mdpi.com/1424-8220/23/14/6459)
6. Drowsy Driving Behaviour Survey: [rosap.ntl.bts.gov/view/dot/1725/dot\_1725\_DS 1.pdf](https://rosap.ntl.bts.gov/view/dot/1725/dot_1725_DS%201.pdf)

# Contributors

1. SAJAL KORDE  
   Enrolment Number: 221B319  
   Email: [221b319@juetguna.in](mailto:221b319@juetguna.in)

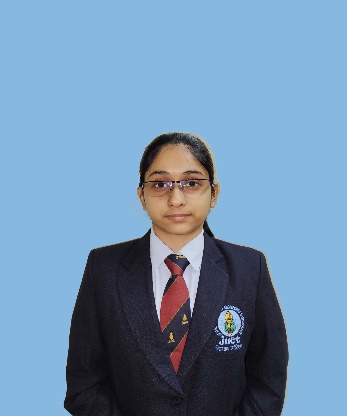
**Personal Email:** [sajal@gmail.com](mailto:sajal@gmail.com)

**Address:**

1. SHRUTI BHARGAVA  
   Enrolment Number: 221B374  
   Email: [221b374@juetguna.in](mailto:221b374@juetguna.in)

**Personal Email:** [shrutibhargava2004@gmail.com](mailto:shrutibhargava2004@gmail.com)

**Address:** Betul, M.P.

1. SNEHIL SHARMA  
   Enrolment Number: 221B387   
   Email: [221b387@juetguna.in](mailto:221b387@juetguna.in)

**Personal Email:** [ssnehil06@gmail.com](mailto:ssnehil06@gmail.com)

**Address:** Bhilwara, Raj.