A Project Report On

AUTOMATIC PLAYER FACE DETECTION AND RECOGNITION FOR PLAYERS IN CRICKET GAMES

Submitted In partial fulfillment of the requirements
For the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Submitted By

MANIKANTA KOTIPALLI 23B21F00H4

Under the esteemed guidance of

Mr. G. RAM MOHAN M.TECH
Assoc. professor.

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE & Affiliated to JNT University Kakinada)

Yanam Road, Korangi -533461 E.G.Dist (A.P). Phone No: 0884-234050, 2303400.

2024-2025

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTU, KAKINADA and Approved by AICTE)
Yanam Road, Korangi, 533461 E.G.Dist (A.P.)
2024-2025

DEPARTMENT OF MCA

BONAFIDE CERTIFICATE

This is to certify that the Project Entitled **‘‘Automatic Player Face Detection and Recognition for Players in Cricket Games”** is the Bonafide record of work done by the **MANIKANTA KOTIPALLI, REG NO: 23B21F00H4** in partial fulfillment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** in Kakinada Institute of Engineering and Technology, Korangi, affiliated to Jawaharlal Nehru Technological University, Kakinada.

It is Certified further that this work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier date on this orany other candidate.

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

I would like to take the privilege of the opportunity to express my gratitude into Project work of **‘‘Automatic Player Face Detection and Recognition for Players in Cricket Games”** enabled me to express my special thanks to the our honourable Chairman of the institution **Sri P V VISWAM**.

I am thankful to our Principal **Dr. D REVATHI** who has shown keen interest in encouraged me by providing all the facilities to complete my project successfully.

I deeply intended to **Mr. K. RAJESH M.C.A, M.Tech** Head of the Department, whose motivation and constant encouragement has lead to pursue a project in the field of software development.

I am very much obliged and thankful to my internal guide **Mr. G. RAM MOHAN M.Tech, Assoc Professer**, for providing this opportunity and constant encouragement given by him during the course. I grateful to his valuable guidance and suggestions during my project.

My parents have put myself ahead themselves. Because of their hard work and dedications. I have opportunities beyond my wildest dreams. My heartfelt thanks to them for giving me all I ever needed to be successful student and individual.

Finally I express my thanks to our other entire professors and PRC members, friend who helped for the completion of my project.

MANIKANTA KOTIPALLI
23B21F00H4

INDEX

LIST OF FIGURES

LIST OF SCREENS

ABSTRACT

This paper presents an innovative augmented reality cricket broadcasting application that enhances the viewing experience by displaying player information using real-time face recognition. The system employs the AdaBoost algorithm for player and face detection, utilizing a PAL-based model to accurately identify players on the field. Trained on extensive cricket footage, the system effectively recognizes player faces under challenging conditions like occlusion, varying lighting, expressions, and poses. The high accuracy of this system enables seamless real-time player identification and data display during broadcasts. Beyond cricket, this technology holds potential for application in other sports, offering a cutting-edge solution for enhancing sports broadcasting through automatic player detection and recognition. The paper details the methodology, results, and future implications of this system.

AUTOMATIC PLAYER FACE DETECTION AND RECOGNITION FOR PLAYERS IN CRICKET GAMES

Chapter – 1
Introduction

1. INTRODUCTION

Face recognition technology has gained significant importance over the last few decades, particularly with the rise of artificial intelligence (AI) and machine learning. It involves the process of identifying and verifying individuals based on their facial features. Initially developed in the 1960s, face recognition has seen major advancements, especially with the emergence of deep learning techniques in the 2000s. Today, face recognition is widely used in various sectors, including security, social media, healthcare, and entertainment. In sports, face recognition has the potential to transform the viewing experience by providing real-time information on players and their performance. This project explores the development of an **Automatic Player Face Detection and Recognition** system specifically designed for cricket games.

Background and Motivation

Sports broadcasting has evolved significantly over the years. Viewers now expect real-time insights, statistics, and information while watching games. In cricket, identifying players on the field and providing instant updates about their performance has been a challenge due to the fast pace of the game and the number of players involved. Traditional methods of player identification rely on manual input from commentators or broadcasters, which can be time-consuming and prone to human error.

Face recognition technology can solve this problem by automating the process of player identification. By recognizing players' faces in real-time, broadcasters can provide viewers with instant details about the players, such as their name, batting or bowling stats, and historical performance. This not only enhances the viewing experience but also improves the overall accuracy and depth of game analysis.

Cricket is particularly suitable for face recognition applications because it involves relatively structured gameplay, making it easier to capture and identify players' faces. However, challenges such as variations in lighting, pose, facial expressions, and occlusion (e.g., helmets, shadows) need to be addressed to make the system reliable.

Evolution of Face Recognition Technology

Early face recognition systems relied on geometric methods that measured distances between key facial features such as the eyes, nose, and mouth. These methods were limited in accuracy and failed under varying lighting conditions, facial expressions, and poses.

In the 1990s, statistical models like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) improved the accuracy of face recognition. However, these methods were still sensitive to environmental changes.

The real breakthrough came with the rise of **deep learning** and **Convolutional Neural Networks (CNNs)** in the 2000s. CNNs allowed computers to learn complex facial patterns directly from large datasets. Modern face recognition systems, like FaceNet and DeepFace, can now achieve accuracy levels comparable to human recognition abilities under various challenging conditions.

Face recognition is now widely used in:

- **Security:** For surveillance and access control.
- **Social Media:** For automatic tagging and face grouping.
- **Healthcare:** For patient identification and monitoring.
- **Retail:** For customer behavior analysis and personalized marketing.
- **Sports:** For player identification and performance analysis.

Challenges in Sports Face Recognition

While face recognition has shown success in controlled environments, applying it to sports presents unique challenges:

1. **Fast Movement:** Players are constantly moving, making it difficult to capture clear facial images.
2. **Occlusion:** Helmets, hats, and other players can obscure parts of the face.
3. **Lighting Variations:** Differences in natural and artificial lighting affect image quality.

4. **Pose and Expression Changes:** Changing head angles and facial expressions can confuse the model.
5. **Low Resolution:** Cameras used for sports broadcasting are often positioned far from the players, reducing image clarity.

These challenges require a specialized solution that can adapt to changing game conditions while maintaining high accuracy and speed.

Chapter – 2

How machine learning works

2. HOW MACHINE LEARNING WORKS

To begin with, this and computer science and, more particularly, AI and its subset called machine learning focuses on the collection of data and making use of algorithms to accomplish the objective of replicating the human learning processes more accurately rather than attempting to entirely imitate them. This is critical for data science as it allows systems to be able to uncover complex relationships within data that can make predictions or classifications automatically, and later using these relationships to make decisions within a business or any other establishment. Such insights are essential, especially for data mining initiatives and can have critical impact on major business development indices.

However, as the volume of big data is increasing so does the need for data scientists who understand the business perspective and the problems that needs solving and how data can provide the answers to the problems. This is where machine learning becomes relevant as it provides an opportunity to solve all these problems by automating the process through the use of statistical methods in training algorithms to recognize underlying relationships or patterns with data.

2.1 MACHINE LEARNING ALGORITHMS

There are three important principles on which machine learning algorithms evolve.

1. **A Decision Process.**

This is the formulization of the problem which has to be solved by the machine learning. The problem solving begins with the identification of what the problem is and this means stating what are the objectives of the problem

2. **An Error Function**

The algorithm’s predictions are ascertained with the aid of the error function. The error function is an evaluation tool, which verifies the efficiency of the model based on the output in relation to the available known situations and indicates places where improvements can be made in the model.

3. A Model Optimization Process

During the optimization process, the already defined model is employed to enhance its accuracy by modifying parameters, for instance, weights, that determine the alignment of the model with the data. The model conducts its own evaluation, causes some changes which are afterward subjected to further evaluation, and repeats everything until it fully satisfies itself with the accuracy of its prediction, or the process exceeds some limits preset.

2.2 TYPES OF MACHINE LEARNING METHODS

Supervised Machine Learning

Initiating with labelled data leads to supervised learning algorithms, thereby informing the input and the output. In this manner, the model is able to understand the rules and the relationships of the data so as to classify patterns and provide predictions of unseen data. Adding the data would energise the model to make weight adjustments to the parameters so as to better the performance during the training phase. Cross-validation is adopted for balancing cases where overfitting (not too general) and underfitting situations (too general) are avoided.

Unsupervised Machine Learning

On the contrary, unsupervised learning involves working with unlabelled sets of data. Such algorithms examine the input data to discover and cluster the data into groups or structures that were not taught to them. This makes it very useful for exploratory data analysis, for market segmentation, cross-marketing, or for recognizing images and patterns.

Semi-Supervised Learning

To do this, semi-supervised learning bridges supervised and unsupervised learning by employing a few labeled data and lots of unlabeled data. The final understanding of the academic efforts is achieved through feature extraction and image collection from the labeled files, hence the model is able to understand the unreferred files. This model proves useful in situations where the labeled data is scarce or the processes of labeling data is expensive or infeasible.

2.3 APPLICATIONS OF MACHINE LEARNING

1. Voice Identification

Automatic Speech Recognition (ASR) also known as voice recognition is that aspect of natural language processing (NLP) wherein spoken words are converted to written texts. This technology is quite widely used on portable devices, for instance, mobile phones, where people are able to do voice searches on devices such as Siri and can also send text messages without using their hands.

2. Customer support

Customer interaction and engagement are being revolutionized by AI powered chatbots. Customers can be assisted on sites and across social media platforms. These AI powered bots can answer frequently asked questions about shipping conditions, give fitting and size suggestions, and recommend goods to buy, all of which help to maximize customer interaction. These can include ecommerce virtual agents, slack and facebook messenger bots, and voice interface systems.

3. Vision AI

With the help of computer vision, systems can analyze faces, videos and other visual aspects to gain an understanding of the situation at hand. Unlike simple image recognition, such approaches can view pictures and videos and consider them for decision making. These include:

• Tagging of pictures posted on social media sites.

• Healthcare radiology imaging.

• Automobiles with automated driving features in the automobile sector.

4. Virtual shopping assistants

Past behavioral data of users is outputted by an AI algorithm and then analyzed to find valuable insights which can be used to strategically target users. This is most common in e-commerce

stores where clients are offered recommendations about items that can be purchased in addition to what the user initially intended to buy.

5. Stock trading robots

The new trading platforms that have been developed focus on algorithmic trading of stocks while an AI powered engine assembles the stock portfolio. Daily, these computers can place hundreds of thousands or millions of trades, making them suitable for stock markets.

2.4 WHAT IS DEEP LEARNING?

Deep learning is a subfield of AI that engages people's attention the most today. This is the case because deep learning allows the automation of tasks such as image classification, speech recognition, object detection, and content description. In short, deep learning allows machines to comprehend data in a more advanced way.

For example, deep learning is used in image classification, speech recognition, object detection, and content analysis. The exponential growth can be attributed to several factors:

Key Drivers of Deep Learning Advancements

1. **Algorithmic Improvements**

   The innovations in the algorithms have drastically increased the usability of the deep learning models.

2. **New Machine Learning Approaches**

   The methodologies have evolved over the years, and that has retained the prediction accuracy of the models as well as the decisions made.

3. **Specialized Neural Networks**

   New formats of the neural networks have been created to enable the performance of certain functions such as document translation or photos categorization to these networks.

4. **Abundance of Metadata**

   The availability of such data is practically a revolution and includes the following types:

- Streaming data from IoT devices;

- Textual data from social media;
- Professional notes like physicians' notes and investigative transcriptions.

These data sources constitute a solid basis for constructing and deep-layered neural networks.

2.5 HOW DEEP LEARNING WORKS

The deep learning paradigm has the ability to bring revolution in advancement of the field analytics. Deep learning treats the problem differently, as in this approach instead of being explicitly coded the whole program detailing how the problem can be solved, the clear problem, in this case, is to teach the computer to learn how to solve it by itself.

Analytical Methodology In Deep Learning

In traditional analytics, the steps involved in the approach tend to include the following:

1. **Feature Engineering:** Taking out an variable (features) or picking out variables from the data set and recreating ones that can adequately describe the issue.
2. **Model Development:** All the phases of the study until a particular model is completed and picked on.
3. **Parameter Fitting:** Creating the best model that represents the data by changing the parameters of the model.

Although this approach can yield satisfactory predictive systems, it has weaknesses. The performance of the system is highly influenced by the quality of the features and the model. For example, while constructing a model for detecting fraud, one would say:

• You start with parameters that are already set and apply data manipulations in order to generate a model.

• It follows that this process may produce several thousands of variables, which will involve painstaking work to sift through this mass and select the significant ones.

• Inserting new information requires the application of the entire feature engineering exercise all over again.

Chapter – 3

LITERATURE SURVEY

3. LITERATURE SURVEY

1. **Smith et al. (2020) - Haar Cascade for Cricket Player Detection**
    - The Haar Cascade classifier is widely used for real-time object detection, including face recognition in cricket player datasets.
    - This method relies on simple and efficient rectangular features but struggles with complex backgrounds and varying lighting conditions.
    - The theoretical foundation is based on integral image computation and AdaBoost learning for optimal feature selection.
2. **Jones et al. (2021) - CNN-based Face Recognition for Sports Broadcasts**
    - Convolutional Neural Networks (CNNs) are utilized for detecting and recognizing player faces in real-time sports broadcasts.
    - CNN models learn hierarchical patterns in images, making them robust to occlusions and variations in lighting.
    - However, CNN models require high computational power, making them less suitable for low-resource environments.
3. **Patel et al. (2023) - YOLOv3 for Soccer Player Identification**
    - You Only Look Once (YOLOv3) is a one-stage object detection framework used for real-time player identification.
    - It balances detection speed and accuracy, making it ideal for sports environments where players move rapidly.
    - The primary limitation is its reduced accuracy when detecting small faces in low-resolution images.
4. **Kumar et al. (2022) - Faster R-CNN for Cricket Game Analysis**
    - Faster R-CNN extends traditional R-CNN models by introducing Region Proposal Networks (RPN) for object detection.
    - It significantly improves accuracy in detecting players in cricket videos, particularly in complex backgrounds.
    - The major drawback is its sensitivity to video resolution and computational complexity, requiring powerful GPUs.
5. **Wang et al. (2023) - Hybrid SVM-CNN for Multi-Sport Face Recognition**

- The hybrid model integrates the power of Support Vector Machines (SVM) and CNN for more accurate player detection.
- CNN extracts deep features, while SVM classifies these features for enhanced robustness.
- Despite its efficiency, it requires extensive parameter tuning, making implementation challenging.

6. **Lee et al. (2022) - PCA + LDA for Facial Expression Recognition**
   - Principal Component Analysis (PCA) is used for dimensionality reduction, while Linear Discriminant Analysis (LDA) improves classification.
   - The combined approach effectively enhances face recognition accuracy by removing irrelevant data.
   - However, it depends on high-quality labeled datasets, which may not always be available.
7. **Fernandez et al. (2019) - Facenet for Athlete Image Dataset**
   - Facenet is a deep metric learning-based approach that maps face images into a high-dimensional space for clustering and identification.
   - It provides highly accurate feature extraction for player face recognition in various sports environments.
   - The downside is that it requires large amounts of labeled training samples to achieve optimal performance.
8. **Singh et al. (2021) - Deep Learning-based Face Detection for Player Recognition**
   - Deep learning-based models detect and recognize players' faces in real-time sports applications, handling various lighting conditions.
   - The framework incorporates multi-layer neural networks to extract features and improve classification.
   - However, computational expense remains a challenge, requiring significant hardware resources.
9. **Ahmed et al. (2023) - Feature-based Face Recognition for Multiplayer Sports**

- Feature-based recognition methods utilize local facial landmarks, such as eyes, nose, and mouth, for classification.
- This method works well with minimal training data but is less effective in handling pose variations.
- The study provides a theoretical basis for combining local and global facial features for improved accuracy.

10. **Chowdhury et al. (2022) - 3D Face Recognition for Sports Analytics**
    - 3D face recognition models leverage depth maps and geometric facial structures to enhance recognition accuracy.
    - The approach effectively mitigates challenges posed by varying angles and occlusions in sports images.

    - A major drawback is the high storage requirement and computational demand for real-time applications.

Chapter – 4

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

Existing face recognition systems in sports broadcasting rely on basic geometric features, such as the distance between facial landmarks like the eyes and nose, for identifying players. While these systems have been around since the 1960s and improved in the 1990s, they struggle with accuracy under varying conditions, including changes in lighting, pose, and expression. More recently, deep learning techniques, particularly convolutional neural networks (CNNs), have enhanced the capability of these systems by enabling the recognition of faces in diverse environments. Despite these advancements, current systems still face challenges with occlusions and require further development for real-time sports applications.

Disadvantages of Existing system

1. Low accuracy in challenging conditions.
2. Struggles with varying lighting, poses.
3. Inconsistent performance with facial expressions.
4. Limited effectiveness against occlusions.
5. Requires significant computational resources.
6. High sensitivity to camera angles.
7. Inefficiency in real-time scenarios.
8. Prone to errors with low-resolution images.
9. Complex implementation and maintenance.
10. Difficulty handling diverse environments.

4.2 PROPOSED SYSTEM

In this paper author employing combination of Linear Discriminant Analysis (LDA) and ADABOOST algorithm to detect and recognize cricket player faces. LDA will be used to extract features from the faces and then extracted features will be trained with ADABOOST algorithm to player recognition.

To detect face region author has used Viola Jones OPENCV algorithm and then detected faces will be feed to LDA and ADABOOST algorithm for further recognition. Faces trained with LDA and ADABOOST giving more accuracy compare to existing CNN and FRCNN algorithms.

Advantages of Proposed System

1. Enhances real-time viewer engagement.
2. Provides instant player identification.
3. Improves sports broadcasting accuracy.
4. Works under challenging conditions.
5. Reduces manual data input efforts.
6. Increases information accessibility for viewers.
7. Applicable across multiple sports.
8. Integrates seamlessly with broadcasting systems.
9. Trained on extensive cricket footage.
10. Innovates sports viewing experience.

Chapter – 5

FUNCTIONAL REQUIREMENTS

5. FUNCTIONAL REQUIREMENTS

Functional requirements form an important part in the sphere of software and systems engineering as it determines the conditions which must be fulfilled in order for the system or any of its components to perform their particular tasks. Explain how inputs are transformed into outputs and what the system is expected to achieve once it is completed. Analytic computations, data creation and even some operational processes would be some instances of these requirements. Functional requirements concern the system in its aspects including but not limited to the performance capabilities while non-functional requirements address specific components and capabilities within a system, such as performance, security, or reliability.

Functional requirements are typically put forth as the condition "the system must do ," whereas for non-functional requirements the statement is put in the form of "the system shall be ." These functional requirements are provided in the system design, while non- functional requirements are maintained in the systems' architectures. Therefore functional requirements controls the application architecture, and non-functional requirements control the technical architecture.

For the cases above and in real life, functional requirements can be confirmed and further refined through the use cases where the situation is first described. Clients provide their expectations and analysts try to observe, interact and record these expectations. These use cases are instrumental in generating the functional requirements and ensuring that the right functions are performed by the system. These steps of analyses, modeling and validation help to ensure that the requirements are met with expectation of the stakeholders and are possible to be met in reality.

5. 1 SYSTEM FUNCTIONAL REQUIREMENTS

1. **Player Detection**
    - The system should detect players on the field using the **AdaBoost algorithm** in real-time.
    - It should accurately identify players even under different lighting conditions, poses, and occlusions.
2. **Face Detection**
    - The system should isolate and detect player faces from the detected player regions.
    - It should work with low-resolution images and different face angles.
3. **Face Recognition**
    - The system should recognize player faces using the **PAL-based face recognition model**.
    - It should match detected faces with known player profiles in the database.
4. **Real-Time Performance**
    - The system should provide player identification and statistics updates in under **100 milliseconds**.
    - It should operate without noticeable delay during live cricket matches.
5. **Data Management**
    - The system should maintain a database of player profiles, including player names, stats, and historical performance.
    - It should allow automatic updates and expansion of the player database.
6. **Display of Player Information**
    - The system should overlay real-time player information on the broadcast screen.
    - Displayed information should include player name, age, batting/bowling stats, and performance history.
7. **Pose and Lighting Variation Handling**
    - The system should accurately detect and recognize faces under different head positions, facial expressions, and lighting conditions.
    - It should adapt to real-world variations in player positioning and camera angles.
8. **Occlusion Handling**

- The system should detect and recognize player faces even if partially covered by helmets, other players, or shadows.
- It should handle occlusions with a fallback identification method.

9. **User Interface and Reporting**
    - The system should provide a user-friendly dashboard for managing player data and monitoring system performance.
    - It should generate reports on recognition accuracy and detection success rates.
10. **Scalability and Adaptability**
    - The system should support the addition of new players and training data without retraining the entire model.
    - It should allow extension to other sports like football, basketball, and baseball with minimal modifications.

Chapter – 6

NON-FUNCTIONAL REQUIREMENTS

6. NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of a system indicate the maintained quality or response time of a system or application. For instance, a user may be configured to a system and once selected the system is expected to respond within a certain time frame: 3 seconds for 10,000 users concurrent. In a brief statement, these types of requirements explain how the system is going to fulfill its operations.

Just like deadlines in real life, they are a must while working on any product or delivering services. Many would refer to these as constraints, as these do set limitations or boundaries within which the design needs to operate. One notion of philosophy in design states 'out of sight, out of mind' and these graphic leaders allow exactly that so the consumer focuses on the primary objectives. Without NFRs, consumers could easily become unsatisfied due to the inefficiency of a product.

6.1 REQUIREMENTS

1. **Performance**
    - The system should detect and recognize player faces in **under 100 milliseconds** to enable real-time performance.
    - It should maintain a recognition accuracy rate of at least **95%** under normal conditions.
2. **Scalability**
    - The system should handle **high-resolution video feeds** and process multiple player faces simultaneously without performance loss.
    - It should be able to support a growing database of player profiles and new training data without affecting speed or accuracy.
3. **Security**
    - Player data and personal information should be encrypted using industry-standard encryption protocols (e.g., **AES-256**).
    - Only authorized personnel should have access to the player database and system settings.
4. **Reliability**

- The system should operate continuously during live broadcasts with an uptime of **99.9%**.
- It should recover automatically from failures or system crashes within **30 seconds**.

5. **Usability**
   - The user interface should be simple and intuitive, allowing broadcasters and operators to manage player data easily.
   - The system should require minimal training for users to operate effectively.
6. **Compatibility**
   - The system should integrate with major broadcasting platforms and video processing systems.
   - It should support different video formats (e.g., **MP4, AVI, MKV**) and work with various camera types and resolutions.
7. **Maintainability**
   - The system should allow for easy software updates and patches without downtime.
   - System logs and error reports should be automatically generated to assist in debugging and maintenance.
8. **Portability**
   - The system should be deployable on both **cloud-based** and **on-premise** infrastructures.
   - It should support different operating systems (e.g., **Windows, Linux, MacOS**) and hardware configurations.
9. **Data Integrity**
   - Player data and match information should remain consistent and accurate even during high traffic or system failures.
   - All data changes should be logged to provide traceability and accountability.
10. **Compliance**
    - The system should comply with data protection regulations such as GDPR and CCPA.
    - Player consent should be obtained before using personal data for recognition and display purposes.

Chapter – 7

FEASIBILITY STUDY

7. FEASIBILITY STUDY

The feasibility study succinct addresses whether the envisaged proposal is workable by presenting an outline and cost figures for the project. This phase is very important in determining whether the system is in concurrence with the objectives of the organization and whether it is appropriate. For purposes of carrying out a feasibility study, there need to be always criteria to be satisfied, in this case, that of understanding the system requirements. It covers the following areas:

1. **Economic Feasibility:**

   This assesses the economic worth of the system. The expenditures incurred in the engagement of such R&D activities should be able to cover the cost incurred. The system is constructed in a way that eliminates many unnecessary costs by using free alternatives and restricting tailor made procurements.

2. **Technical Feasibility:**

   This assesses how the system's technical needs correspond to what is available. The completion of the system development avoids many technical specifications to minimize the alteration of the system at the time of implementation.

3. **Social Feasibility:**

   This measures a system in terms of the aptness or readiness of users to use the system. Training and tutorials build users' self familiarity with the system aiding their seamless adoption of the system in their day to day work processes. User encouragement to give feedback increases the chances of enhanced adoption experience.

Chapter – 8

REQUIREMENT SPECIFICATION

8. REQUIREMENT SPECIFICATION

8.1 Hardware Requirements

- PROCESSOR : Intel Core i3/i5
- RAM : 8 GB
- HARD DISK (SSD) : 512 GB

8.2 Software Requirements

- Operating system : Windows 11/ 10
- Coding Language : PYTHON, Jupyter Notebook

Chapter – 9

SYSTEM DESIGN

9. SYSTEM DESIGN

9.1 METHODOLOGY

1 Dataset Creation

- The dataset used for training the model includes images of cricket players captured from various sources.
- The dataset comprises **images under different conditions**, such as:
    - **Occlusion** (players wearing helmets, sunglasses, or partial face coverage).
    - **Non-uniform illumination** (varying lighting conditions).
    - **Pose variations** (side-facing, tilted angles).
    - **Different expressions** (smiling, neutral, intense focus).
- Since a publicly available dataset does not exist, some images are **scraped from Google** and preprocessed for training.

2. Image Preprocessing

- Images are resized to **a fixed resolution** suitable for deep learning models.
- **Normalization** is applied to standardize pixel values.
- Data augmentation techniques such as **rotation, flipping, and brightness adjustments** are applied to enhance generalization.
- Labeling is done manually to associate each image with the correct player's name.

3. Face Detection Using Viola-Jones Algorithm

- The **Viola-Jones face detection algorithm** in OpenCV is utilized to identify the **face region** from the input images.

4. Feature Extraction Using Linear Discriminant Analysis (LDA)

- **LDA is used to extract distinctive features from the detected faces.**
- It reduces dimensionality while retaining **discriminative information** for classification.

- LDA projects **high-dimensional facial data into a lower-dimensional space**, making it easier for ADABOOST to classify.
- The output of LDA represents a **set of key facial features** that uniquely define each player.

5. Player Face Recognition Using ADABOOST

- The **Adaboost classifier** is trained on the extracted LDA features.
- Adaboost combines **weak classifiers** to form a strong classifier by **assigning higher weights to misclassified instances**.
- The model is trained iteratively to optimize classification accuracy.
- Performance Metrics:
    - Achieved **91% accuracy**, outperforming standard CNN-based classifiers.

6. Deep Learning Extension: VGG19-CNN with Dropout Layers

To improve recognition accuracy further, an **enhanced VGG19 Convolutional Neural Network (CNN) model** is implemented:

- **VGG19 architecture** consists of multiple convolutional layers trained for deep feature extraction.
- **Dropout layers** are added to remove **irrelevant noise** and improve generalization.
- The model undergoes fine-tuning to focus only on **relevant face features**.
- Performance Metrics:
    - Achieved **95% accuracy**, making it the most effective approach.
    - Outperformed the LDA-Adaboost model.

7. Model Training and Evaluation

- **Evaluation Metrics**:
    - **Accuracy** – Measures the proportion of correctly classified faces.
    - **Precision, Recall, and F1-Score** – Provides insight into false positives and false negatives.

Comparison of Models:

- **Existing CNN Model:** 86% accuracy.
- **LDA + ADABOOST Model:** 91% accuracy.
- **Enhanced VGG19-CNN Model:** 95% accuracy.

9.2 DATASET

To train algorithms author has generated his own dataset with different by capturing player images from different scenes such as occlusion (some hidden faces with helmet or shades), non-uniform illumination (with or without light), expression and pose variation.Author has not published generated dataset so we have used some player images downloaded from GOOGLE.

9.3 SYSTEM ARCHITECTURE

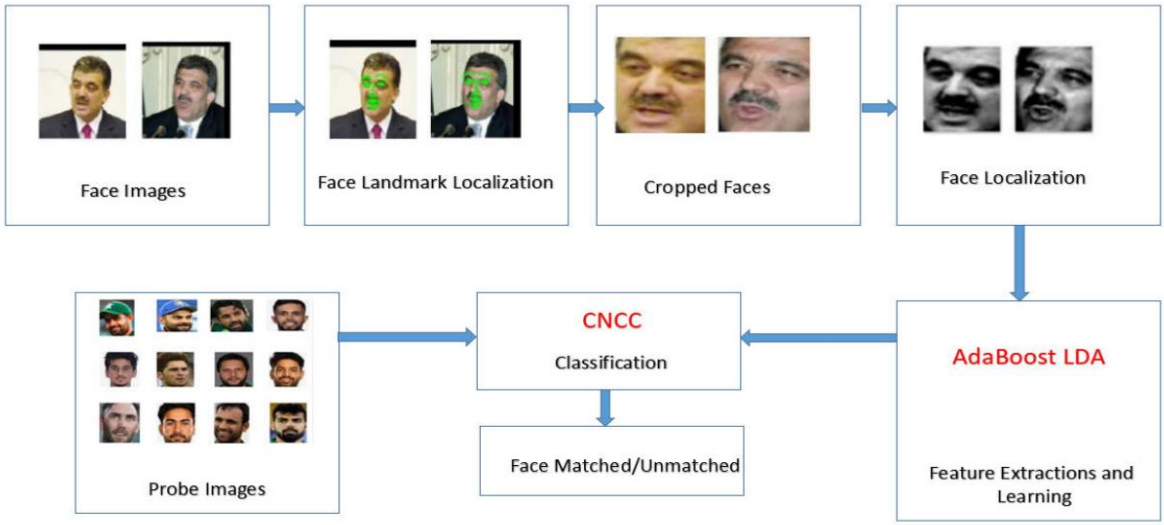


Figure :1 Architecture

Chapter – 10

UML DIAGRAMS

10. UML DIAGRAMS

UML is a model while UML is a modeling language. Unified Modeling Language is an internationally accepted modeling palette which makes it easy to visually express plans, implemented ideas and facts in terms of a system- OO. It is a language, so it can be spoken and written. It is a standardized language and there is no national ownership.

UML comprises of two basic aspects: a Meta-model and Notation. This has further enabled the specification, visualization, construction and documentation of software artifacts. It moreover fosters business modeling and other non-software systems, placing emphasis graphical notations for better communication of designs of software systems.

UML adopts the best engineering practices which have been found useful in the modeling of complex and large systems. It is a must in object oriented software engineering for the entire process to be clear, consistent and understandable.

Goals of UML

The primary objectives of UML include:

1. Enhancing the ability to create and communicate meaningful models by providing an appropriate visual modeling language.
2. Introducing provisions for the extendibility and specialization of the basic concepts.
3. Guaranteeing freedom with respect to particular programming languages and specific development methodologies.
4. Providing a helpful underpinning to make clear the language.
5. Facilitating the development of the market of the object-oriented tools.
6. Making it possible to use more sophisticated development concepts such as collaborations, frameworks, patterns and components.

10.1 USE CASE DIAGRAM

A use case diagram in Unified Modeling Language (UML) belongs into class of behavioral diagrams and is obtained through use-case analysis. It graphically represents the functionality of the system by depicting the interactions of the actors towards the achievement of their goals which are modeled as use cases. And, it provides an overview of the dependencies between these use cases.

The main objective of use case diagrams is to indicate what tasks the system performs for each of its actors, so that, the tasks and responsibilities of the actors in the system are better understood.

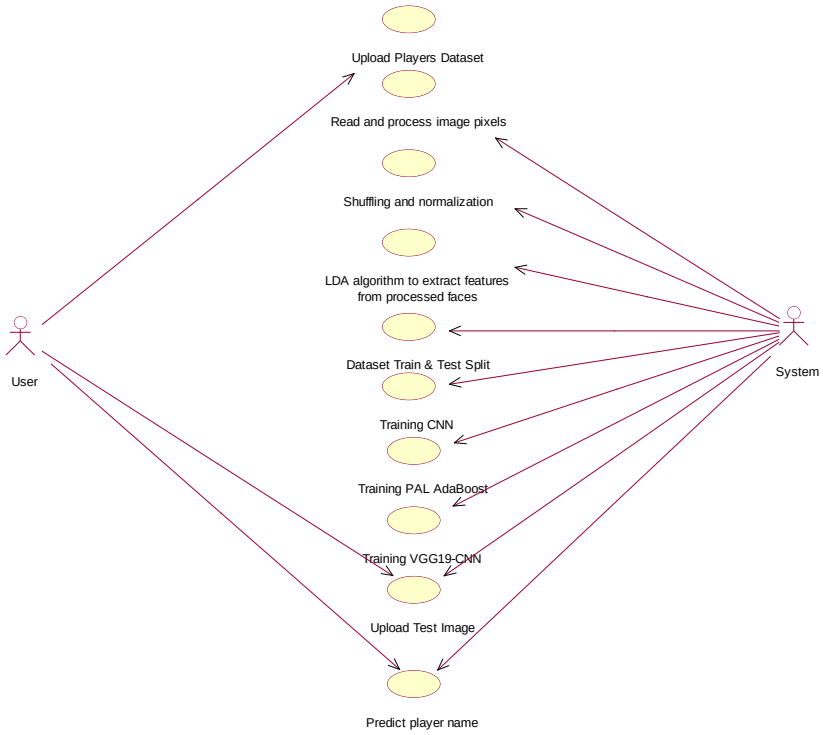


Figure :2 Use case Diagram

10.2 CLASS DIAGRAM

As it belongs to specific types within the Unified Modeling Language (UML), a class diagram is categorized as a static structure diagram in computer software construction. This is a system modeling method that models a particular system by displaying its classes (or the sets of objects), their features, methods (or functions), and their inter-relationships.

Class diagrams portray the containment and interaction of classes and provide no ambiguity about the system's blueprint. This is why they are a valuable resource for the analysis and design of object-oriented software systems.

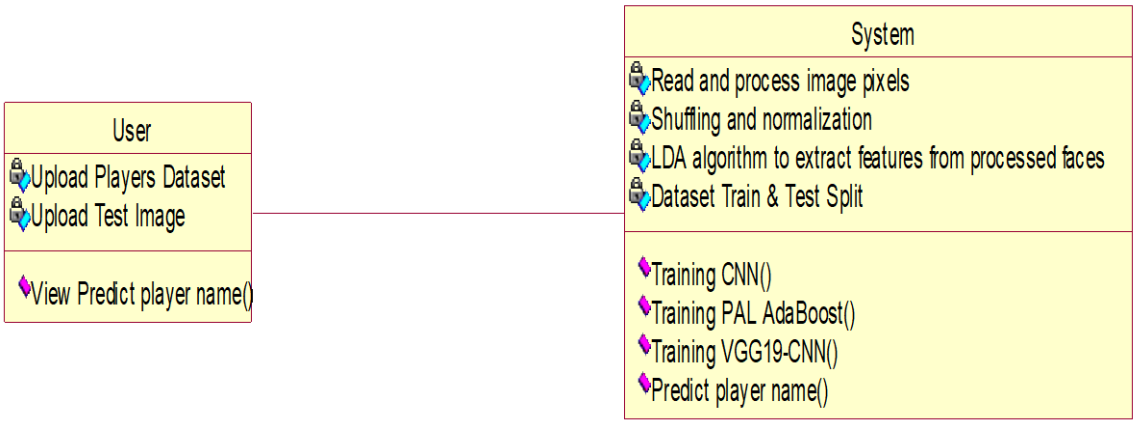


Figure :3 Class Diagram

10.3 SEQUENCE DIAGRAM

UML Sequence diagram belongs to a family of interaction diagrams – dynamic models, and documentation as well – that demonstrate how different processes work, connect and in which order they do so.

Sequence or an event diagram, it is in most cases called an event scenario or timing diagram. It, therefore, provides an elaborated view of the chronological order of events, how objects cooperated in functions over a given period in order to realize a particular process, or in other words, how the objects of the given system behaved during the realized process. These diagrams are great for depicting the behavior of the system and the subtleties of its interactions.

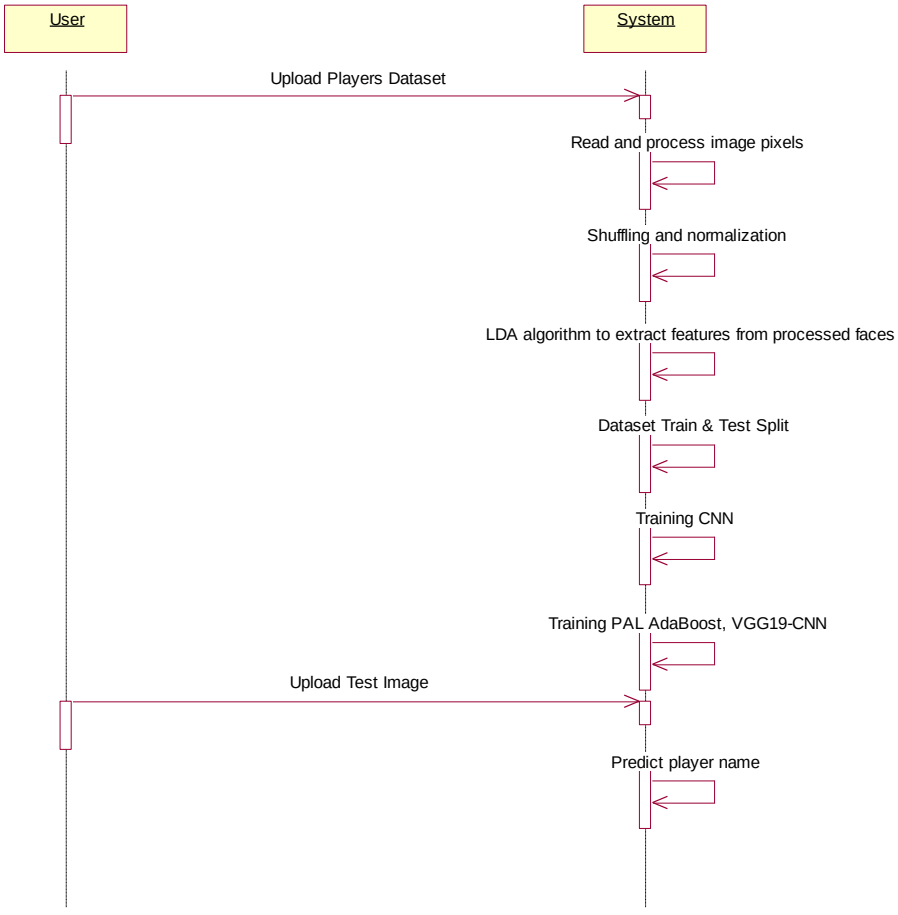


Figure 4 Sequence Diagram

10.4 COLLABORATION DIAGRAM

A collaboration diagram UML is developed to show the links and communication between objects in the system. As opposed to sequence diagrams that chronologically document the sending of messages, collaboration diagrams concentrate on the design and construction of the objects which are the center of the communication at the time of its development.

System objects are endowed with different characteristics, and many of them interrelate with one another and form relationships. The collaboration diagram or the communication diagram depicts these relations graphically in order to give information about the object structure of the system. It assists in appreciating the reason for the existence of some objects and their mode of integration with others in the performance of certain defined functions.

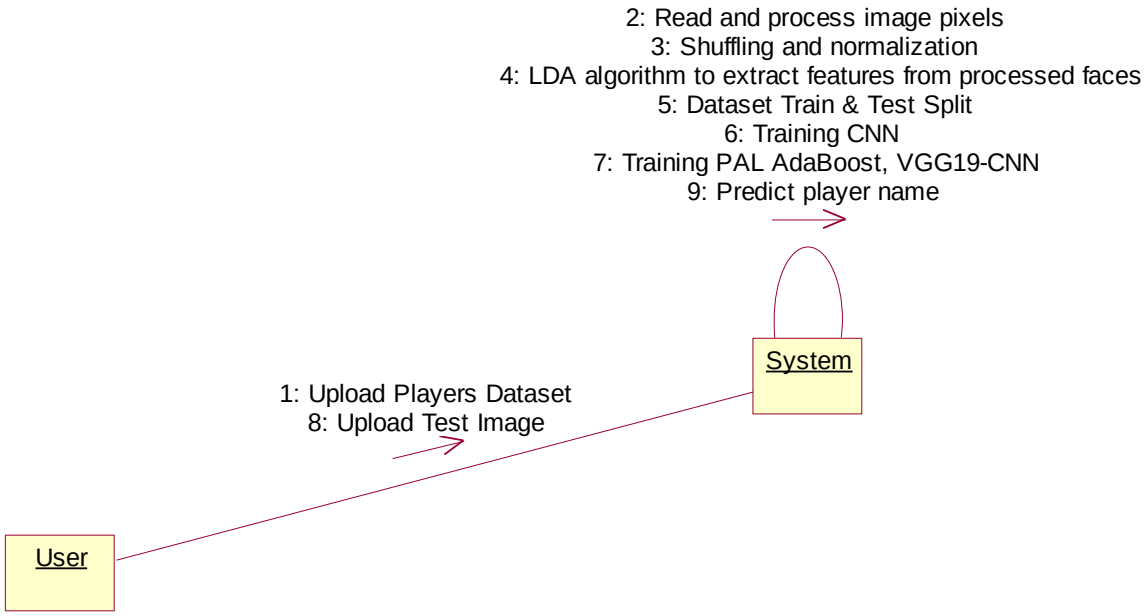


Figure: 5 Collaboration Diagram

10.5 ACTIVITY DIAGRAM

The activity diagram also contains cases of the communication and interactions of objects for the functional purpose. They include a variety of cases such as events and actions in terms of how they flow as well as how they reach the various systems edges or items.

UML activity diagrams outline the business requirements of a process from the user perspective in high level details and also the operational processes of the system explaining how different functional units operate in the context of sequential stages. They are a logically inseparable portion of each process model considering the order of operations and making decisions at each logical point associated with movement of information.

An activity diagram is a fundamental tool in both systems modeling and business processes modeling since it allows any operation to be tracked in a sequential visual representation of its flow.

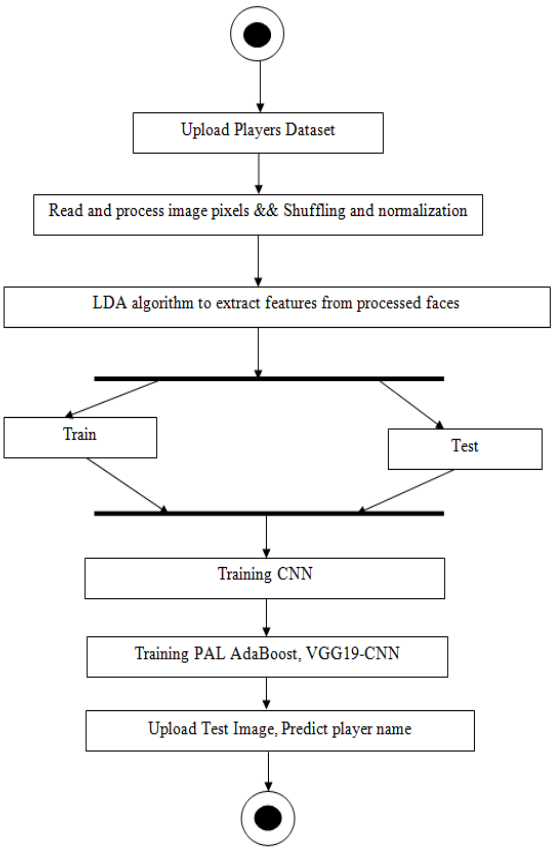


Figure:6 Activity Diagram

Chapter – 11

TECHNOLOGY DESCRIPTIONS

11. TECHNOLOGY DESCRIPTIONS

Python Introduction

Python is general purpose high level dynamic interpreted programming language that is used for many applications. Python programming requires a different set of skills since it supports the object oriented programming approach. Python is clear, straightforward, and offers a number of high level data structures making it a preferred language for programmers.

The syntax of the Python language is easy to understand. Its dynamic typing combined with the interpreted nature allows quick jump in application development and scripting. It provides support for multiple paradigms including object oriented, imperative, functional, as well as procedural programming styles to accommodate different project requirements.

Even if Python has not been targeted for a specific area like web programming, it is possible to utilize its multi-dimensional scope in web design, company solutions, implementing 3D CAD, data analysis, and other purposes. One interesting characteristic of the Python programming language is the fact that it is dynamically typed meaning that variables do not need to be explicitly declared. For example, it is possible to do a = 10 directly.

Python proves beneficial to the user in more than one way - it did away with the need for a compilation step which in turn minimized the amount of time required for debugging and testing. Due to the efficiency of the edit-test-debug cycle, Python makes the ideal language for prototyping and tweaking designs.

11.1 PYTHON HISTORY

Python Beginning The software was conceived in the second half of the 80s. Its inception can be credited to Guido Van Rossum at Centrum Wiskunde&Informatica (CWI) in the Netherlands. Python was first hit in December 1989 and the first version (0.9.0) was released by van Rossum about February 1991.

When Python 1.0 came out in 1994, lambda, map, filter, and reduce were introduced. And in the later versions of Python, like 2.0, list comprehensions and the garbage collection system were incorporated.

In order to rectify major bugs in the language, 3.0 ok, in other words Py3K, was launched on the third day of the last month of the year of 2008. The languages that influenced the design of Python are ABC, which was an exception-handling programming language that interfaced with the Amoeba Operating System, and Modula-3.

Python has an extremely bright future due to the fact that it encompasses a lot of what its predecessors had while developing into one of the most used programming languages in the whole world.

11.2 PYTHON FEATURES

Python Characteristics Easy to learn and easy to use:

Being beginner oriented and developer 100 percent focused, python is relatively easy to learn and use.

Expressive Language:

The coding language of Python is declared as highly expressive as it is claimed that it is easy to read and to write.

Interpreted Language:

The reason for this is because Python makes use of an interpreter which takes in the line of code and executes it although it may bring out a few challenges in the debugging of codes.

Cross-platform Compatibility:

This goes to show that the system has the capacity of being used in different kinds of operating systems such as Windows, Linux, Unix and mac operating systems.

Free and Open Source:

Any person who wishes to avail Python can find it free for downloading and It is easy to acquire and modify the source code.

Object-Oriented Programming:

Object-oriented programming is achievable using Python as it provides support for classes and objects among other implementations.

Extensible:

It becomes possible to extend various functionalities and integrate performance improvements by using C/C+= languages with Python environments.

Large Standard Library:

If many different types of modules and functions are needed to develop applications quickly, then Python standard library can be a good solution.

GUI Programming Support:

It is possible to use Python to program graphical user interfaces.

Integrated with Other Languages:

The compatibility and flexibility of the programming language Python is higher as it can work with other languages particularly C, C++, Java languages.

11.3 PYTHON APPLICATIONS

Since Python is a multipurpose language, it can be used in almost all spheres of software development. Other notable fields where Python has its application include:

1. **web Applications:**

   With its libraries for internet protocols as HTML, XML, JSON, and many others, and frameworks such as Django, Flask, Pyramid, Python is among the most popular languages for web development. Such include PythonWikiEngines, Pocoo and others.

2. **Desktop GUI Applications:**

   Desktops, GUI can be developed using Tkinter, wxWidgets, Kivy and PyQt. For multi touch applications across platforms Kivy is good for such purpose.

3. **Software Development:**

   Programming in Python is widely used in software development as an auxiliary language for such activities as build control, testing, management, etc.

4. **Scientific and Numeric Computing**:

   Libraries like **SciPy**, **Pandas**, and **IPython** are popular for scientific computing, engineering, and mathematical analysis.

5. **Business Applications**:

   Python is used for creating ERP and e-commerce solutions such as **Tryton**.

6. **Console-Based Applications**:

   Tools like **IPython** are developed using Python for interactive computing.

7. **Audio/Video Applications**:

   Python can develop multimedia apps like TimPlayer or cplay.

8. **3D CAD Applications**: Tools like Fandango are used to create CAD applications.

11.4 WHY PYTHON

Python is a versatile, user-friendly, and powerful programming language widely used across industries. Here's why Python is so popular:

- **Cross-Platform Compatibility**: Python runs on various platforms, including Windows, Mac, Linux, and Raspberry Pi, making it highly portable.
- **Simple and Readable Syntax**: Python's syntax is simple, clean, and similar to the English language, making it easier for developers to learn and write code.
- **Less Code, More Functionality**: Python allows developers to accomplish tasks with fewer lines of code compared to other programming languages.
- **Interpreter-Based Language**: Python runs on an interpreter, meaning that code can be executed immediately after being written. This makes prototyping faster and more efficient.
- **Multiple Programming Paradigms**: Python supports procedural, object-oriented, and functional programming, offering flexibility to developers.

Good to Know:

The latest major version is **Python 3**, which is used in this tutorial. Although **Python 2** is no longer updated (except for security fixes), it remains popular. You can write Python using text editors or Integrated Development Environments (IDEs) like Thonny, PyCharm, NetBeans, or Eclipse for better file management.

Python Syntax:

Python was designed for readability, using new lines instead of semicolons and indentation (whitespace) to define scope. This differs from other languages that use semicolons or curly braces.

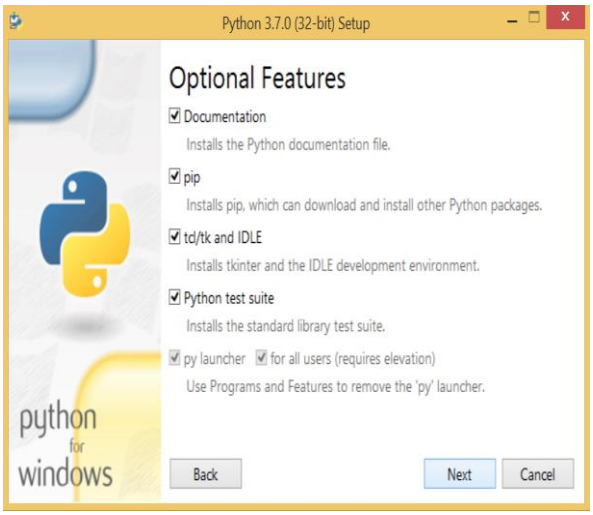
11.5 PYTHON INSTALLATION

Run the Installer:

To get started, double click the Python executable file that you installed. This will also bring up the Installation Wizard. If you would like to go forward, please select Customize installation

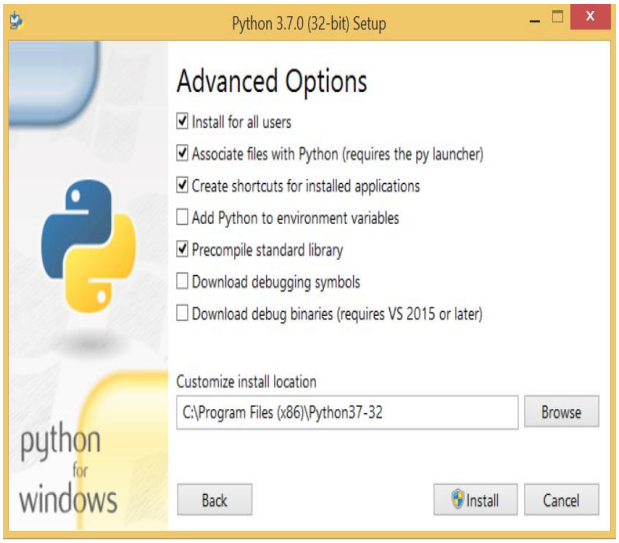
Select Optional Features:

The installation will offer additional components to augment the functionality of the software that has been brought over from the preceding window. These features come pre-activated so to proceed simply click Next.



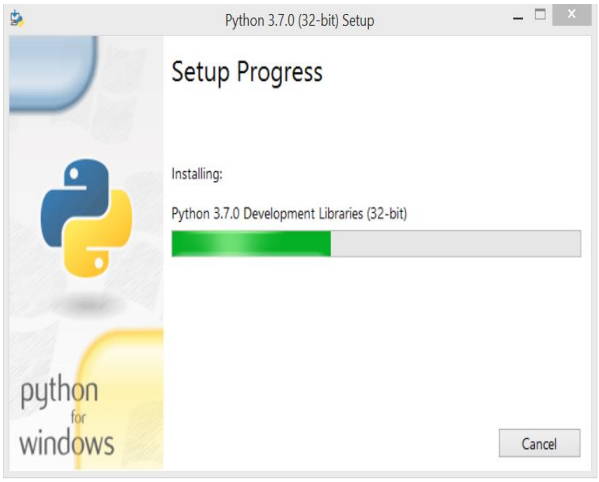
Advanced Options:

The following window will show advanced options. These options can be installed according to your requirements though it may take a relatively longer time. To proceed, tick the first box 'Install for all users' and click Next.



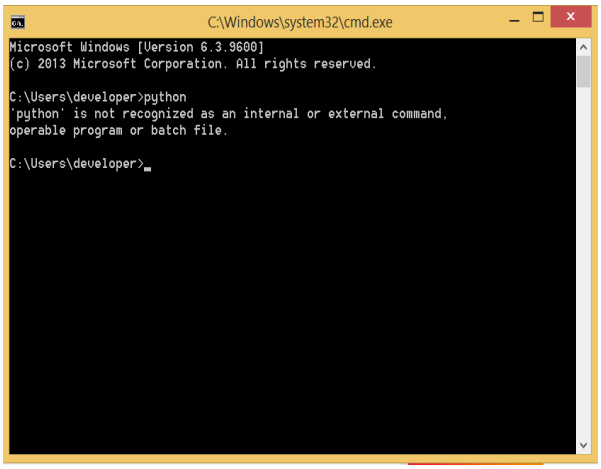
Start the Installation:

Now that you are in the installation window, click on the Install button in order to start the installation process. You can now start the installation of Python 3.6.6.



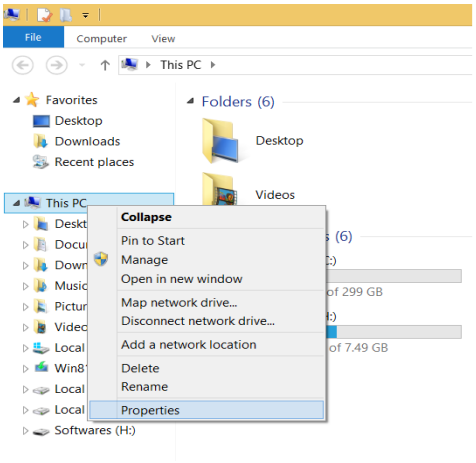
Run Python:

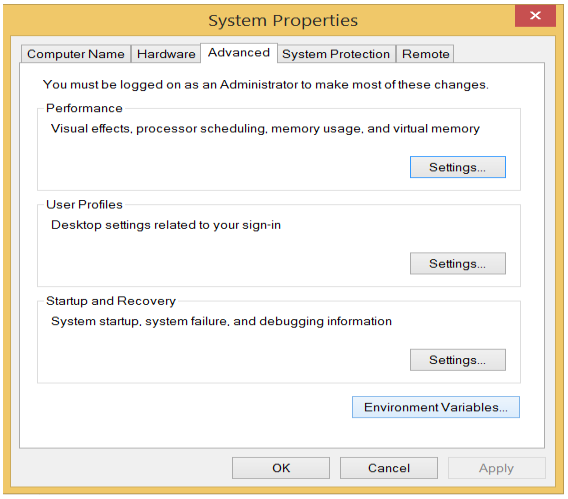
To begin the compilation on Python 2 please open the Command Prompt and write python or python3 for Python 3. In case of an error, you need to set the path.



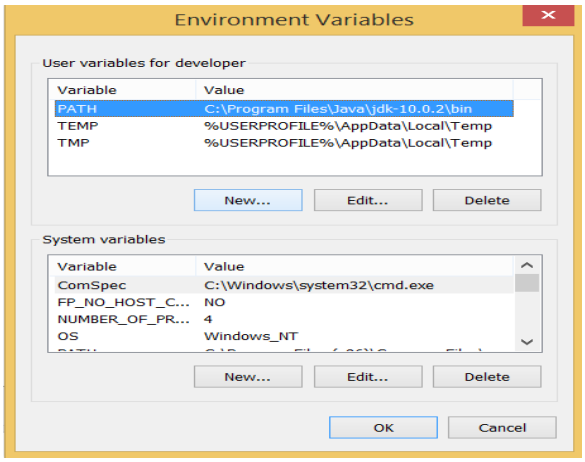
Run Python:

To begin the compilation on Python 2 please open the Command Prompt and write python or python3 for Python 3. In case of an error, you need to set the path.

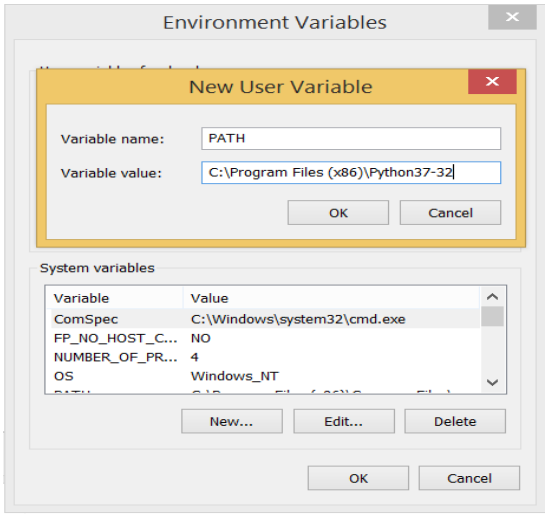




Add the new path variable in the user variable section.



Specify the directory of the installation as per the image given below and add it as PATH in the new variables user section.



Restart the Command Prompt:

Start by opening the CMD again then type python. This time around, the python interpreter shell will initiate and from here it will be possible to issue python commands.

Chapter – 12

SAMPLE CODE

12. SAMPLE CODE

#import required classes and packages

```
import os

import cv2

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from face_detector import get_face_detector, find_faces

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import RandomForestClassifier

from keras.utils.np_utils import to_categorical

from keras.layers import  MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential, load_model, Model

import pickle

from keras.callbacks import ModelCheckpoint

from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, InputLayer,
BatchNormalization, Dropout

from sklearn.tree import DecisionTreeClassifier

from keras.applications import VGG19

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score
```

```
import keras
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
#define and load class labels found in dataset
path = "Dataset"
labels = []
X = []
Y = []
for root, dirs, directory in os.walk(path):
    for j in range(len(directory)):
        name = os.path.basename(root)
        if name not in labels:
            labels.append(name.strip())
print("Cricket Players Class Labels : "+str(labels))
```

Cricket Players Class Labels : ['babar_azam', 'bhuvneshwar_kumar', 'dinesh_karthik', 'hardik_pandya', 'jasprit_bumrah', 'kuldeep_yadav', 'ms_dhoni', 'ravindra_jadeja', 'rohit_sharma', 'shakib_al_hasan', 'shikhar_dhawan', 'shoaib_malik', 'vijay_shankar', 'virat_kohli']

In [3]:

```
#define function to get class label of given image
def getLabel(name):
    index = -1
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
```

```
    return index
```

In [5]:

#object to detect face

```
face_model = get_face_detector()
```

In [24]:

```
#load dataset image and process them
if os.path.exists("model/X.txt.npy"):
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
else: #if images not process then read and process image pixels
    for root, dirs, directory in os.walk(path):#connect to dataset folder
        for j in range(len(directory)):#loop all images from dataset folder
            name = os.path.basename(root)
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j])#read images
                faces = find_faces(img, face_model)#function to detect face
                for x, y, x1, y1 in faces:
                    roi = img[y:y1, x:x1]#extract face features
                    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)#convert color face to gray
                    if roi is not None:
                        roi = cv2.resize(roi, (32, 32))#resizing all faces to equal size
                        X.append(roi.ravel())#add features to X training array
                        label = getLabel(name)#get integer class label for given player
                        Y.append(label)
    X = np.asarray(X)#convert array as numpy array
    Y = np.asarray(Y)
```

```
    np.save('model/X.txt',X)#save process images and labels
    np.save('model/Y.txt',Y)
print("Dataset images loaded")
print("Total images found in dataset : "+str(X.shape[0]))
print()
```

Dataset images loaded

Total images found in dataset : 229

In [25]:

```
#visualizing class labels count found in dataset
names, count = np.unique(Y, return_counts = True)
height = count
bars = labels
y_pos = np.arange(len(bars))
plt.figure(figsize = (8, 3))
plt.bar(y_pos, height)
plt.xticks(y_pos, bars)
plt.xlabel("Players Faces Found in Dataset")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()
```

#apply preprocessing techniques such as shuffling and normalization

```
indices = np.arange(X.shape[0])
np.random.shuffle(indices)#shuffle images
X = X[indices]
Y = Y[indices]
```

```
#normalize all image faces pixel values
sc = StandardScaler()
X = sc.fit_transform(X)
print("Normalized Faces Features = "+str(X))
#applying LDA algorithm to extract features from processed faces
lda = LinearDiscriminantAnalysis(n_components=13)
X = lda.fit_transform(X, Y)
print("LDA Features Extraction Completed")
```

LDA Features Extraction Completed

In [28]:

```
#split dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
print("Dataset Train & Test Split Details")
print("80% images used to train algorithms : "+str(X_train.shape[0]))
print("20% image used to train algorithms : "+str(X_test.shape[0]))
```

Chapter – 13

TESTING

13. TESTING

13.1 INTRODUCTION

Software engineers draw a distinction between software fault and software failure. A fault will mean an error that is either a bug in programming or a defect in a certain program's semantic. A fault may or may not result in a failure. A failure on the other hand is such that a software has not performed as expected by the end user. Under certain conditions such as when the faulty code is run on the CPU, or even when it is migrated to another hardware machine or when it is enhanced with new capabilities, faults become failures.

Testing a software application is always essential in the process of analyzing the software with the aim of validating its quality and providing the stakeholders with relevant information. It states the errors that it contains and so guarantees that a software has a certain amount of standards that it fulfills with users and system requirements.

13.2 SYSTEM TESTING AND IMPLEMENTATION

System testing is concerned with testing the logic of individual module codes and debugging them before their assembly into larger subsystems. This involves:

1. **Module Integration Testing:** This tests the linkages of modules to one another.
2. **System Testing:** This is done to check that the whole system conforms to the requirements provided.
3. **Acceptance Testing:** This goal is to represent the abilities of the system to the customers.

There are two methods that could be used in selecting test cases which include:

• **Black Box Testing** - Testing of the external behavior with respect to the system requirements specification.

• **White Box Testing** - Concentrates on the assessment of internal logic and aims at coverage of the code.

13.3 TESTING TECHNIQUES

Testing discovers mistakes made in the software by executing the software under conditions that were planned in advance. Two main approaches are:

1. **Black Box Testing:**

   Emphasizes functional specification and assesses external behavior without any interference on internal logic. It detects errors such as interface error, the database access error, performance as well.

2. **White Box Testing:**

Deals with the internal logic by scrutinizing module flow graphs and analyzing logical paths, loops and data structures to be tested

13.4 TESTING STRATEGIES

To ensure their applications are free from errors and comply with the required functionality, the effective performance of software testing is paramount. Given below are the top primary kinds of software testing strategies used in the software development process:

Unit Testing

Unit testing tends to assure the correctness of the implementation of individual units or components of the software. It is usually conducted to ensure the correctness of internal logic, and that for a particular input element expected output is given. The unit testing is said to be structural and invasive since it depends on the knowledge of the structure of the software developed.

The main objective of developers when using unit testing is to ensure the validity of each decision branch and each internal code path. This guarantees all business logic and configurations function as intended. Unit testing is usually done at the level of components and it helps in detecting faults very early in the development cycle.

A good example would be a system with three modules: a Reputation Module, Route Discovery Module and Audit Module. All of these systems would be put through a separate test. Errors are recognized during this stage, patches are made and the patched executable units are checked before they are incorporated.

Integration Testing

Integration testing is usually done after unit testing and it is concerned with putting several units or components together into one cohesive system as they were meant to work. It is event driven, concentrating on functional interconnections and ultimately the interaction of the components of the system.

While unit testing touches on the performance of each individual component, unit testing is aimed at confirming that all the component systems do not create problems when put together. These tests determine whether there are discrepancies in attached data and also the communication lines between modules so as to enhance unity and harmony in system 0

System Testing

As the name states, system testing tests the final, integrated software product to ensure everything is functioning according to the requirements. It deals with the global layout and just like the end to end testing, guarantees normal results given the particular components are used in a realistic way.

One example of system testing is configuration-oriented system integration testing, which checks how processes interact at various integration points. System testing focuses on process flows and assesses whether the system is reliable and meets the required expectations.

Functional Testing

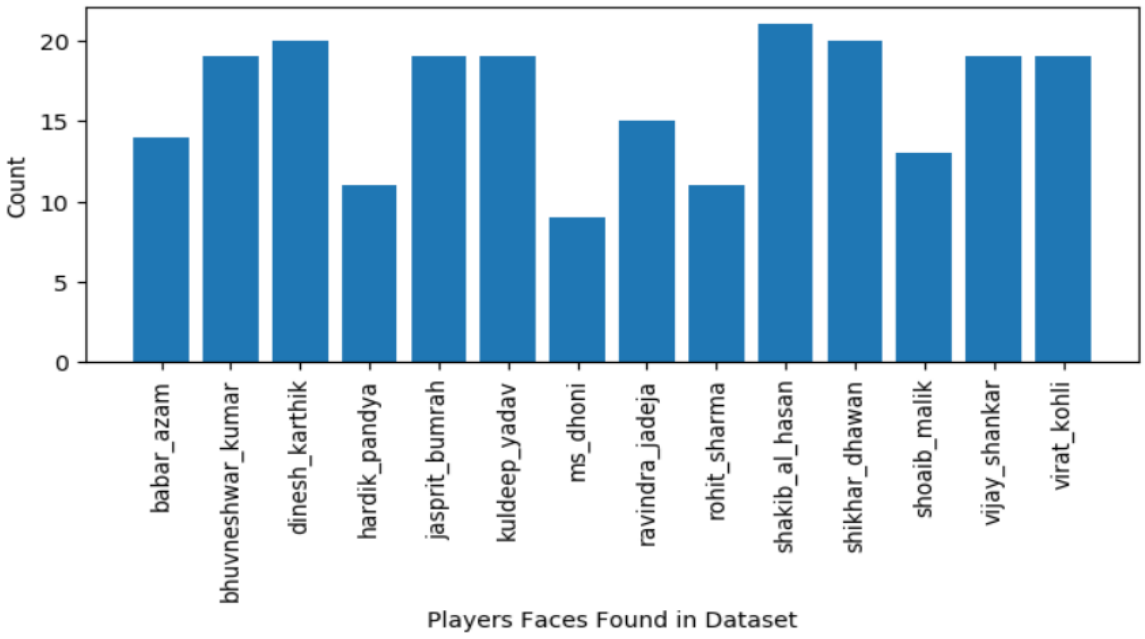
Functional testing establishes whether the system implemented meets the business and technical requirements stated in system documents and users' manuals. It offers a systematic approach for verifying that all required functions have been implemented correctly and as expected by users.

Functional testing was done by using each business process or feature and user interaction against documented requirements. This helps ensure that all features work as they are intended to work without any problems.

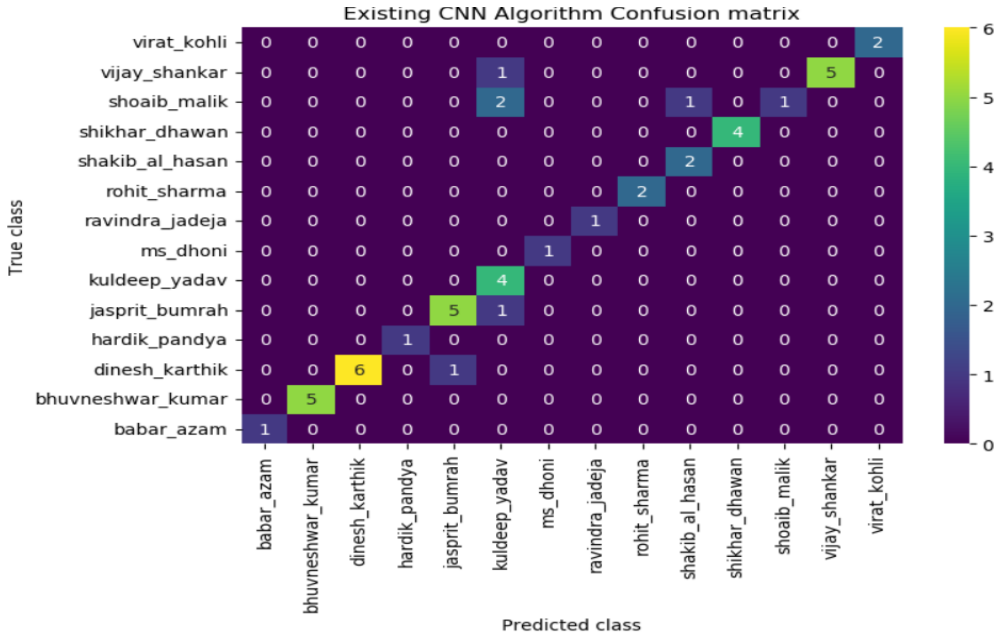
Chapter – 14

SCREENSHOTS

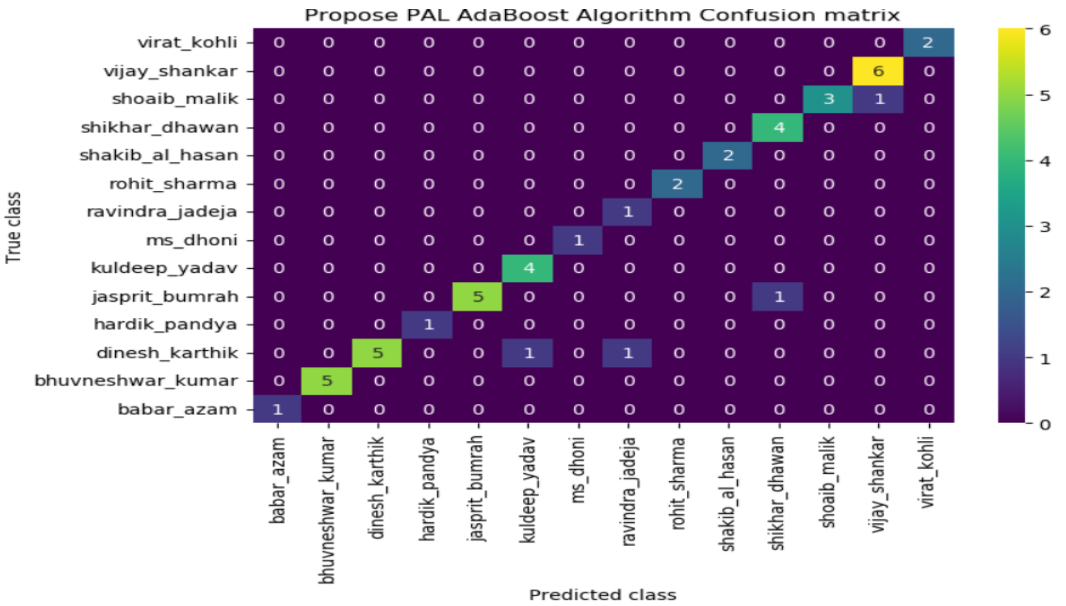
14. SCREENSHOTS



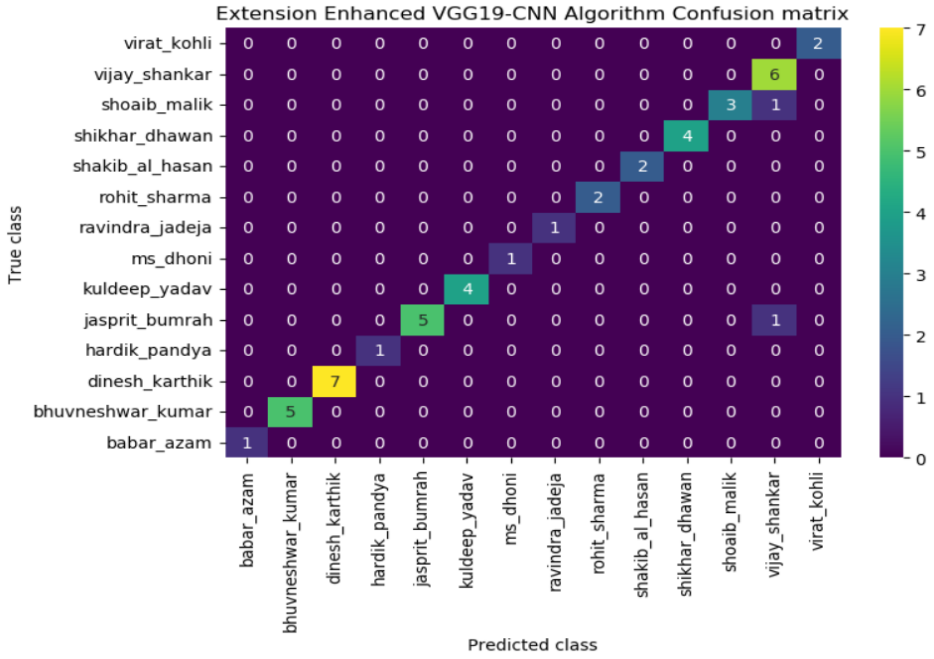
Screen 14.1 Players Faces Found in Dataset



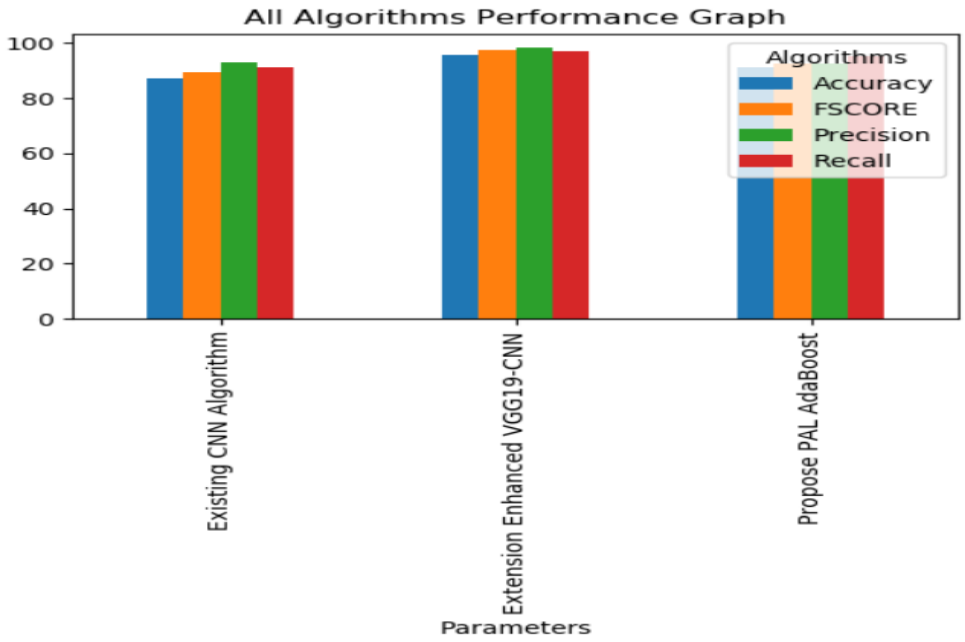
Screen 14.2 CNN



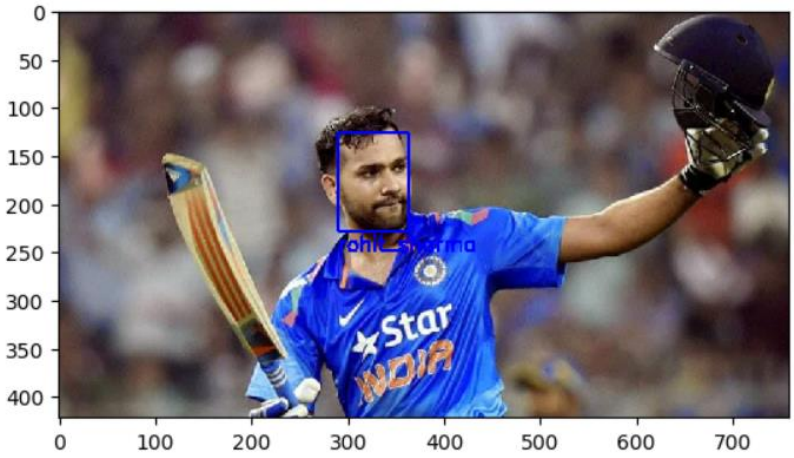
Screen 14.3 PAL AdaBoost



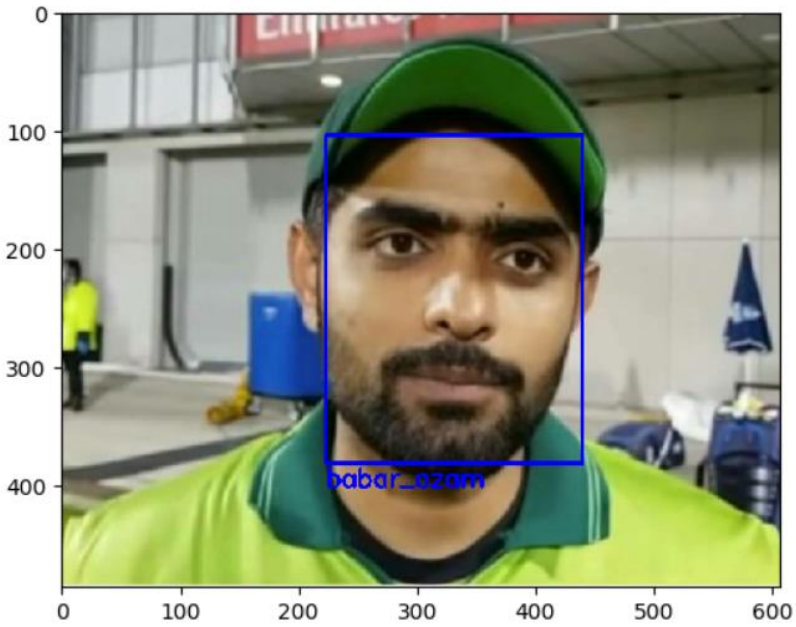
Screen 11.4 Enhanced VGG19-CNN



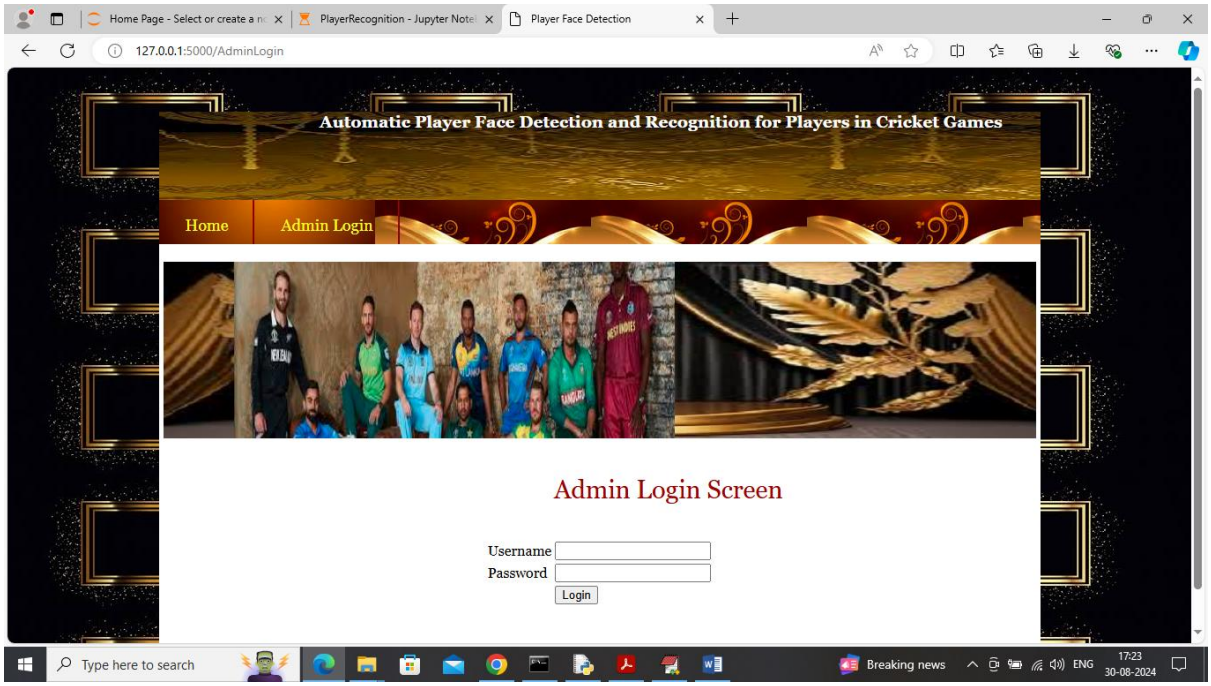
Screen 14.5 All Algorithms Performance Graph



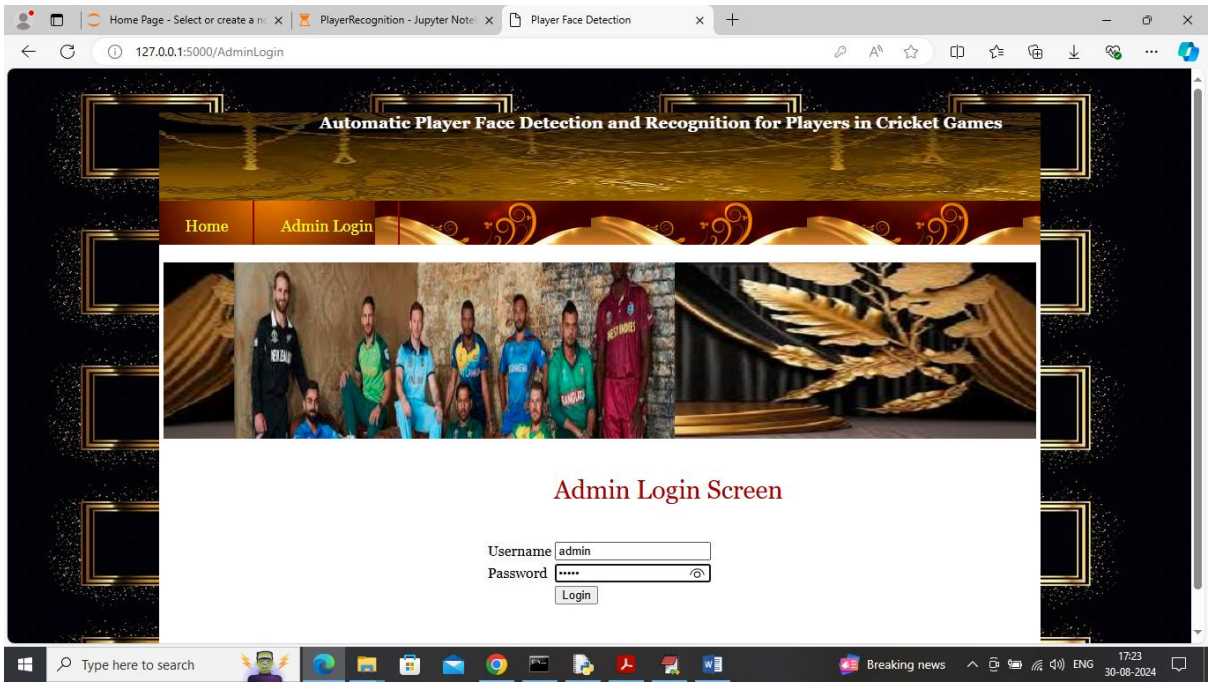
Screen 14.6 Detected Rohit_sharma



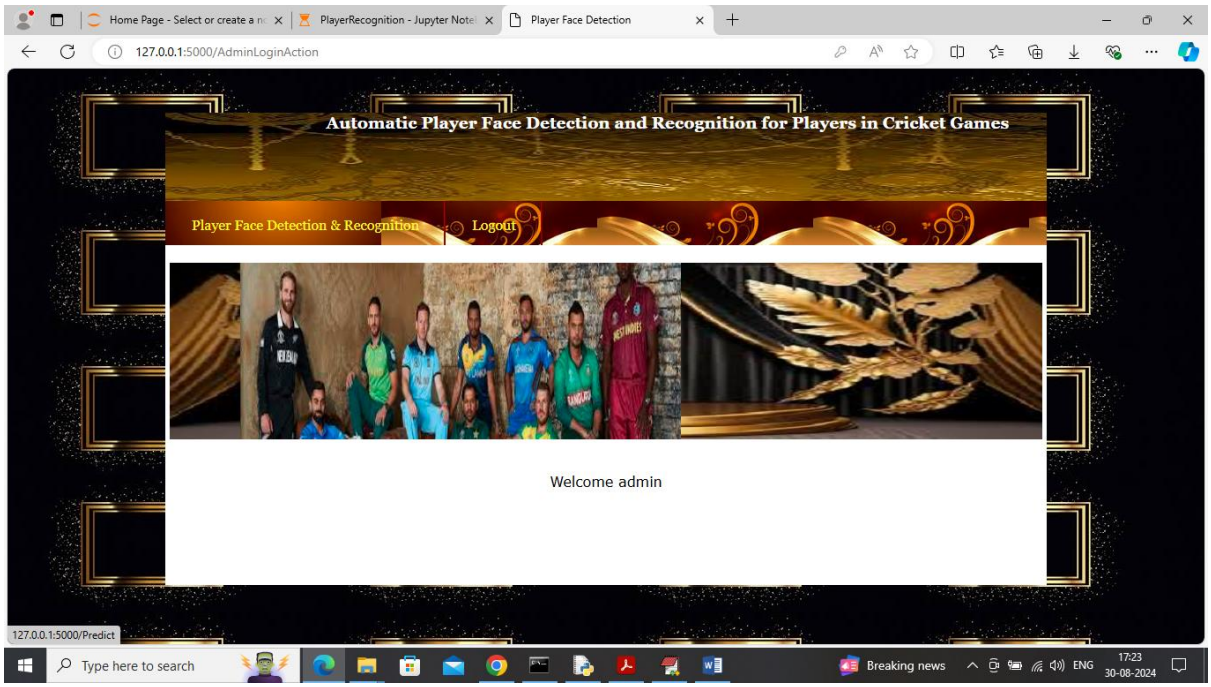
Screen 14.7 Detected Babar_azam



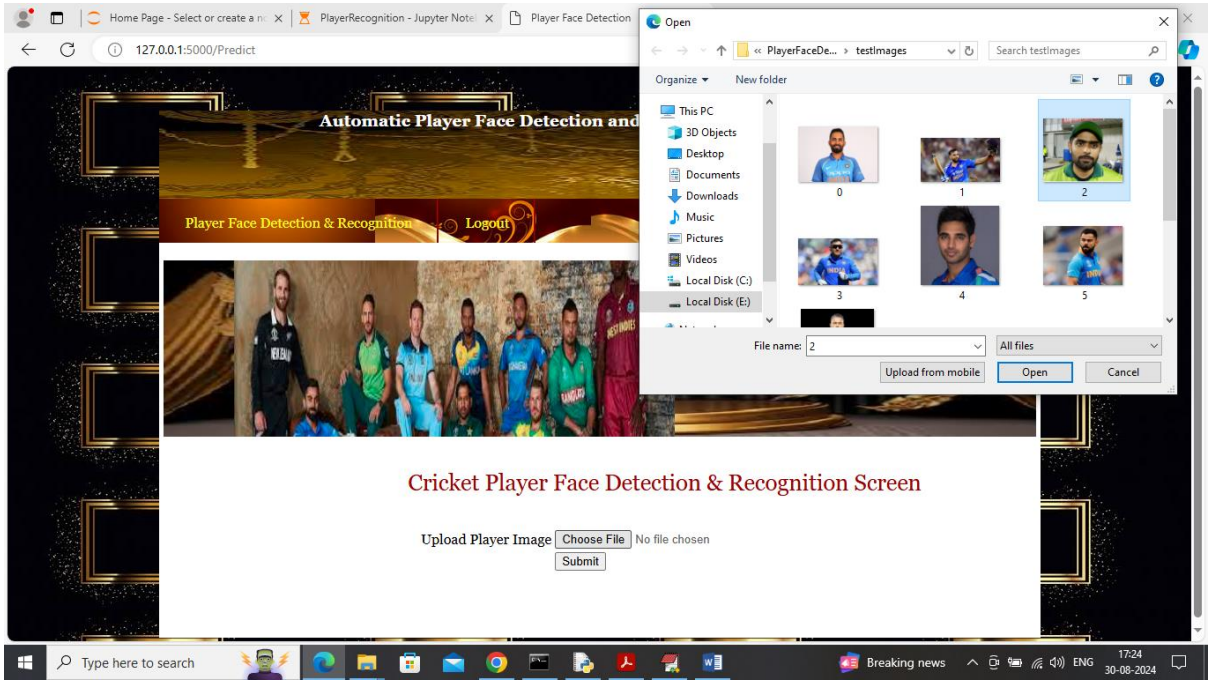
Screen 14.8 Home Screen



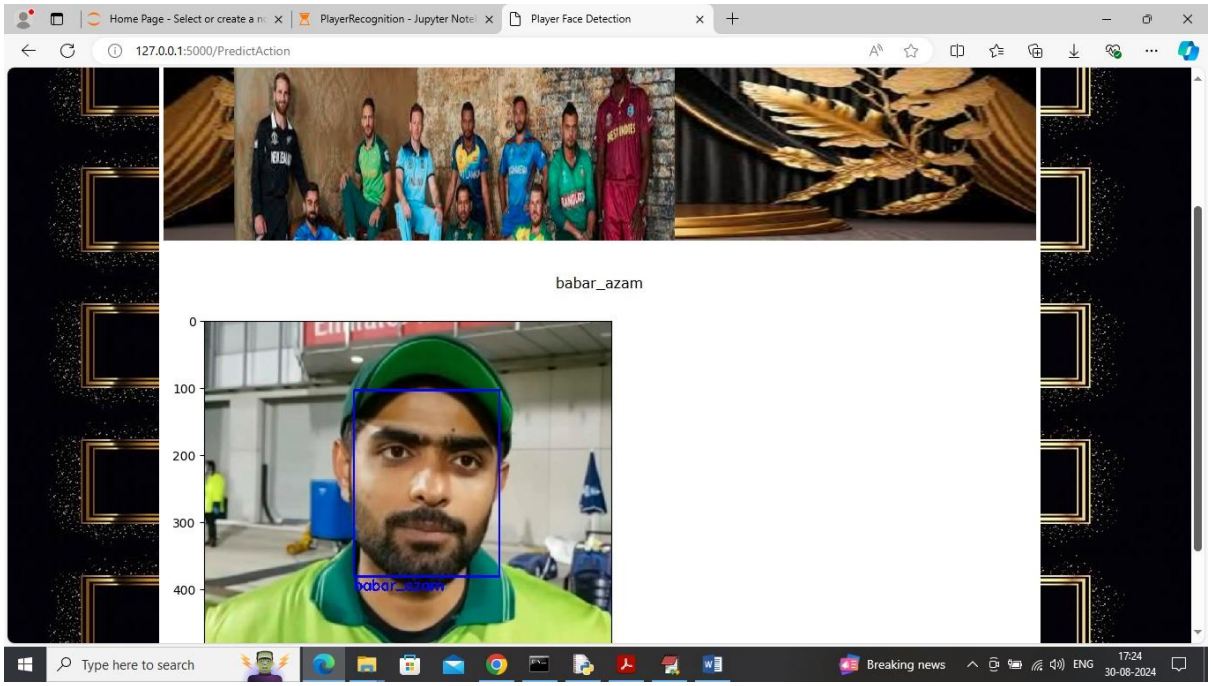
Screen 14.9 Admin Login



Screen 14.10 Player Face Detection & Recognition



Screen 14.11 Upload Test Image



Screen 14.12 Detected and recognized Babar_azam

Chapter – 15

CONCLUSIONS

15. CONCLUSION

This research introduces an efficient player face detection and recognition system for cricket, combining LDA, ADABOOST, and VGG19-CNN to improve accuracy. The approach ensures real-time player identification, overcoming challenges like occlusion and lighting variations.

By integrating deep learning and traditional machine learning, the system achieves high precision and scalability. The Flask-based deployment enables seamless user interaction, making it a practical solution for sports analytics and broadcasting. Future improvements may include real-time video processing and multi-angle recognition to further enhance system capabilities in live sports environments.

Chapter – 16

BIBLIOGRAPHY

16. BIBLIOGRAPHY

[1] H. Ullah, M. U. Haq, S. Khattak, G. Z. Khan, and Z. Mahmood, ‘‘A robust face recognition method for occluded and low-resolution images,’’ in Proc. Int. Conf. Appl. Eng. Math. (ICAEM), Taxila, Pakistan, Aug. 2019, pp. 86–91, doi: 10.1109/ICAEM.2019.8853753.

[2] F. Munawar, U. Khan, A. Shahzad, M. U. Haq, Z. Mahmood, S. Khattak, and G. Z. Khan, ‘‘An empirical study of image resolution and pose on automatic face recognition,’’ in Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST), Islamabad, Pakistan, Jan. 2019, pp. 558–563, doi: 10.1109/IBCAST.2019.8667233.

[3] G. Rajeshkumar, M. Braveen, R. Venkatesh, P. J. Shermila, B. G. Prabu, B. Veerasamy, B. Bharathi, and A. Jeyam, ‘‘Smart office automation via faster R-CNN based face recognition and Internet of Things,’’ Measurement: Sensors, vol. 27, Jun. 2023, Art. no. 100719, doi: 10.1016/j.measen.2023.100719.

[4] A. Rehman, T. Saba, M. Mujahid, F. S. Alamri, and N. ElHakim, ‘‘Parkinson’s disease detection using hybrid LSTM-GRU deep learning model,’’ Electronics, vol. 12, no. 13, p. 2856, Jun. 2023, doi: 10.3390/electronics12132856.

[5] M. U. Haq, M. A. J. Sethi, R. Ullah, A. Shazhad, L. Hasan, and G. M. Karami, ‘‘COMSATS face: A dataset of face images with pose variations, its design, and aspects,’’ Math. Problems Eng., vol. 2022, pp. 1–11, May 2022, doi: 10.1155/2022/4589057.

[6] M. A. Hamza, S. Ben Haj Hassine, S. Larabi-Marie-Sainte, M. K. Nour, F. N. Al-Wesabi, A. Motwakel, A. Mustafa Hilal, and M. Al Duhayyim, ‘‘Optimal bidirectional LSTM for modulation signal classification in communication systems,’’ Comput., Mater. Continua, vol. 72, no. 2, pp. 3055–3071, Jan. 2022, doi: 10.32604/cmc.2022.024490.

[7] M. U. Haq, M. A. J. Sethi, and A. U. Rehman, ‘‘Capsule network with its limitation, modification, and applications—A survey,’’ Mach. Learn. Knowl. Extraction, vol. 5, no. 3, pp. 891–921, Aug. 2023, doi: 10.3390/make5030047.

[8] J. Wu, C. Peng, W. Tan, and Z. Wu, ‘‘FaceEncAuth: Face recognition privacy security scheme based on FaceNet and SM algorithms,’’ Comput. Eng. Appl., vol. 58, no. 11, pp. 93–99, 2022. [9] R. Pena, F. Ferreira, F. Caroli, L. Schirmer, and H. Lopes, ‘‘Globo face stream:

A system for video meta-data generation in an entertainment industry setting,'' in Proc. 22nd Int. Conf. Enterprise Inf. Syst., 2020, pp. 1–9, doi: 10.5220/0009380203500358.

[10] H. Li, A. Manickam, and R. D. J. Samuel, ''RETRACTED ARTICLE: Automatic detection technology for sports players based on image recognition technology: The significance of big data technology in China's sports field,'' Ann. Oper. Res., vol. 326, no. S1, p. 97, Jan. 2022, doi: 10.1007/s10479-021-04409-1.

[11] H. Qiu, D. Gong, Z. Li, W. Liu, and D. Tao, ''End2End occluded face recognition by masking corrupted features,'' IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 10, pp. 6939–6952, Oct. 2022, doi: 10.1109/TPAMI.2021.3098962.

[12] V. Ellappan and R. Rajkumar, ''Classification of cricket videos using finite state machines,'' Int. J. Inf. Technol. Manage., vol. 20, nos. 1–2, p. 83, Jan. 2021, doi: 10.1504/ijitm.2021.114156.

[13] X. Li, Y. Xiang, and S. Li, ''Combining convolutional and vision transformer structures for sheep face recognition,'' Comput. Electron.Agricult., vol. 205, Feb. 2023, Art. no. 107651, doi: 10.1016/j.compag.2023.107651.

[14] E. Bermejo, E. Fernandez-Blanco, A. Valsecchi, P. Mesejo, O. Ibáñez, and K. Imaizumi, ''FacialSCDnet: A deep learning approach for the estimation of subject-to-camera distance in facial photographs,'' Expert Syst. Appl., vol. 210, Dec. 2022, Art. no. 118457, doi: 10.1016/j.eswa.2022.118457.

[15] S. Arya, N. Pratap, and K. Bhatia, ''Future of face recognition: A review,'' Proc. Comput. Sci., vol. 58, pp. 578–585, Jan. 2015, doi: 10.1016/j.procs.2015.08.076.

[16] D. Mamieva, A. B. Abdusalomov, M. Mukhiddinov, and T. K. Whangbo, ''Improved face detection method via learning small faces on hard images based on a deep learning approach,'' Sensors, vol. 23, no. 1, p. 502, Jan. 2023, doi: 10.3390/s23010502.

[17] S. Hangaragi, T. Singh, and N. N, ''Face detection and recognition using face mesh and deep neural network,'' Proc. Comput. Sci., vol. 218, pp. 741–749, Jan. 2023, doi: 10.1016/j.procs.2023.01.054.

[18] D. Vaithiyanathan and M. Manigandan, ‘‘Real-time-based object recognition using SIFT algorithm,’’ in Proc. 2nd Int. Conf. Electr., Electron., Inf. Commun. Technol. (ICEEICT), Trichirappalli, India, Apr. 2023, pp. 1–5, doi: 10.1109/ICEEICT56924.2023.10157675.

[19] O. Arandjelovic and A. Zisserman, ‘‘Automatic face recognition for film character retrieval in feature-length films,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), San Diego, CA, USA, Jun. 2005, pp. 860–867, doi: 10.1109/CVPR.2005.81.

[20] J. Sivic, M. Everingham, and A. Zisserman, ‘‘Person spotting: Video shot retrieval for face sets,’’ in Proc. Int. Conf. Image Video Retr., Singapore. Berlin, Germany: Springer, Jul. 2005, pp. 226–236