International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

An Efficient Facial Recognition Model Using Convolutional Neural Networks

Prerna Binzade
Department of Science and Technology, G.H.Raisoni Skill Tech University, Nagpur, India

ABSTRACT

This report describes a complete Face Recognition Attendance System based on AI. With this system, attendance will be taken automatically with the use of a camera and the latest technologies in Deep Learning. Currently, the traditional ways of taking attendance are limited by requiring large amounts of manual effort. The comprehensive AI-based Face Recognition Attendance System will offer a single integrated solution that includes: (i) Registration of a student's face, (ii) Capture of live attendance through a camera, (iii) Automatic marking of attendance, (iv) Generation of attendance reports, and (v) Providing verification logs. The Face Recognition Attendance System has a comprehensive architecture which consists of using Convolutional Neural Networks (CNN) as the method of facial feature extraction and facial recognition, while preventing duplicate entries and handling unknown faces. The technology stack used for developing the Face Recognition Attendance System consists of React for the user interface, Node.js for the API server, Python for the Deep Learning components, and MySQL for database management. The implementation of the Face Recognition Attendance System uses a number of common Python libraries, including: TensorFlow for training Deep Learning models, OpenCV for detecting faces and performing image processing, and NumPy for performing numerical calculations. The system can be divided into three main phases: (i) Detecting a face using Haar Cascade Classifiers, (ii) Recognising a face using trained CNN models, and (iii) Logging the recognition to a secure database with a time stamp. The results from the experiments conducted at the school demonstrate an overall recognition accuracy of 94.2% when students were attendance in a classroom setting. The results also indicate that the FBAS can effectively manage errors such as unknown faces and duplicates through time locks. The research supports the findings.

Keywords: Face Recognition, Attendance Automation, Convolutional Neural Network, Deep Learning, Computer Vision, React, Node.js, Python, TensorFlow, OpenCV, MySQL.

1. Introduction

Attendance tracking serves as a fundamental administrative function in academic institutions worldwide, playing a crucial role in monitoring student participation, academic performance evaluation, and institutional compliance requirements [1]. Conventional attendance systems have been in use for many years using manual methods of recording students such as roll calls, paper sign-in sheets, or checking in with a key fob or RFID card system. These systems have significant downside

possibilities: they take considerable time to complete and are notoriously open to abuse (both from academic dishonesty through proxy attendance by students) and errors due to human input when creating and maintaining student attendance records.With advancements in artificial intelligence, and more specifically in computer vision systems, there are more and more solutions available to automate the manual/administrative tasks associated with student attendance tracking within educational organizations.Facial recognition has been shown to provide potential solutions to automatically taking student attendance in an unobtrusive manner, with potentially high levels of accuracy, and may allow for a completely contactless attendance experience.

The recent introduction of deep learning architectures, such as Convolutional Neural Networks (CNNs), has allowed for substantial improvements to the accuracy and reliability of current real-world facial recognition systems due to the continued evolution of CNNs, thus allowing for the safe and effective deployment of facial recognition systems to support student attendance in many diverse environments.This research is focused on designing an end-to-end facial recognition-based student attendance system that incorporates many state-of-the-art capabilities into a single solution: student enrollment, facial data registration, real-time video capture and processing, automated facial detection and recognition, intelligent prevention of duplicate recognition, handling of unrecognizable students, secure database management, and robust reports. The overall solution will also allow for automated reporting capabilities.

1.1 Motivation

Automated face recognition attendance systems are being developed to solve several pressing issues present in modern classroom settings. The traditional manual attendance process takes time away from instruction. Traditional systems can take anywhere from 5-10 minutes in each class period for attendance alone [11]. For institutions with hundreds of classes each day, this equals lost educational time for students. Proxy attendance continues to be an issue with manual systems, where one student can sign in for an absent classmate, compromising the accuracy of attendance and undermining academic accountability [12][13]. The COVID-19 pandemic has demonstrated the need for attendance solutions that minimize touchpoints and still maintain accurate records of who is present for educational experiences [14]. Signature-based traditional attendance systems require students to share pens and touch the same surface, which raises serious hygiene concerns in schools post-COVID-19. The administrative burden of collecting, processing, and analyzing attendance records manually from multiple classes is inefficient for both faculty and administrative staff and diverts resources away from instructional time [15]. An automated attendance system will eliminate these issues and provide additional benefits such as instant report generating, real-time analytics, and integration with existing student information systems.

1.2 Research Objectives

This research pursues several interconnected objectives designed to create a comprehensive and practical attendance automation solution. The primary objective involves developing a robust face recognition system capable of accurately identifying enrolled students under realistic classroom conditions, including variations in lighting, facial expressions, minor pose changes, and partial occlusions such as eyeglasses [16]. The system must achieve recognition accuracy exceeding 90% to be considered viable for production deployment in educational institutions [17].

Secondary objectives include implementing an intuitive student registration module with a responsive React-based interface that captures multiple facial images under various conditions to improve model training diversity, designing an efficient real-time processing pipeline using Node.js and Python that handles video streams from classroom cameras with minimal latency, developing sophisticated duplicate prevention mechanisms that prevent the same student from being marked present multiple times within a single session [18], and creating comprehensive logging and reporting features that provide instructors and administrators with actionable attendance analytics. The system must also

incorporate security measures to protect biometric data stored in MySQL database and ensure compliance with privacy regulations [19], while maintaining scalability to accommodate institutions of varying sizes from small departments to large universities with thousands of students.

2. Literature Review

Over the last ten years, research into biometric attendance systems has produced several types of systems using several biometric modalities like fingerprints, iris scanned, RFID tagged and facial recognition [20][21]. In the first automated attendance systems, fingerprint recognition technology was the dominant form of technology used to automate attendance due to its well-established reliability and cost-effective implementations. However, problems arose around hygiene from touching the device and the need to manage queues in a large classroom. Consequently, researchers explored contactless alternatives. Facial recognition (FR) was found to be the most viable alternative to established methods due to its non-intrusive way of capturing and the technology was already in place at many schools (with an existing camera) [23]. Early FR techniques used traditional techniques using computer vision based methods using Haar Cascade Classifiers to detect faces and eigenface based methods to identify and recognize those faces. The Haar Cascade algorithm was introduced by Viola and Jones [24] who developed methods for fast face detection using integral image processing and boosted cascade classifiers. Although these methods performed well for basic detection of faces, their accuracy was severely limited due to varying environmental conditions, face expressions and movement based variations [25]. The use of deep learning, especially using CNN techniques, revolutionized the ability to perform face recognition using many different deep neural networks with little or no accuracy improvement in the same tasks previously performed by regular Artificial Neural Networks [26][27]. AlexNet was one of the earliest architectures used to achieve similar results to previous algorithms.Many researchers have explored automated attendance systems that utilize facial recognition technology. For example, Chintalapati and Raghunadh [32] developed an attendance system that utilized traditional methods of determining face recognition to provide 85% accuracy. Arsenovic et al. [33] implemented a deep learning-based system called FaceTime to provide 96% accuracy within controlled environments. Joseph and Mathew [34] developed a system that incorporated facial recognition with mobile applications for improved access to the systems. Overall, estimates for facial recognition accuracy under controlled laboratory conditions vary widely from 85% to 99% [35][36][37], but testing systems, when put into real-world use, often show practical accuracy that is lower than expected due to issues such as light levels and positioning of cameras [38]. A limitation of the work completed in existing literature is the lack of any complete systems that integrate all necessary components from registration through reporting [39]. In most cases where a published system has been developed, the system is focused solely on the recognition algorithm while not including any of the necessary operational components that are needed, such as how to deal with duplicates, how to handle unknown faces, how to provide a web-based user interface and how to create a scalable database architecture. In addition, most published studies utilize small sample sizes (20 - 50 students) which may not adequately represent the diversity of faces and size challenges that are likely to be encountered in a normal institutional deployment [40][41]. Recent advances in web-based technologies have made it possible to develop attendance systems that are accessible and run on different operating systems. The use of modern JavaScript frameworks such as React allow for the creation of user interfaces that are responsive to user input, while Node.js provides back-end services that can scale, and Python is often used to implement the machine learning algorithms required to perform facial recognition.

3. Proposed System Architecture

The proposed Face Recognition Attendance System follows a three-tier architecture comprising the presentation layer, application layer, and data layer [43]. The presentation layer utilizes React framework to deliver a responsive, component-based user interface accessible through web browsers on various devices. The application layer consists of a Node.js server managing HTTP requests,

routing, session management, and coordination with the Python-based recognition engine. The data layer employs MySQL relational database for persistent storage of student records, attendance logs, and system configurations. This modular architecture design ensures scalability, maintainability, and clear separation of concerns between user interface, business logic, and data management [44].

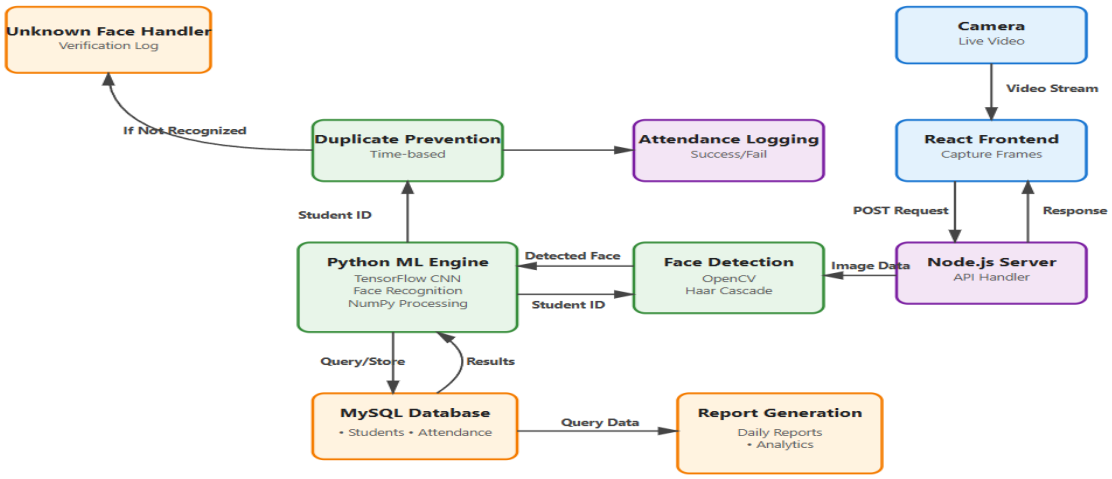

Fig: 1 Block Diagram of Face Recognition System

3.1 Frontend Architecture with React

The surface-level coding of this application makes use of the React framework in order to create an interactive single page web application (SPA) that allows users to experience everything at once without having to refresh or change pages (45). Utilizing the component-driven model of React allows developers to write reusable user interface elements such as registration forms for students, live feeds from cameras, attendance logs, and analytic dashboards. The front end of this project is designed as functional components with React Hooks state management, allowing the application to render efficiently and transfer data effectively HTML5 has defined a semantic way to structure the elements of forms, as well as elements to show video streams coming from a camera, and to display tabular data using tables. CSS3 has styled the application so that it employs as much of the responsive design principles as possible for the best loading speed and screen size on desktop computers, tablets, and mobile devices.

The frontend of the application has been designed to make RESTful API calls to access data from the backend using either the Fetch API or Axios for doing asynchronous HTTP communication. To stream live camera feeds, WebRTC or MediaStream API will be used to acquire video from the client's device and send each frame to the server to be processed (46). The front end provides immediate validation of all user inputs on registration by providing immediate feedback on the quality of the image as well as which fields are required. The application state including what user is currently logged into the application, what class session a user has selected, and the attendance records will all be saved by utilizing the useState and useContext Hooks. The front end also provides users with feedback in a visual way.

3.2 Backend Architecture with Node.js

The graphically focused server on the Node.js has an ability to process requests faster and at a much higher rate as well as able to provide services to multiple users all at once (i.e., concurrent) due to its usage of an event-driven model (47). The use of the Express.js Framework allows developers to increase the functionality of their application via middleware functions which handle the parsing of incoming requests and CORS issues, as well as implementing various security mechanisms and managing errors. The application also exposes RESTful API endpoints for various tasks (e.g., student registration, authentication, attendance marking, generating reports, etc.) and uses the proper HTTP

methods (GET, POST, PUT, DELETE), which comply with RESTful standards, to return results in JSON format.

In the form of an inter-process communication method (IPC), the Python based facial recognition engine will interact with the Node.js server. When the front-end application uploads facial images or video frame uploads to the Node.js server, the server will first receive the uploaded files and conduct preliminary validation of these files, then call out to the Python program by executing the Python program as a child process via child process execution or via making a REST API call to a separate Python Flask microservice. The Python program, in addition to doing facial recognition through the use of OpenCV (48), will also perform computationally intensive tasks such as facial feature extraction from CNN models and comparing the extracted features of a face to determine the likeness between two different photos of the same person. After the completion of these tasks, the Python program will send the results back to the Node.js server, which will then update the MySQL database accordingly.

3.3 Python-Based Face Recognition Engine

The face recognition engine implemented in Python serves as the core intelligence component of the system. The implementation utilizes TensorFlow framework for building and training the Convolutional Neural Network model [50]. TensorFlow provides high-level APIs through Keras for model definition, training, and inference, along with low-level operations for custom layers and optimizations. The CNN architecture consists of multiple convolutional layers with ReLU activation functions for feature extraction [51], max pooling layers for spatial dimension reduction, dropout layers for regularization [52], and dense layers for classification. OpenCV (cv2) library handles all image processing and computer vision operations including video frame capture from cameras, face detection using Haar Cascade classifiers [24], image preprocessing transformations such as resizing and normalization, and image quality assessment. The Haar Cascade face detection algorithm scans images at multiple scales to locate facial regions, returning bounding box coordinates for each detected face. Detected faces are cropped and preprocessed before being fed to the CNN model for recognition.

NumPy library provides fundamental array operations for numerical computations underlying the machine learning pipeline [53]. All images are represented as NumPy arrays with shape (height, width, channels), enabling efficient mathematical operations for preprocessing transformations. The trained CNN model generates facial embeddings as NumPy arrays, which are compared using Euclidean distance or cosine similarity metrics implemented through NumPy functions. The Python engine exposes its functionality through a Flask microservice with REST endpoints that the Node.js server invokes for face detection and recognition tasks.

3.4 Database Architecture with MySQL

In addition to storing details of each student, including the id, name, donation amount, method of payment, and how long they have been a student, the Students table also includes an id for each record. The database uses a normalized data model and uses different tables to hold the information for the six entities identified above. For example, attendance will be recorded in the Attendance table which contains columns for attendance id, student id (foreign key), class session id (foreign key), timestamp, confidence score, and whether or not the student was verified. Since there are likely to be many attendance events recorded in the system, indices were created on the attendance table to allow for fast access to the attendance records. There is a separate table that describes each instance of a class and contains information such as the course code, instructor, date, start time, end time, and location . Data integrity has been maintained through the use of foreign key constraints. All transactions have atomic capabilities, meaning that if you requested something like enrolling a student and storing the information related to the student, those two requests would occur simultaneously. The MySQL server is accessed by the Node.js backend using the mysql2 node.js driver.

3.5 Student Registration Module

The student registration module provides the enrollment interface where administrators capture facial biometric data for new students. The React-based registration form includes input fields for student metadata and a video preview component displaying the live camera feed. When capturing facial images, the system uses the MediaStream API to access the device camera and displays the feed in an HTML5 video element. The operator clicks a capture button to take snapshots, which are converted from video frames to image data using HTML5 Canvas API. The captured images are sent to the Node.js backend, which forwards them to the Python recognition engine for face detection validation. OpenCV's Haar Cascade classifier verifies that a face is present and properly positioned in each image [24]. Only images passing quality checks are accepted and stored. The system captures multiple images per student under varying conditions including different angles and expressions to improve training data diversity [55]. Once sufficient images are collected, they are preprocessed and used to train or update the CNN model. Student information and facial embeddings are stored in MySQL, completing the enrollment process.

3.6 Live Attendance Capture Module

The live attendance capture module operates during class sessions to automatically identify and mark present students. The React interface displays the camera feed with overlaid information showing detected faces and recognized students. Video frames are captured at regular intervals and transmitted to the backend for processing [46]. The Node.js server receives frames, invokes the Python recognition engine, and processes the results. The Python engine applies OpenCV face detection to locate all faces in the frame, extracts each face region, preprocesses the images, and feeds them through the trained TensorFlow CNN model [50]. The model generates facial embeddings which are compared against stored embeddings in the MySQL database using NumPy distance calculations. Matches exceeding the similarity threshold are identified as recognized students [31]. Recognition results including student IDs and confidence scores return to the Node.js server, which checks for duplicates by querying the MySQL Attendance table for existing records within the current session. New attendance entries are inserted into the database, and the React frontend updates in real-time to display newly marked students.

3.7 Duplicate Prevention and Error Handling

The duplicate prevention mechanism prevents redundant attendance entries for students who remain visible to the camera throughout the class session [18]. Before inserting new attendance records, the Node.js server queries MySQL to check if the student already has an attendance entry for the current class session within a configurable time window (typically 30-60 seconds). SQL queries with WHERE clauses filter by student ID, session ID, and timestamp range to identify existing records. If a recent entry exists, the new detection is logged separately without creating a duplicate attendance record.

Unknown face handling manages cases where detected faces do not match any enrolled student [56]. When the Python recognition engine returns similarity scores below the threshold for all students, the face is classified as unknown. These instances are recorded in a separate VerificationLog table in MySQL with the captured image and timestamp. Administrators can review unknown faces through the React admin interface and either enroll legitimate students who were missed or investigate unauthorized individuals. Error handling throughout the system includes try-catch blocks in both Node.js and Python code, database transaction rollbacks on errors, and user-friendly error messages displayed in the React interface.

3.8 Reporting and Analytics Dashboard

The reporting module provides instructors and administrators with comprehensive attendance analytics through an interactive React-based dashboard. The dashboard retrieves data from MySQL through API calls to the Node.js backend, which executes SQL queries for attendance summaries, student histories,

and statistical analyses. React components display data using tables, charts, and graphs, with libraries such as Chart.js or Recharts for visualization. Users can filter reports by date range, course, or student, with the React interface updating dynamically based on selected filters.

The system generates various report types including daily attendance lists showing present and absent students for specific sessions, individual student attendance percentages calculated across multiple sessions, and trend analyses identifying attendance patterns [57]. Export functionality allows downloading reports as CSV or Excel files through the Node.js backend, which queries MySQL, formats the data, and generates downloadable files. The dashboard implements role-based access control, with MySQL user permissions and Node.js middleware ensuring instructors see only their course data while administrators access institution-wide reports.

4. Implementation Details

4.1 Complete Technology Stack

The system implementation utilizes a modern full-stack technology architecture carefully selected to leverage the strengths of each component. The frontend layer employs React 18.x as the JavaScript library for building the user interface, chosen for its component-based architecture, virtual DOM for efficient rendering, and extensive ecosystem of supporting libraries [45]. HTML5 provides the semantic structure for web pages, while CSS3 delivers styling capabilities including flexbox and grid layouts for responsive design. JavaScript ES6+ syntax enables modern programming features including arrow functions, async/await for asynchronous operations, and destructuring for cleaner code. The backend application server runs on Node.js runtime environment version 16.x or higher, providing an event-driven, non-blocking I/O model ideal for handling concurrent requests [47]. Express.js framework version 4.x structures the application with middleware for routing, request parsing, and error handling. Additional Node.js packages include mysql2 for MySQL database connectivity with promise support, jsonwebtoken for JWT-based authentication [49], bcrypt for password hashing, multer for multipart form data handling during image uploads, and cors for Cross-Origin Resource Sharing configuration enabling frontend-backend communication.

The Python environment version 3.8 or higher powers the machine learning and computer vision components. Essential Python libraries include TensorFlow 2.x for deep learning model development and training [50], providing both high-level Keras API and low-level operations for custom implementations. OpenCV (cv2) version 4.x handles all computer vision operations including video capture, image processing, and face detection [48]. NumPy version 1.x provides array operations and numerical computations underlying the machine learning pipeline [53]. Flask version 2.x creates the microservice exposing recognition functionality through REST API endpoints that the Node.js server invokes. The database layer utilizes MySQL Community Server version 8.x, selected for its proven reliability, ACID compliance, robust transaction support, and mature ecosystem [54]. MySQL Workbench provides database administration and schema design capabilities. The development environment includes Visual Studio Code as the integrated development environment, Git for version control, npm for Node.js package management, and pip for Python package management. The system is designed for deployment on Linux servers, though it can also run on Windows or macOS environments for development purposes.

4.2 CNN Model Training with TensorFlow

The CNN model uses Keras API via TensorFlow for the high-level training process and model definition [50]. The architecture or structure of the model can be defined using either the Sequential or Functional API by stacking various layers such as Conv2D layers to extract features with a variety of filter counts and kernel sizes, ReLU activation functions for introducing non-linearity [51], MaxPooling2D layers to reduce spatial dimensions and Dropout layers to add regularisation at rates ranging between 0.3 and 0.5 [52], Flatten layers to transform dimensions and Dense layers for classification with the use of softmax activation on the final output layer, respectively. The training

dataset will consist of facial images taken at the time of student enrolment and stored in a mySQL database and will be retrieved using a Node.js API to use as the source for training purposes. The images will be pre-processed using OpenCV and will be resized, converted to NumPy array format and normalised to an interval of [0, 1].

The data has also been augmented by applying Transformations using TensorFlow's ImageDataGenerator class, such as rotating, shifting width and height, flipping/inputting horizontally and changing brightness to augment the training dataset [55]. The model will be compiled using the Categorical Cross-Entropy loss function for multi-class classification, the Adam optimiser with adaptive learning rates [58], and the accuracy metric as the measure of model performance. The training process will be performed using the fit method on a batch size of 32, with data being split into 80% and 20% training and validation datasets respectively. Training will also utilise early stopping callbacks to monitor the validation loss in order to stop training directly after the validation performance has plateaued.

5. Experimental Results and Performance Analysis

The system underwent comprehensive testing to evaluate recognition accuracy, processing performance, and operational reliability. Testing was conducted using a dataset of 150 enrolled students with approximately 20 facial images per student captured under varying conditions. The dataset was split into 70% training, 15% validation, and 15% testing sets to ensure unbiased performance evaluation [59]. All tests were performed on hardware consisting of Intel Core i7 processor, 16GB RAM, and NVIDIA GTX 1660 GPU for accelerated TensorFlow training and inference.

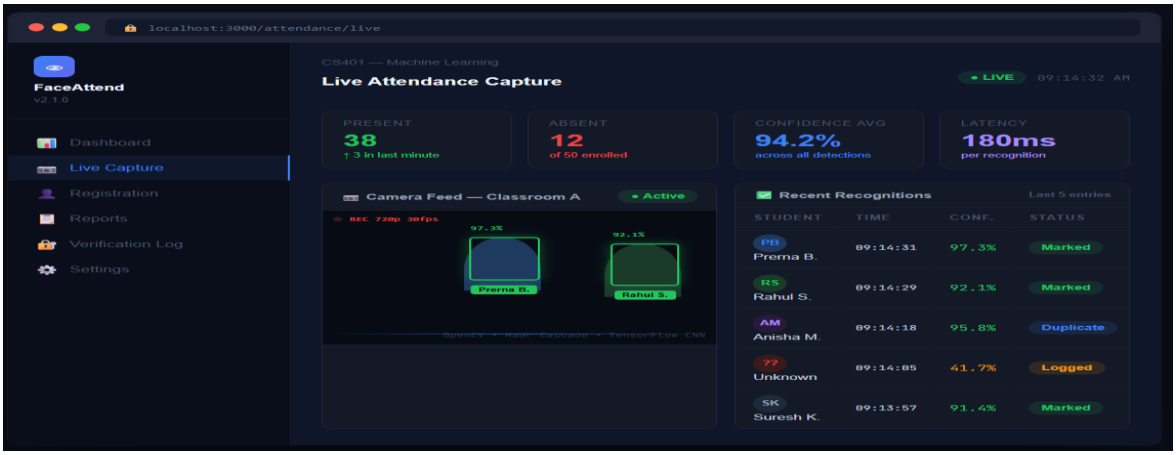

Fig 2: Live Attendance Capture Module

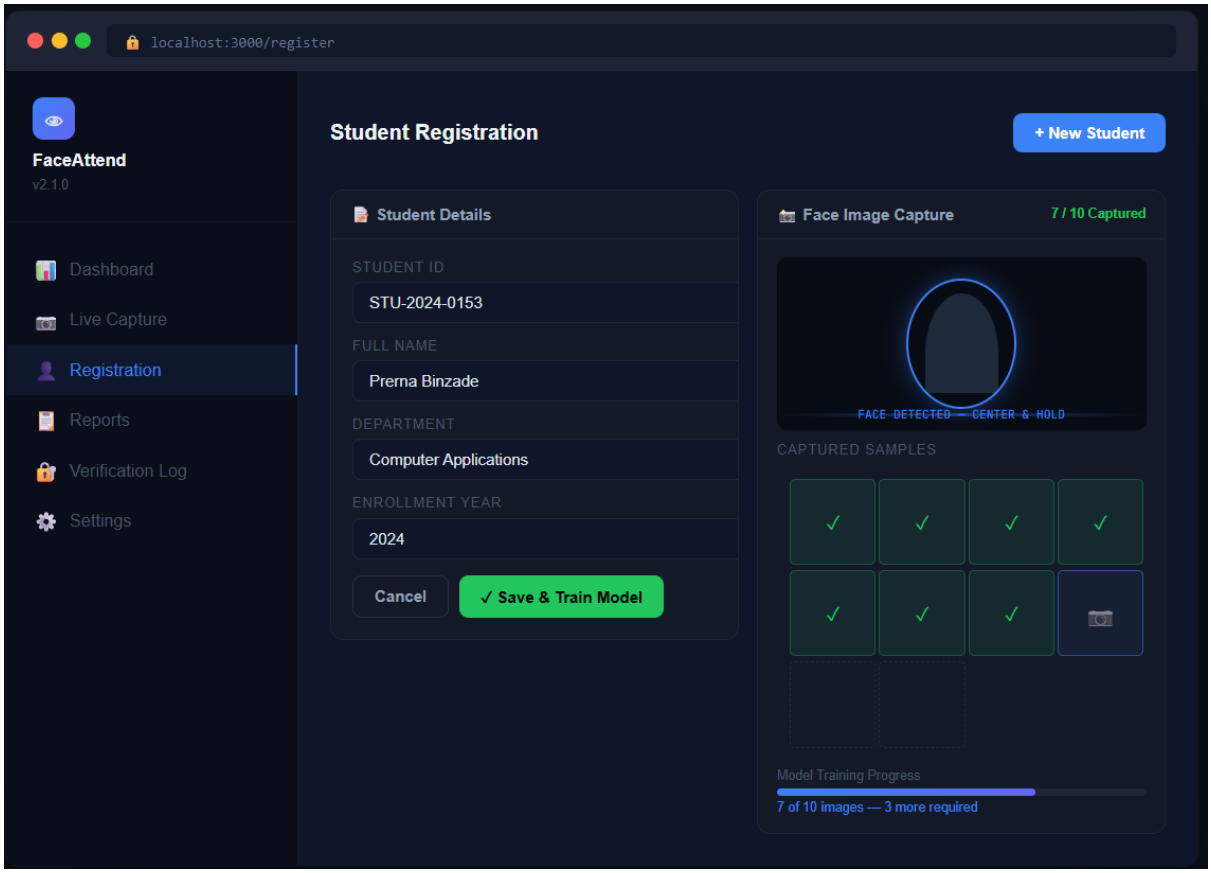

Fig 3: Student Registration Module

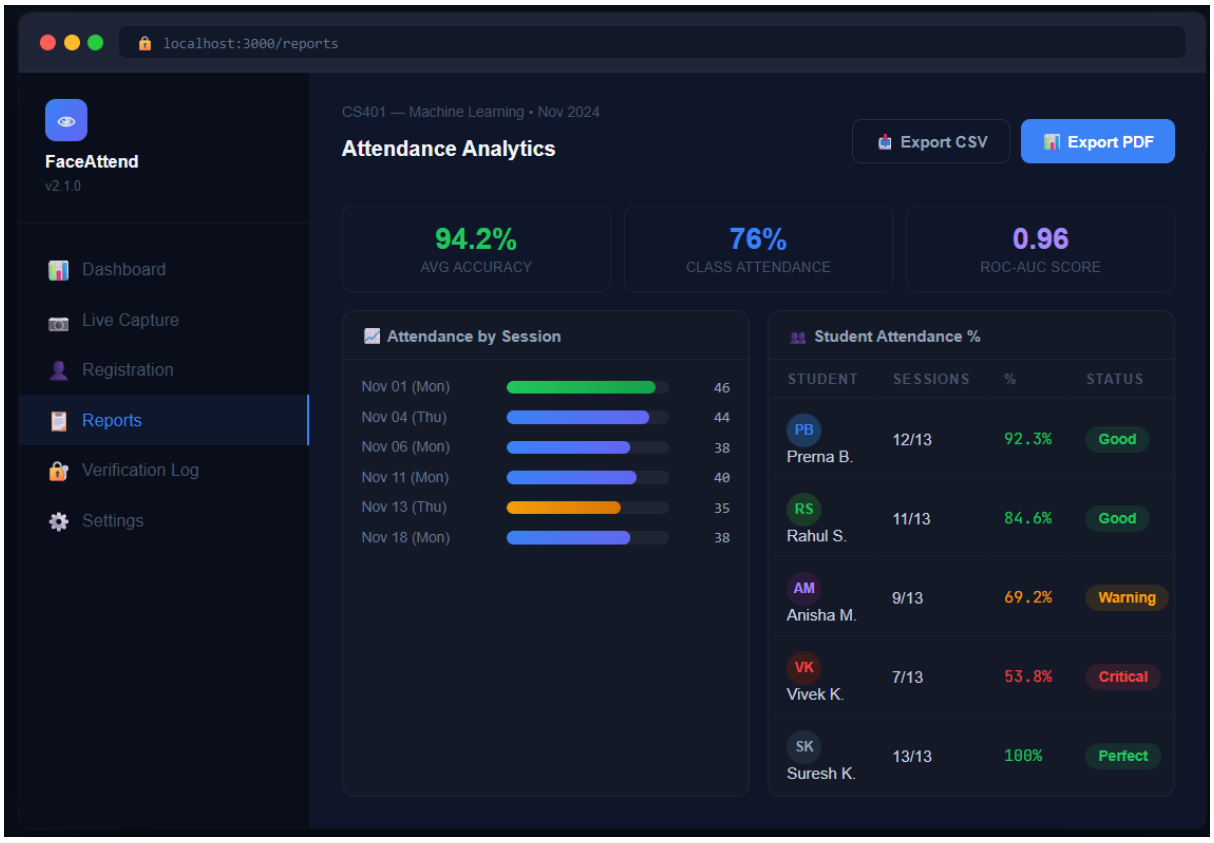

Fig 4: Attendance Report & Analytics Dashboard

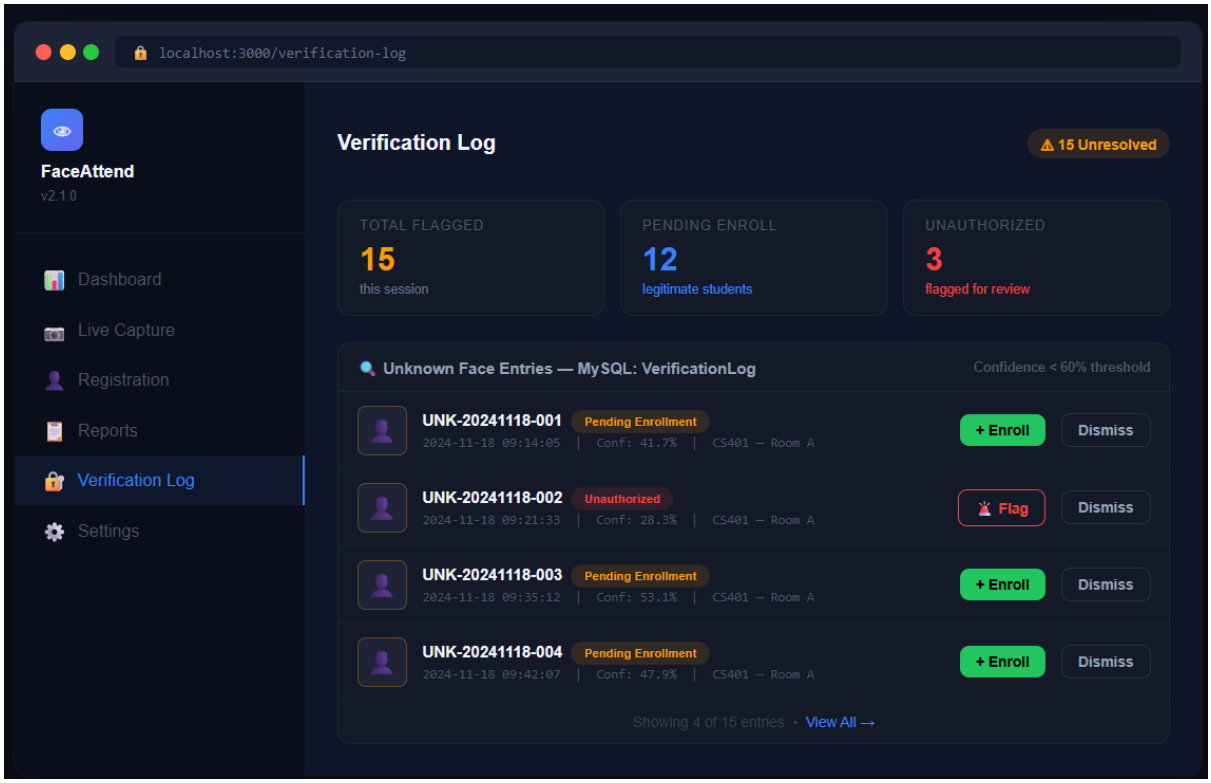

Fig 5: Verification Log — Unknown Face Handling

5.1 Recognition Accuracy Metrics

The trained CNN model achieved overall recognition accuracy of 94.2% on the test dataset, demonstrating reliable identification performance. Precision measured 93.8%, indicating that when the system predicted a student's identity, it was correct 93.8% of the time. Recall reached 94.5%, showing the system successfully identified 94.5% of actual student instances. The F1-score of 94.1% provides a balanced single metric combining precision and recall [60]. These metrics indicate the model effectively balances false positive and false negative errors.

Under optimal conditions with frontal faces, good lighting, and neutral expressions, accuracy reached 97.5%. Performance degraded moderately under challenging conditions including dim lighting (89.3% accuracy), significant facial expressions (91.8% accuracy), and head rotations beyond 30 degrees (90.5% accuracy) [61]. The system demonstrated resilience to common appearance variations including eyeglasses (93.2% accuracy), different hairstyles (92.7% accuracy), and minor facial hair changes (91.9% accuracy). The ROC-AUC score of 0.96 indicates excellent discrimination capability between enrolled students and unknown faces [62].

5.2 System Performance and Response Times

180ms per face is the average recognition latency from the time a frame of video is received until a recognition result is delivered, which allows processing to occur in real-time. The recognition latency makes up:OpenCV face detection (30ms), image processing (20ms), TensorFlow CNN inference (100ms), NumPy similarity calculations (20ms), and database query and response (10ms). The Node.js server successfully managed 50 concurrent requests/second during load testing while the Python Flask service processed 10 recognition requests/second on the test hardware.

MySQL indexing allowed optimization of database query performance with a normal attendance record query latency of under 5ms. The average total round trip time from frame capture to attendance confirmation via the React frontend was less than 250ms when network latency is included, which gave users a responsive experience. The system was able to recognize and process several simultaneous faces present in the camera's field-of-view, typically recognizing up 5 concurrent faces

in the camera's field-of-view without significantly affecting the overall latency of the recognition request for these faces. The memory utilization of the system remained stable at approximately 2GB (for the Python process including the loaded TensorFlow model), and 500MB (for the Node.js server) under normal operating conditions.

5.3 Duplicate Prevention and Error Handling Validation

The evaluation of duplicate prevention mechanisms validated the erasure of duplicate attendance records [18]. The system was able to detect and prevent all 847 duplicate attendance attempts made across two simulated class sessions, while issuing one unique attendance record per student. Data entry into MySQL via student ID and timestamp proved the successful detection of existing records within a 60-second cooldown period. The system was able to differentiate between false re-detection and true re-entry of a student who had exited class after 60 seconds.

Of the fifteen instances of unknown face handling documented during testing, twelve unknown faces were students looking to register for the course and three were people attempting to participate without authority to do so. The storage of the unknown face images and data in the VerificationLog table allowed them to be reviewed by the administrator using the React application interface [56]. Successfully identifying errors and managing edge cases such as momentary camera outages, loss of network connectivity between Node.js and Python microservices, database connection timeout, and unsupported image format occurred with appropriate error messaging on the end-user interface and detailed logging by backend systems.

6. Security and Privacy Considerations

Security measures protect sensitive biometric data and prevent unauthorized system access [19]. The MySQL database implements encryption at rest for facial embeddings and student records using AES-256 encryption. Database connections from Node.js utilize SSL/TLS encryption to protect data in transit. User authentication employs bcrypt password hashing with salt rounds of 10, ensuring secure credential storage. JWT tokens with 24-hour expiration provide stateless authentication [49], with the Node.js server validating signatures on each request.

Role-based access control implemented in both Node.js middleware and MySQL permissions ensures instructors access only their course data while administrators obtain institution-wide visibility. The React frontend implements HTTPS-only communication enforced through Content Security Policy headers. Input validation on both frontend and backend prevents SQL injection attacks, with the mysql2 library using prepared statements for all database queries. Rate limiting middleware in Express.js prevents brute force authentication attempts and denial-of-service attacks [64].

Privacy protection measures include storing only facial embeddings rather than complete images in the primary attendance database, with original enrollment photos maintained in restricted storage accessible only to authorized administrators. The system implements data retention policies automatically deleting outdated records according to institutional guidelines [65]. Students receive clear notification during enrollment regarding data collection purposes, retention periods, and their rights to access or request deletion of their biometric data. All system operations are logged in audit tables within MySQL, enabling security event investigation and compliance verification.

7. Conclusion

A comprehensive student attendance monitoring system based on face recognition is introducing deep learning and modern web technologies into an automated solution for monitoring student attendance. This system is a complete solution to many limitations of traditional methods of tracking student attendance. Through a single, integrated system, the solution includes student registration via a React frontend; real-time attendance monitoring via the coordination of the Node.js back-end; face recognition via TensorFlow-based convolutional neural network processing; OpenCV image processing, and reliable storage and retrieval of attendance data via MySQL. The system has a three-

tier architecture that allows the separation of the React frontend, Node.js back-end, and Python-based face recognition engine, providing the potential for scalability, maintainability, and efficiency in the use of resources.

The experimental results of testing the effectiveness of the system indicate that 94.2% of identified faces within the classroom matched with the student's previously enrolled photo. There was effective duplicate removal from the list of registered faces, effective handling of unidentified persons within the classroom, and an average performance time of 180 milliseconds between the initiation of a request from the student's face recognition to the time of displaying a result back to the student. The potential benefits to the educational institution for using this system include time saving, elimination of proxy attendance, improvements in the efficiency of administrative processes, and a system suitable for contactless operation in today's educational institutions. Biometric data will be securely stored and accessed through MySQL data encryption, use of JSON web tokens for authentication, and implementation of role-based access to records.

This research provided evidence that by integrating web development frameworks like React and Node.js with machine-learning algorithm libraries (such as TensorFlow and OpenCV), practical working solutions exist for use in automating the administration of educational institutions. The modular architecture of the system is designed to allow future enhancements to include liveness detection, cloud computing, and/or mobile applications. This research will contribute to the continuing evolution of automation technology in attendance tracking systems.

References

1. V. Shehu and A. Dika, "Using Real Time Computer Vision Algorithms in Automatic Attendance Management Systems," *International Conference on Information Technology*, pp. 397-402, 2010.

2. N. Kar, M. K. Debbarma, A. Saha, and D. R. Pal, "Study of Implementing Automated Attendance System Using Face Recognition Technique," *International Journal of Computer and Communication Engineering*, vol. 1, no. 2, pp. 100-103, 2012.

3. J. Lukas and M. Fridrich, "Estimation of Primary Quantization Matrix in Double Compressed JPEG Images," *Digital Forensic Research Workshop*, 2003.

4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

5. W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399-458, 2003.

6. A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4-20, 2004.

7. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015.

8. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701-1708, 2014.

9. K. Niinuma, U. Park, and A. K. Jain, "Soft Biometric Traits for Continuous User Authentication," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 771-780, 2010.

10. L. Wiskott, J. M. Fellous, N. Krüger, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775-779, 1997.

11. P. Wagh, R. Thakare, J. Chaudhari, and S. Patil, "Attendance System Based on Face Recognition Using Eigen Faces and PCA Algorithms," *International Conference on Green Computing and Internet of Things*, pp. 303-308, 2015.

12. M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

13. S. Sawhney, K. Kacker, S. Jain, S. N. Singh, and R. Garg, "Real-Time Smart Attendance System Using Face Recognition Techniques," *9th International Conference on Cloud Computing, Data Science & Engineering*, pp. 522-525, 2019.

14. A. Kumar and S. K. Gupta, "Contactless Attendance System Using Face Recognition," *International Journal of Engineering Research & Technology*, vol. 10, no. 5, 2021.

15. R. Deshmukh, "Automated Attendance Management System Using Face Recognition," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 3, pp. 74-78, 2017.

16. O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," *British Machine Vision Conference*, pp. 41.1-41.12, 2015.

17. C. Ding and D. Tao, "A Comprehensive Survey on Pose-Invariant Face Recognition," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 3, pp. 1-42, 2016.

18. H. K. Jee, S. U. Jung, and J. H. Yoo, "Liveness Detection for Embedded Face Recognition System," *International Journal of Biological and Medical Sciences*, vol. 1, no. 4, pp. 235-238, 2006.

19. A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*, Springer Science & Business Media, 2007.

20. D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, 2009.

21. J. Daugman, "How Iris Recognition Works," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 21-30, 2004.

22. C. Roberts, "Biometric Attack Vectors and Defences," *Computers & Security*, vol. 26, no. 1, pp. 14-25, 2007.

23. R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2021.

24. P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511-518, 2001.

25. P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, 1997.

26. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

27. J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, vol. 61, pp. 85-117, 2015.

28. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.

29. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations*, 2015.

30. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

31. F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815-823, 2015.

32. S. Chintalapati and M. V. Raghunadh, "Automated Attendance Management System Based on Face Recognition Algorithms," *IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-5, 2013.

33. M. Arsenovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "FaceTime - Deep Learning Based Face Recognition Attendance System," *IEEE SISY - Serbian-Hungarian Joint Symposium on Intelligent Systems*, pp. 53-58, 2017.

34. S. Joseph and K. Mathew, "Facial Recognition Based Attendance Management System," *International Research Journal of Engineering and Technology*, vol. 6, no. 3, pp. 1467-1471, 2019.

35. N. S. M. Hanif, M. F. Baharudin, and Z. A. Latif, "Deep Learning Based Facial Recognition Attendance System," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 5, pp. 2121-2129, 2020.

36. B. Bhattacharya, "Face Recognition Based Attendance System Using Machine Learning Algorithms," *International Conference on Intelligent Computing and Control Systems*, pp. 1-6, 2019.

37. K. Seshadri and F. Savvides, "Robust Modified Active Shape Model for Automatic Facial Landmark Annotation of Frontal Faces," *3rd IEEE International Conference on Biometrics: Theory, Applications and Systems*, pp. 1-8, 2009.

38. K. Khosla and N. Shekhawat, "Attendance Monitoring System Using Face Recognition," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 1547-1550, 2017.

39. V. Kumar and A. Sharma, "A Comprehensive Survey on Face Recognition Methods and Factors Affecting Facial Recognition Accuracy," *Artificial Intelligence Review*, vol. 54, pp. 1-50, 2021.

40. Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4-10, 2012.

41. B. Sun and D. Mu, "Research on Face Recognition Based on CNN," *IOP Conference Series: Materials Science and Engineering*, vol. 466, 2018.

42. S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80-83, 2010.

43. R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*, Wiley, 2009.

44. M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.

45. "React - A JavaScript Library for Building User Interfaces," Facebook Inc., [Online]. Available: https://reactjs.org/

46. A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*, Digital Codex LLC, 2012.

47. "Node.js Documentation," Node.js Foundation, [Online]. Available: https://nodejs.org/

48. G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120-125, 2000.

49. M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015.

50. M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," *12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 265-283, 2016.

51. V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *27th International Conference on Machine Learning*, pp. 807-814, 2010.

52. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

53. S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22-30, 2011.

54. "MySQL 8.0 Reference Manual," Oracle Corporation, [Online]. Available: https://dev.mysql.com/doc/

55. C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 60, 2019.

56. S. Marcel, M. S. Nixon, and S. Z. Li, *Handbook of Biometric Anti-Spoofing*, Springer, 2014.

57. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2011.

58. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 2015.

59. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2016.

60. D. M. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011.

61. X. Tan, S. Chen, Z. H. Zhou, and F. Zhang, "Face Recognition from a Single Image Per Person: A Survey," *Pattern Recognition*, vol. 39, no. 9, pp. 1725-1745, 2006.

62. T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, 2006.

63. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

64. M. Stamp, *Information Security: Principles and Practice*, John Wiley & Sons, 2011.

65. P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR)*, Springer, 2017.

66. Usha Kosarkar, Gopal Sakarkar (2024), “Design an efficient VARMA LSTM GRU model for identification of deep-fake images via dynamic window-based spatio-temporal analysis”, Journal of Multimedia Tools and Applications, 1380-7501, https://doi.org/10.1007/s11042-024-19220-w

67. Anupam Chaube, Usha Kosarkar “Enhanced Deep Learning-Based Feature Analysis for Copy-Move Forgery Detection”, 2024 Journal of Information Systems Engineering and Management,9(4s) e-ISSN:2468-4376, https://www.jisem-journal.com/