Tribhuvan University

Faculty of Humanities and Social Sciences

Orchid International College

Face Recognition System Using SVM

A PROJECT REPORT

Submitted to

Department of Humanities and Social Science

Orchid International College

In partial fulfillment of the requirements for the Bachelor in Computer Application.

Submitted by

Saugat Balami(93902052)

September 2024

Under the Supervision of

Er.Dhiraj Kumar Jha

Tribhuvan University

Faculty of Humanities and Social Sciences

Orchid International College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project be prepared under my supervision by Saugat Balami entitled "Face Recognition System" in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

____

SUPERVISOR

Er. Dhiraj Kumar Jha

Project Coordinator

Department of IT

Orchid International College

Bijaychowk, Gaushala

Tribhuvan University

Faculty of Humanities and Social Sciences

Orchid International College

LETTER OF APPROVAL

This is to certify that this project was prepared by Saugat Balami entitled "**Face Recognition System"** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

SIGNATURE of Supervisor	**SIGNATURE of Internal Examiner**
Er. Dhiraj Kumar Jha Department of IT Orchid International College	Er. Nipun Thapa Department of IT Orchid International College

SIGNATURE of External Examiner
External Examiner **Tribhuvan University** **Kritipur , Nepal**

ABSTRACT

The Face Recognition System has gained significant importance in various fields, including security, authentication, and surveillance. This project presents the development of a Face Recognition system using Support Vector Machines (SVM) with a Radial Basis Function (RBF) kernel for classification. The system employs the Multi-task Cascaded Convolutional Networks (MTCNN) for face detection and the InceptionResnetV1 model for extracting facial embeddings, which provides a robust and efficient face representation. The model is further calibrated using the CalibratedClassifierCV from the scikit-learn library to improve the performance by providing probabilistic outputs for better decision making. The system was evaluated using a benchmark facial recognition dataset, demonstrating high accuracy in recognizing and verifying individuals. This project aims to create a reliable face recognition solution that can be effectively utilized in real-world scenarios, balancing accuracy and computational efficiency. The final model can be deployed for various applications including access control systems and security frameworks.

Keywords: Face Recognition System ,SVM ,RBF , Scikit-learn

ACKNOWLEGEMENT

We express our heartfelt gratitude to the dedicated individuals who played a pivotal role in the successful completion of this project. Their unwavering support and guidance were instrumental in shaping our journey and achieving our goals.

First and foremost, we extend our deepest appreciation to **Er. Dhiraj Kumar Jha**, our esteemed project coordinator. His expertise, encouragement, and valuable insights provided us with a strong foundation, propelling us forward throughout the project.

List of Table

LIST OF FIGURES

LIST OF ABBREVIATIONS

DOM – Domain Object Model

UC – Use Case

UI – User Interface

VS – Visual Studio

CSS – Cascading style sheet

HTML – Hyper Text Markup Language

SQL – Structured Query Language

MTCNN –Multi-Task Cascaded Convolutional Neural Networks

SVC – Support Vector Machine

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

Face recognition systems have become an essential component in various sectors such as security, access control, and user authentication. The ability of machines to recognize and verify human faces in real-time has seen significant advancement due to the development of powerful machine learning algorithms and deep learning models. This project focuses on developing an efficient and accurate Face Recognition System using a combination of state-of-the-art technologies, including deep learning for face detection and embeddings, and Support Vector Machines (SVM) for classification.

The core of the system involves the use of Multi-task Cascaded Convolutional Networks (MTCNN) for detecting and aligning faces from input images. Once detected, the face embeddings are extracted using the InceptionResnetV1 model, which is known for providing robust facial features by utilizing deep neural network architectures. These embeddings are then used to train a Support Vector Machine (SVM) model with a Radial Basis Function (RBF) kernel, chosen for its ability to effectively handle non-linear relationships in the data.

To further refine the predictions and provide probabilistic outputs for improved decision-making, the CalibratedClassifierCV from the scikit-learn library is applied, although its role in the system requires further exploration. The final system is integrated into a Flask-based web application, allowing for easy interaction and deployment of the face recognition model in real-world scenarios.

The system’s performance was evaluated on standard facial recognition datasets, demonstrating high accuracy in identifying and verifying individuals. This project aims to contribute to the growing field of biometric recognition systems by providing a scalable, efficient, and accurate face recognition solution.

1.2 Problem Statement

As security threats and identity fraud continue to rise, traditional methods of user authentication, such as passwords and keycards, are proving inadequate. There is a need for more secure, reliable, and user-friendly biometric systems that can recognize individuals with a high degree of accuracy. This project addresses the challenge of developing a face recognition system capable of identifying individuals in real-time with a high degree of accuracy and efficiency, making it applicable for use in security systems, access control, and other verification applications. In addition to security, the management of attendance in institutions and workplaces remains a time-consuming and error-prone process. Traditional methods of recording attendance, such as manual registers and RFID cards, are not only inefficient but also susceptible to manipulation, such as proxy attendance. To overcome these issues, face recognition technology offers a more reliable and automatic solution for attendance management. A face recognition-based attendance system can streamline the process, making it more efficient by automatically marking attendance when a person's face is detected and recognized. This reduces the possibility of errors and ensures that the attendance records are accurate and tamper-proof.By integrating the face recognition system with an attendance management framework, this project also aims to solve the challenges associated with existing methods, providing a secure, accurate, and seamless process for attendance tracking in schools, offices, and other institutions.

1.3 Objectives

The primary objectives of this project are as follows:

- To design and implement a face recognition system using modern machine learning and deep learning techniques.
- To achieve real-time face detection and recognition using efficient models and algorithms.
- To employ MTCNN for accurate and efficient face detection and alignment.
- To explore the application of the system in attendance management, providing a secure and efficient method for automated attendance tracking.

1.4 Scope and Limitation

1.4.1 Scope

The scope of this project encompasses the development and deployment of a real-time face recognition system with the following capabilities and features:

Real-Time Face Detection and Recognition: The system is designed to detect and recognize faces in real-time, providing instant feedback and results.

Accurate Face Detection Using MTCNN: The Multi-task Cascaded Convolutional Networks (MTCNN) algorithm is employed for effective face detection and alignment in images, ensuring precision in identifying facial regions.

Robust Feature Extraction with InceptionResnetV1: To generate high-quality face embeddings, the system utilizes the InceptionResnetV1 model, which effectively captures facial features for reliable recognition.

Classification Using SVM with RBF Kernel: The system leverages Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel to classify faces based on the extracted embeddings, ensuring accurate and efficient recognition.

Web Application Interface: A user-friendly web interface is developed using Flask, enabling users to interact with the face recognition system by uploading images and verifying identities through a local server.

1.4.2 Limitations

While the face recognition system achieves its primary goals, certain limitations exist:

Generalization to Diverse Conditions: The model's performance may degrade under extreme variations in lighting, facial occlusions, or expressions. The system requires further enhancement to handle diverse and challenging conditions effectively.

Limited Dataset and Training: The current system is trained on a specific dataset, which may limit its generalization to new faces or large-scale datasets. For optimal performance in broader applications, retraining or fine-tuning on additional data is recommended.

Scalability Challenges: The system is initially designed to operate on a local server, which may face performance and scalability issues when deployed in larger or distributed environments without significant optimization.

1.3 Development Methodology

The development of the face recognition system follows an Iterative and Incremental Development approach. This methodology emphasizes building the system in small, manageable increments, with each iteration refining and improving upon the previous version. This approach is particularly effective for machine learning projects, as it allows for continuous evaluation, model improvement, and feature enhancement based on feedback and testing.

The key stages of this methodology as applied to this project are as follows:

1. **Requirement Gathering and Analysis**: The initial phase involves understanding the requirements of the face recognition system, including the functionality, data sources, performance metrics, and application needs. A thorough analysis of existing systems and technologies like MTCNN, InceptionResnetV1, and SVM is performed to ensure the feasibility of the project.
2. **System Design and High-Level Architecture**: The architecture of the system is designed to handle real-time face detection and recognition. Key components such as the face detection module (MTCNN), feature extraction module (InceptionResnetV1), and classification module (SVM with RBF kernel) are defined, and their interactions are mapped out. The system's integration into a web-based application using Flask is also planned in this phase.
3. **Incremental Development and Testing**: The system is developed in a series of iterative cycles. Each cycle focuses on a specific component or feature of the system:
   - **Iteration 1**: Implementing and testing the face detection module using MTCNN to ensure accurate identification of facial regions.
   - **Iteration 2**: Developing the face embedding extraction using InceptionResnetV1 and verifying that the embeddings effectively represent facial features.
   - **Iteration 3**: Training the SVM model with an RBF kernel for classification, followed by testing its performance on known datasets to ensure high accuracy.
   - **Iteration 4**: Integrating all components into a Flask web application and enabling real-time recognition through an easy-to-use interface.

   Each iteration includes thorough testing and validation to confirm the effectiveness of the components before moving on to the next phase.
4. **Model Evaluation and Fine-Tuning**: After the initial development of the system, the face recognition model is evaluated for accuracy, efficiency, and scalability. Parameters such as the SVM hyperparameters, embedding

thresholds, and performance of **CalibratedClassifierCV** are fine-tuned to optimize the system's accuracy and response time.

5. **Integration and Deployment**: The final system is integrated as a whole and deployed as a web application using Flask. This allows users to interact with the system through a local server, enabling real-time face detection and recognition. The web interface is tested for usability, and any performance bottlenecks are addressed.
6. **Final Testing and Validation**: Extensive testing is carried out on the complete system to validate its functionality in various real-world scenarios. The system is tested for different conditions like lighting variations, facial occlusions, and real-time performance to ensure reliability.
7. **Documentation and Reporting**: The project documentation is prepared in parallel with development to ensure all iterations, changes, and final system functionalities are well-recorded. The final project report includes all phases, technical details, and results, following the prescribed format.

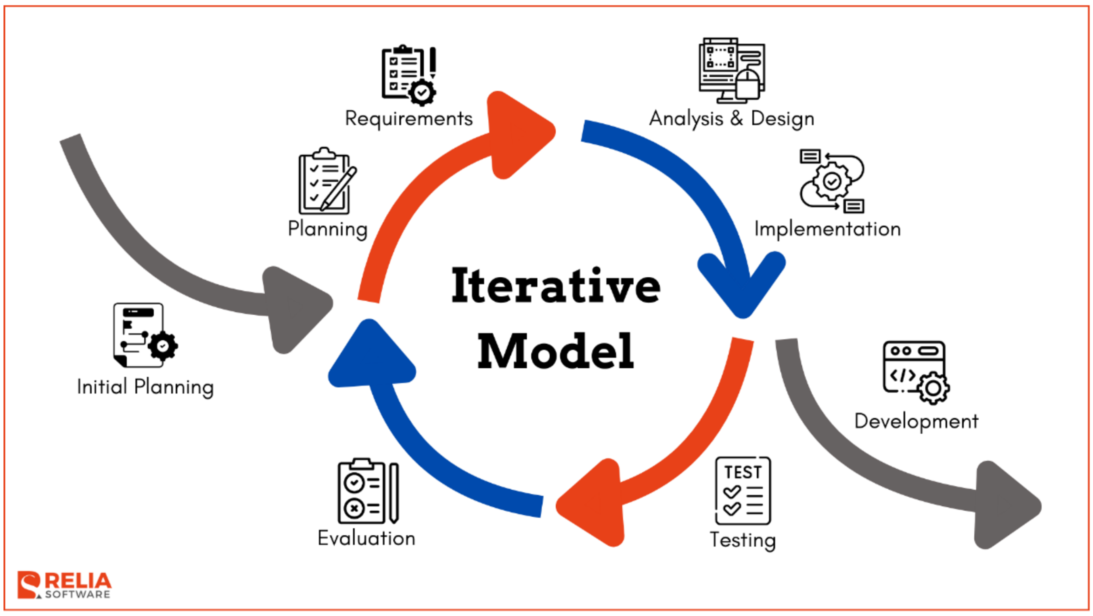


Figure 1: Iterative model

1.6 Report Organization

This is the report organization for the house rental system which also includes charts/diagrams to illustrate the system architecture and design. Furthermore, it contains information regarding the tools and technologies used to build the system.

Table 1 Outline of the Report

Introduction	Introduction Problem Statement Objective Scope and Limitations Development Methodology Report Organization
Background Study	Background Study Literature Review
System Analysis and Design	System Analysis Feasibility Analysis System Design
Implementation and Testing	Tools and Technology Test Cases
Conclusion and Recommendations	Lessons Learnt Conclusion Future Recommendations

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Face recognition technology has become a crucial area of research in computer vision and biometrics, with applications spanning security, access control, and attendance management. The process involves face detection, feature extraction, and classification. In this project, the Multi-task Cascaded Convolutional Networks (MTCNN) is used for accurate face detection and alignment. MTCNN's deep convolutional network efficiently locates faces and their key landmarks in real-time.

For feature extraction, InceptionResnetV1 is utilized to produce robust face embeddings, capturing unique facial features that are crucial for classification. This approach outperforms traditional techniques like Eigenfaces and Fisherfaces, offering better robustness against variations in expression, lighting, and pose. The extracted face embeddings are then classified using a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel, chosen for its ability to handle non-linear separations in complex face data effectively.

To improve interpretability and decision-making, CalibratedClassifierCV is applied to generate probabilistic outputs, which helps refine recognition accuracy. The entire system is developed as a web application using Flask, allowing users to interact through a local server and enabling real-time face recognition. The project aims to deliver a reliable and efficient face recognition system applicable to real-world scenarios, such as security systems, authentication, and attendance tracking.

2.2 Literature Review

Face recognition has been a key area of research within pattern recognition, with multiple approaches developed to address the challenge of accurately identifying faces under various conditions, such as lighting, pose, and expression. Early techniques, like geometric feature-based approaches, relied on detecting and analyzing specific facial features (e.g., eyes, nose, mouth) and their relative properties, such as distances and angles [1]. These methods often faced limitations due to their dependence on reliable feature extraction, which could be inconsistent under different conditions.

Subsequently, template matching and neural network-based approaches emerged as more practical alternatives, as they operated directly on the image pixels without the need for explicit feature extraction. One pioneering method in this category was the Eigenfaces approach [2], which used Principal Component Analysis (PCA) to construct a face-space where each face was represented as a vector of weights. This method, while a significant improvement, struggled to handle changes in viewpoint, lighting, and facial expressions effectively.

The work of Guo et al. [3] explored the application of Support Vector Machines (SVMs) for face recognition, demonstrating that SVMs can be particularly effective in pattern recognition tasks involving faces. SVMs operate by identifying an optimal separating hyperplane between two classes, maximizing the margin between them for better generalization. For face recognition, the SVM approach is enhanced through a binary tree recognition strategy, which enables handling multiple classes by building a decision tree structure for classification.

Guo et al. [3] evaluated this SVM-based approach on both the Cambridge ORL face database and a larger compound dataset, showing that SVMs performed better than the Eigenfaces approach using the Nearest Center Classification (NCC) criterion. The binary tree strategy optimized the recognition process by reducing the number of comparisons required to classify new faces. These experiments demonstrated that SVM-based face recognition had a lower error rate, highlighting its potential for robust and accurate face classification.

The findings of Guo et al. [3] indicate that SVMs are effective in handling the high-dimensional and non-linearly separable nature of face data, offering a more robust solution than traditional template-matching techniques. The binary tree structure for multi-class recognition further improves the efficiency of the SVM-based approach, making it suitable for real-world applications in face recognition systems.

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

The system analysis of the system is done by conducting requirement analysis, feasibility analysis, object modeling, dynamic modeling and process modeling as follows:

3.1.1 Requirement Analysis

The requirement analysis is broken down into two parts, namely functional requirements, and non-functional requirements. Both of these are discussed below:

3.1.1.1 Functional Requirement

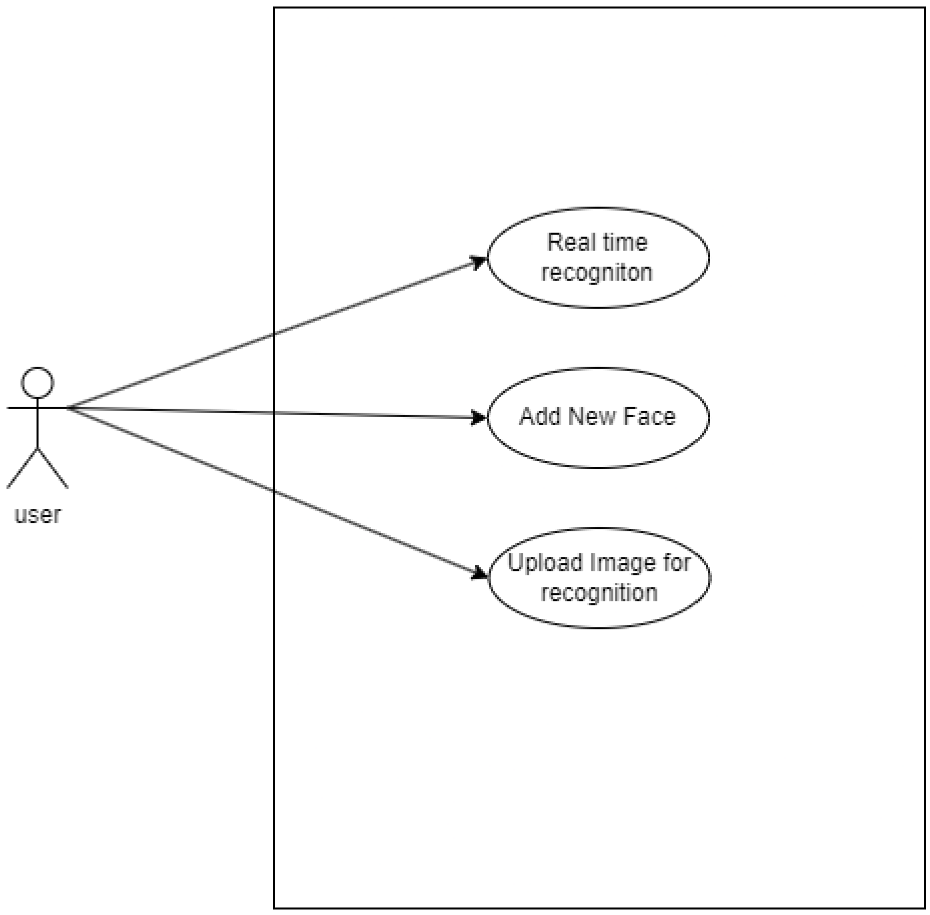


Figure 3.1: Use Case Diagram

Table 3.2: Real time recognition

Use Case ID	UC-001
Use Case Name	Real time face recognition
Actor	User
Brief Description	The user interacts with the system, which captures live video feeds or images. It detects faces in real-time using MTCNN, matches them with previously trained face data, and recognizes individuals based on the stored labels.
Triggers	The user initiates the face recognition process by opening the camera or uploading a live video feed.
Preconditions	• The face recognition system is connected to a functional camera or video feed. • The system must have access to trained face data. • The user's face must be clearly visible in the camera. • Lighting conditions must be sufficient for face detection.
Normal Flow	• The user starts the recognition system. • The system opens the camera and begins processing the video feed. • The system detects faces in real-time using MTCNN. • Detected faces are processed and compared with the trained dataset. • If a match is found, the system displays the recognized person's label (e.g., name). • The system logs the recognition data, including timestamps and any relevant metadata.
Alternate Flows	• If no faces are detected, the system alerts the user and continues scanning until a face is detected.

Table 3.1: Add New Face

Use Case ID	UC-001
Use Case Name	Real time face recognition
Actor	User
Brief Description	The user uploads 8 images of a new individual's face to the system for training. The system processes the images and adds the face data to the recognition dataset.
Triggers	The user selects the option to add a new face to the system. The user uploads 8 images of a new individual.
Preconditions	• The system is operational, and the user is authorized to add new face data. • The user must have 8 clear and varied images of the individual (from different angles or expressions). • The face in the images must be clearly visible, with good lighting.
Normal Flow	• The user selects the option to add a new face to the system. • The system prompts the user to upload 8 images. • The user selects and uploads 8 images of the new individual.
Alternate Flows	• If the user uploads fewer than 8 images, the system prompts the user to upload additional images to meet the requirement.

3.1.1.2 Non-Functional Requirements

The non-functional requirements of Face Recognition system are as follows:

Performance Requirements:

- The system shall perform efficiently to enable real-time face detection and recognition without noticeable delays.

Usability Requirements:

- The user interface shall be intuitive, ensuring that users can operate the system easily, with minimal training or assistance.

Availability Requirement:

- The system is available 100% for the user and is used 24hrs a day and 365 days a year.
- The system shall be operational 24hrs a day and 7 days a week.

Environmental Requirements:

- The system shall require a modern web browser to function successfully.
- The system shall operate in a standard networked environment with reliable internet connectivity.

Compatibility Requirements:

- The system shall be compatible across major platforms, including Windows, macOS, and Linux, and shall work under the required environment (e.g., using popular browsers such as Chrome, Firefox, and Safari).
- It shall function smoothly on both desktop and mobile devices.

3.1.2 Feasibility Analysis

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

3.1.2.1 Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different type of frontend and backend platform.

3.1.2.2 Operational Feasibility

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to a layman. Besides, a proper training has been conducted to let know the essence of the system to the users so that they feel comfortable with new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

3.1.2.2 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of the system. The tangible benefits proposed that the manual work and burden is reduced maximum as possible, resulting the reduction in manpower requirement and cost incurred on manpower as well. The system provides many benefits that can't be measured in terms of money for e.g., user friendliness, more efficient user response, maintenance of database, etc.

3.1.2.3 Legal Feasibility

The face recognition system must comply with relevant data protection and privacy laws, such as the **GDPR** or **CCPA**, to ensure lawful processing of biometric data. Key considerations include:

3.1.2.4 Schedule Feasibility

Schedule feasibility is the degree to which a deadline for a strategy, plan, project or process is realistic and achievable. The feasibility of this project is high as the system can be completed within the time limit.

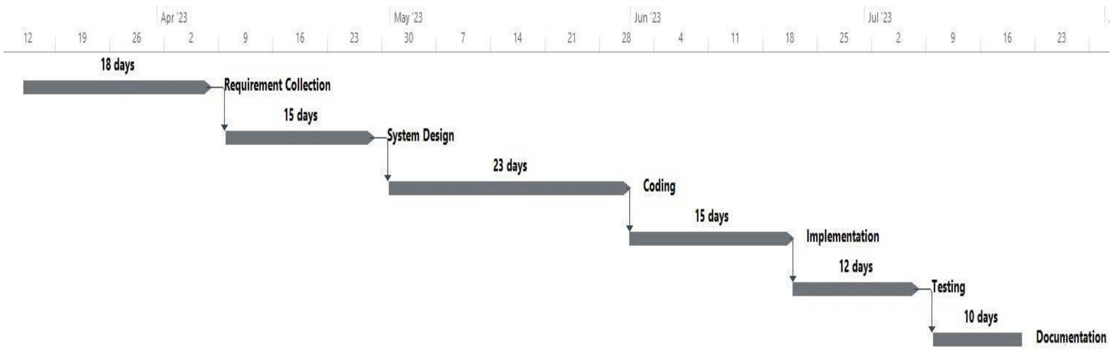


Figure-3.1: Gantt Chart

The above Gantt Chart shows the schedule for the project. We can see that the first 18 days were dedicated to requirement collection. The completion of requirement collection allowed the project to move to System Design which then took 15days. Then the project moved on to Coding. We coded for 23days and then started the implementation which took 15days. Then the project moved to testing phase which took 12days then finally we documented the project for 10 days.

3.1.3 Sequence Diagram

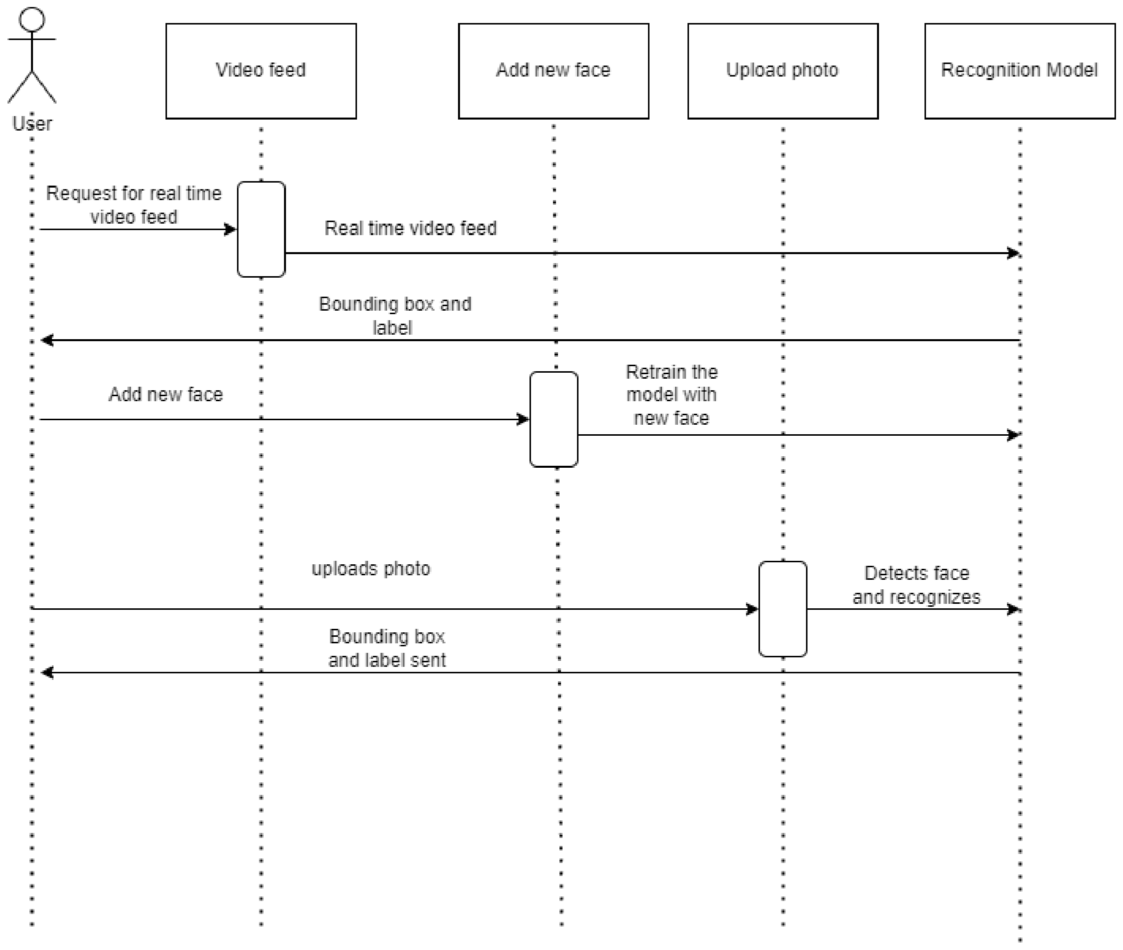


Figure-3.2: Sequence diagram

3.1.5 Activity Diagram

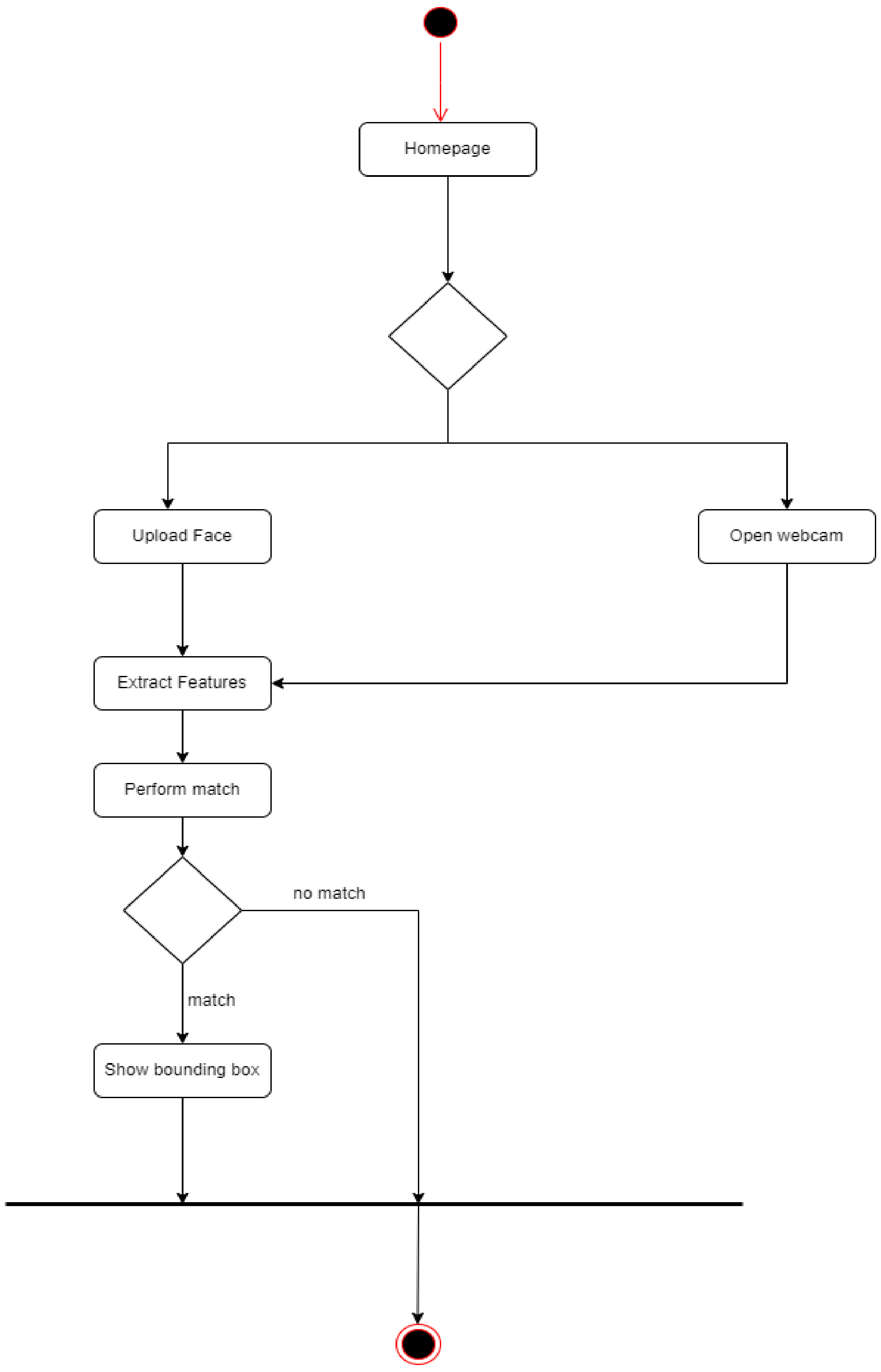


Figure-3.3: Activity diagram

3.2.1 Interface design

The interface design for all the major pages of Face Recognition is shown as follows:

Home Page:

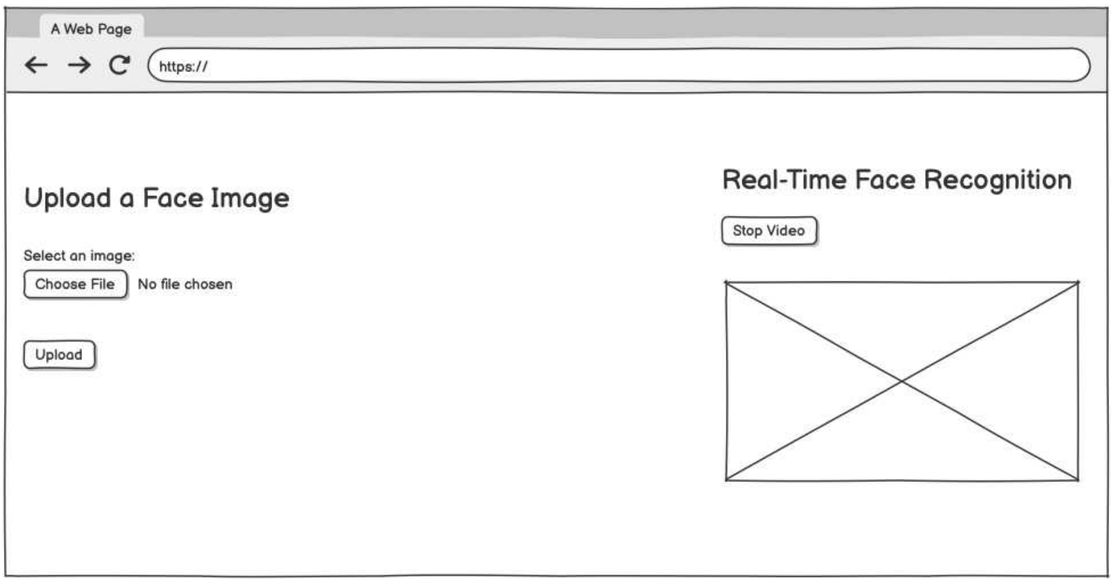


Figure 3.4: Homepage

Add New Face:

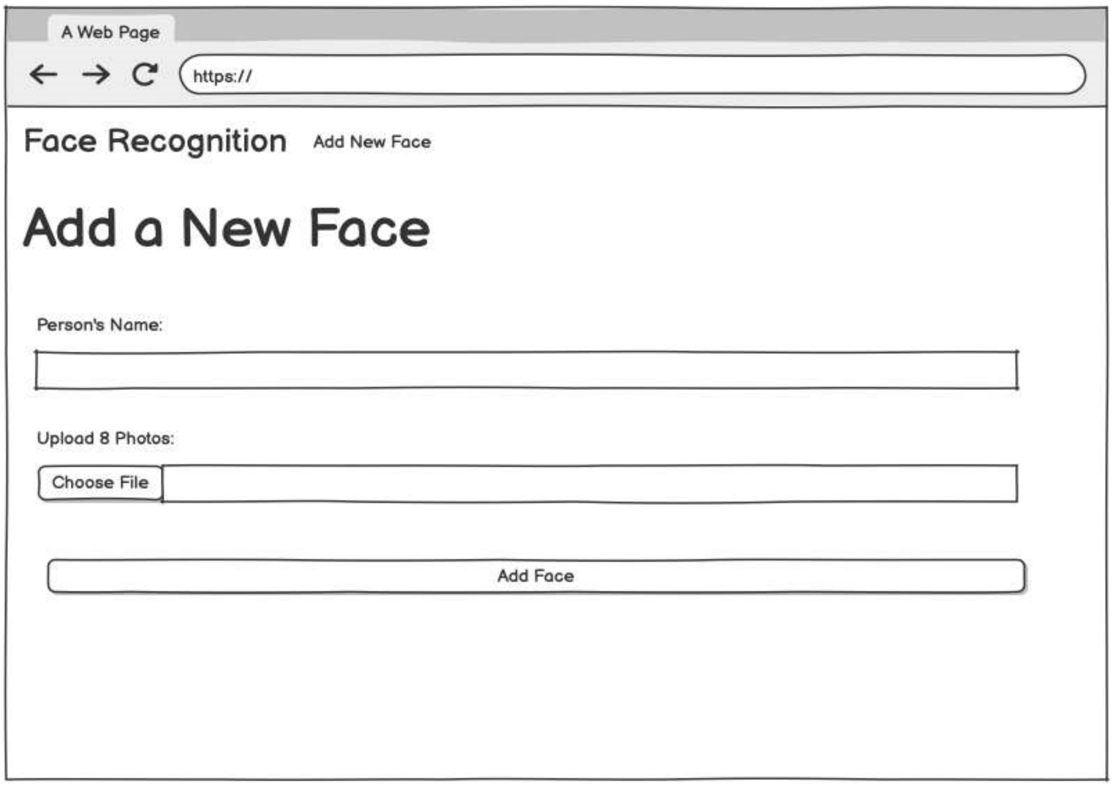


Figure-3.5: Upload new face page

3.2.2 Algorithm details

The detail information about the algorithms that has been used in this system are as follow:

Face Detection Using MTCNN (Multi-task Cascaded Convolutional Networks)

MTCNN is a robust algorithm designed for detecting faces and their landmarks (e.g., eyes, nose, mouth) in images with high precision. It achieves this through a three-stage cascade of neural networks — P-Net, R-Net, and O-Net — each improving upon the results of the previous stage to produce accurate face localization and alignment. Below is a detailed breakdown of how each network contributes to the face detection process.

1. P-Net (Proposal Network)

The Proposal Network (P-Net) is the first stage of MTCNN and is responsible for quickly generating candidate face regions from an input image.

Sliding Window Approach: P-Net slides a small window across the entire input image to evaluate patches of the image for potential faces.

Bounding Boxes and Facial Landmarks: For each window or patch, P-Net predicts a bounding box (coordinates of the face) and facial landmark positions (e.g., eyes, nose, mouth). This stage produces many candidate regions with rough localization.

Non-Maximum Suppression (NMS): To reduce overlapping and redundant bounding boxes, P-Net uses Non-Maximum Suppression, which removes less confident bounding boxes in favor of the best one. This stage filters out non-face regions and refines the bounding box locations.

2. R-Net (Refine Network)

The Refine Network (R-Net) takes the candidate bounding boxes proposed by P-Net and further refines them to improve the accuracy of face localization.

Refinement of Bounding Boxes: R-Net reprocesses the candidate face regions generated by P-Net to refine their bounding box coordinates, making the face localization more precise.

Filtering False Positives: This stage is responsible for eliminating false positives (regions that are wrongly classified as faces by P-Net) by performing a secondary check to ensure that only high-quality face proposals are passed on to the next stage.

Non-Maximum Suppression (NMS): Once again, R-Net applies NMS to eliminate redundant bounding boxes and retain only the best proposals.

3. O-Net (Output Network)

The Output Network (O-Net) is the final stage of the MTCNN cascade, providing the most accurate face localization and landmark detection.

Fine-Grained Refinement: O-Net processes the refined face proposals from R-Net and outputs more precise bounding box coordinates and facial landmark locations. The network has a deeper architecture and greater capacity than the previous stages, allowing it to fine-tune the face location and detect landmarks accurately.

Five Facial Landmarks: O-Net predicts the positions of five key facial landmarks — the left eye, right eye, nose, left corner of the mouth, and right corner of the mouth — which are essential for aligning faces for subsequent processing (e.g., feature extraction).

Final NMS Application: O-Net applies Non-Maximum Suppression one last time to produce the final set of non-overlapping bounding boxes around detected faces, with their corresponding landmark locations.

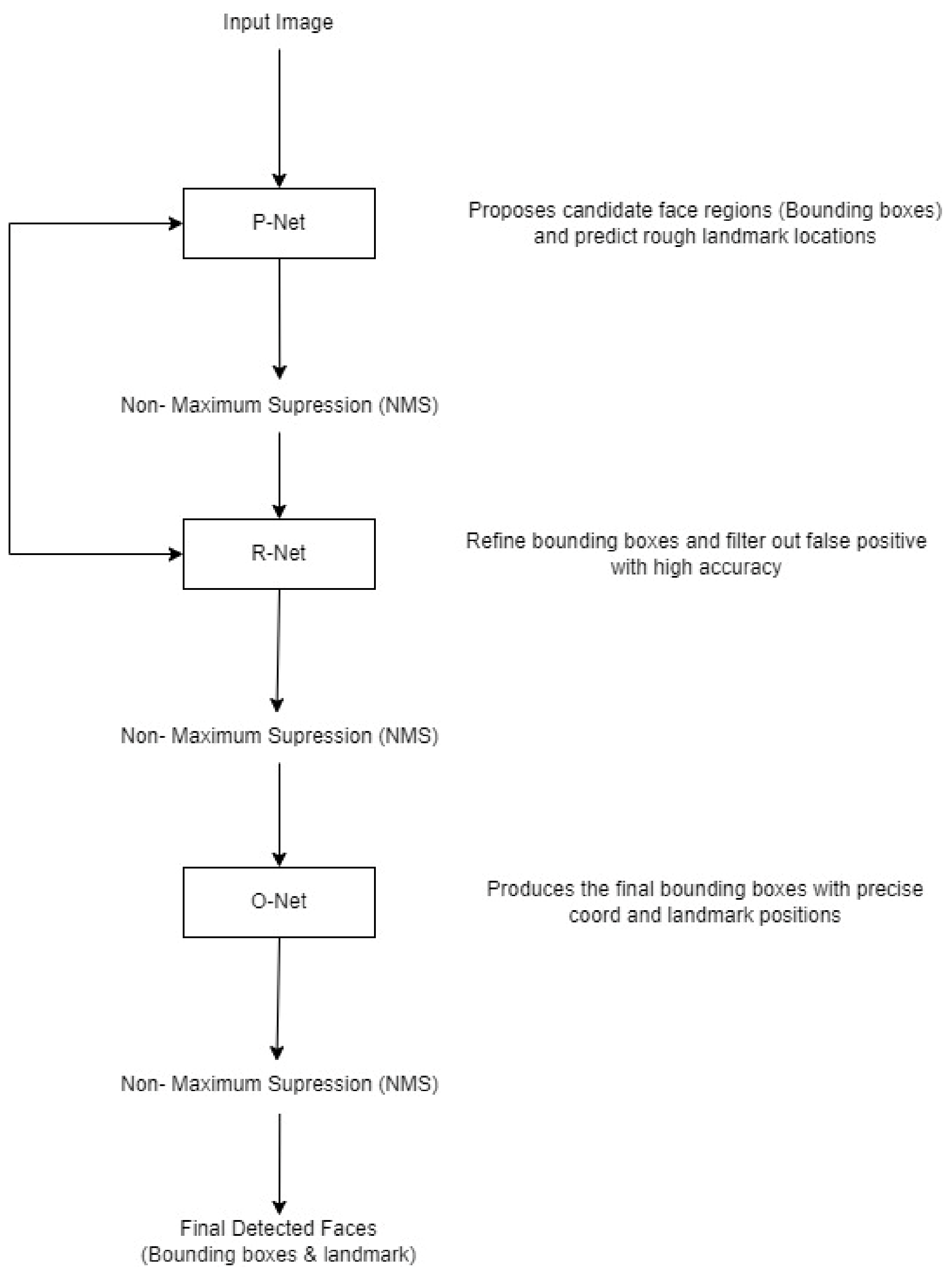

Input Image
P-Net
Proposes candidate face regions (Bounding boxes) and predict rough landmark locations
Non- Maximum Supression (NMS)
R-Net
Refine bounding boxes and filter out false positive with high accuracy
Non- Maximum Supression (NMS)
O-Net
Produces the final bounding boxes with precise coord and landmark positions
Non- Maximum Supression (NMS)
Final Detected Faces (Bounding boxes & landmark)

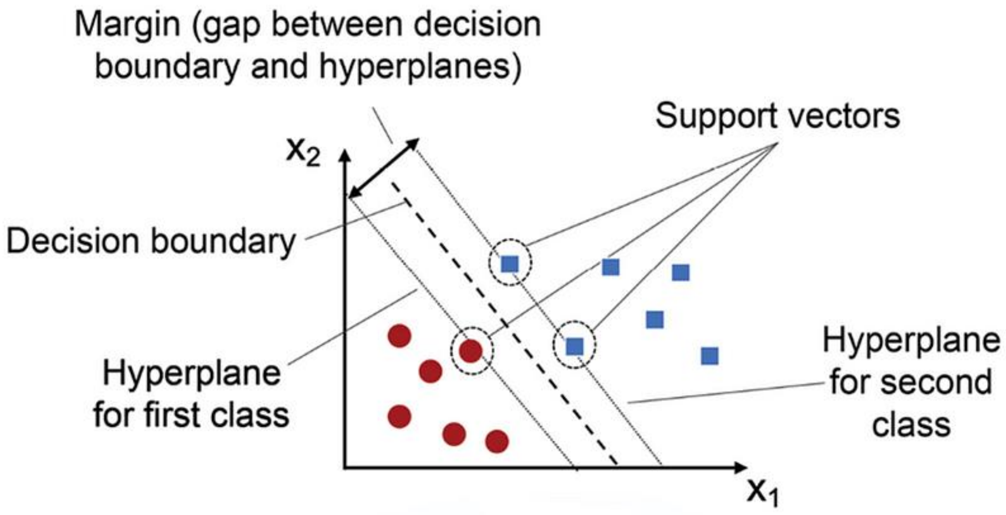
Figure-3.6: MTCNN

Support Vector Machines (SVM) in Face Recognition

SVM is a supervised learning algorithm widely used for classification tasks due to its ability to handle high-dimensional and complex data efficiently. The objective of an SVM is to find the optimal separating hyperplane that divides the data into classes with the maximum margin. For face recognition, the SVM is trained to classify face embeddings (features extracted from face images) into distinct identities.

1. The Concept of Hyperplane and Margin

- Hyperplane: A decision boundary that separates the data into different classes. In a 2D space, it is a line, while in higher dimensions, it is a plane.
- Margin: The distance between the hyperplane and the nearest data points from each class, known as support vectors. The goal of SVM is to find the hyperplane that maximizes this margin, leading to better generalization and separation of classes.


Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
Decision boundary
Hyperplane for first class
Hyperplane for second class
x_1

2. Linear vs. Non-Linear Classification

In simple cases, when the data is linearly separable, an SVM can draw a straight line (or hyperplane in higher dimensions) to separate the classes. However, face data is often non-linearly separable, which means a more complex boundary is needed to distinguish between classes (identities).

To address this, the SVM uses kernel functions, which transform the data into a higher-dimensional space where a linear hyperplane can be used to separate the classes. A commonly used kernel for face recognition is the **Radial Basis Function (RBF)** kernel, which maps data into a higher dimension, enabling non-linear separation.

3. SVM Training and Classification Process

3.1. Training Phase: The SVM is trained using labeled face embeddings (generated by a model like InceptionResnetV1). Each embedding corresponds to a face, and its label represents the identity of the person.

- The algorithm finds the optimal hyperplane that maximizes the margin between different classes (different people in this case).
- If the data is not linearly separable, the RBF kernel is applied to transform the embeddings into a higher dimension where a linear separation is possible.

3.2. Classification Phase: When a new face embedding is inputted to the SVM:

- The algorithm calculates which side of the hyperplane the embedding falls on.
- The SVM then assigns the embedding to the appropriate class (identity) based on the decision boundary.

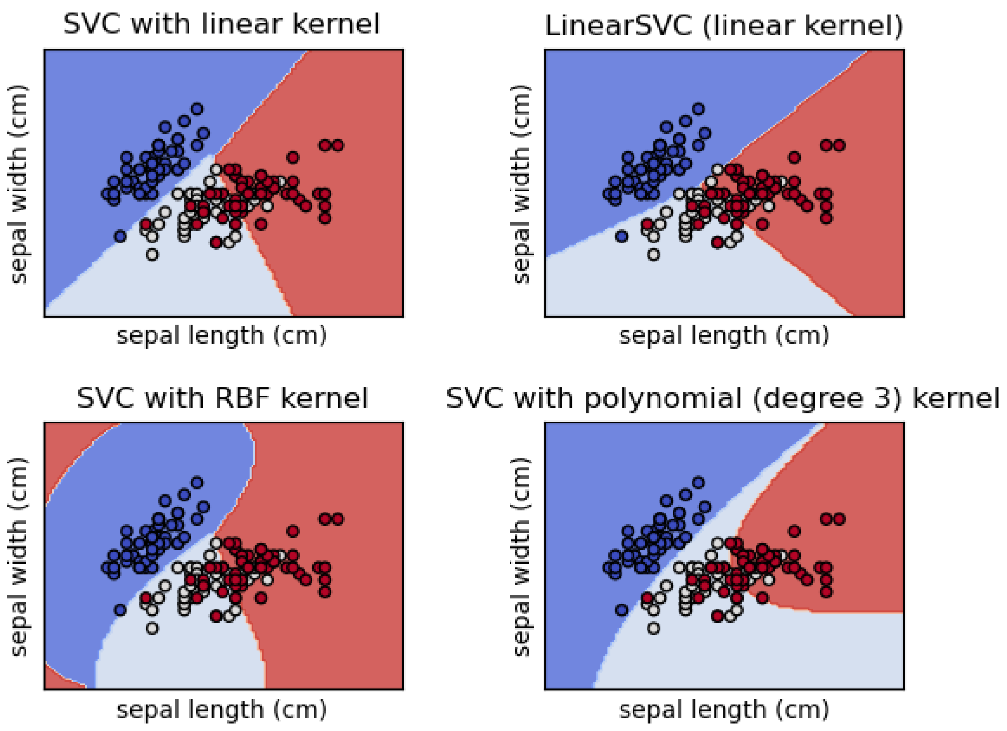


Figure-3.7: SVC(Support Vector Machine)

Popular kernel functions in SVM

The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, i.e.it converts no separable problems to separable problems. It is mostly useful in non-linear separation problems. Simply put the kernel, does some extremely complex data transformations and then finds out the process to separate the data based on the labels or outputs defined/

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

The development of the face recognition system involved using several tools and technologies for different stages of design, development, and deployment. Below is an overview of the tools and techniques used for various components of the system:

4.1.1 Tools Used

The tools used for the implementation of Face are listed below:

Draw.io

It was used for designing the system designs such as system flowchart, ER diagram, relational model, architectural design, use case diagram etc.

Python and Flask

Flask, a micro web framework written in Python, was used for developing the web application interface of the face recognition system. Flask enables rapid web development and allows easy integration of machine learning models for real-time face detection and recognition.

MTCNN (Multi-task Cascaded Convolutional Networks)

MTCNN was utilized for the face detection process. The library provides pre-trained models to efficiently detect faces and facial landmarks within images. MTCNN was chosen due to its ability to handle varying poses, expressions, and lighting conditions.

PyTorch & InceptionResnetV1

PyTorch was the deep learning framework used for developing and running the feature extraction model. InceptionResnetV1, a pre-trained deep convolutional neural network, was employed to generate facial embeddings from detected faces. These embeddings form the feature vectors that are used for classification.

Scikit-learn (SVM and CalibratedClassifierCV)

The Support Vector Machine (SVM) classifier was implemented using scikit-learn, a machine learning library in Python. The RBF kernel was chosen to handle the complex feature space of facial embeddings. Additionally, CalibratedClassifierCV was used to transform the raw SVM outputs into calibrated probabilities, providing a measure of confidence in the predictions.

SQLite

SQLite, a lightweight SQL database engine, was used for managing and storing face embeddings and user information. It provided an easy-to-use, serverless, and zero-configuration database to keep track of identities and their corresponding face embeddings.

HTML, CSS, and JavaScript

These were used to build the front-end of the web application. HTML was utilized to structure the web pages, CSS for styling and layout, and JavaScript for dynamic functionalities and client-side validations. Together, they provided an interactive and user-friendly interface for the face recognition application.

Bootstrap

Bootstrap, a popular front-end framework, was used to ensure responsive design and a clean, modern layout. Components like navbars, cards, buttons, and grids from Bootstrap were used to enhance the web application's UI and ensure compatibility across devices.

Visual Studio Code (VS Code)

All the coding, debugging, and project management were performed using Visual Studio Code, a versatile code editor that supports various programming languages and has numerous extensions for Python, HTML, and JavaScript development.

4.2 Testing

The testing of Thrift store is done by testing the unit modules like login module and sign-up module.

Table 4.1 Test Cases for Unit Testing

Test Case ID	Test Description	Input	Expected Output	Actual Output	Status (Pass/Fail)
TC-001	Face Detection on a Single Face	Image with a single face	Accurate detection of face with bounding box	Detected face accurately	Pass
TC-002	Face Detection on Multiple Faces	Image with multiple faces	Detection of all faces with individual bounding boxes	Detected all faces with individual	Pass
TC-003	Extracting Face Embeddings	Image of a detected face	Generation of a fixed-size feature vector for the face	Generated embedding of face	Pass
TC-004	Face Recognition Accuracy	Face embedding from the dataset	Correct identification of the person with a confidence score	Corrected identified the person	Pass
TC-005	Real-Time Face Recognition	Live video stream	Continuous detection and recognition of faces with minimal delay	Continuous detection and recognition of face	Pass

TC-006	Response to Unknown Face	Image of a face not in the dataset	Output "Unknown" or a low confidence score for identification	Unknown shown to unknown face	Pass

CHAPTE R 5: CONCLUSION AND FUTURE RECOMMENDATIONS

5.1 Conclusion

The face recognition system developed in this project effectively demonstrates the use of modern machine learning and deep learning techniques to detect and recognize faces in real-time. By utilizing MTCNN for face detection and InceptionResnetV1 for feature extraction, the system is able to generate robust face embeddings. These embeddings are then classified using a Support Vector Machine (SVM) with an RBF kernel, ensuring accurate identification of individuals. The web application built using Flask provides an interactive interface for testing and deployment, making the system practical for real-world applications like access control and attendance management.

The project achieves its primary objective of creating a reliable, efficient, and scalable face recognition system. The overall performance of the system, including its ability to handle different lighting conditions, multiple faces, and unknown individuals, was evaluated and validated through thorough testing. The real-time nature of the system adds to its utility in practical use cases, highlighting the effectiveness of the algorithms and tools selected for implementation.

5.2 Lessons Learned / Outcomes

Throughout the development of this project, several valuable lessons and outcomes emerged:

- **Understanding of Face Recognition Workflow**: The project provided an in-depth understanding of the face recognition workflow, from data preprocessing and face detection to feature extraction and classification.
- **Effectiveness of Deep Learning Models**: The use of deep learning models like MTCNN for detection and InceptionResnetV1 for feature extraction was key to the system's performance, proving their robustness in handling variations in pose, lighting, and facial expressions.

- **Role of SVM in Classification**: SVMs, especially with the RBF kernel, showed high accuracy in classifying complex face embeddings, reinforcing their importance as reliable classifiers for high-dimensional data.
- **Integration of Machine Learning in Web Applications**: Building a Flask-based web application provided practical experience in integrating machine learning models into web environments for real-time interaction.
- **Model Calibration for Improved Confidence**: Using **CalibratedClassifierCV** to produce probabilistic outputs for SVM classifications added an additional layer of reliability, allowing for more informed decision-making in recognition.

5.3 Future Recommendations

While the system achieved its primary goals, there are several potential areas for enhancement and future exploration:

- **Improving Generalization and Accuracy**: To further improve the system's accuracy, especially in challenging scenarios, additional training data covering more diverse faces and conditions can be used. Fine-tuning the InceptionResnetV1 model on a domain-specific dataset may also boost recognition performance.
- **Handling Occlusions and Extreme Conditions**: Incorporating techniques to handle occlusions (e.g., masks, glasses) or extreme conditions (such as poor lighting or low resolution) can further improve robustness. Methods like data augmentation, GAN-based image enhancement, or adaptive feature extraction may be helpful.
- **Scaling to Larger Databases**: For larger-scale applications, the system should be optimized for scalability. This could involve leveraging distributed storage solutions, optimizing database queries, or implementing more efficient embedding storage and retrieval mechanisms.
- **Deployment in Cloud and Mobile Environments**: To extend usability, the system can be deployed on cloud platforms for better accessibility and scalability. Additionally, optimizing the model to run on edge devices, such as mobile phones, can enable real-time face recognition on portable devices.

- **Enhanced Security and Privacy Measures**: Given the sensitivity of biometric data, future versions of the system should incorporate privacy-preserving mechanisms, such as encryption of stored embeddings, secure transmission protocols, and access control policies.
- **User Experience and Application Expansion**: The web application's UI/UX can be improved to provide more feedback to users, better visualization of results, and seamless integration into existing applications like attendance management systems or security frameworks.

REFERENCES

[1] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of human faces," *Proceedings of the IEEE*, vol. 59, no. 5, pp. 748-760, May 1971.

[2] M. A. Turk and A. P. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

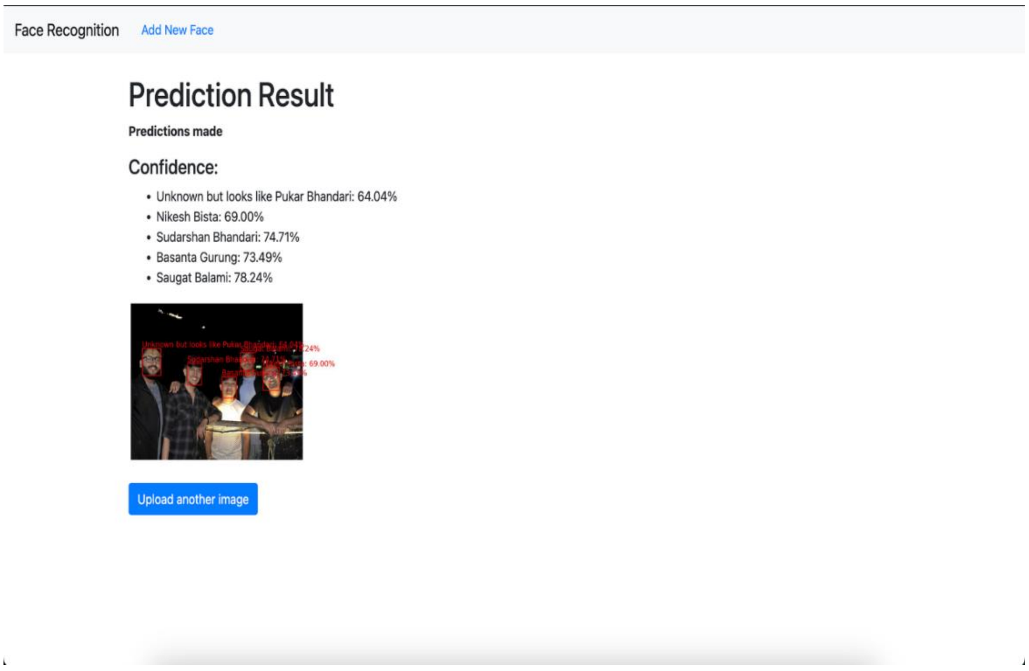
[3] G. Guo, S. Z. Li, and K. Chan, "Face Recognition by Support Vector Machines," presented at the Cambridge ORL face database, 1970. [Online]. Available: https://www.researchgate.net/publication/2427763_Face_Recognition_by_Support_Vector_Machines.

[4] R. Pramoditha, "Coding a convolutional neural network (CNN) using Keras Sequential Api," *Medium*. Available: https://towardsdatascience.com/coding-a-convolutional-neural-network-cnn-using-keras-sequential-api-ec5211126875 (accessed Jul. 29, 2023).

[5] "ResNet: The basics and 3 resnet extensions," *Datagen*. Available: https://datagen.tech/guides/computer-vision/resnet/ (accessed Jul. 29, 2023).

[6] "Python 3.11.4 documentation," *Python Software Foundation*. Available: https://docs.python.org/3/ (accessed Jan. 29, 2023).

APPENDICES

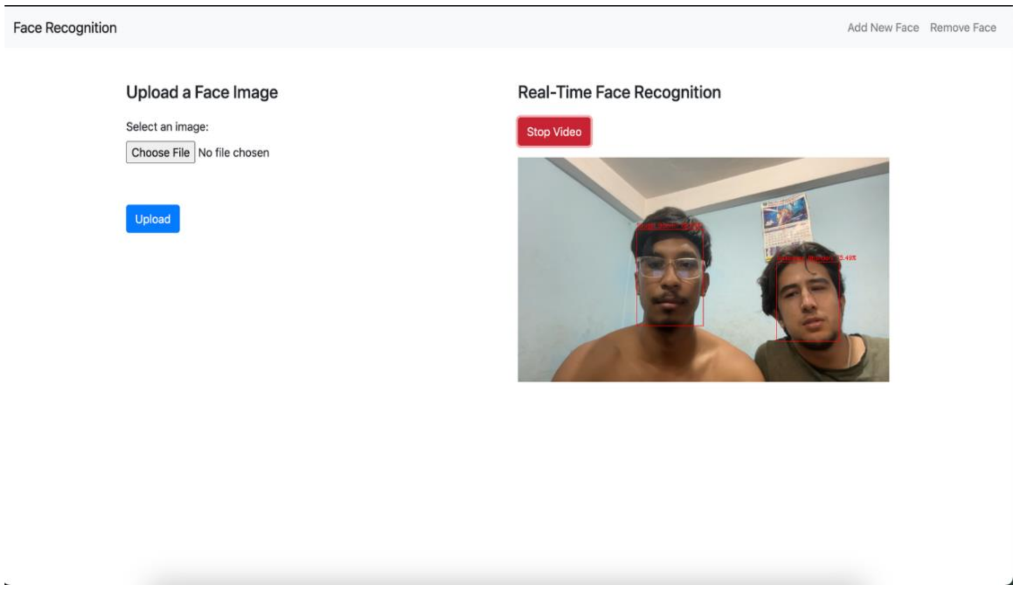


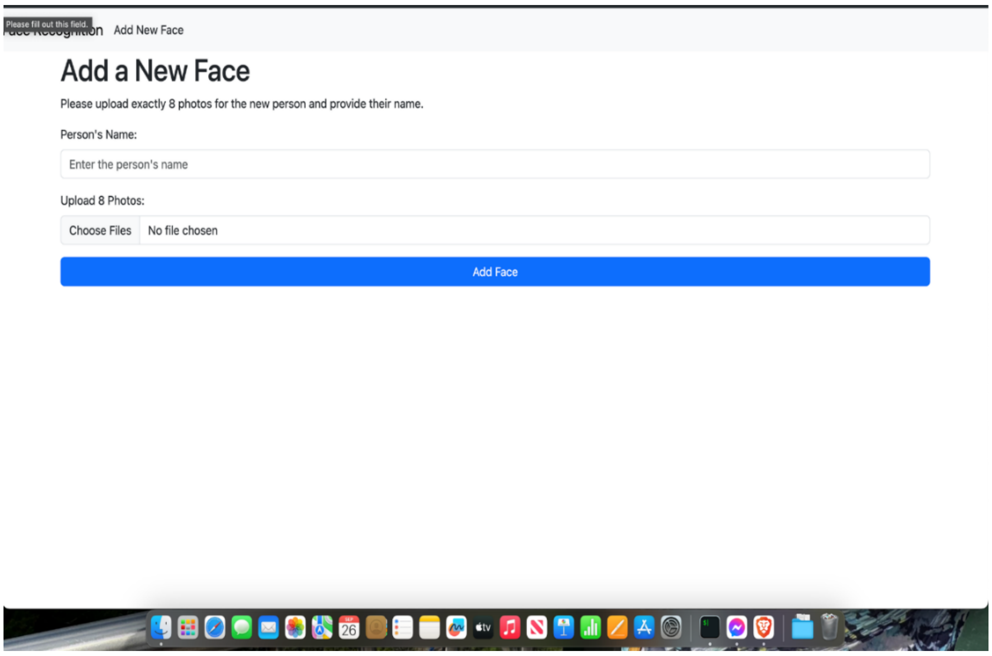
Prediction Result

Predictions made

Confidence:

- Unknown but looks like Pukar Bhandari: 64.04%
- Nikesh Bista: 69.00%
- Sudarshan Bhandari: 74.71%
- Basanta Gurung: 73.49%
- Saugat Balami: 78.24%




Please fill out this field.

Face Recognition Add New Face

Add a New Face

Please upload exactly 8 photos for the new person and provide their name.

Person's Name:

Enter the person's name

Upload 8 Photos:

Choose Files No file chosen