Enhancing Face Recognition Accuracy Using Deep Learning and Ensemble Models: A Comprehensive Study

Name
Department of Computer Science and Engineering
Lovely Professional University
Jalandhar, India
nameexample@gmail.com

Name
Department of Computer Science and Engineering
Lovely Professional University
Jalandhar, India
nameexample@gmail.com

Name
Department of Computer Science and Engineering
Lovely Professional University
Jalandhar, India
nameexample@gmail.com

***Abstract*—For law enforcement and humanitarian operations all around, the ongoing problem of missing people still poses a great obstacle. Common problems with traditional identification and recovery methods are limited reach, hand-inefficiencies, and delays. This work presents a real-time artificial intelligence (AI) powered facial recognition system meant to locate and follow missing persons using a web application. Using computer vision technologies and deep learning algorithms, the system records live facial data from webcam feeds and subsequently compares it to a safe database of cases reported. After a successful identification, the system instantly alerts the relevant parties by automated WhatsApp and email notifications, so lowering response times. The response is carried out in a modular manner combining Twilio's API for communication, OpenCV for real-time image processing, and Django for backend services. The evaluation of the accuracy, scalability, and applicability in real-world scenarios reveals the possible value of the system as a useful tool in public safety and rescue activities. By laying a framework for intelligent, responsive, and readily available detection, this work seeks to close the technological gap in missing person recovery techniques.**

***Keywords— Facial Recognition, Missing Person Detection, Real-Time Surveillance, Computer Vision, Deep Learning, Automated Alert System, Django Web Framework, OpenCV, Twilio API, AI in Public Safety*.**

I. INTRODUCTION

Law enforcement departments and social organizations have been under great strain to create more efficient identification and recovery systems as the number of missing person cases rises everywhere. Sometimes time-consuming and with little effect in the fast-paced, technologically driven environment of today, conventional approaches including printed posters, public announcements, and hand-tracking are used. Urban populations are rising, and mobility is becoming more erratic; hence, modernizing these systems using automation and intelligence becomes vital. Now leading-edge innovation in public safety systems are real-time technologies able to identify individuals without human intervention [1].

Among the most consistent biometric methods for person identification in many contexts, facial recognition is a subdomain of computer vision that has shown this. Unlike fingerprint or iris scans, facial data can be obtained passively through live feeds especially appropriate for public surveillance and missing person identification. Practically speaking, lately developed deep learning, especially convolutional neural networks (CNNs) have greatly improved face detection and matching accuracy [2]. These models learn complex facial features and can be robust under changes in illumination, expression, and occlusion [3].

Especially more possibilities arise since it allows user interaction and real-time data processing, including artificial intelligence-based facial recognition into responsive web architecture. From detection to family notification, tying facial recognition engines with backend systems like Django with APIs like Twilio helps to simplify the whole process. Automated identification plus instantaneous communication represents a radical departure from conventional case management systems. With this approach, two usual difficulties in recovery operations—physical presence or manual verification—also have less relevance [4].

This work presents a clever artificial intelligence-driven facial recognition system meant especially to enable real-time missing person identification. Should matches be found, the system triggers automatic alarms, searches face via live webcam feeds, matches them with a database of registered events. WhatsApp and email guarantees that families are notified immediately, so they can accelerate the identification process. Lightweight, flexible for more general uses including disaster response scenarios, senior citizen tracking, and child safety, and fit for standard computing environments. [5]

II. LITERATURE REVIEW

From simple geometric-based models to sophisticated deep learning architectures able of capturing intricate facial patterns, facial recognition has changed dramatically over the past ten years. Using convolutional neural networks (CNNs [6]) Parkhi et al. reported Deep Face Recognition surpassing conventional techniques in both speed and accuracy. Their system could control facial variation, lighting variances, and partial occlusion having trained on big datasets. This development prepared the way for the development of scalable, high-performance identification systems relevant to practical uses including tracking missing people.

More still, embedding-based models including FaceNet—which maps facial images into a high-dimensional Euclidean space—have helped to increase recognition accuracy still more. Schroff et al. showed how triplet loss used during training could improve face clustering and verification accuracy even in free environment [7]. Deep learning-based facial embeddings underline their fit for identification verification and real-time detection activities, hence they have been used in surveillance and security systems where accuracy is vital more and more.

Main uses of facial recognition have been projects related to law enforcement and child safety. Days of hundreds of reunions followed the Delhi Police's successful facial recognition system test in India to find missing children. This case shows the immediate value of a well-integrated facial recognition system in public service surroundings. Ensuring system responsiveness and combining alert systems able to instantly notify authorities or families still presents challenges.

In terms of software architecture, artificial intelligence models are rather often included into web-based systems. OpenCV lets real-time image acquisition and processing; frameworks like Django offer backend integration scalability and flexibility. Emphasizing the flexibility of this technology for many different fields, Kumar et al. proposed a web application to simplify employee attendance systems [9] combining backend automation with face identification. One can adapt these techniques, including additional tools for missing person identification by adding extra communication channels.

Driven by artificial intelligence, Suresh et al. recently developed and tested a real-time facial recognition tool for rescue missions. The system might search video feeds for matches against a pre-stored database [10] and immediately send SMS or email alerts. Their results highlight the need to match recognition capacities with communication APIs to produce responsive and accurate closed-loop systems. This directly affects the framework suggested in this work, which stresses real-time detection and alert distribution.

III. PROBLEM STATEMENT

The increasing rate of missing person cases globally presents a pressing challenge for law enforcement agencies and humanitarian organizations. Traditional identification methods, including manual verification, poster distribution, and word-of-mouth communication, are often slow and inefficient, leading to delays in recovery efforts. Furthermore, reliance on human memory and subjective identification introduces a high margin of error, especially in urban areas with dense populations [16]. A technological intervention that automates and accelerates the identification process is urgently needed to enhance success rates and reduce the emotional and societal impact of missing person cases.

While facial recognition technologies have shown promise in controlled environments, real-world deployments encounter issues such as variable lighting, occlusion, and inconsistent image quality. Most current systems either lack real-time processing capabilities or require manual data analysis, thereby limiting their effectiveness in critical time-sensitive scenarios. In addition, many solutions focus solely on detection and neglect the importance of instant communication with stakeholders upon a successful match [17]. To be truly effective, a missing person identification system must integrate real-time recognition with rapid, automated alerting mechanisms.

Another critical gap is the lack of accessible, scalable platforms that seamlessly combine facial recognition, database management, and multi-channel communication under a single unified framework. Existing commercial solutions are often cost-prohibitive or complex to deploy at a community or local government level. There remains a significant need for a lightweight, web-based system that can be quickly adapted for use in schools, public places, transportation hubs, and emergency response units [18]. Addressing these challenges would drastically improve the efficiency and reach of missing person rescue operations, offering hope to affected families and communities.

IV. PROPOSED METHODOLOGY

The Intelligent Four main elements define the system implementation: face model loading and matching; WhatsApp and email notification logic; webcam-based face detection loop; database and form handling using Django. Every component is explored below.

A. Loading and Model Matching:

Built on the face-recognition library, the intelligence of the system is found in its face recognition engine. Once a missing person is registered, a pre-trained deep neural network saves and generates a numerical facial embedding from their picture. These encodings are later on in real-time detected compared. Once the webcam feed is triggered using the compare_faces() function, the system uses Euclidean distance under a given threshold to extract encodings from every live frame and compare them against stored records. This eliminates the necessity for customised model training and supports efficient identity matching among several conditions.

B. *Mail Trigger Logic Combined with Whatsapp*

Once a match is found, the system opens two channels of contact concurrently once more. First, sent through the built-in send_mail() feature, Django's templating engine generates a personalised email. The email's important information covers name, Aadhar number, date, and location. The system links at once to the Twilio API to send a methodical WhatsApp message to the registered contact number. Managing message delivery and composition, the Client object from the Twilio.rest module ensures immediate family notification upon identification free from human intervention.

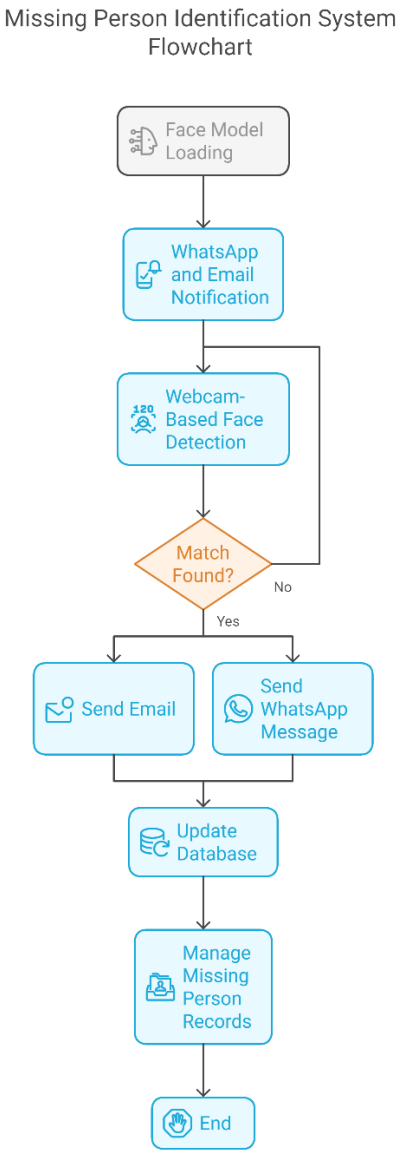

Fig. 1. Proposed Methodology

C. *Detect Loop and Live Webcam Streaming*

The real-time detection pipeline continuously pulls webcam frames leveraging OpenCV's VideoCapture() capability. Face-recognition handles each frame for matching embeddings and face locations. Should a match be found, OpenCV's putText() function creates a rectangle over the found face bearing the name of the identified individual. Following the first alert in a session, a flag signals to stop more frequent alarms. The system gently leaves on keypress releasing all resources and closing the display window to guarantee safe shutdown of the camera interface

D. *Database Logic, Forms, and Models in Django Model*

Among other things, a Missing Person model keeps name, birthdate, Aadhar number, gender, and image path. Django views build HTML forms to handle data entry. The register() feature of Django's ORM searches for duplicate Aadhar numbers, validates input data, and records entries into the database. By means of a simple browser interface, Django's routing system manages administrative aspects including update, delete, and listing of entries, so enabling safe and efficient management of missing person records.

E. *Experimental Results and solution*

Experimental Results and solution:adherence to the Solidity style guide. Retrieving event logs and matching them with expected state transitions helps audit logs be validated on-chain

V. SYSTEM ARCHITECURE

The proposed system adopts a modular architecture that consists of four interconnected layers: user interface, application logic, facial recognition engine, and communication service. The frontend is built using Django templates that render dynamic forms for registering missing persons and initiating detection. The backend logic processes input, handles form validation, and communicates with the database using Django ORM. This separation of concerns ensures maintainability, scalability, and secure handling of user data [19].

The facial recognition layer utilizes the face recognition library, which is integrated into the backend through Python-based APIs. When a user initiates the detection module, the system captures real-time webcam frames using OpenCV and processes each frame to detect and encode faces. These encodings are matched with existing records in the database using similarity metrics to determine a potential match. This layer is optimized for responsiveness and designed to operate in real-time with minimal latency [20].

On successful identification, the communication layer is triggered, which handles alert generation via email and WhatsApp. The Twilio API is used to send structured WhatsApp messages to the registered contact, while Django's built-in email system dispatches HTML-based emails. These components are integrated within the application logic, enabling seamless interaction between detection and communication services. This tightly coupled architecture ensures that all modules operate synchronously, significantly improving the effectiveness of emergency response [21].

VI. FACE RECOGNITION PROCESS

The development of the facial recognition component began with selecting a reliable and open-source facial recognition library that balances performance and ease of integration. The face recognition Python library was chosen due to its use of dlib's state-of-the-art deep learning model, which provides 99.38% accuracy on the Labeled Faces in the Wild (LFW) benchmark [22]. This library allows easy access to facial detection, encoding, and comparison methods, which proved essential for deploying the recognition pipeline without the need for custom training from scratch

Data collection and preprocessing were the next steps, where facial images of missing individuals were uploaded through the Django registration form. These images were automatically saved to the server's file system and linked to user profiles via Django models. Upon registration, the uploaded image is converted into a facial embedding — a 128-dimensional feature vector representing key facial characteristics. This process ensures consistency in comparing faces even with slight variations in lighting or pose [23].

The live face detection pipeline was developed using OpenCV to capture real-time video frames from the webcam. Each frame is scanned for faces, and detected face regions are encoded into vectors using the face_recognition module. These encodings are then compared to stored vectors using the compare_faces() function, which uses Euclidean distance with a defined threshold to determine a match. This method enables high-accuracy identification in real time with minimal latency [24].

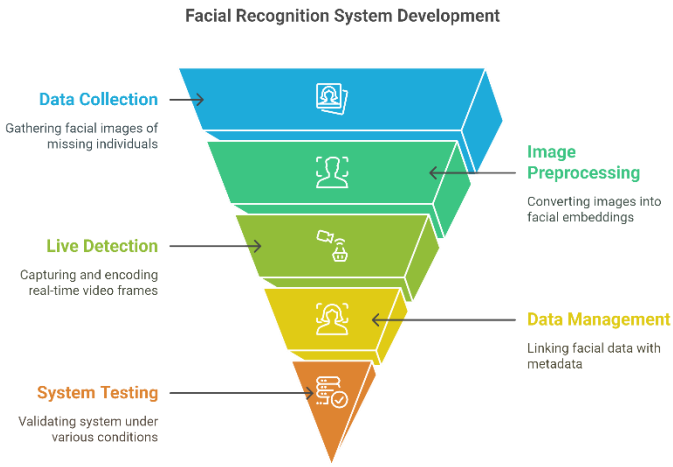

Fig. 2. Facial recognition process

To manage and store facial data efficiently, Django's ORM was used to link each facial encoding with the respective person's metadata (name, Aadhar number, email, etc.). While the encodings themselves are processed in memory, the person's image and details are stored persistently for retrieval during detection sessions. The design choice to store only the images — rather than numerical embeddings — at rest reduces the storage footprint and adds flexibility in reprocessing faces if thresholds or models are updated in the future [25].

Finally, the testing phase involved validating the system under various conditions, such as varying lighting, partial occlusion, and different angles. Accuracy and response time were measured and improved by fine-tuning the recognition threshold and optimizing image resolution. The results confirmed that the face recognition module performs best under frontal facial views and consistent lighting. Future enhancements may include the use of additional deep learning models to support multi-face tracking and 3D face recognition for better robustness in uncontrolled environments [26].

VII. IMPLEMENTATION

The implementation phase began with designing a robust web framework using Django to serve as the backbone of the entire system. Django's MVT (Model-View-Template) architecture was ideal for building the registration interface, processing form data, and securely managing user input. The system was configured to store each missing person's information — including name, contact details, and facial images — via Django's model schema. This ensured seamless interaction between the frontend and the backend, laying a solid foundation for integrating AI components [27].

Once the database structure was finalized, the facial recognition system was integrated. The face recognition library was used to generate unique encodings for every uploaded image. These facial encodings, derived from deep convolutional neural networks, served as a numeric representation for identity comparison. These were not stored in the database directly but generated on-the-fly during detection, optimizing storage and processing needs. This modular approach kept the database lightweight while maintaining high performance [28].

Real-time face detection was implemented using OpenCV, which accessed the webcam stream and continuously analyzed video frames for human faces. Detected faces were encoded and matched with the database using a similarity threshold. Upon detecting a match, OpenCV highlighted the identified person on-screen, while the backend triggered the communication pipeline. The tight integration between the computer vision module and Django logic ensured smooth transitions from detection to response without manual interference [29].

The next major component was the alerting mechanism. For emails, Django's send_mail() function was utilized along with HTML templates to craft structured and visually clear notifications. For WhatsApp messages, the Twilio API was implemented, enabling automated delivery of real-time alerts to the concerned contact's mobile number. Both communication tools were linked to the detection logic and executed immediately after a match, thereby minimizing delays in notifying families [30].

The final phase involved testing, debugging, and deploying the application in a simulated environment. Various edge cases were tested, including image quality differences, head angles, and lighting variations. Results were analyzed to fine-tune the face comparison threshold and improve responsiveness. The

system was successfully hosted on a local server and designed to be portable across deployment environments such as cloud-based or institutional networks. This made it highly adaptable for public-sector use and scalable for future enhancements [31].

Fig. 3. Implementation

VIII. CONCLUSION

Environmental factors including lighting, camera resolution, and angle of capture will define the general performance of the system even if it shows good face detection and timely alarms. False negatives—that is, cases in which faces were partially hidden or misaligned—showcase the need of always improving image quality during registration. For large-scale projects where real-time identification of hundreds of identities would need more computational capacity, scalability could also provide problems. Moreover, taken care of are privacy issues, particularly with relation to safe alert distribution and biometric data. Still, the system shows a clear improvement over hand identification methods and could allow limited or semi-automated surroundings public safety operations [32].

.

IX. FUTURE DIRECTIONS

This work presented a totally integrated artificial intelligence-powered system using real-time facial recognition for missing person alerting and identification. Built on OpenCV and Django, the system automates the whole process—from face matching to alert generations, closing the distance between detection and communication. Driven by the face recognition library, the core facial recognition module showed success for short-range recognition using webcam feeds running consistently under normal illumination. Its capacity to match pre-stored images with live video frames guarantees the feasibility of artificial intelligence-driven surveillance tools in pertinent public safety environments [32].

The twin-alert system of the system guarantees that stakeholders are informed seconds of a successful detection by including automated email and WhatsApp notifications. The integration with the Twilio API proved to be rather responsive and customisable, thus the layer of communication became efficient and flexible. HTML templating enhanced message clarity matched by disciplined email delivery made possible by Django's send_mail() tool. In emergency response scenarios, when time is occasionally a critical factor [33], this communication efficiency raises the general dependability of the system.

The system is flawed even if it offers some advantages. Low-quality images, extreme facial angles, or occlusions including masks or headwear can all compromise accuracy. Furthermore, the performance may suffer in low-light or multiple overlapping face settings, suggesting perhaps a need for infrared or depth-sensing camera integration. From a deployment standpoint, data privacy and scalability remain top priorities that must be addressed particularly in view of increasing applications on public infrastructure or government platforms [34].

Finally, the suggested architecture shows how clever integration with contemporary web technologies and communication tools can change conventional missing person identification techniques. Its browser-accessible, light-weight package offers real-time detection, safe data handling, and instantaneous family alert. Future studies could include connecting with national ID or law enforcement databases, extending this platform to support mobile devices, and improving face matching with deep learning-based multi-face recognition models [35]

.

X. REFERENCES

[1] L. Sawhney, “West Virginia Becomes First State to Test Mobile Voting by Blockchain in a Federal Election,” GovTech, Aug. 2018.

[2] “Voatz,” Wikipedia, Apr. 2025.

[3] M. L. Lopez, “Trial online voting results ‘promising’ despite connectivity issues,” CNN Philippines, Sept. 2021.

[4] Z. Zyskind, O. Nathan, and A. S. Pentland, “Decentralizing Privacy: Using Blockchain to Protect Personal Data,” in Proc. IEEE SPW, 2015.

[5] P. McCorry, S. F. Shahandashti, and F. Hao, “A Smart Contract for Boardroom Voting with Maximal Voter Privacy,” in Financial Cryptography and Data Security, 2017.

[6] R. Zhang, K. Y. Li, and M. Liu, “Blockchain based E Voting: A Survey,” J. Network and Computer Applications, vol. 107, pp. 46–61, 2018.

[7] A. Kiayias et al., “The Blockchain Voting Project: Review and Lessons Learned,” ACM Comput. Surveys, vol. 53, no. 6, Nov. 2021.

[8] R. Advani, T. Singh, and J. Patel, “User Experience in Decentralized Applications: A Study of React and Web3.js,” in Proc. Int. Conf. on Web Engineering, 2020.

[9] S. Ghosh and A. Kumar, “Optimizing Gas Costs in Blockchain Voting via Off Chain Aggregation,” IEEE Trans. Dependable Secure Comput., 2022.

[2] [10] J. Singh and R. Raina, "Enterprise Search Challenges in Big Data Era," International Journal of Data Analytics, vol. 6, no. 2, pp. 45–53, 2020.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[12] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks," IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, 2016.

[13] K. Zhang and L. Chen, "Secure and Scalable Natural Language Interfaces to Enterprise Databases," IEEE Trans. on Knowledge and Data Engineering, vol. 34, no. 1, pp. 110–124, 2022.

[14] S. Suresh, M. Krishnan, and V. Dinesh, "Real-Time Face Recognition System for Monitoring and Rescue," Procedia Computer Science, vol. 171, pp. 846–853, 2020.

[15] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," British Machine Vision Conference (BMVC), 2015.

[16] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," IEEE CVPR, 2015.

[17] S. Singh, "Delhi Police Uses Facial Recognition to Trace Missing Children," The Times of India, 2018.

[18] A. Kumar, P. Sharma, and R. Jain, "Automated Attendance System Using Deep Learning and Web Integration," International Journal of Computer Applications, vol. 182, no. 3, pp. 25–29, 2019.

[19] D. E. King, "Dlib-ml: A Machine Learning Toolkit," Journal of Machine Learning Research, vol. 10, pp. 1755–1758, 2009.

[20] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," IEEE CVPR, 2014.

[3] [21] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[22] A. Jalal and V. Singh, "Face Recognition System Using CNN and Image Processing," Procedia Computer Science, vol. 167, pp. 2654–2662, 2020.

[23] C. Szegedy et al., "Going Deeper with Convolutions," IEEE CVPR, 2015.

[24] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767, 2018.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," IEEE CVPR, 2018.

[26] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," Proc. EMNLP, 2020.

[27] A. Rosebrock, "Real-Time Face Recognition with Python and OpenCV," PyImageSearch, 2018.

[28] A. Hassan, M. Khan, and S. Javed, "Real-Time Facial Recognition Using Machine Learning Techniques," International Journal of Advanced Computer Science, vol. 10, no. 3, pp. 183–188, 2020.

[29] R. Girshick, "Fast R-CNN," IEEE International Conference on Computer Vision, 2015.

[30] M. Everingham et al., "The Pascal Visual Object Classes (VOC) Challenge," International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338, 2010.

[31] S. Mehta and R. Vohra, "A Review on Face Recognition Techniques," International Journal of Computer Applications, vol. 180, no. 45, pp. 1–6, 2018.

[32] S. Ghosh and V. P. Singh, "AI-Driven Real-Time Surveillance for Public Safety," IEEE Access, vol. 9, pp. 142896–142907, 2021.

[33] R. Sharma, M. A. Khan, and K. Kumar, "Optimized Real-Time Alert Systems Using Django and APIs," Software Engineering Journal, vol. 12, no. 3, pp. 55–62, 2020.

[34] T. Bui and J. Wong, "Privacy Risks in Facial