Non-Invasive Blood Group Detection Using Deep Learning Model

A Project work submitted in the partial fulfilment of the requirement for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. SAI RAM	Y21CSE279056	K. PRAVEEN KUMAR	Y21CSE279047
K. RADHA KRISHNA	Y21CSE279048	CH. DILEEP KUMAR	L22CSE279003

Under The Esteemed Guidance of

Dr. Md. Ali Mirza

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KRISHNA UNIVERSITY COLLEGE OF ENGINEERNG AND TECHNOLOGY

(Accredited by NAAC with B+ & Approved by AICTE, New Delhi)

RUDRAVARAM, MACHILIPATNAM, KRISHNA DISTRICT, A.P

2024-2025

KRISHNA UNIVERSITY COLLEGE OF ENGINEERNG AND TECHNOLOGY

(Accredited by **NAAC** with **B+** & Approved by **AICTE**, New Delhi)

RUDRAVARAM, MACHILIPATNAM, KRISHNA DISTRICT, A.P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"Non-Invasive Blood Group Detection Using Deep Learning Model"** is a bonafide work done by **K. SAI RAM(Y21CSE279056), K. PRAVEEN KUMAR (Y21CSE279047), K. RADHA KRISHNA (Y21CSE279048), CH. DILEEP KUMAR (L22CSE279003)** Submitted in partial fulfilment of the requirements for the award of the degree of "**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING**" during the academic year 2024-2025.

Project guide
Dr. Md. Ali Mirza
(Asst. Professor)

Department Coordinator
Mr. M. Sai Somayajulu
(Asst. Professor)

Signature of External Examiner

ACKNOWLEDGEMET

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with success. This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped us for the completion of my project work.

We are extremely grateful to our guide **Dr. Md. Ali Mirza Sir, Department of Computer Science & Technology, Krishna University College of Engineering and Technology,** for giving us an opportunity to work on a project with his timely suggestions and guidance.

We are extremely grateful to **Mr. M. Sai Somayajulu, Assistant Professor and Coordinator of Computer Science & Engineering Department, Krishna University College of Engineering and Technology,** for his motivation and valuable advice during this period.

We are extremely grateful to **Dr. R. Vijaya Kumari, Principle(i/c) and Head of the Department, Department of Computer Science & Technology, Krishna University College of Engineering and Technology,** for her motivation and valuable advice during this period.

We also thankful to **Registrar Prof. N. Usha** , **Rector Prof. M.V. Basaveswara Rao, and H'ble Vice-Chancellor Prof. K. Ramji Koona** ,Krishna University, Machilipatnam, for providing all the resources to carry out this work.

We thank all other **teaching faculty and non-teaching staff** of the **C.S.E** department, who rendered help directly or indirectly for their cooperation and encouragement.

We express our special thanks to **Librarian and All the library staff** of Krishna University College of Engineering and Technology, for providing the necessary library facilities.

Sincerely,

K. SAI RAM **Y21CSE279056**

K. PRAVEEN KUMAR **Y21CSE279047**

K. RADHA KRISHNA **Y21CSE279048**

CH. DILEEP KUMAR **L22CSE279003**

DECLARATION

This is to certify that the work reported in thesis titled " **Non-Invasive Blood Group Detection Using Deep Learning Model**", submitted in the Department of **COMPUTER SCIENCE AND ENGINEERING(CSE)**, Krishna University College of Engineering and Technology, Machilipatnam in the partial fulfilment of degree for the award of Bachelor of Technology, and is a bonafide work done by us.

The reported results are based on the project work entirely done by us and not copied from any other source.

Also, we declare that the matter embedded in this thesis has not been submitted by us in full or partial thereof for the award of any degree/diploma of any other institution or University previously.

TEAM MEMBERS,

K. SAI RAM	**Y21CSE279056**
K. PRAVEEN KUMAR	**Y21CSE279047**
K. RADHA KRISHNA	**Y21CSE279048**
CH. DILEEP KUMAR	**L22CSE279003**

INDEX

ABSTRACT

Determining blood type is essential, especially in critical scenarios like medical emergencies, diagnostic procedures, transfusion compatibility, organ transplants, and prenatal care. Conventional blood group determination relies on serological techniques, which, although accurate, involve invasive methods and require laboratory facilities. Additionally, manual testing performed by technicians can be subject to human error. To overcome these challenges, this project focuses on developing a precise and efficient blood typing system using pre-captured palm images. The proposed system integrates advanced image processing techniques and machine learning, particularly Convolutional Neural Networks (CNNs), to analyze fingerprint images and identify unique patterns associated with blood group phenotypes.

A CNN model is trained on an extensive dataset of labelled fingerprint images and demonstrates impressive accuracy in predicting blood groups. The proposed system offers a non-invasive, fast, and reliable alternative to traditional blood typing methods, significantly improving the speed and accessibility of blood group identification. This advancement enhances medical diagnostics, blood transfusion services, and resource-limited healthcare environments. This research opens up new opportunities in biometric technologies and plays a crucial role in advancing healthcare delivery by incorporating cutting-edge technology. Overall, the results reflect a major step forward in integrating biometric data with healthcare technologies, paving the way for further research and innovation in the field.

List of Figures

1.INTRODUCTION

1. INTRODUCTION

1.1 Introduction

This introduction outlines the shift from traditional blood typing to a technology-driven solution that leverages widely available biometric data. The need for rapid blood group identification has intensified in recent years, driven by demands in both medical and forensic applications. Traditional methods, while effective, are time-consuming and impractical in emergencies, often requiring hours to yield results. In contrast, the proposed system processes images in real-time, delivering fast and reliable outcomes. The use of Python Flask, OpenCV, TensorFlow, and Scikit-learn as tools underscores the technical sophistication of this approach, blending open-source technologies with cutting-edge machine learning. This research not only addresses practical healthcare challenges but also opens new avenues in biometric technology applications. The system's non-invasive nature eliminates the discomfort and risks associated with blood draws, making it suitable for a broader population. Furthermore, its scalability allows deployment in diverse settings, from urban hospitals to rural clinics. This introduction sets the stage for understanding how the fusion of image processing and CNNs can transform blood typing into a seamless, efficient process. By reducing dependency on lab facilities and trained personnel, the system democratizes access to critical diagnostic information. Ultimately, this project aims to bridge the gap between biometric innovation and healthcare delivery, offering a glimpse into the future of medical diagnostics.

An innovative and painless method for blood group determination involves analyzing fingerprints, combining advanced deep learning algorithms with medical data. This approach identifies blood type by examining the unique configurations of ridges and valleys in a person's fingerprint, information vital for medical professionals. In contrast to conventional blood tests, which can cause discomfort due to the use of needles, this fingerprint-based method offers a more comfortable and less invasive experience [1].

The patterns found in fingerprints are considered one of the most dependable and distinctive forms of identification, remaining consistent throughout a person's life [2]. Even in legal settings, fingerprints are often regarded as crucial evidence. The likelihood of two individuals possessing identical minute details within their fingerprints is exceedingly small, roughly one in 64 million, including identical twins. Furthermore, the configuration of ridges is both unique and unchanging from birth. Blood group, an inherited characteristic, also remains constant. Using fingerprints in analysis helps to significantly reduce the risk of infections. While traditional blood tests, which involve extracting blood via needles and subsequent expensive antibody procedures, are required for disease diagnosis and blood collection [3].

One of the main obstacles in creating a predictive model for blood groups is the scarcity of diverse fingerprint samples. Research on using fingerprints as a biometric for predicting blood groups and identifying age-related diseases is still in its early stages [4].

Fingerprint patterns are generally categorized into loops, whorls, and arches, depending on the direction and arrangement of the ridges. Loops, the most prevalent type, are characterized by ridges forming a loop-like shape. Whorls are defined by circular or spiral ridge formations, with several variations. Arches, which are simpler, feature flowing ridges without the presence of deltas. The

analysis of these patterns, including ridge orientation and the number of deltas, is crucial for accurate classification in both forensic and biometric applications [6].

Types of Fingerprint Patterns

1. **Loop**:
   Loop patterns are the most prevalent among fingerprint patterns. In this type, the ridges enter from one side of the finger, curve to form a loop, and then exit through the same side they came in. Loops are further categorized based on ridge flow direction:
   - **Ulnar Loops**: Ridges flow toward the little finger.
   - **Radial Loops**: Ridges flow toward the thumb.
2. **Whorl**:
   Whorl patterns consist of ridges arranged in circular or spiral formations. This pattern can be further divided into subtypes, including:
   - Plain Whorls
   - Central Pocket Loops
   - Double Loops
   - Accidental Whorls
3. **Arch**:
   Arch patterns are characterized by ridges flowing in a wave-like motion from one side of the finger to the other. Unlike loops and whorls, arch patterns do not feature prominent deltas (triangular ridge formations). They are analyzed and classified based on the arrangement and orientation of ridges, as well as specific characteristics like ridge count and the presence of deltas.

Fig-1.1.a Fingerprint Patterns

System Architecture:

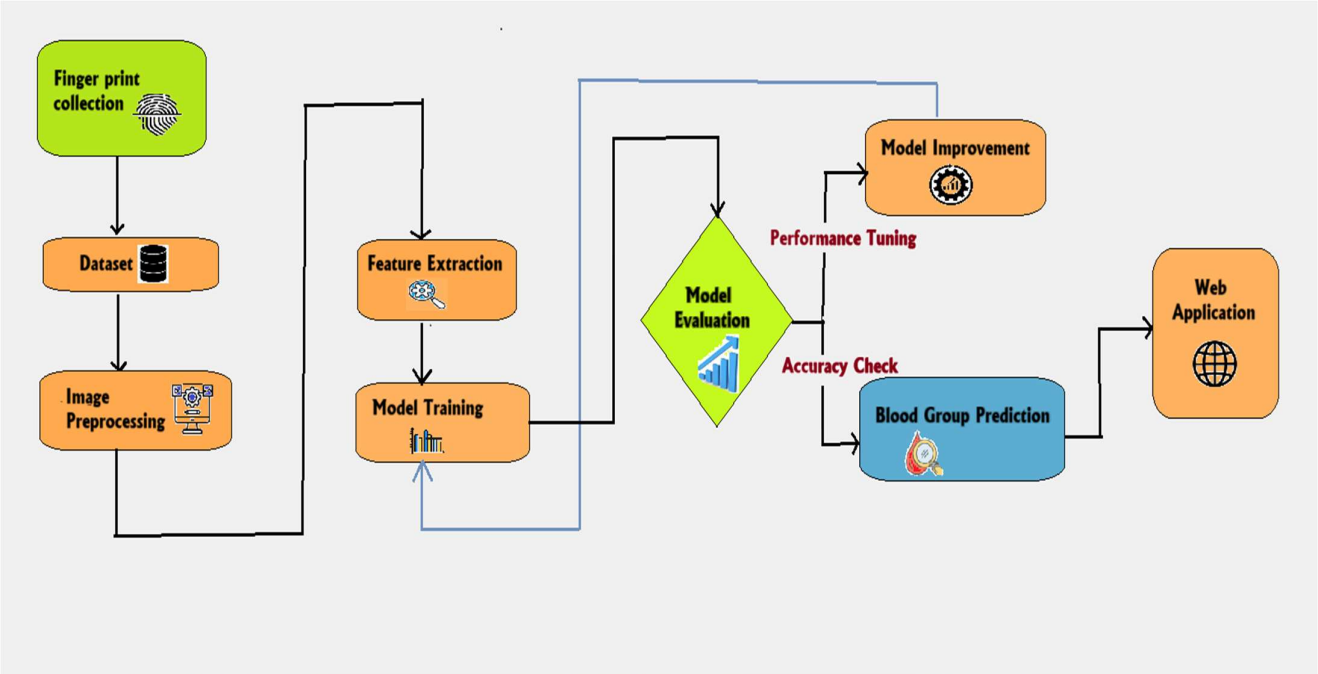


Fig-1.1.b Project Basic Architecture

- Fingerprint Collection: The process starts with collecting fingerprint samples. These are likely captured using specialized devices to ensure clarity and accuracy.
- Dataset Creation: The collected fingerprints are stored in a structured format to form a dataset, which serves as the foundation for further analysis.
- Image Preprocessing: Fingerprint images undergo preprocessing steps, such as noise reduction, enhancement, and segmentation, to improve their quality and extract useful information.
- Feature Extraction: Important features from the fingerprints, like ridge patterns, minutiae points, or unique identifiers, are extracted to create a representation suitable for machine learning.
- Model Training: The extracted features are fed into a machine learning model, which is trained to identify patterns and correlations, such as blood group prediction based on fingerprint traits.
- Model Evaluation: The trained model is evaluated for its performance using metrics like accuracy and precision to ensure reliability.
- Model Improvement: If the model's performance is unsatisfactory, it undergoes performance tuning, optimization, or retraining to enhance accuracy and efficiency.
- Blood Group Prediction: Once the model is refined, it predicts blood groups based on input fingerprints. The accuracy of these predictions is checked to validate the system.
- Web Application Integration: The final, optimized model is integrated into a web application, enabling users to access the blood group prediction feature seamlessly.

1.2 Problem Statement

Traditional blood typing relies on invasive serological techniques that require blood samples and laboratory facilities, making it impractical for rapid use in emergencies or resource-limited settings. These methods are time-consuming, depend on trained personnel, and are prone to human error, risking delays or inaccuracies in critical medical scenarios like transfusions or forensic investigations. The lack of portable, non-invasive alternatives restricts timely blood group identification, particularly in underserved regions or urgent situations. This project addresses these challenges by developing a system that uses fingerprint image analysis and machine learning to determine blood types efficiently and accurately. The goal is to overcome the limitations of conventional methods, offering a fast, reliable, and accessible solution for modern healthcare and forensic needs.

1.3 Motivation of the Project

The motivation for this project stems from the pressing need to overcome the limitations of traditional blood typing methods, which are invasive, time-consuming, and resource-intensive, particularly in critical medical and forensic scenarios. In emergencies, such as trauma cases or mass casualty events, the ability to quickly determine blood type can mean the difference between life and death, yet current techniques often fail to deliver timely results. The reliance on blood samples and laboratory infrastructure excludes many underserved regions from accessing this vital diagnostic tool, inspiring the pursuit of a more inclusive solution. Additionally, the potential for human error in manual testing whether through misinterpretation or procedural mistakes poses risks that modern technology can mitigate. This project is driven by the desire to create a non-invasive alternative that leverages fingerprint images, a biometric resource already widely collected, to streamline blood group identification. The growing demand for rapid diagnostics in healthcare and forensics further fuels the motivation, as delays in blood typing can hinder transfusion compatibility or criminal investigations. Advances in machine learning, particularly Convolutional Neural Networks (CNNs), provide a powerful tool to analyze complex patterns, making this an opportune moment to explore their application in medical diagnostics.

1.4 Objective

The primary objective of this project is to develop a precise, efficient, and non-invasive blood typing system that utilizes pre-captured palm images, specifically fingerprint patterns, to determine blood groups with high accuracy. By integrating advanced image processing techniques and machine learning, particularly Convolutional Neural Networks (CNNs), the system aims to identify unique ridge patterns associated with blood group phenotypes. This research seeks to eliminate the need for invasive blood draws, offering a safer and more comfortable alternative for patients across various medical contexts. Another key objective is to enhance the speed of blood group identification, making it viable for emergency situations where rapid results are critical. The system is designed to achieve high performance metrics, including accuracy, precision, recall, and F1-score, ensuring reliability comparable to traditional serological methods. By training a CNN model on an extensive dataset of labeled

fingerprint images, the project aims to establish a robust correlation between biometric patterns and blood types. The objective includes creating a scalable solution that can be implemented in diverse settings, from well-equipped hospitals to remote clinics with limited resources. Additionally, the system strives to reduce dependency on laboratory facilities and trained personnel, democratizing access to blood typing technology. The use of tools like Python Flask, OpenCV, TensorFlow, and Scikit-learn reflects the goal of building a technically sophisticated yet practical application.

2. SYSTEM REQUIREMENTS & SPECIFICATIONS

2. SYSTEM REQUIREMENTS & SPECIFICATIONS

2.1 SOFTWARE REQUIREMENTS:

• Operating system: Windows 10 Pro
• Coding Language: Python 3.12.3, HTML, CSS, JS
• Web Framework: Flask
• Packages Used:

Tensorflow-2.18.0
Sickit-learn-1.2.3
Pandas-2.2.2
Pillow-10.3.0
Opencv-python-4.10.0.84
Keras-3.3.3
Numpy

2.2 HARDWARE REQUIREMENTS:

Requirement specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be in on a server, which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

Hardware Requirements

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics. Hard disk speed: To improve performance, install windows and Xampp Server on a solid state drive (SSD). Hard disk space: Minimum of 800MB up to 210 GB of available space, depending on features installed, typical installations require 20-50 GB of free space. Processor: 1.8 GHz or faster processor. Quad-core or better recommended RAM: 4 GB to 8 GB of RAM recommended (2.5 GB minimum if running on a virtual machine) Graphic card: Video card that supports a minimum display resolution of 720p (1280 by 720) Sublime Text will work best at a resolution of WXGA (1366 by 768) or higher.

• System: Pentium i3 Processor.
• Hard Disk: 500 GB.
• Monitor: 15" LED
• Input Devices: Keyboard, Mouse
• Ram: 4 GB

2.3 Functional Requirements

(What the system should do):

1. **Image Acquisition:**
   - **Expanded:** The system must provide a clear and straightforward mechanism for users (or other system components) to input pre-captured palm or fingerprint images. This could involve:
     - Accepting image files in common formats (e.g., JPEG, PNG).
     - Integrating with a camera or image capture device (if real-time capture is a future consideration, though the abstract suggests pre-captured).
     - Providing clear instructions or prompts for image selection or upload.
     - Handling potential issues with image input, such as invalid file formats or corrupted files, with informative error messages.
2. **Image Preprocessing:**
   - **Expanded:** Before analysis, the system needs to prepare the input image to ensure optimal performance of the feature extraction and CNN model. This involves a series of automated steps, potentially including:
     - **Noise Reduction:** Applying algorithms to minimize artifacts or unwanted variations in the image that could interfere with feature detection.
     - **Normalization:** Adjusting the image's brightness, contrast, or pixel intensity range to ensure consistency across different images and improve model robustness.
     - **Resizing:** Scaling the image to a specific size expected by the CNN model.
     - **Orientation Correction:** Potentially rotating or aligning the fingerprint image to a standard orientation for consistent feature extraction.
     - **Image Enhancement:** Applying filters or techniques to sharpen edges, highlight ridge patterns, or improve the overall clarity of the fingerprint details.
3. **Feature Extraction:**
   - **Expanded:** This crucial step involves identifying and isolating distinctive characteristics or patterns within the preprocessed fingerprint image that the CNN model can learn to associate with different blood groups. This might involve:
     - Employing techniques to analyze ridge patterns (minutiae like ridge endings, bifurcations), overall ridge flow, or other textural features.
     - Converting these identified patterns into a numerical or vector representation that can be fed into the CNN.
     - This stage is often implicitly handled by the convolutional layers of the CNN itself, which automatically learn relevant features. However, the design might involve specific initial feature engineering steps before feeding into the CNN.
4. **Blood Group Prediction:**
   - **Expanded:** This is the core functionality of the system. Based on the extracted features and the trained CNN model, the system must:
     - Process the feature representation through the layers of the CNN.
     - Generate a probability distribution across the different possible blood groups (A+, A-, B+, B-, AB+, AB-, O+, O-).
     - Select the blood group with the highest probability as the predicted blood type.
     - Provide a clear indication of the predicted blood group.

5. **CNN Model Integration:**
    - **Expanded:** The system must seamlessly incorporate the pre-trained Convolutional Neural Network model. This includes:
        - Loading the trained model (weights and architecture) into the system's memory.
        - Ensuring compatibility between the image processing pipeline and the input requirements of the CNN model.
        - Providing the necessary software libraries and dependencies (e.g., TensorFlow, Keras) to run the CNN model.
        - Potentially allowing for updates or replacement of the CNN model with improved versions.
6. **Result Display:**
    - **Expanded:** The system needs to present the predicted blood group to the user in a clear and understandable format. This could involve:
        - Displaying the predicted blood group label (e.g., "A+").
        - Optionally displaying the confidence level or probability associated with the prediction.
        - Providing a user-friendly interface (if applicable) to view the results.
        - Logging or storing the prediction results for future reference or analysis.
7. **Performance Evaluation:**
    - **Expanded:** To assess the effectiveness of the system, it must be able to calculate and display key performance metrics. This requires:
        - Comparing the system's predictions against known correct blood groups in a test dataset.
        - Calculating and presenting metrics such as:
            - **Accuracy:** The overall percentage of correct predictions.
            - **Precision:** For each blood group, the proportion of correctly predicted instances out of all instances predicted as that group.
            - **Recall:** For each blood group, the proportion of correctly predicted instances out of all actual instances of that group.
            - **F1-score:** The harmonic mean of precision and recall, providing a balanced measure of performance.
        - Potentially displaying these metrics in a tabular or graphical format.
8. **Data Input (for Training/Testing):**
    - **Expanded:** For developing and evaluating the CNN model, the system needs a robust mechanism to manage the labeled dataset. This includes:
        - Allowing the input of fingerprint images along with their corresponding blood group labels.
        - Organizing and storing this data in a structured manner.
        - Potentially providing tools for data exploration, visualization, and quality control.
        - Facilitating the splitting of the dataset into training, validation, and testing sets.

9. **Model Training (Potentially):**
    - **Expanded:** While the abstract mentions a "trained" CNN, the system might include functionality to retrain or fine-tune the model with new data to improve its performance or adapt to different image qualities or populations. This would involve:
        - Implementing the training algorithm for the CNN.

- Allowing the selection of training parameters (e.g., learning rate, number of epochs).
- Monitoring the training process (e.g., loss and accuracy on training and validation sets).
- Saving the trained model weights.

10. **User Interface (if applicable):**
    - **Expanded:** If a user-facing application is developed (as suggested by the mention of Python Flask), it should provide an intuitive way for users to interact with the system. This includes:
        - A clear method for uploading or selecting fingerprint images.
        - Visual feedback during the processing stages.
        - A clear display of the predicted blood group and potentially the confidence level.
        - Easy navigation and a user-friendly design.

2.4 Non-Functional Requirements

(How well the system should perform):

1. **Accuracy:**
    - **Expanded:** The system must achieve a high degree of correctness in its blood group predictions. The specific acceptable accuracy level would depend on the intended use case (e.g., screening vs. diagnostic). This needs to be quantitatively defined (e.g., achieve >95% accuracy on a benchmark dataset).
2. **Performance/Speed:**
    - **Expanded:** The time taken to process an input image and generate a blood group prediction should be minimal. This is crucial for real-world applications, especially in time-sensitive scenarios. Specific performance targets (e.g., prediction within X seconds) should be defined.
3. **Reliability:**
    - **Expanded:** The system should consistently produce accurate results under various conditions (e.g., different image qualities, variations in fingerprint capture). It should be robust to minor variations in input and should not produce erroneous results frequently. Measures to ensure reliability might include rigorous testing and validation.
4. **Non-Invasiveness:**
    - **Expanded:** The system must function solely based on pre-captured images, eliminating the need for any physical contact or extraction of biological samples from the individual at the time of analysis. This is a fundamental requirement highlighted in the abstract.
5. **Scalability (Potential):**
    - **Expanded:** If the system is intended for widespread use, it should be able to handle a large volume of image inputs and prediction requests without significant degradation in performance. This might involve considerations for the underlying infrastructure and the efficiency of the algorithms.
6. **Usability (if applicable):**
    - **Expanded:** If a user interface is part of the system, it should be easy for the intended users (e.g., medical professionals, technicians) to understand and operate without extensive training. The interface should be intuitive, efficient, and free from ambiguity.
7. **Maintainability:**

- **Expanded:** The system's design and implementation should allow for easy modification, updates, and bug fixes. The codebase should be well-structured, documented, and adhere to good programming practices to facilitate ongoing maintenance and evolution.

8. **Security (Potential):**
   - **Expanded:** If the system handles potentially sensitive biometric data (fingerprint images) and predicted medical information (blood group), robust security measures must be implemented to protect this data from unauthorized access, modification, or disclosure. This includes considerations for data storage, transmission, and access control.
9. **Portability (Potential):**
   - **Expanded:** Depending on the intended deployment environment (e.g., desktop application, web service, mobile app), the system might need to be portable across different operating systems (Windows, macOS, Linux, Android, iOS) or devices. This would influence the choice of programming languages, frameworks, and deployment strategies.

3. SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Existing System:

The traditional method for blood group detection typically involves a manual procedure carried out in a laboratory setting. The existing systems for blood group detection primarily rely on serological methods, which involve the agglutination reaction between antigens and antibodies. These traditional methods, although accurate, are labour-intensive, time-consuming, and require skilled personnel and laboratory infrastructure.
The main objective of this project is to develop a quick and efficient way to determine the blood type. In today's world where blood transfusion plays an important role in trauma and operational cases, it is really important to transfuse the correct blood type into the patient or else the blood can clump, which may be fatal. This project aims to address these limitations by developing a rapid and efficient blood group detection method using fingerprint image processing.
The importance of accurate blood typing is underscored by the critical role of blood transfusions in emergency and surgical situations. Administering the wrong blood type can have fatal consequences. Traditional lab testing can be prone to inaccuracies and delays, especially in urgent scenarios. The proposed image processing approach minimizes human intervention, thereby enhancing accuracy and reducing errors. Furthermore, it significantly accelerates the blood typing process, which is crucial in time-sensitive situations.

This research investigates the feasibility of establishing a reliable correlation between fingerprint patterns and blood groups to enable accurate fingerprint-based blood typing. The project will initially evaluate existing Convolutional Neural Network (CNN) architectures, and based on the performance, a custom model may be developed to optimize accuracy.
Advancements in biometrics have paved the way for exploring fingerprint image processing as a non-invasive and rapid alternative for blood group detection. While the hypothesis of a correlation between unique fingerprint ridge patterns and blood groups is promising, current systems face challenges such as the need for extensive datasets, substantial computational resources, and robust algorithms for accurate classification.

3.2 disadvantages:

The concept of non-invasive blood group detection using deep learning models offers an exciting potential in healthcare, but it also comes with some challenges and disadvantages:

1. **Data Quality and Availability**: For deep learning models to work effectively, they need large amounts of high-quality data. Gathering enough labeled, high-quality datasets of blood samples with varied characteristics for training the model could be a significant hurdle. The lack of sufficient and diverse data could result in poor model performance or biases.
2. **Accuracy and Reliability**: Although deep learning can achieve impressive results, non-invasive methods (such as using sensors or imaging techniques) may not always be as precise as traditional invasive methods like blood tests. The models might struggle with detecting blood groups in certain scenarios, such as with people having rare or uncommon blood types.
3. **Complexity of Model Training**: Training deep learning models can be computationally expensive and time-consuming. The model needs to be trained on a variety of real-world scenarios, which could include challenges like different lighting conditions, skin tones, or other environmental factors that affect sensor data.

4. **Regulatory Challenges**: Implementing such non-invasive systems in healthcare would require approval from medical regulatory bodies, which can be a lengthy and expensive process. `Ensuring the safety and effectiveness of these models in clinical environments is crucial.
5. **User Variability**: Non-invasive detection methods may face difficulty in accounting for variations in individuals. Factors such as skin tone, body temperature, or even environmental conditions could affect the sensor readings and lead to inconsistent results across different users.
6. **Ethical and Privacy Concerns**: Non-invasive methods may raise privacy concerns, as they could involve the collection of personal and potentially sensitive data (like facial images or biometric data) for blood group determination. Safeguards would need to be in place to protect this information.
7. **Cost of Implementation**: Developing and implementing a reliable deep learning-based non-invasive blood group detection system could be costly in terms of both hardware (e.g., sensors) and the computational resources required for deep learning models. These costs may make it difficult for the technology to be widely adopted, particularly in low-resource settings.
8. **Dependence on External Factors**: Non-invasive systems might be influenced by external factors such as lighting, angle of data collection, or other environmental variables, which could degrade the model's performance in uncontrolled settings.
9. **Generalization Across Populations**: The model's ability to generalize across different demographics (age, ethnicity, medical conditions) may be limited if the training data is not sufficiently diverse. This could reduce the model's performance for some groups of people.

3.3 Proposed System:

CNNs have shown remarkable capabilities in image analysis and pattern identification, making them a promising technology for fingerprint-based blood group detection. By training on extensive datasets of labeled fingerprint images, CNNs can learn to identify complex patterns and relationships associated with blood groups. However, the development and implementation of such systems face challenges, including the demand for significant computational power, complex network designs, and large, accurately labeled datasets.

Despite their potential, using CNNs for fingerprint-based blood group detection remains relatively unexplored. Research in this area is limited, and practical applications are scarce. Current systems lack the necessary reliability and accuracy for widespread use in medical diagnostics. There is a notable absence of research on optimizing CNN architectures specifically for this purpose, as well as on establishing comprehensive, annotated fingerprint datasets. My research addresses this gap by achieving higher accuracy than previously reported systems.

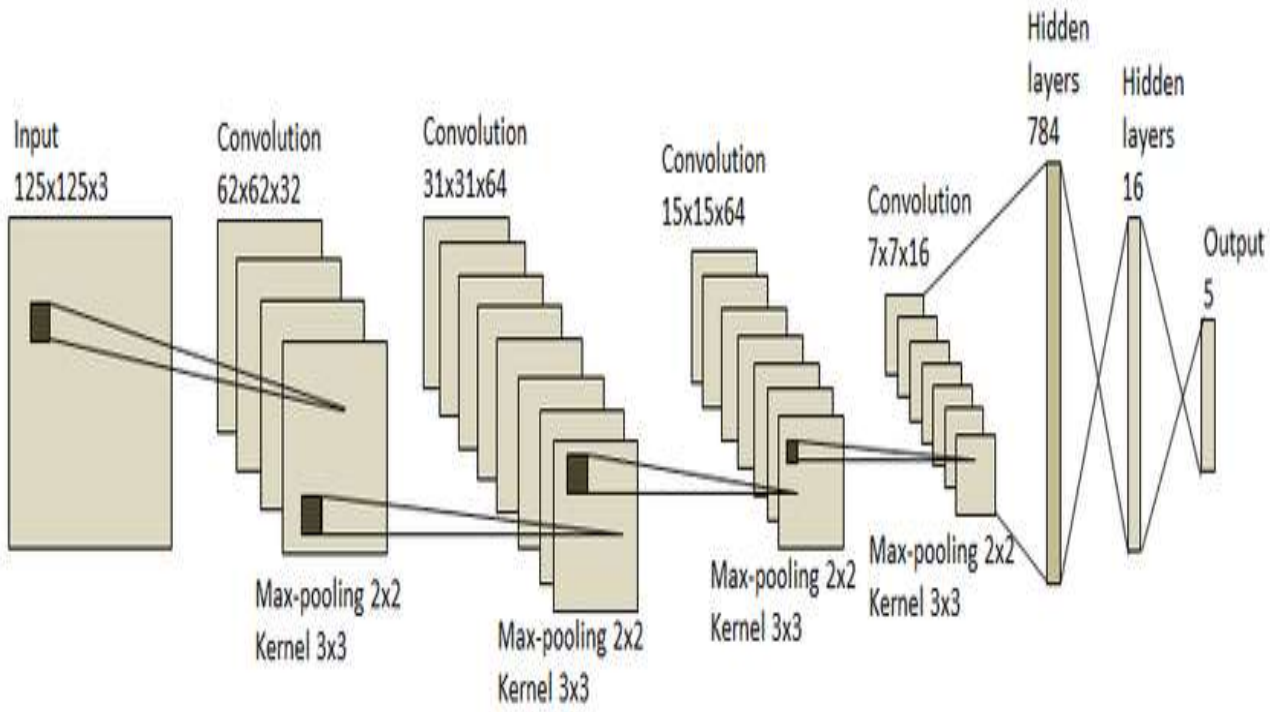


Fig-3.3.a CNN Model Basic Architecture

Methodology

The fundamental motive of the project is to use the relationship among details and blood type to create an accurate fingerprint-based blood group test and evaluate the feasibility of the concept. First the model is evaluated using existing CNN architectures and upon observing the performance a custom model can be constructed for better performance.

Feature Extraction :

The feature extraction plays a major role in latent fingerprint matching. For the though transform algorithm minutiae points are needed. Minutiae points are extracted for matching. They are specific points in a fingerprint image, determined by the termination or the bifurcation of the ridge lines.

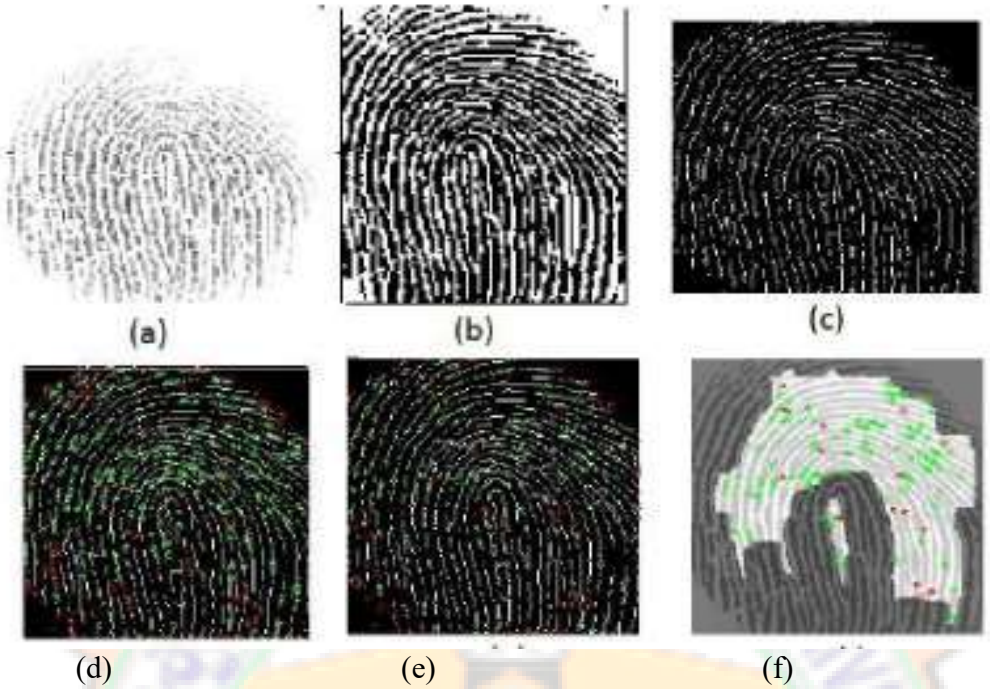


Fig-3.3.b Feature Extraction steps

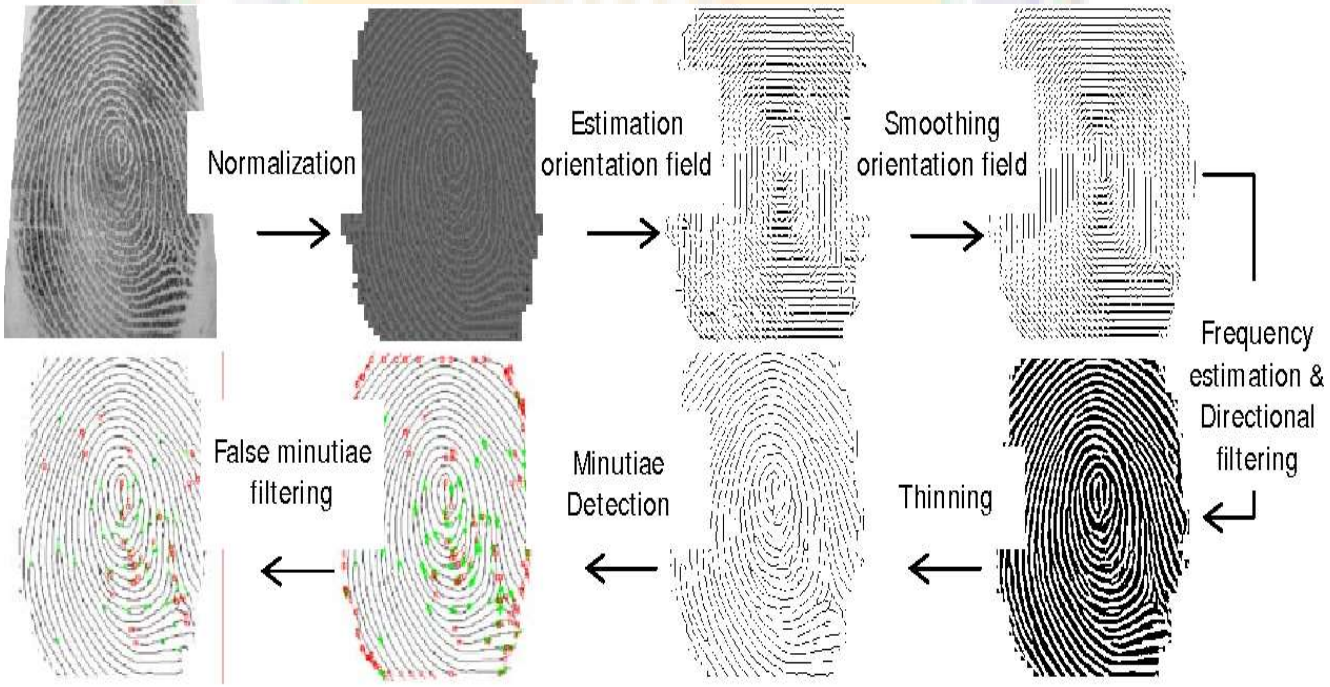


Fig-3.3.c Pre-processing and Feature Extraction Process[11]

(a)Input image
(b)Smoothing
(c)Thinning
(d)minutiae identification
(e)removal of false minutiae
(f)Region of Interest

Feature extraction techniques are used to obtain a feature vector which is a reduced representation of the input data which contains information about the most important features of data. This feature vector is sent to a Convolutional Neural Network for classification.

Thinning Image: It refers to the technique of reducing the thickness of lines or edges in a binary image to a single-pixel width while preserving the structural information and connectivity of the shapes. Thinning is often employed to simplify the representation of shapes, making them more suitable for further analysis or pattern recognition tasks. In the current context we can obtain minimal representation of fingerprints that retains the shape and vital information of the fingerprint.

Minutiae Detection: Minutiae points are specific locations where ridge patterns in a fingerprint exhibit unique characteristics. The most common types of minutiae are bifurcations and ridge endings. Bifurcations take place when a ridge divides into two, while ridge endings occur at points where a ridge concludes. Minutiae detection is the process of identifying and locating these minutiae points in a fingerprint image. This process is typically part of the feature extraction phase in fingerprint recognition systems.

Real Minutiae: The term real minutiae refer to the actual minutiae points that exist on a person's fingertip. These are the biometrically relevant features that contribute to the uniqueness of a fingerprint. Real minutiae are the points used in fingerprint matching algorithms to distinguish one fingerprint from another.

Data Augmentation: Augmentation is a method employed in machine learning and deep learning to expand data size by introducing alterations or enhancements to existing data, with the aim of enhancing the efficiency, generalization, and robustness of machine learning models, particularly in situations with restricted data availability. Due to the limited number of documents containing fingerprints and related blood, we need to expand the existing data to come up with a better model.

Improving the accuracy of Latent Matching Approach using texture features

We propose to automate the process of fingerprint matching after reconstruction of latent fingerprints. We have done the Hough transform matching after the enhancement of input images. After feature extraction we have to align the minutiae point for matching. From that alignment method the similarity between the minutiae points is calculated using Hough transform. We incorporate texture-based features like entropy, correlation, contrast, homogeneity and energy for improving the accuracy.

Segmentation, normalisation, local orientation estimation, ridge frequency estimation, region mask estimation and filtering are the procedure done during enhancement of latent fingerprint. Segmentation leads to the calculation of variance for each pixel. Normalisation is the process, that changes the range of pixel intensity value. After that local orientation of each pixel in a fingerprint image is computed. Ridge frequency is obtained from the extraction of the ridge map. Classification of pixels into retrievable or unretrievable is called masking. Filtering helps for removing noises.

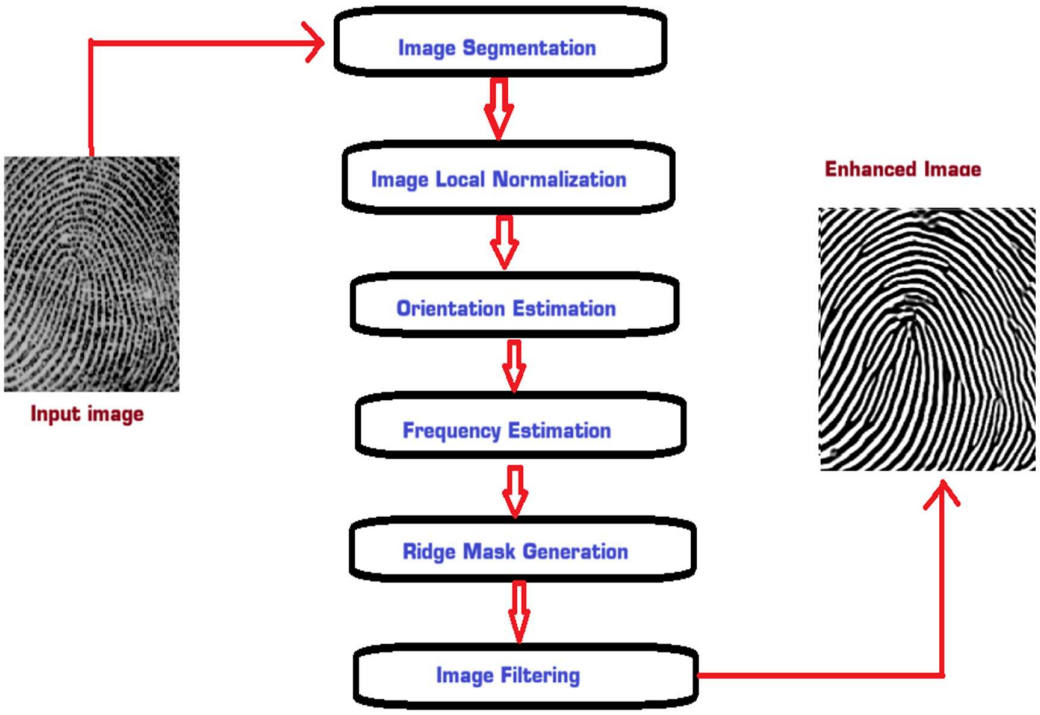


Fig-3.3.d Enhancement steps

3.4 Advantages: Non-invasive blood group detection using deep learning models offers several advantages that could significantly impact healthcare, making it more efficient, accessible, and user-friendly. Here are some of the key benefits:

1. **Minimized Invasiveness**: Traditional blood group determination involves a blood sample, which can be uncomfortable, time-consuming, and potentially risky. Non-invasive detection eliminates the need for needles or blood draws, providing a more comfortable and less traumatic experience for patients.
2. **Quick and Convenient**: Non-invasive methods can offer real-time or near-real-time results without the need for laboratory procedures. This can be particularly advantageous in emergency situations, where quick blood type identification is essential for transfusions or other medical treatments.
3. **Cost-Effective**: Over time, non-invasive methods could lower the cost of blood group testing by reducing the need for expensive lab equipment and procedures, as well as the labor costs involved in collecting and processing blood samples.
4. **Remote Accessibility**: Non-invasive detection could be performed using portable devices that can be easily deployed in remote or underserved areas where access to healthcare facilities and traditional blood testing is limited. This can increase access to blood type testing in regions with fewer medical resources.

5. **Enhanced Safety**: Since no blood is drawn, there's no risk of infections from needles or contamination, making it a safer alternative for blood group testing. It could be especially beneficial for individuals with compromised immune systems or those who are wary of needles.
6. **Patient Comfort and Compliance**: People who are afraid of needles (trypanophobia) or those with conditions requiring frequent blood tests (e.g., diabetics) might find non-invasive blood group detection more acceptable, leading to higher compliance with testing protocols.
7. **Automation and Integration**: Deep learning models can automate the detection process, reducing human error and increasing the reliability of results. The integration of AI-powered tools with medical devices could enable seamless data collection and analysis, streamlining workflows in healthcare settings.
8. **Non-invasive Monitoring for Various Conditions**: Once developed, such deep learning-based systems could potentially be adapted for continuous monitoring or tracking of blood groups in people with conditions that might require regular blood transfusions, such as thalassemia or sickle cell anemia.
9. **Improved Efficiency**: Since the deep learning models can be trained on large datasets, they could potentially improve over time, becoming more accurate and adaptable to different environmental factors, demographics, and individual differences.
10. **Non-Disruptive for Routine Check-ups**: Unlike blood tests that may require patients to visit a healthcare facility, non-invasive blood group detection could be incorporated into routine check-ups or screenings, making it more convenient for patients to get tested without disrupting their daily lives.
11. **Potential for Integration with Other Health Monitoring**: With the rise of wearable health technology, non-invasive blood group detection could be integrated into devices such as smartwatches or fitness trackers, enabling users to monitor their blood type alongside other health metrics like heart rate, blood oxygen levels, or glucose levels.
12. **Faster Results in Critical Care**: In critical care scenarios (e.g., surgeries, accidents, or emergency blood transfusions), knowing a patient's blood group immediately is vital. A non-invasive, rapid detection method can save valuable time in urgent situations, potentially saving lives.

4.SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 SOFTWARE TOOLS: To develop a road lane detection system using OpenCV, you will primarily need the following software tools and libraries:

1. Python- Python is a popular programming language widely used in computer vision and machine learning applications. It provides a simple and powerful syntax, making it suitable for rapid prototyping and development.
2. OpenCV (Open Source Computer Vision Library) -OpenCV is a comprehensive open-source library for computer vision and image processing tasks. It offers a wide range of functions and algorithms for tasks such as image loading, manipulation, feature extraction, object detection, and more.
3. NumPy- NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, mathematical functions, and linear algebra operations, which are commonly used in image processing and machine learning tasks.
4.Matplotlib or OpenCV's GUI functions Matplotlib is a plotting library for Python that can be used to visualize images, plots, and graphs. Alternatively, OpenCV provides its own set of GUI functions for displaying images and interacting with users, such as `cv2.imshow()` and `cv2.waitKey()`.
5.Development Environment You can use any integrated development environment (IDE) or text editor of your choice for writing and running Python code. Popular options include PyCharm, Visual Studio Code, Atom, Sublime Text, and Jupyter Notebook.

4.1.1 PYTHON TECHNOLOGY

Python is a versatile and widely-used programming language known for its simplicity, readability, and flexibility. It has become one of the most popular languages in various fields, including web development, data science, machine learning, artificial intelligence, and computer vision. Python's ease of use and extensive ecosystem of libraries make it well-suited for developing road lane detection systems using OpenCV. Here's an overview of Python technology and its features in the context of road lane detection:

1.Simplicity and Readability -Python's simple and readable syntax makes it easy to write and understand code, even for beginners. This simplicity accelerates the development process and reduces the time needed to implement road lane detection algorithms.

2 . Interpreted and Dynamic -Typing Python is an interpreted language, meaning that code is executed line-by-line by an interpreter rather than compiled into machine code. It also supports dynamic typing, allowing variables to be dynamically typed at runtime, which contributes to its flexibility.

3. Extensive Library Ecosystem -Python boasts a vast ecosystem of libraries and frameworks for various tasks, including image processing, computer vision, and machine learning. OpenCV is one such library that provides a wide range of functions and algorithms for image manipulation, feature extraction, object detection, and more. Leveraging OpenCV in Python simplifies the implementation of road lane detection algorithms, as it offers high-level functions and intuitive interfaces for common tasks.

4.Platform Independence -Python is platform-independent, meaning that code written in Python can run on different operating systems without modification. This platform independence facilitates the deployment of road lane detection systems across various environments, including Windows, Linux, and

macOS.

5. Integration with C/C++ and Other Languages Python can be easily integrated with other programming languages, such as C/C++, through language bindings and APIs. OpenCV, for example, is implemented in C++ 35 but provides Python bindings, allowing developers to access its functionalities directly from Python. This seamless integration enables the use of efficient C/C++ code alongside Python for performance-critical tasks in road lane detection.

6. Interactive Development Environment Python offers interactive development environments (IDEs) and tools that streamline the development process. IDEs like PyCharm, Visual Studio Code, and Jupyter Notebook provide features such as code autocompletion, debugging, and visualization tools, enhancing productivity and facilitating experimentation with road lane detection algorithms

7. Community Support and Documentation Python has a large and active community of developers, researchers, and enthusiasts who contribute to its development and maintenance. The availability of extensive documentation, tutorials, forums, and online resources makes it easy to learn Python and address challenges encountered during road lane detection system development. Overall, Python's simplicity, extensive library ecosystem, platform independence, integration capabilities, interactive development environments, and community support make it an ideal choice for developing road lane detection systems using

OpenCV

Its combination of ease of use and powerful features empowers developers to create robust and efficient solutions for detecting lanes on roads. OPEN CV OpenCV (Open Source Computer Vision Library) is a powerful open-source library for computer vision and image processing tasks. It provides a wide range of functions and algorithms that are essential for developing road lane detection systems and other computer vision applications. Here's an overview of OpenCV and its features in the context of road lane detection:

1.Image Loading and Manipulation

OpenCV allows you to load images from various sources, including files, cameras, and video streams. Once loaded, you can manipulate images using a variety of functions, such as resizing, cropping, rotating, and converting between different color spaces (e.g., RGB, grayscale, HSV).

2.Feature Extraction and Edge Detection

OpenCV provides functions for extracting features from images, such as corners, keypoints, and descriptors. These features can be used for tasks like object recognition, tracking, and lane detection. Additionally, OpenCV includes algorithms for edge detection, including the popular Canny edge detector, which is commonly used as a preprocessing step in lane detection systems.

3.Line and Shape Detection

OpenCV offers functions for detecting lines, circles, and other shapes in images. The Hough Line Transform, for example, can be used to detect straight lines in an image, making it suitable for identifying lane markings in road lane detection systems.

4.Image Filtering and Noise Reduction

OpenCV provides a variety of filters for image smoothing and noise reduction, such as Gaussian blur, median blur, and bilateral filter. These filters can help improve the quality of images and enhance the performance of lane detection algorithms by reducing noise and artifacts.

5.Machine Learning and Deep Learning Integration

OpenCV integrates with popular machine learning and deep learning frameworks, such as TensorFlow,

PyTorch, and scikit-learn. This integration allows you to leverage pre trained models and train custom models for tasks like object detection, lane detection, and semantic segmentation.

6.Performance Optimization

OpenCV is highly optimized for performance, with many of its core functions implemented in C/C++ for efficiency. Additionally, OpenCV provides support for parallel processing and hardware acceleration using technologies like OpenCL and CUDA, enabling real-time processing of video streams and large-scale image analysis.

7.Cross-Platform Compatibility

OpenCV is cross-platform and supports various operating systems, including Windows, Linux, macOS, Android, and iOS. This allows you to develop road lane detection systems that can run on a wide range of devices and platforms.

8.Documentation and Community Support

OpenCV has extensive documentation, tutorials, and examples to help you get started with developing computer vision applications. Additionally, it has a large and active community of developers, researchers, and enthusiasts who contribute to its development and provide support through forums, mailing lists, and online communities. Overall, OpenCV provides a comprehensive set of tools and functionalities for developing road lane detection systems and other computer vision applications. Its ease of use, performance, and extensive capabilities make it a popular choice among developers for tackling a wide range of image processing and computer vision tasks.

NUMPY:

NumPy is a fundamental package for numerical computing in Python. It provides support for multidimensional arrays, mathematical functions, linear algebra operations, and more. In the context of road lane detection using OpenCV, NumPy plays a crucial role in handling and processing image data efficiently. Here's how NumPy can be utilized in road lane detection applications:

1.Image Representation

NumPy arrays are commonly used to represent images in Python. Images loaded using OpenCV are typically represented as NumPy arrays, allowing for seamless integration between OpenCV and NumPy. NumPy arrays enable efficient manipulation of image data, such as accessing individual pixels, performing element-wise operations, and applying filters and transformations.

2.Data Manipulation

NumPy provides a wide range of functions for manipulating and processing multidimensional arrays. These functions can be used to preprocess images before applying lane detection algorithms. For example, NumPy functions can be used to resize, crop, rotate, and transpose images, as well as convert between different color spaces (e.g., RGB to grayscale).

3.Array Operations

NumPy supports a variety of array operations, including arithmetic operations, element-wise operations, and broadcasting. These operations are useful for performing mathematical computations on image data, such as applying filters, calculating gradients, and performing morphological operations. For example, NumPy functions can be used to apply Gaussian blur, compute derivatives, and perform thresholding operations on images.

4.Indexing and Slicing

NumPy provides powerful indexing and slicing capabilities for accessing and modifying elements of

arrays. This allows for efficient extraction of regions of interest (ROIs) from images and manipulation of pixel values based on specific criteria. For example, NumPy indexing and slicing can be used to extract the region containing lane markings from an image and modify the pixel values to enhance the visibility of lanes.

5.Integration with OpenCV

NumPy arrays seamlessly integrate with OpenCV, allowing for efficient data exchange between the two libraries. Images loaded using OpenCV can be converted to NumPy arrays for processing, and the results can be converted back to OpenCV format for display or further processing. This interoperability simplifies the development of road lane detection systems by leveraging the strengths of both OpenCV and NumPy. Overall, NumPy is an essential tool for road lane detection using OpenCV, providing powerful array manipulation capabilities that are indispensable for processing and analyzing image data efficiently. By combining the functionalities of NumPy and OpenCV, developers can implement robust and efficient road lane detection algorithms capable of handling a wide range of image processing tasks.

4.1.2 IMAGE PROCESSING

Image processing plays a crucial role in road lane detection using OpenCV. It involves various techniques and algorithms to preprocess the input images, extract lane markings, and identify the lanes on the road. Here's an overview of common image processing steps used in road lane detection:

1.Image Acquisition:

The first step is to acquire the input images or video frames from a camera or video file. OpenCV provides functions for reading images and video streams from different sources, such as files, cameras, and network streams.

2.Color Space Conversion:

Convert the input image from the BGR (Blue-Green-Red) color space to a suitable color space for lane detection. Converting to grayscale (single-channel) or HLS (Hue Lightness-Saturation) color space is common, as it simplifies subsequent processing steps and enhances lane detection accuracy.

• **Grayscale Conversion**:

Convert the input image to grayscale, which represents the intensity (luminance) values of the pixels without considering color information. Grayscale images are simpler to process and analyze compared to color images, making them suitable for lane detection algorithms that rely primarily on intensity differences between lane markings and road surfaces.

• **Color Space Adjustment:**

Adjust the color channels or components of the input image to enhance the visibility of lane markings. This may involve increasing the contrast, adjusting the brightness, or modifying specific color channels to improve the differentiation between lanes and background. For example, in the HLS color space, adjusting the saturation channel can help enhance the color contrast of lane markings.

• **Histogram Equalization**:

Apply histogram equalization to adjust the intensity distribution of the input image, making dark regions brighter and bright regions darker. Histogram equalization can help enhance the contrast and visibility of lane markings, especially in images with uneven lighting conditions or low contrast.

• **Adaptive Thresholding:**

Apply adaptive thresholding techniques to segment the input image into binary regions based on local

intensity variations. Adaptive thresholding can help distinguish between lane markings and road surfaces by dynamically adjusting the threshold values in different image regions.

4.1.3 Algorithm Used:

CNNs have shown remarkable capabilities in image analysis and pattern identification, making them a promising technology for fingerprint-based blood group detection. By training on extensive datasets of labeled fingerprint images, CNNs can learn to identify complex patterns and relationships associated with blood groups. However, the development and implementation of such systems face challenges, including the demand for significant computational power, complex network designs, and large, accurately labeled datasets.

Despite their potential, using CNNs for fingerprint-based blood group detection remains relatively unexplored. Research in this area is limited, and practical applications are scarce. Current systems lack the necessary reliability and accuracy for widespread use in medical diagnostics. There is a notable absence of research on optimizing CNN architectures specifically for this purpose, as well as on establishing comprehensive, annotated fingerprint datasets. My research addresses this gap by achieving higher accuracy than previously reported systems.

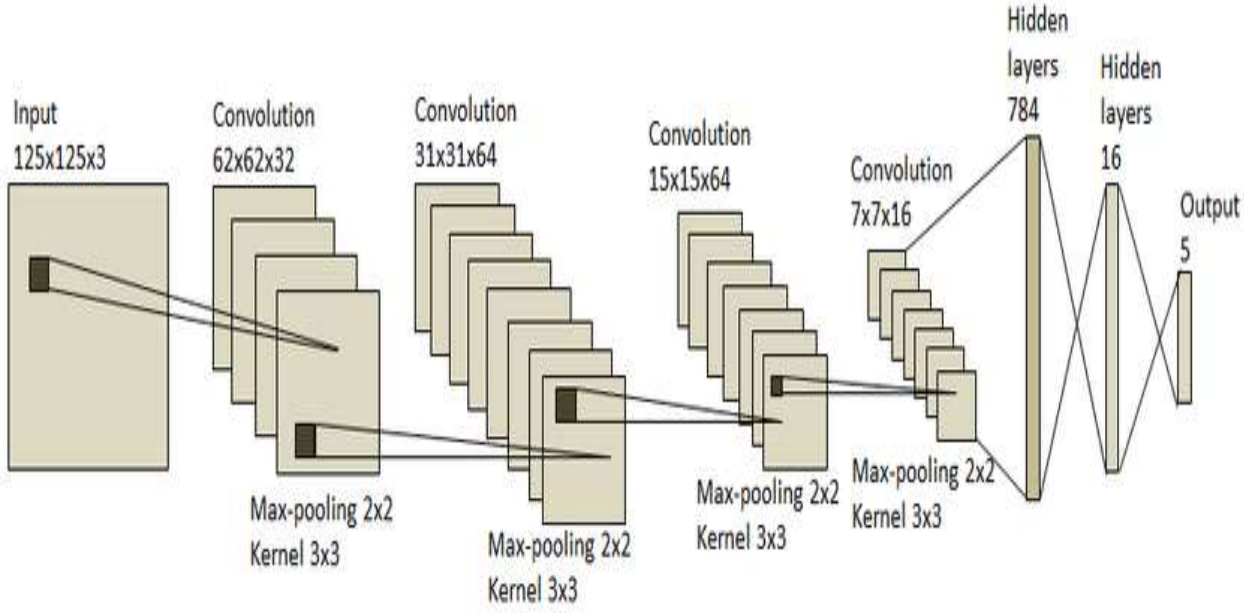


Fig- CNN Model Basic Architecture

5. INTRODUCTION TO UML

5. INTRODUCTION TO UML

Unified Modelling Language (UML) is a standardized modelling language used in software engineering for visualizing, specifying, constructing, and documenting the artifacts of a system. It provides a set of graphical notations for creating visual models of software systems, allowing developers to communicate effectively and understand the structure, behaviour, and interactions within a system.
UML was developed in the 1990s by Grady Booch, James Rumbaugh, and Ivar Jacobson, who merged their individual modelling languages (Booch, OMT, and OOSE, respectively) to create a unified standard. Over time, UML has evolved into a comprehensive language with a wide range of diagrams and concepts.
The primary goals of UML include:
1. Standardization: UML provides a standard notation for modelling software systems, ensuring consistency and interoperability among different tools and methodologies.
2. Abstraction: UML allows developers to abstract away complex details of a system and focus on its essential aspects, making it easier to understand and communicate system design.
3. Visualization: UML diagrams provide visual representations of various aspects of a system, such as its structure, behaviour, interactions, and architecture, making it easier to communicate complex ideas and concepts.
4. Specification: UML can be used to specify the requirements, design, and architecture of a software system, helping stakeholders to understand the system's functionality and characteristics.
5.Documentation: UML diagrams serve as documentation for software systems, capturing important design decisions, relationships, and constraints, which can be referred to throughout the software development lifecycle.
Some of the key concepts and diagrams in UML include:
• Class Diagram: Describes the static structure of a system, including classes, attributes, operations, and relationships.
• Use Case Diagram: Represents the functional requirements of a system from the perspective of users, actors, and use cases. • Sequence Diagram: Illustrates the interactions between objects or components over time, showing the sequence of messages exchanged.
• Activity Diagram: Models the flow of control within a system, depicting the behaviour and activities performed by objects or components. • State Machine Diagram: Describes the behaviour of an object or system in response to events, transitions, and states.
• Component Diagram: Represents the physical structure of a system, including components, interfaces, and dependencies.
• Deployment Diagram: Depicts the deployment of software components on hardware nodes, illustrating the physical architecture of a system. Overall, UML is a versatile and powerful modelling language that plays a vital role in software development, helping teams to visualize, specify, and design software systems effectively.

5.1 COMPONENTS OF UML:

Unified Modelling Language (UML) consists of several components, including: 1.Diagrams UML provides a variety of diagrams to represent different aspects of a system's structure and behaviour. Some of the most commonly used UML diagrams include:
• Class Diagram: Describes the static structure of a system, including classes, attributes, operations, and relationships between classes.
• Use Case Diagram: Represents the interactions between users (actors) and the system, depicting the functional requirements from the users' perspective.

• Sequence Diagram: Illustrates the interactions between objects or components over time, showing the sequence of messages exchanged.
• Activity Diagram: Models the flow of control within a system, depicting the behaviour and activities performed by objects or components.
• State Machine Diagram: Describes the behaviour of an object or system in response to events, transitions, and states.
• Component Diagram: Represents the physical components of a system, including software modules, libraries, and dependencies.
• Deployment Diagram: Depicts the deployment of software components on hardware nodes, illustrating the physical architecture of a system. .Elements UML defines various elements that can be used to construct diagrams. These elements represent different concepts and artifacts within a system, such as classes, objects, actors, use cases, interfaces, components, and nodes.

5.2 Activity Diagram:

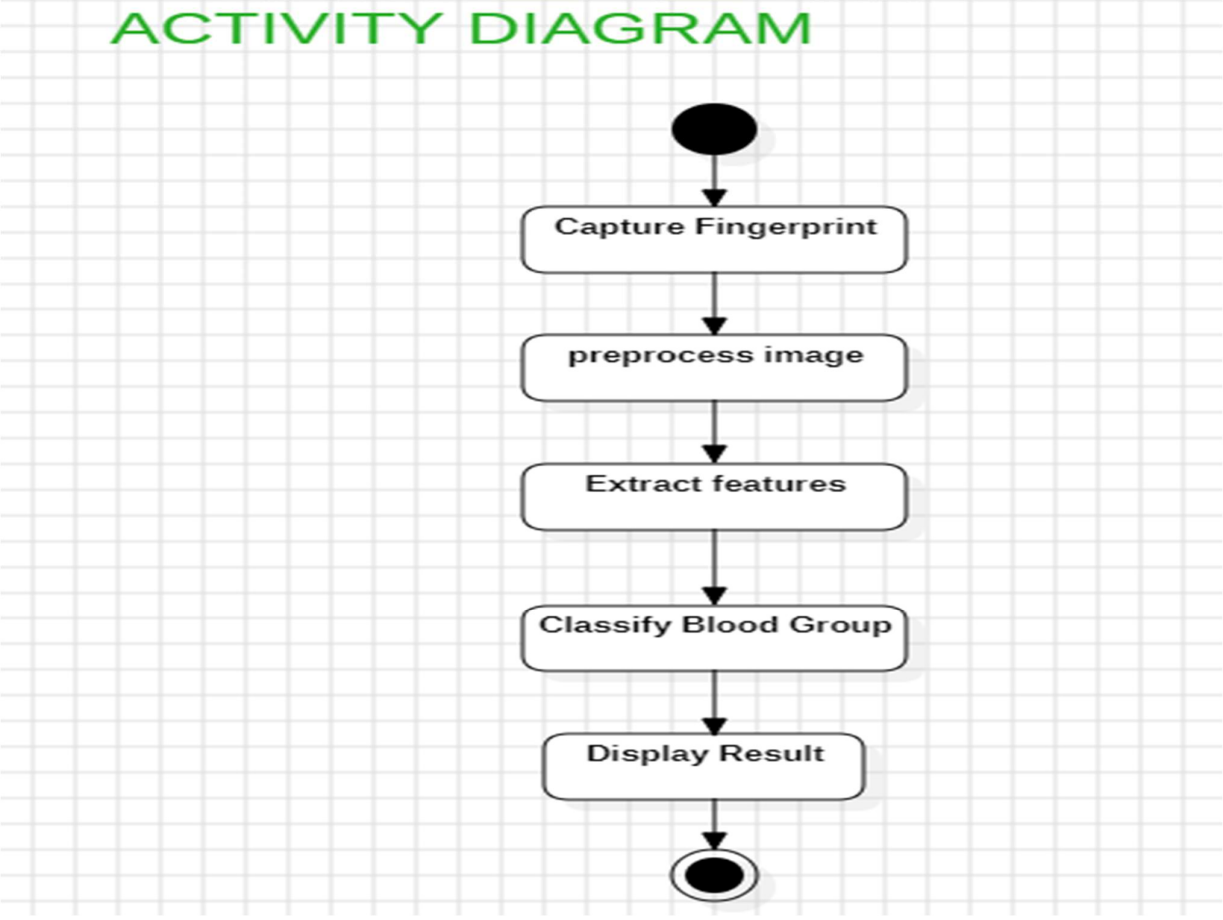


Fig-5.2.a Activity Diagram

5.3 Sequence Diagram

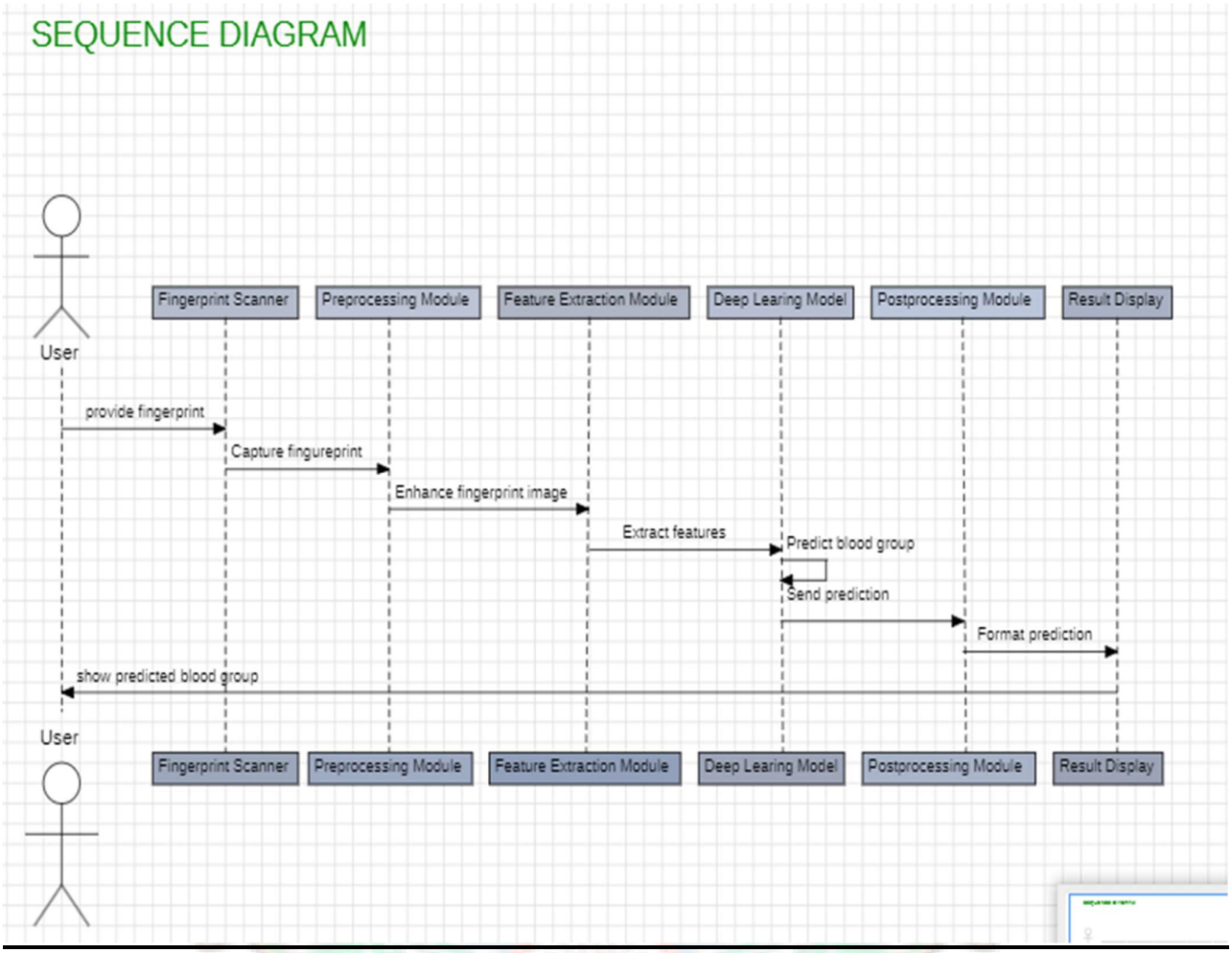


Fig-5.3.a Sequence Diagram

6.IMPLEMENTATION

6.1 MODULES USED:

In blood group detection using OpenCV, several modules and libraries are commonly utilized to perform various tasks such as image processing, edge detection, visualization, and more. Here are some of the key modules commonly used in road lane detection with OpenCV:

1.cv2

This is the main module of OpenCV in Python. It provides functions and classes for image processing, computer vision algorithms, and I/O operations. Some commonly used functions from this module include `cv2.imread()`, `cv2.cvtColor()`, `cv2.Canny()`, `cv2.HoughLines()`, `cv2.destroyAllWindows()`. `cv2.imshow()`, `cv2.waitKey()`, and

• Image Processing:

OpenCV's cv2 module provides a wide range of functions and classes for image processing tasks such as loading, saving, and manipulating images. This includes operations like resizing, cropping, rotating, and converting images between different color spaces.

• Feature Extraction:

It offers functions for extracting features from images, such as corners, keypoints, and descriptors, which can be useful for identifying lane markings or distinguishing between different road features

. • Edge Detection:

OpenCV provides edge detection algorithms like the Canny edge detector (cv2.Canny()) for detecting edges in images. Edges correspond to changes in intensity, which can help identify lane boundaries.

2.numpy (np)

NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, mathematical functions, and linear algebra operations. NumPy arrays are commonly used to represent images and perform various image processing operations efficiently.

• Array Operations:

NumPy arrays are the primary data structure used for representing images and performing numerical computations in Python. NumPy provides functions for array manipulation, mathematical operations, and linear algebra, which are essential for various image processing tasks.

• Efficient Processing:

NumPy's vectorized operations allow for efficient processing of large image datasets, making it well-suited for tasks like filtering, transformation, and feature extraction in road lane detection.

• Matrix Operations:

NumPy's linear algebra capabilities are often leveraged for performing matrix operations such as matrix multiplication, inversion, and decomposition. These operations are useful for applying geometric transformations, filtering, and feature extraction in road lane detection algorithms.

3.matplotlib Matplotlib

is a plotting library for Python that is often used for data visualization and image display. It provides functions for creating plots, charts, and images, as well as interactive features for exploring data. Matplotlib is commonly used to display images, visualize intermediate results, and evaluate the performance of lane detection algorithms.

• Data Visualization:

Matplotlib is commonly used for visualizing images, intermediate results, and performance metrics in road lane detection. It provides functions for creating plots, histograms, and heatmaps, which can help developers analyze and evaluate the performance of lane detection algorithms.

• Interactive Visualization:

Matplotlib's interactive features allow users to explore images and results dynamically, enabling detailed inspection and analysis of lane detection outputs.

• **Plotting:**
Matplotlib provides functions for creating plots, histograms, and scatter plots, which can be used to visualize quantitative data and performance metrics related to road lane detection. Plots can help developers analyze the behavior of lane detection algorithms and identify areas for improvement.
• **Customization:**
Matplotlib offers extensive customization options for adjusting plot styles, colors, labels, and annotations. This flexibility allows developers to create visually appealing and informative visualizations for presenting lane detection results and findings.

4.math

The math module provides mathematical functions and constants in Python. It is often used for geometric calculations, angle conversions, and other mathematical operations in road lane detection algorithms.
• **Geometric Calculations:** The math module provides mathematical functions and constants for performing geometric calculations in road lane detection algorithms. This includes angle conversions, distance calculations, and other 52 geometric operations required for processing image data and analyzing lane markings.
• **Trigonometric Functions:** The math module provides trigonometric functions such as math.cos() and math.sin() for performing calculations involving angles and geometric transformations. These functions are often used in road lane detection algorithms for computing slopes, angles, and distances between points.

5. scikit-image (skimage)

scikit-image is a collection of algorithms for image processing in Python. It provides functions for various image processing tasks such as filtering, segmentation, feature extraction, and morphology. Some functions from this module may be used for advanced image processing and preprocessing steps in road lane detection.
• **Advanced Image Processing:** scikit-image offers a collection of algorithms for advanced image processing tasks such as filtering, segmentation, and feature extraction. While OpenCV provides basic image processing functions, scikit image extends the capabilities with more specialized and advanced algorithms that can enhance the performance of road lane detection systems

6. User

• **Converted into images**: Here the system can covert the videos into images which is uploaded by the user.
• **Pre-processing the images:** Pre-process the data converted into images.
• **Calculate the pixels into meters:** Here the system can convert the pixels into meters by calculating the pixels.
• **Calculate the measurement of curves:** Here the system can measure the curves on roads.
• **Detecting the road line:** With the help of curve detection, system can detect the road line of uploaded road videos.
. • Upload road video: Here the user can upload the data is road videos
• **View results:**Here the user can view the output data of road lane line after detection. These are some of the key modules used in road lane detection using OpenCV. Depending on the specific requirements of the lane detection algorithm and the complexity of the application, additional modules and libraries may be utilized for tasks such as machine learning, deep learning, data visualization, and performance evaluation.

6.2 SOURCE CODE:

app.py

```
from flask import Flask, request, jsonify, render_template, flash, redirect, url_for,
session
from flask_mysqldb import MySQL
from wtforms.validators import ValidationError
import numpy as np
import tensorflow as tf
import cv2
import os
import ssl
from werkzeug.utils import secure_filename
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Disable SSL verification
ssl._create_default_https_context = ssl._create_unverified_context

app = Flask(_name_)

# MySQL Configuration
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'blooddatabase'
app.secret_key = 'your_secret_key_here'

mysql = MySQL(app)

# Load the pre-trained model
model = tf.keras.models.load_model('./notebook/my_model.keras')

# Define allowed extensions
ALLOWED_EXTENSIONS = {'bmp'}

# Check allowed file extensions
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

def preprocess_image(file_path):
```

```
    img = load_img(file_path, target_size=(64, 64))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

@app.route('/about.html')
def homee():
    if 'user_id' in session:
        return render_template('about.html')#index
    return redirect(url_for('login'))

@app.route('/home.html', methods=['GET', 'POST'])
def home():
    if 'user_id' in session:
        return render_template('home.html')#home
    return redirect(url_for('login'))

@app.route('/register.html', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']

        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM users WHERE email = %s", (email,))
        user = cursor.fetchone()
        if user:
            cursor.close()
            flash('Email Already Taken', 'danger')
            return redirect(url_for('register'))

        cursor.execute('INSERT INTO users (name, email, password) VALUES (%s, %s,
%s)', (name, email, password))
        mysql.connection.commit()
        cursor.close()

        flash('Registration Successful!', 'success')
        return redirect(url_for('login'))

    return render_template('register.html')
```

```
@app.route('/login.html', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM users WHERE email = %s AND password =
%s', (email, password))
        user = cursor.fetchone()
        cursor.close()

        if user:
            session['user_id'] = user[0]
            return redirect(url_for('homee'))
        else:
            flash("Login failed. Please check your email and password","danger")
            return redirect(url_for('login'))

    return render_template('login.html')

@app.route('/chart.html', methods=['GET'])
def chart():
    if 'user_id' in session:
        return render_template('chart.html')
    return redirect(url_for('login'))

@app.route('/', methods=['GET'])
def about():
    if 'user_id' in session:
        return render_template('index.html')#about
    return redirect(url_for('login'))

@app.route('/api/data', methods=['POST'])
def get_data():
    data = request.json
    return jsonify({"status": "success", "data": data})

class]

        if confidence < CONFIDENCE_THRESHOLD:
            return jsonify({
```

```
                'error': 'Prediction confidence is too low. Please try with a clearer fingerprint image.',
                'confidence': confidence
            }), 400

        return jsonify({
            'predicted_class': predicted_class,
            'predicted_label': predicted_label,
            'confidence': confidence
        })
    except Exception as e:
        return jsonify({'error': str(e)}), 500
    finally:
        if os.path.exists(file_path):
            os.remove(file_path)
@app.route('/portfolio_details.html', methods=['GET', 'POST'])
def blood_group():
    blood_group = request.args.get('portfolio-content')
    group_data = details.get(blood_group)
    if group_data:
        return render_template('portfolio_details.html', group=group_data)
    else:
        return render_template('404.html', message="Blood group details not available.")
if _name_ == '_main_':
    if not os.path.exists('uploads'):
        os.makedirs('uploads')
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blood Group Detection</title>
    <style>
        /* General Styles */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f9f9f9;
            color: #333;
        }

        h1, h2 {
            text-align: center;
            color: #444;
        }

        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }

        /* Form Section */
        .form-section {
            margin-bottom: 30px;
        }

        .form-group {
            margin-bottom: 15px;
        }
```

```
label {
    display: block;
    font-weight: bold;
    margin-bottom: 5px;
}

input[type="text"], input[type="tel"], select, input[type="file"], button {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="file"] {
    padding: 5px;
}

button {
    background-color: #2f4faf;
    color: #fff;
    border: none;
    cursor: pointer;
    font-size: 16px;
    font-weight: bold;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #3126d1;
}

.preview-container {
    text-align: center;
    margin-top: 10px;
}
;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```
        .result-section h2 {
            margin-bottom: 20px;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 10px;
        }

        table th, table td {
            padding: 10px;
            text-align: left;
            border: 1px solid #ddd;
        }

        table th {
            background-color: #3831b9;
            color: white;
        }

        table td img {
            max-width: 100px;
            max-height: 100px;
            border-radius: 4px;
        }

        .alert {
            color: red;
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Blood Group Detection From Fingerprint</h1>
">Mobile</label>
                <input type="tel" id="mobile" placeholder="Enter your mobile number"
pattern="[0-9]{10}" required>
            </div>
            <div class="form-group">
                <label for="gender">Gender</label>
                <select id="gender" required>
```

```
                <option value="" disabled selected>Select your gender</option>
                <option value="Male">Male</option>
                <option value="Female">Female</option>
                <option value="Other">Other</option>
            </select>
        </div>
        <div class="form-group">
            <label for="age">Age</label>
            <select id="age" required>
                <option value="" disabled selected>Select your age</option>
                <!-- Age range -->
                <script>
                    for (let i = 18; i <=100; i++) {
                        document.write(<option value="${i}">${i}</option>);
                    }
                </script>
            </select>
        </div>
        <div class="form-group">
            <label for="fingerprint">Upload Fingerprint</label>
            <input type="file" id="fingerprint" accept="image/*"
onchange="previewFingerprint()" required>
        </div>
        <div class="preview-container">
            <img id="fingerprintPreview" alt="Fingerprint Preview" />
        </div>
        <button type="button" onclick="submitDetails()">Detect Blood
Group</button>
    </form>
</div>

<!-- Result Section -->
<div class="result-section" id="resultSection" style="display: none;">
    <h2>Detection Result</h2>
    <table>
        <thead>
            <tr>
                <th>Field</th>
                <th>Value</th>
            </tr>
        </thead>
        <tbody id="resultTableBody"></tbody>
```

```
        </table>
    </div>
<!--Flash Messages Section  -->
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        <div class="flash-container">
            {% for category, message in messages %}
                <div class="flash-message flash-{{ category }}">
                    {{ message }}
                </div>
            {% endfor %}
        </div>
    {% endif %}
{% endwith %}

</div>

<script>
    function previewFingerprint() {
        const fileInput = document.getElementById('fingerprint');
        const previewImage = document.getElementById('fingerprintPreview');
        const file = fileInput.files[0];

        if (file) {
            const reader = new FileReader();
            reader.onload = function (e) {
                previewImage.src = e.target.result;
            };
            reader.readAsDataURL(file);
        }
    }

    function submitDetails() {
        const name = document.getElementById('name').value;
        const mobile = document.getElementById('mobile').value;
        const gender = document.getElementById('gender').value;
        const age = document.getElementById('age').value;
        const fingerprintInput = document.getElementById('fingerprint');
        const fingerprintFile = fingerprintInput.files[0];

        if (!name || !mobile || !gender || !age || !fingerprintFile) {
            alert("Please fill in all fields.");
```

```
            return;
        }

        const formData = new FormData();
        formData.append('file', fingerprintFile);

        fetch('/predict', {
            method: 'POST',
            body: formData
        })
        .then(response => response.json())
        .then(data => {
            const resultSection = document.getElementById('resultSection');
            const resultTableBody = document.getElementById('resultTableBody');

            resultTableBody.innerHTML = `
                <tr><td>Name</td><td>${name}</td></tr>
                <tr><td>Mobile</td><td>${mobile}</td></tr>
                <tr><td>Gender</td><td>${gender}</td></tr>
                <tr><td>Age</td><td>${age}</td></tr>
                <tr><td>Fingerprint</td><td><img
src="${document.getElementById('fingerprintPreview').src}" style="max-width:
100px; max-height: 100px;" alt="Uploaded Fingerprint" /></td></tr>
                <tr><td>Confidence</td><td>${data.confidence}</td></tr>
                <tr><td>Blood Group</td><td>${data.predicted_label}</td></tr>
            `;

            resultSection.style.display = 'block';
        })
        .catch(error => {
            console.error('Error:', error);
            alert('An error occurred. Please try again.');
        });
    }
  </script>
</body>
</html>
```

Home.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Home</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- Favicons -->
  <link href="../static/images/favicon.png" rel="icon">
  <link href="../static/images/apple-touch-icon.png" rel="apple-touch-icon">

  <!-- Fonts -->
  <link href="https://fonts.googleapis.com" rel="preconnect">
  <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0
,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&family=Inter:wght@100;200;3
00;400;500;600;700;800;900&family=Nunito:ital,wght@0,200;0,300;0,400;0,500;0,60
0;0,700;0,800;0,900;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap"
rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="../static/vendor/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/aos/aos.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">

  <!-- Main CSS File -->
  <link href="../static/css/main.css" rel="stylesheet">

  <!-- ======================================================
  * Template Name: QuickStart
  * Template URL: https://bootstrapmade.com/quickstart-bootstrap-startup-website-
template/
  * Updated: Jun 29 2024 with Bootstrap v5.3.3
  * Author: BootstrapMade.com
```

```
 * License: https://bootstrapmade.com/license/
 ========================================================== -->
 <style>
            .logout {
        position: absolute;
        top: 20px;
        right: 20px;
    }
    .logout a {
;
        border-radius: 10px;
        border: 1px solid white;
    }
    button {
        width: 100%;
        padding: 10px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }
    button:hover {
        background-color: #45a049;
    }



.logout-link i {
  margin-right: 5px; /* Space between icon and text */
}


.mobile-nav {
  transition: opacity 2.0s ease-in-out; /* Adjust the duration (0.5s) and easing as needed
*/
  opacity: 1; /* Initially visible */
}

.mobile-nav.hidden {
  opacity: 0; /* Hidden state */
}
```

```
  </style>
</head>


<body class="index-page">

  <header id="header" class="header d-flex align-items-center fixed-top">
    <div class="container-fluid container-xl position-relative d-flex align-items-center">

      <a href="/home" class="logo d-flex align-items-center me-auto">
        <img src="../static/images/logo.png" alt="">
        <h1 class="sitename">Bloodgroup Prediction</h1>
      </a>

      <nav id="navmenu" class="navmenu">
       <!--     <ul>
          <li><a href="home.html" class="active">Home</a></li>
          <li><a href="about.html">About</a></li>
          <li><a href="/team">Team</a></li>
          {% if 'user_id' in session %}
            <li><a href="/predict_blood_group">Predict</a></li>
            <li><a href="/Accurancy">Accurancy</a></li>
           <li><a href="/profile"><i class="bi bi-person-circle"></i> Profile</a></li>
             <li><a href="/logout" class="logout-link">
    <i class="bi bi-box-arrow-right"></i> Logout
</a></li>

        {% else %}
          <li><a href="login.html">Login</a></li>
          <li><a href="register.html">Sign Up</a></li>
        {% endif %}

              </ul>  -->
              <ul class="nav-links">
                <li><a href="home.html">Home</a></li>
                <li><a href="about.html">About</a></li>
                <li><a href="chart.html">Chart</a></li>
                <li><a href="login.html">Login</a></li>
                <li><a href="register.html">Register</a></li>
             </ul>
        <i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
```

```
    </nav>

<!--     <a class="btn-getstarted" href="about">Get Started</a>-->

   </div>
  </header>


<main class="main">

   <!-- Hero Section -->
   <section id="hero" class="hero section">
    <div class="hero-bg">
     <img src="../static/images/hero-bg-light.jpg" alt="">
    </div>
    <div class="container text-center">
     <div class="d-flex flex-column justify-content-center align-items-center">
      <h1 data-aos="fade-up">Welcome to <span>Bloodgroup
Prediction</span></h1>
      <p data-aos="fade-up" data-aos-delay="100">New way of Predicting Blood
Group with Fingerprint<br></p>
<!--       <div class="d-flex" data-aos="fade-up" data-aos-delay="200">-->
<!--        <a href="login" class="btn-get-started">Get Started</a>-->
<!--        <a href="https://www.youtube.com/watch?v=LXb3EKWsInQ"
class="glightbox btn-watch-video d-flex align-items-center"><i class="bi bi-play-
circle"></i><span>Watch Video</span></a>-->
<!--       </div>-->
      <img src="../static/images/homepic.jpg" class="img-fluid hero-img" alt="" data-
aos="zoom-out" data-aos-delay="300">
     </div>
    </div>

   </section><!-- /Hero Section -->

   <!-- Featured Services Section -->
<section id="featured-services" class="featured-services section light-background">

  <div class="container">

   <div class="row gy-4">

    <div class="col-xl-4 col-lg-6" data-aos="fade-up" data-aos-delay="100">
```

```
        <div class="service-item d-flex">
          <!-- Updated AI Icon -->
          <div class="icon flex-shrink-0"><i class="bi bi-cpu"></i></div> <!-- AI-related
icon -->
          <div>
            <h4 class="title"><a href="#" class="stretched-link">AI Model
Development</a></h4>
            <p class="description">Our team develops advanced CNN models that analyze
patterns in medical data to predict blood groups accurately.</p>
          </div>
        </div>
      </div><!-- End Service Item -->

      <div class="col-xl-4 col-lg-6" data-aos="fade-up" data-aos-delay="200">
        <div class="service-item d-flex">
          <
        </div>
      </div><!-- End Service Item -->

      <div class="col-xl-4 col-lg-6" data-aos="fade-up" data-aos-delay="300">
        <div class="service-item d-flex">
          <div class="icon flex-shrink-0"><i class="bi bi-graph-up-arrow"></i></div>
          <div>
            <h4 class="title"><a href="#" class="stretched-link">Model Evaluation &
Optimization</a></h4>
            <p class="description">We evaluate our CNN models using various metrics and
optimize them for higher accuracy and faster predictions for blood group
classification.</p>
          </div>
        </div>
      </div><!-- End Service Item -->

    </div>
  </div>
</section><!-- /Featured Services Section -->
    </footer>

  <!-- Scroll Top -->
  <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-
center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Preloader --> <div id="preloader"></div><!-- </html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blood Group Detection</title>
    <style>
        /* General Styles */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f9f9f9;
            color: #333;
        }

        h1, h2 {
            text-align: center;
            color: #444;
        }

        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }

        /* Form Section */
        .form-section {
            margin-bottom: 30px;
        }

        .form-group {
            margin-bottom: 15px;
        }

        label {
```

```
    display: block;
    font-weight: bold;
    margin-bottom: 5px;
}

input[type="text"], input[type="tel"], select, input[type="file"], button {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="file"] {
    padding: 5px;
}

button {
    background-color: #2f4faf;
    color: #fff;
    border: none;
    cursor: pointer;
    font-size: 16px;
    font-weight: bold;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #3126d1;
}

.preview-container {
    text-align: center;
    margin-top: 10px;
}

img#fingerprintPreview {
    max-width: 150px;
    max-height: 150px;
    border: 1px solid #ccc;
```

```
    border-radius: 8px;
    margin-top: 10px;
}

/* Result Section */
.result-section {
    margin-top: 30px;
    padding: 20px;
    background-color: #f1f1f1;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.result-section h2 {
    margin-bottom: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 10px;
}

table th, table td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ddd;
}

table th {
    background-color: #3831b9;
    color: white;
}

table td img {
    max-width: 100px;
    max-height: 100px;
    border-radius: 4px;
}

.alert {
    color: red;
```

```
        text-align: center;
      }
    </style>
</head>
<body>
    <h1>Blood Group Detection From Fingerprint</h1>

    <div class="container">
      <!-- Form Section -->
      <div class="form-section">
        <h2>
>
                <option value="" disabled selected>Select your gender</option>
                <option value="Male">Male</option>
                <option value="Female">Female</option>
                <option value="Other">Other</option>
              </select>
            </div>
            <div class="form-group">
              <label for="age">Age</label>
              <select id="age" required>
                <option value="" disabled selected>Select your age</option>
                <!-- Age range -->
                <script>
                  for (let i = 18; i <=100; i++) {
                    document.write(<option value="${i}">${i}</option>);
                  }
                </script>
              </select>
            </div>
            <div class="form-group">
              <label for="fingerprint">Upload Fingerprint</label>
              <input type="file" id="fingerprint" accept="image/*"
onchange="previewFingerprint()" required>
            </div>
            <div class="preview-container">
              <img id="fingerprintPreview" alt="Fingerprint Preview" />
            </div>
            <button type="button" onclick="submitDetails()">Detect Blood
Group</button>
        </form>
      </div>
```

```
    <!-- Result Section -->
    <div class="result-section" id="resultSection" style="display: none;">
        <h2>Detection Result</h2>
        <table>
            <thead>
                <tr>
                    <th>Field</th>
                    <th>Value</th>
                </tr>
            </thead>
            <tbody id="resultTableBody"></tbody>
        </table>
    </div>
<!--Flash Messages Section  -->
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        <div class="flash-container">
            {% for category, message in messages %}
                <div class="flash-message flash-{{ category }}">
                    {{ message }}
                </div>
            {% endfor %}
        </div>
    {% endif %}
{% endwith %}

</div>

<script>
    function previewFingerprint() {

    function submitDetails() {
        const name = document.getElementById('name').value;
        const mobile = document.getElementById('mobile').value;
        const gender = document.getElementById('gender').value;
        const age = document.getElementById('age').value;
        const fingerprintInput = document.getElementById('fingerprint');
        const fingerprintFile = fingerprintInput.files[0];

        if (!name || !mobile || !gender || !age || !fingerprintFile) {
            alert("Please fill in all fields.");
```

```
            return;
        }

        const formData = new FormData();
        formData.append('file', fingerprintFile);

        fetch('/predict', {
            method: 'POST',
            body: formData
        })
        .then(response => response.json())
        .then(data => {
            const resultSection = document.getElementById('resultSection');
            const resultTableBody = document.getElementById('resultTableBody');

            resultTableBody.innerHTML = `
                <tr><td>Name</td><td>${name}</td></tr>
                <tr><td>Mobile</td><td>${mobile}</td></tr>
                <tr><td>Gender</td><td>${gender}</td></tr>
                <tr><td>Age</td><td>${age}</td></tr>
                <tr><td>Fingerprint</td><td><img
src="${document.getElementById('fingerprintPreview').src}" style="max-width:
100px; max-height: 100px;" alt="Uploaded Fingerprint" /></td></tr>
                <tr><td>Confidence</td><td>${data.confidence}</td></tr>
                <tr><td>Blood Group</td><td>${data.predicted_label}</td></tr>
            `;

            resultSection.style.display = 'block';
        })
        .catch(error => {
            console.error('Error:', error);
            alert('An error occurred. Please try again.');
        });
    }
  </script>
</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login form</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='login.css') }}">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
   <!--<h1>Login Form</h1>  - ->
    <form action="{{ url_for('login') }}" method="POST">
       <!-- <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Login</button>    - ->
        <h1>Login</h1>

        {% with messages = get_flashed_messages(with_categories=True) %}
        {% if messages %}
          <div>
            {% for category, message in messages %}
              <div class="alert alert-{{ category }} alert-dismissible fade show"
role="alert">
                {{ message }}

              </div>
            {% endfor %}
          </div>
        {% endif %}
      {% endwith %}

      <div class="input-box">
        <input type="email" name="email" placeholder="Email" required><br>
        <i class='bx bxs-user'></i>
      </div>
      <div class="input-box">
        <input type="password" name="password" placeholder="Password" required>
```

```
      <i class='bx bxs-lock-alt' ></i>
    </div>
    <div class="remember-forgot">
      <label><input type="checkbox">Remember Me</label>
      <a href="#">Forgot Password</a>
    </div>
    <button type="submit" class="btn">Login</button>
    <div class="register-link">
      <p>Dont have an account? <a href="{{ url_for('register') }}">Register page</a>
</p>
    </div>
  </form>

</body>
</html>

-->

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login form</title>
  <link href="../static/images/favicon.png" rel="icon">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/login.css') }}">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/boxicons@2.1.4/css/boxicons.min.css"
rel="stylesheet">
</head>
<body>
 <div style="background-image: url('{{ url_for('static', filename='images/login.jpg')
}}'); background-size: cover; background-position: auto; height: 99vh; width: 210vh;">
  <div style="text-align: center; padding-top: 100px; color: white;">
  <div class="form-container">
    <h1>Login</h1>
    <form action="{{ url_for('login') }}" method="POST">
      {% with messages = get_flashed_messages(with_categories=True) %}
      {% if messages %}
       <div>
        {% for category, message in messages %}
```

```
            <div class="alert alert-{{ category }} alert-dismissible fade show"
role="alert">
              {{ message }}
            </div>
          {% endfor %}
        </div>
      {% endif %}
      {% endwith %}

="password" placeholder="Password" required id="password">
          <i class='bx bxs-show' id="togglePassword" style="cursor: pointer;"></i>
      </div>
      <div class="remember-forgot">
        <label><input type="checkbox">Remember Me</label>
        <a href="#">Forgot Password</a>
      </div>
      <button type="submit" class="sparkle-button">Login</button>
      <div class="register-link">
        <p>Don't have an account? <a href="{{ url_for('register') }}">Register
page</a></p>
      </div>
    </form>
  </div>

  <script>
    const togglePassword = document.getElementById('togglePassword');
    const passwordInput = document.getElementById('password');

    togglePassword.addEventListener('click', function () {
      const type = passwordInput.getAttribute('type') === 'password' ? 'text' :
'password';
      passwordInput.setAttribute('type', type);
      this.classList.toggle('bx-show');
    });
  </script>
</body>
</html>
```

Register.html

```
<!--

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Register form</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='register.css') }}">
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
   <!-- Add Boxicons CDN - ->
   <link href="https://cdn.jsdelivr.net/npm/boxicons@2.1.4/css/boxicons.min.css"
rel="stylesheet">

</head>
<body>
  <!-- Add the image at the top - ->
  <div style="background-image: url('{{ url_for('static', filename='register.jpg') }}');
background-size: cover; background-position: auto; height: 99vh; width: 210vh;">
   <div style="text-align: center; padding-top: 100px; color: white;">

   <form action="{{ url_for('register') }}" method="POST">
      <h1>Register Form</h1>

         <!-- Flash Messages - ->
   {% with messages = get_flashed_messages(with_categories=True) %}
   {% if messages %}
    <div>
      {% for category, message in messages %}
       <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
         {{ message }}
       </div>
      {%
ainer" style="text-align: center; margin-top: 20px;">
   {% if user %}
      <a href="/logout" class="btn-btn-dark">Logout</a>
   {% endif %}
</div>
```

```
      <input type="text" name="name" placeholder="Name" required><i class='bx bxs-
user'> </i></input>    <br>
      <input type="email" name="email" placeholder="Email" required><br>
      <input type="password" name="password" placeholder="Password" required><i
class='bx bxs-lock-alt' ></i><br>   </div>
      <button type="submit">Register</button>
      <p><a href="{{ url_for('login') }}">Login page</a></p>
   </form>
  </div>
</body>
</html>



<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Register form</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='register.css') }}">
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
   <link href="https://cdn.jsdelivr.net/npm/boxicons@2.1.4/css/boxicons.min.css"
rel="stylesheet">

</head>
<body>
  <div style="background-image: url('{{ url_for('static', filename='register.jpg') }}');
background-size: cover; background-position: auto; height: 99vh; width: 210vh;">
   <div style="text-align: center; padding-top: 100px; color: white;">  <h1>Register
Form</h1>

      {% with messages = get_flashed_messages(with_categories=True) %}
      {% if messages %}
        <div>
          {% for category, message in messages %}
            <div class="alert alert-{{ category }} alert-dismissible fade show"
role="alert">
              {{ message }}
            </div>
```

```
        {% endfor %}
      </div>
    {% endif %}
  {% endwith %}

    <div class="logout-container" style="text-align: center; margin-top: 20px;">
      {% if user %}
        <a href="/logout" class="btn-btn-dark">Logout</a>
      {% endif %}
    </div>

    <form action="{{ url_for('register') }}" method="POST" style="background-color:
rgba(0, 0, 0, 0.5); padding: 20px; border-radius: 10px;">  <input type="text"
name="name" placeholder="Name" required><i class='bx bxs-user'></input> </i>
<br>
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password" required><i
class='bx
('login') }}">Login page</a></p>
    </form>
  </div>
 </div>
</body>
</html>



<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register form</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='register.css') }}">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Add Boxicons CDN - ->
  <link href="https://cdn.jsdelivr.net/npm/boxicons@2.1.4/css/boxicons.min.css"
rel="stylesheet">

</head>
<body>
```

```
  <div style="background-image: url('{{ url_for('static', filename='register.jpg') }}');
background-size: cover; background-position: auto; height: 99vh; width: 210vh;">
    <div style="text-align: center; padding-top: 100px; color: white;">
    <form action="{{ url_for('register') }}" method="POST">
       <h1>Register Form</h1>

          <!-- Flash Messages - ->
    {% with messages = get_flashed_messages(with_categories=True) %}
    {% if messages %}
     <div>
       {% for category, message in messages %}
        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
          {{ message }}
        </div>
       {% endfor %}
     </div>
    {% endif %}
  {% endwith %}
  <!-- Logout Button Section - ->
  <div class="logout-container" style="text-align: center; margin-top: 20px;">
    {% if user %}
       <a href="/logout" class="btn-btn-dark">Logout</a>
    {% endif %}
</div>

       <input type="text" name="name" placeholder="Name" required></input>    <br>
       <input type="email" name="email" placeholder="Email" required><br>
       <input type="password" name="password" placeholder="Password" required><i
class='bx bxs-lock-alt' ></i><br>   </div>
       <button type="submit">Register</button>
       <p><a href="{{ url_for('login') }}">Login page</a></p>
    </form>
   </div>
</body>
</html>

-->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register form</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/register.css') }}">
    <link href="../static/images/favicon.png" rel="icon">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/boxicons@2.1.4/css/boxicons.min.css"
rel="stylesheet">
</head>
<body>
  <div style="background-image: url('{{ url_for('static', filename='images/register.jpg')
}}'); background-size: cover; background-position: auto; height: 99vh; width: 210vh;">
    <div style="text-align: center; padding-top: 100px; color: white;">
    <form action="{{ url_for('register') }}" method="POST">
       <h1>Register Form</h1>

       <!-- Flash Messages -->
       {% with messages = get_flashed_messages(with_categories=True) %}
       {% if messages %}
         <div>
           {% for category, message in messages %}
             <div class="alert alert-{{ category }} alert-dismissible fade show"
role="alert">
               {{ message }}
             </div>
           {% endfor %}
         </div>
       {% endif %}
       {% endwith %}
             <!-- Logout Button Section --> <
" placeholder="Email" required><br>
       <input type="password" name="password" placeholder="Password" required><i
class='bx bxs-lock-alt'></i><br>

       <!-- Centered Buttons -->
       <div style="margin-top: 20px;">
          <button type="submit" class="sparkle-button">Register</button>
          <p><a href="{{ url_for('login') }}" class="sparkle-button">Login page</a></p>
       </div>
    </form>
   </div>
</body></html>
```

6.3 SCREENS & OUTPUTS

Fig-6.3.a Register page

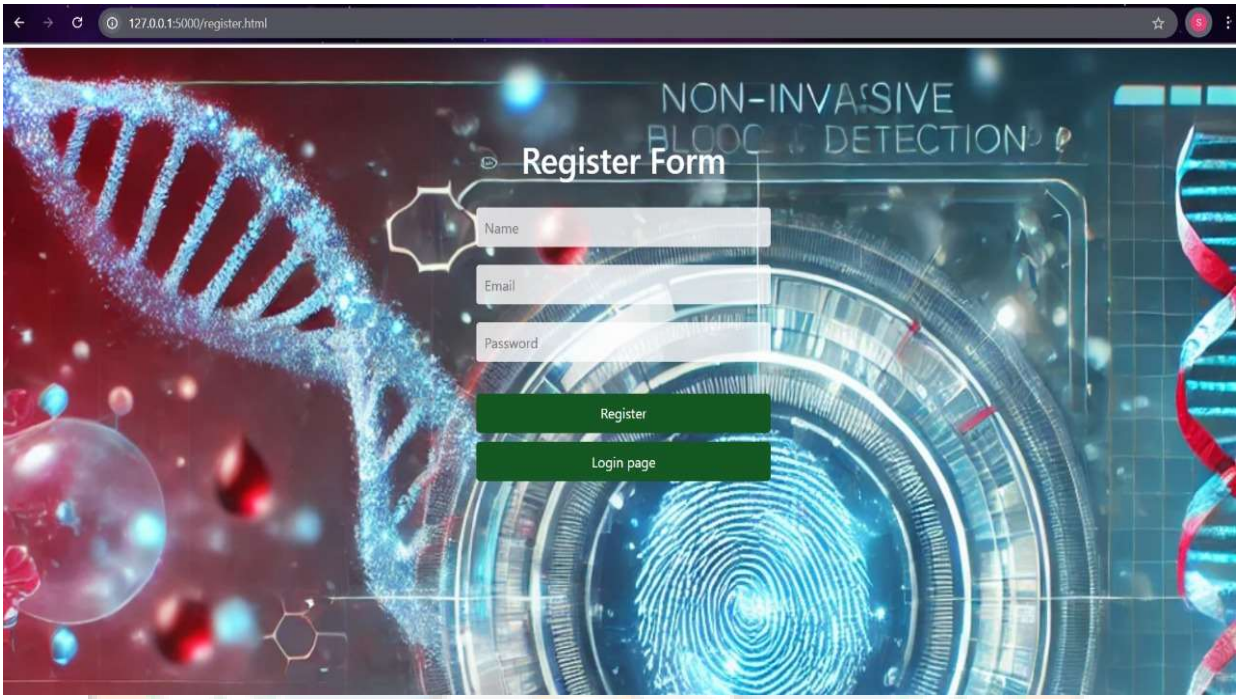


Fig-6.3.b Login Page

Fig-6.3.c Home Page

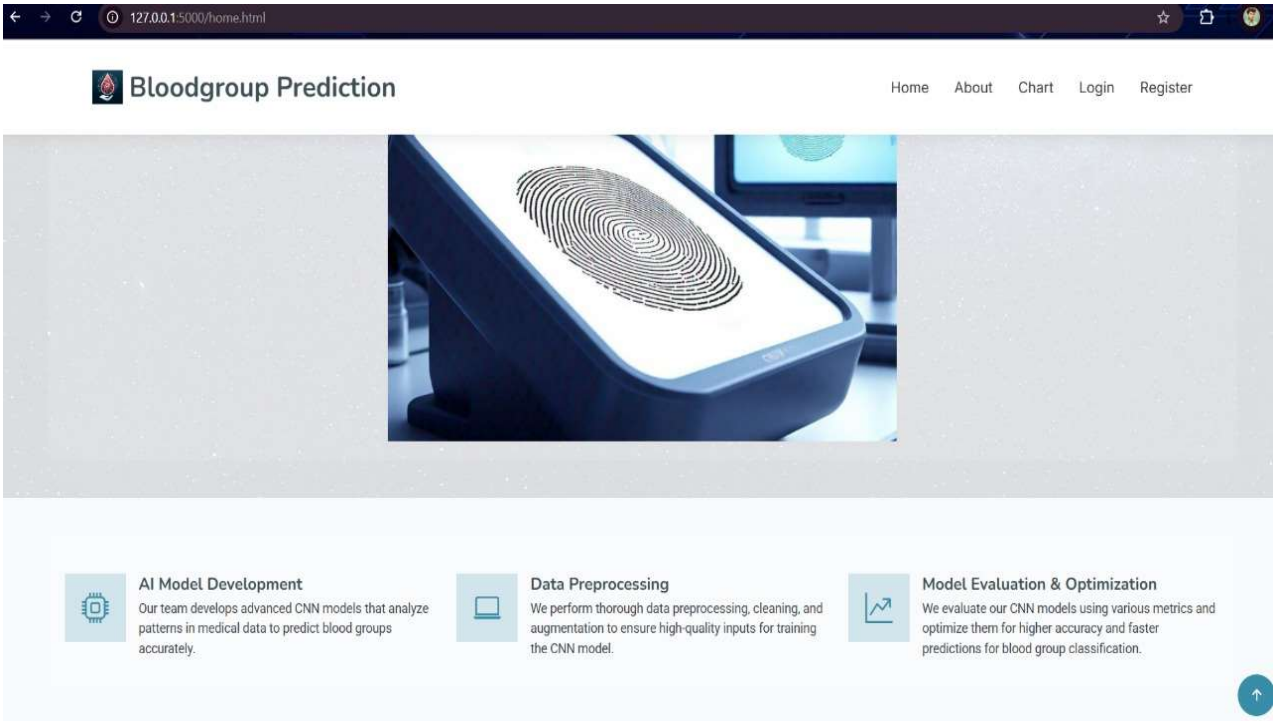


Fig-6.3.d About Page

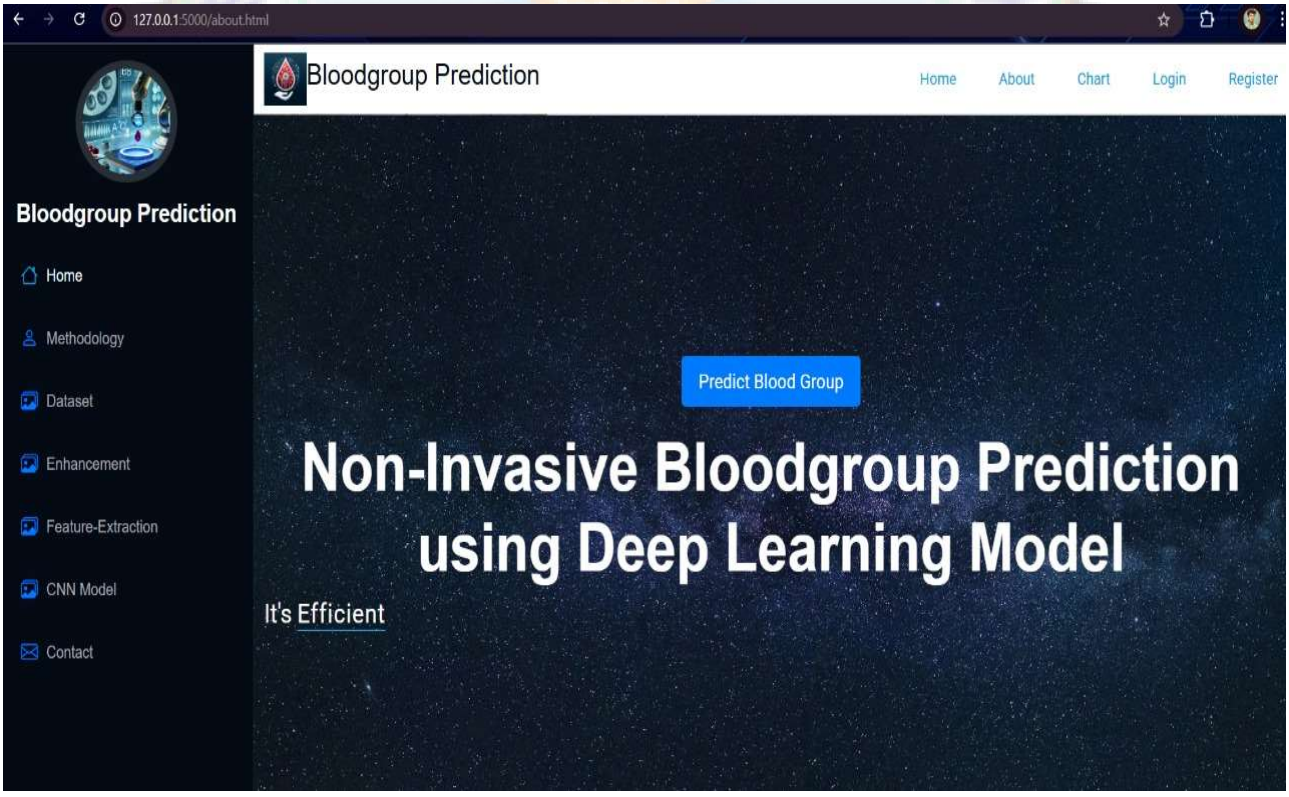
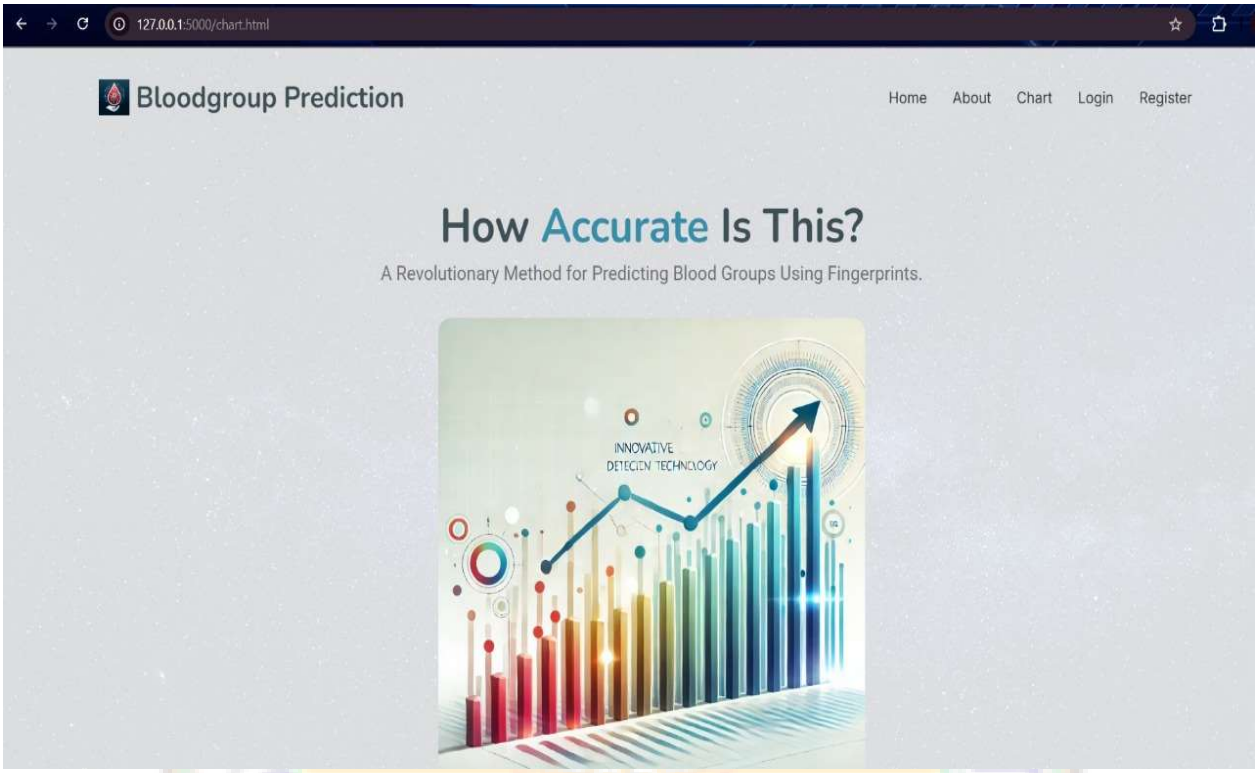


Fig-6.3.e Chart Page



6.3.1 CNN MODEL PERFORMANCE

Fig-6.3.1.a Data set Classification

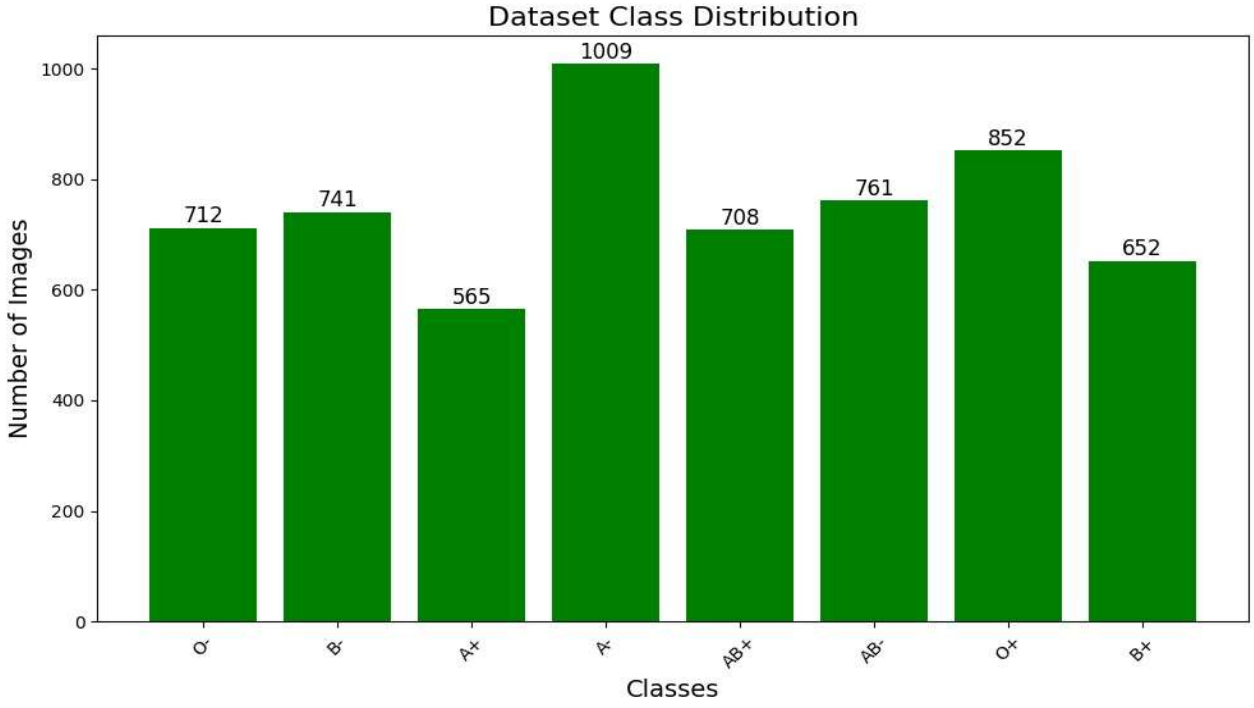


Fig-6.3.1.b Model accuracy of Trained dataset

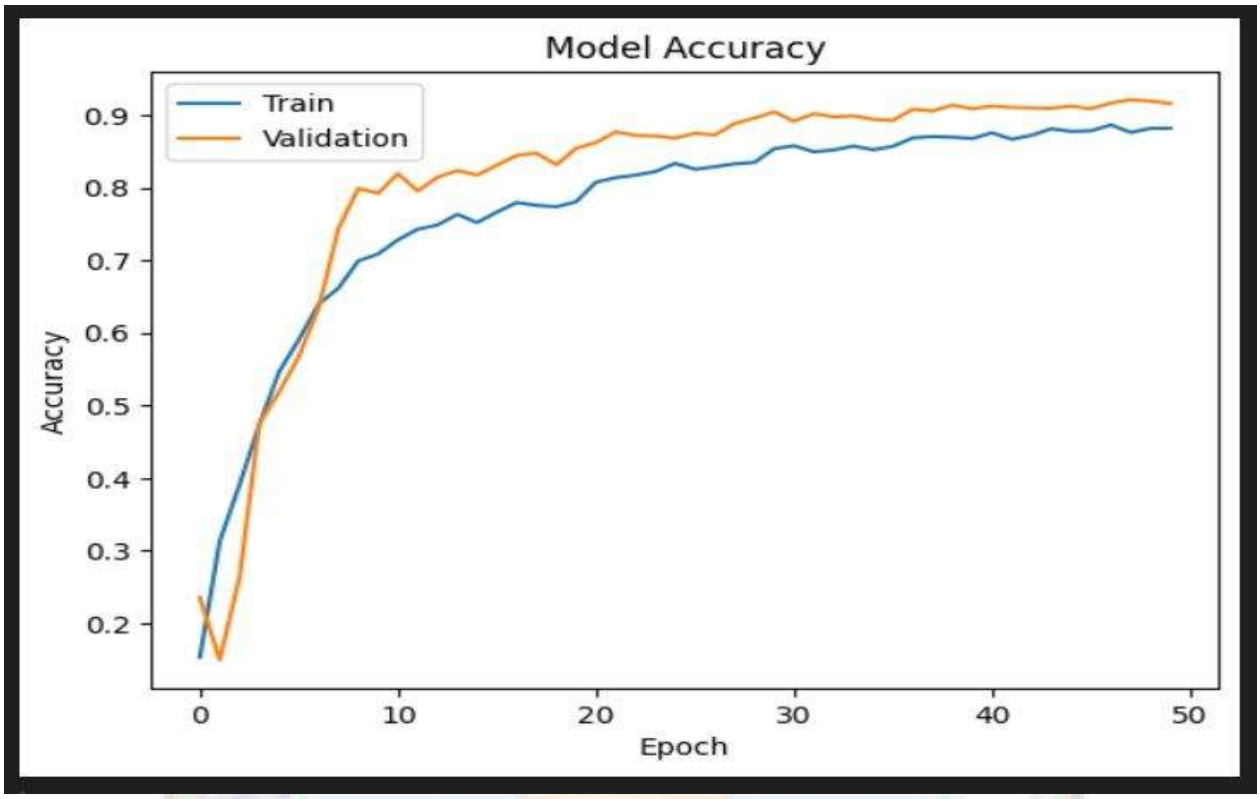
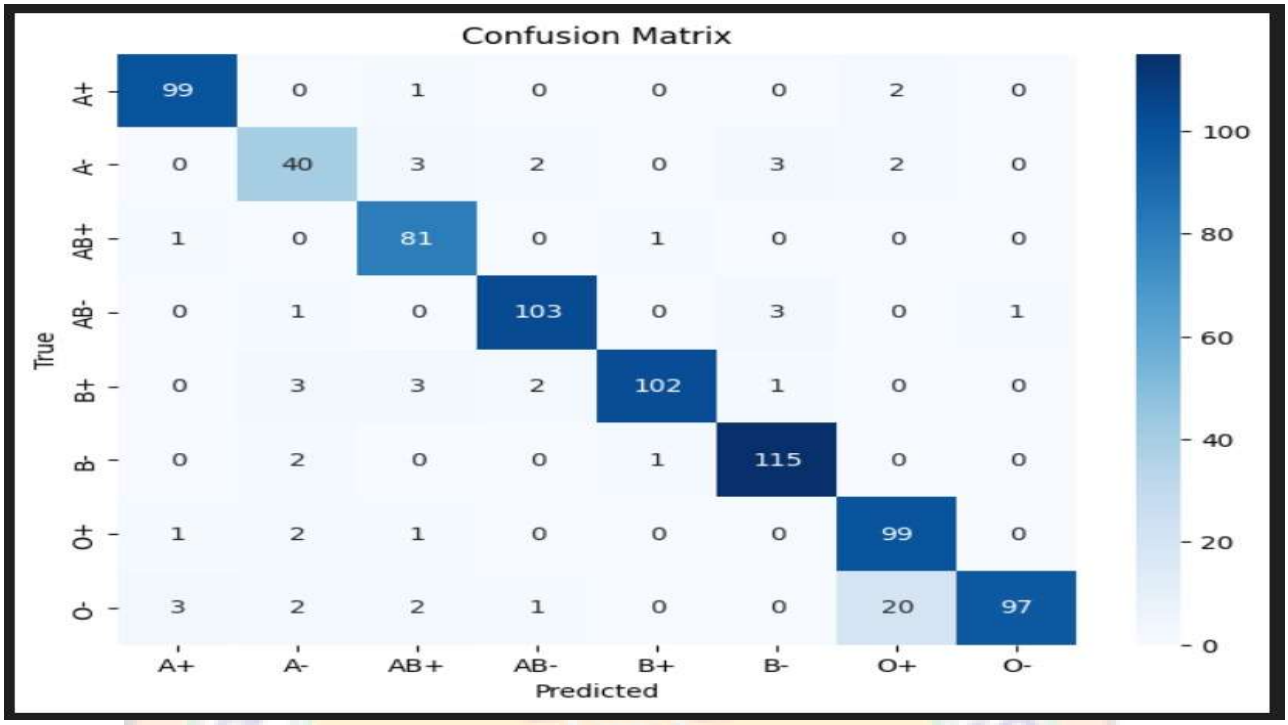


Fig-6.3.1.c Dataset Classification Report

Classification Report:

	precision	recall	f1-score	support
A+	0.95	0.97	0.96	102
A-	0.80	0.80	0.80	50
AB+	0.89	0.98	0.93	83
AB-	0.95	0.95	0.95	108
B+	0.98	0.92	0.95	111
B-	0.94	0.97	0.96	118
O+	0.80	0.96	0.88	103
O-	0.99	0.78	0.87	125
accuracy			0.92	800
macro avg	0.91	0.92	0.91	800
weighted avg	0.93	0.92	0.92	800

Fig-6.3.1.d Confusion matrix



6.3.2 Results

Fig-6.3.2.a User Form to Enter and predict Blood group

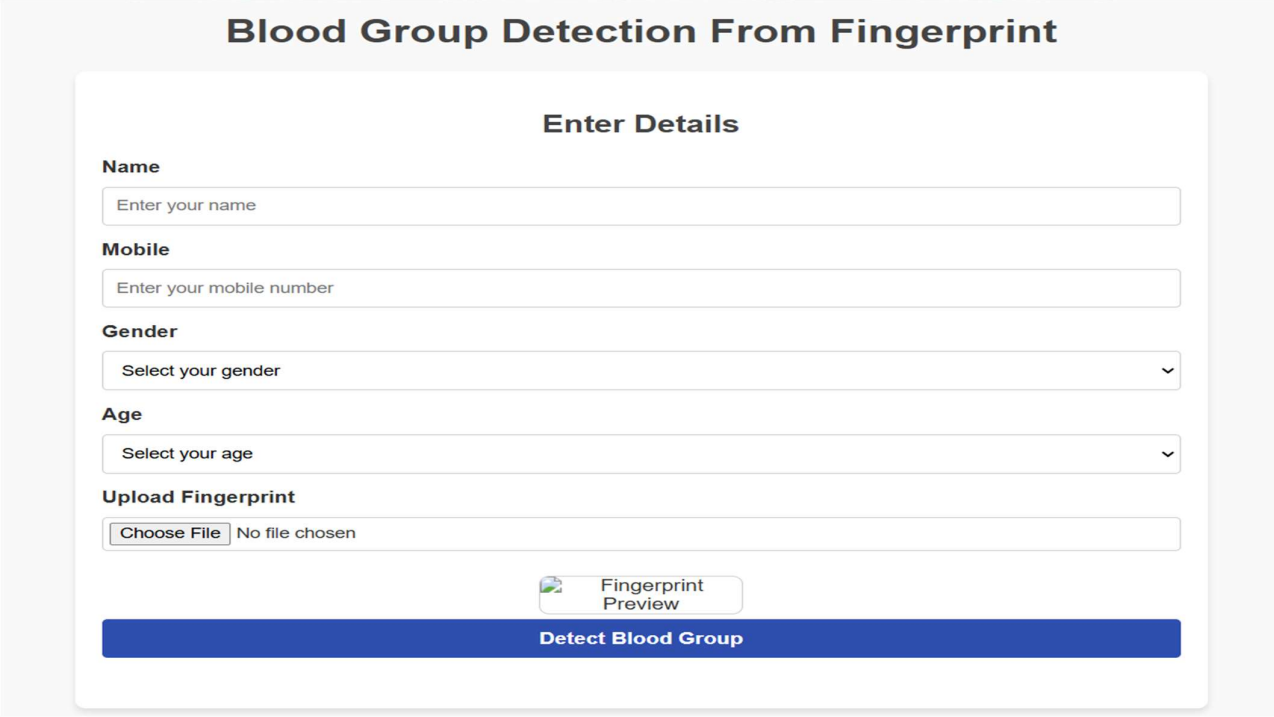


Fig-6.3.2.b Result of A+

Detection Result

Field	Value
Name	Sai
Mobile	9836255234
Gender	Male
Age	22
Fingerprint	
Confidence	0.9075931906700134
Blood Group	A+

Fig-6.3.2.c Result of A-

Detection Result

Field	Value
Name	ashok
Mobile	8762345688
Gender	Male
Age	23
Fingerprint	
Confidence	0.7593255043029785
Blood Group	A-

Fig -6.3.2.d Result of O+

Detection Result

Field	Value
Name	tarun
Mobile	9236567798
Gender	Male
Age	31
Fingerprint	
Confidence	0.8669450283050537
Blood Group	O+

Fig -6.3.2.e Result of O-

Detection Result

Field	Value
Name	ravi
Mobile	8008667798
Gender	Male
Age	31
Fingerprint	
Confidence	0.9902979731559753
Blood Group	O-

Fig -6.3.2.f Result of B-

Detection Result

Field	Value
Name	bargav
Mobile	8008293579
Gender	Female
Age	20
Fingerprint	
Confidence	0.8723762035369873
Blood Group	B-

Fig -6.3.2.g Result of AB+

Detection Result

Field	Value
Name	bargavi
Mobile	6300293579
Gender	Female
Age	20
Fingerprint	
Confidence	0.7690436840057373
Blood Group	AB+

Fig -6.3.2.h Result of AB-

Detection Result

Field	Value
Name	Mahesh
Mobile	6732935790
Gender	Male
Age	31
Fingerprint	
Confidence	0.9750720262527466
Blood Group	AB-

Fig -6.3.2.i Result of B+

Detection Result

Field	Value
Name	swathi
Mobile	9842935790
Gender	Female
Age	20
Fingerprint	
Confidence	0.9446220993995667
Blood Group	B+

7.CONCLUSION

CONCLUSION

While previous research has highlighted the challenges of accurately predicting blood groups solely from fingerprint patterns due to shared patterns and the complex interplay of various factors, our study demonstrates the potential of utilizing advanced techniques like Convolutional Neural Networks (CNNs) to overcome these limitations.

Our CNN model, achieving an impressive 90% accuracy, suggests that deep learning algorithms can effectively discern subtle patterns and features in fingerprint images that may be associated with blood types. This breakthrough underscores the importance of leveraging sophisticated computational approaches to unravel the intricate relationship between fingerprints and blood groups.

Further research is needed to fully understand the underlying mechanisms and validate the generalizability of our findings. However, our results provide a compelling foundation for future investigations and pave the way for the development of innovative, non-invasive blood typing methods with enhanced accuracy and reliability. This could potentially revolutionize blood typing practices, particularly in emergency medical care and resource-limited settings.

8.REFERENCES

REFERENCES

[1] Vijaykumar, Patil N., and D. R. Ingle. "A Novel Approach to Predict Blood Group using Fingerprint Map Reading." 2021 6th International Conference for Convergence in Technology (I2CT). IEEE, 2021.
[2] https://www.irjet.net/archives/V11/i3/IRJET-V11I3169.pdf
[3] Dr. D.Siva Sundhara Raja and J. Abinaya, "A Cost-Effective Method for Blood Group Detection Using Fingerprints", International Journal of Advance Study and Research Work ,Volume 2, March 2019 .
[4] Tariq, A. H., Hussein, A. K., Sara, A. Q., & Tasnim, A. H. (2023). The association between blood groups in Omani population., Sulta University, Muscat, College of Medicine and Health Sciences 10. fingerprint patterns
[5] Smith, J., Johnson, A., & sir Lee, C. (2020). Finger Based Biometric identification Blood Group and Predict Using Learning Techniques.
[6] Jeevesh Gupta, A., Mr. Agarwal, S., & Jain, R. hi (2019). Blood Group Machine Learning.
[7] Verma, A., & Alia Bhatt, V. (2018). Automation of Blood prediction Using Fingerprint Images Group Identification with Fingerprint Analysis.
[8] Nandakumar, R., & Subrama, K. (2017). Blod Group identification Using Deep CNN.
[9] Ali, Mouad MH, et al. "Fingerprint recognition for person identification and verification based on minutiae matching." 2016 IEEE 6th international conference on advanced computing (IACC). IEEE, 2016.
[10] Fernandes, Jose, et al. "A complete blood typing device for automatic agglutination detection based on absorption spectrophotometry." IEEE Transactions on Instrumentation and Measurement 64.1 (2014): 112-119
[11]https://www.researchgate.net/publication/379049931_Blood_group_determination_using_fingerprint
[12]Referencepaper
https://www.irjmets.com/uploadedfiles/paper//issue_7_july_2024/60089/final/fin_irjmets1720690577.pdf
[13]Reference paper
https://www.irjmets.com/uploadedfiles/paper//issue_7_july_2024/60089/final/fin_irjmets1720690577.pdf

[14]Reference paper-
https://www.ieeexpert.com/python-projects/blood-group-detection-using-fingerprint-with-image-processing/