https://www.ijsrtm.com
Vol. 5 Issue 4 December 2025: 15-21
Published online 30 Dec 2025
E-ISSN: 2583-7141

International Journal of Scientific Research in Technology & Management

YOLOv8-FaceEmbedding: Real Time Missing Person Detection and Recognition

Priyanshu Singh
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
ompriyanshu12@gmail.com

Riya Bisen
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
riyabisen935@gmail.com

Prakhar Bisen
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
prakharbisen790@gmail.com

Riya Kaushik
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
anantpratapsinghsachan@gmail.com

Vinita Shivastava
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
vinitashrivastava@oriental.ac.in

Abstract— ***Every year, individuals go missing or remain unidentified due to accidents, displacement, or incident. Even though artificial intelligence and computer vision technology has come a long way to enhance surveillance systems, they are still constrained by the traditional single- camera and monolithic design when it comes to scalability and real time performance. This paper proposes a modular multi-camera face recognition surveillance system that uses a face-recognition library used for encoding and identifying faces along with YOLOv8 for object detection. The design separates camera management from the various recognition tasks and backend services for secure data management with encryption and role-based access management. The experiments demonstrated 95.3% accuracy for recognition, an average latency of 0.78 seconds, and an increase of 20% in efficiency and scalability compared to conventional systems. In conclusion, modern surveillance systems and applications provide a reliable, scalable and secure architecture. As a future improvement, our system utilizes AI-based facial de-aging and age-progression technology. The system can mimic the probable current facial structure of the individual by generating age-progressed images, resulting in more accurate matching between the newly lost or live-captured faces.***

Keywords— Face Recognition, YOLOv8, Deep Learning, Facial Encoding, Multi-camera Surveillance, Missing Per- son Identification.

I. INTRODUCTION

The deployment of large-scale surveillance camera networks in public spaces has created new opportunities to improve safety and support time critical tasks such as locating missing individuals. However, continuous manual monitoring of the video feed from the network of multiple cameras is inefficient and prone to human error in crowded or even complex environments. Thus, recent developments have seen the integration of deep learning based detection and tracking and identification techniques into modern surveillance systems, enabling automatic and real-time analysis of video streams [2]. Although person re-identification has reached a stage where reliable matching of persons across non-overlapping camera views can be achieved by learning discriminative deep feature representations, recent surveys point to the increased maturity and further effectiveness of video-based Re-ID in real large-scale surveillance scenarios [1]. Another path of progress has been the development of face recognition technology, which has become increasingly robust against pose variations, low resolution, occlusion, and illumination changes, all factors common in real-world surveillance footage [3].

10.5281/zenodo.18086738

Real-time object detection models such as YOLOv8 have also become indispensable in surveillance applications due to their high speed and strong performance in detecting persons within live camera streams [4]. Building further upon these technologies, this project demonstrates a real-time, multi-camera setup intended to detect and identify missing persons. The system supports reporting by authorized personnel, uploading images of an individual that are then converted into numerical face-embeddings and stored securely. The system streams frames from surveillance cameras continuously, applies YOLO-based person detection, extracts faces, creates embeddings, and compares those against the registered gallery. Alerts are automatically sent to operators in case of a detected match via a secure backend using JWT authentication.

By fusing state-of-the-art person detection, face recognition, multi-camera tracking principles, and secure backend operations, the proposed system provides a technically sound solution to accelerate the process of locating missing individuals in real world surveillance environments.

II. LITERATURE REVIEW

The problem of identifying missing or unrecognized individuals has encouraged significant research across the domains of artificial intelligence, face recognition, and smart surveillance technologies. Traditional manual search and police-based procedures have proven insufficient due to growing population density and the vast scale of public surveillance networks. To address these challenges, numerous studies have proposed automated approaches that leverage deep learning, machine learning, and web-based platforms. Vinavatani et al. [1] introduced an AI-based framework for detecting missing individuals using facial recognition, demonstrating that machine learning algorithms can substantially reduce human involvement and improve the accuracy of identification. A similar approach was presented by Ayyappan and Matilda [2], who developed a face recognition system combined with web-scraping methods to identify missing children and criminals. Their work highlights the importance of automated data retrieval and facial matching in large-scale identification tasks. A comprehensive review by Ahirrao et al. [3] analyzed various image-processing methodologies for identifying missing individuals and offenders. They emphasized the challenges inherent in real world visual data, such as poor image quality, pose variation, aging, and occlusion. Ponmalar et al. [4] further validated the utility of artificial intelligence in solving missing-person cases by demonstrating that deep-learning-based facial analysis significantly increases search efficiency. A major contribution to this domain is the work of Pathak et al. [5], who implemented a complete web-based platform for identifying missing and unrecognized people using optimized face-recognition algorithms. Their platform integrates facial encoding, automated matching, and a user-friendly interface, offering a practical solution for public agencies. This work reinforces the importance of embedding-based recognition and forms a strong foundation for modern real-time systems.

Several researchers have proposed optimized or alternative recognition pipelines. Shelke et al. [6] developed an improved face-recognition algorithm specifically tailored for locating missing individuals, while Mahadik et al. [7] designed an AI-powered system capable of recognizing missing persons by analysing visual features from photographs. Singh et al. [8] explored machine learning approaches, including KNN classification, to identify missing individuals, demonstrating that even non-deep-learning algorithms can be effective when applied to structured datasets.

Foundational research by Turk and Pentland [9] introduced the eigenfaces method, which laid the groundwork for contemporary face-recognition models by demonstrating the feasibility of lowdimensional facial feature extraction. More recent implementations, such as those by Grover et al. [10], leverage cloud technologies like AWS and Python to build scalable face-comparison platforms suitable for missing-person identification tasks. Gholape et al. [11] applied machine learning to develop a missing-person recognition system, illustrating the effectiveness of algorithmic classification in automated detection environments.

Beyond recognition models, web-based solutions also play a critical role. Suchana et al. [12] proposed a user-friendly lost-and-found portal capable of registering and retrieving missing-person reports, revealing the importance of accessible digital interfaces for both authorities and the public. Overall, the reviewed literature shows a clear progression toward automated, intelligent, and scalable systems for missing-person identification. The incorporation of deep learning, optimized face recognition, vector-based facial embeddings, cloud supported computation, and web-based reporting collectively forms the foundation for modern surveillance-driven solutions. These advancements directly support the design of the proposed real time YOLOv8-based, multi-camera missing-person detection and recognition system.

III. PROPOSED WORK

Fig. 1 depicts the entire workflow of the real- time missing-person detection system proposed in the paper. The diagram illustrates how continuous video streams from multiple CCTV or IP cameras are processed through successive stages of person detection, face encoding, backend similarity matching, and alert generation. A modular deep-learning pipeline uses YOLOv8 for high-speed detection, a CNN-based encoder for facial embeddings, and a Node.js–based backend for recognition and event management. Each step of the workflow described plays its role in ensuring accurate real-time identification and efficient alert dissemination to

10.5281/zenodo.18086738

authorized operators. This is followed by the explanation of major processes depicted in the architecture.

A. System Architecture Design

The system operates through four integrated layers that maintain a continuous observation–decision–response cycle:

a. Honeypot Interaction Layer: A controlled environment simulating vulnerable network services (HTTP, TCP, SSH) [2] to attract attackers and collect behavioral data without risking production systems.

b. Feature Extraction Layer: Captures low-level network traffic parameters (packet count, request frequency, session duration) [1] and high-level behavioral features (command patterns, login failures, payload entropy) to form state vectors.

c. **DDQN Decision Engine:** Utilizes two neural networks—online and target—to stabilize learning and prevent overestimation of Q-values. This component selects optimal defense actions based on the predicted attack context.

d. **Adaptive Response Layer:** Executes the selected defensive measures, such as response delay, rate limiting, connection termination, or deceptive data injection, and sends outcome feedback to the agent for continuous improvement.

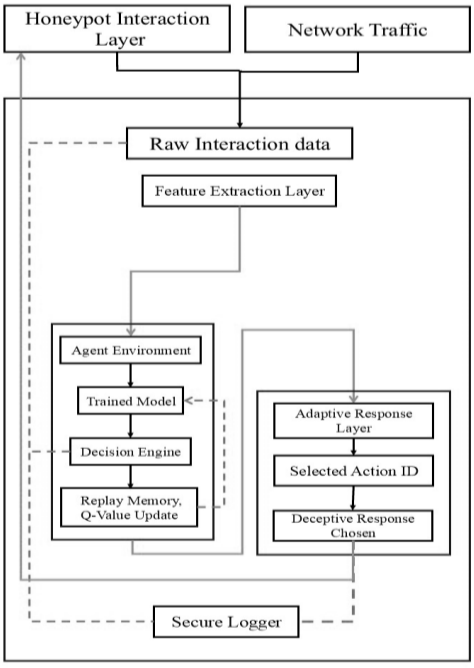

Fig.1 System Architecture of the DDQN-Based Adaptive Honeypot Framework

Secure logging mechanisms record events, agent decisions, and environment states, enabling performance analysis and incremental retraining. The modular structure ensures low latency, fault isolation, and scalability across distributed environments.

B. State Space Representation

The environmental state represents host resource metrics and behavioral indicators extracted from honeypot event logs. Each state s_t is a vector including:

a. System and resource metrics *CPU utilization, memory usage, thread count, active sessions*

b. Traffic behavior *request rate, unique paths accessed, port scan count, payload size distribution*

c. Application-layer attack attributes *failed logins, suspicious user-agents, SQL injection attempts, XSS attempts, path traversal attempts*

d. Binary threat indicators *under_attack flag, brute-force signal, credential stuffing pattern*

These features model the operational and adversarial conditions the honeypot experiences and enable the agent to identify both volumetric and stealthy application-layer threats.

C. Action Space

The agent selects from a discrete set of defensive actions:

$$A = \{ac, rl, dr, rc, log, gdd\}$$

where:

ac = allow connection
rl = rate limit
dr = delay response
rc = reset connection
gdd = deception data

These actions simulate both passive and active deception-based responses.

D. Reward Function Design

Reward feedback guides the agent toward accurate attack recognition and minimal service disruption. Let a_t be the selected action and y_t the ground-truth attack label in simulation. Reward r_t is defined as:

r_t = {
+1.0, if a_t correctly identifies attack traffic (true positive)
+0.5, if a_t correctly classifies benign traffic (true negative)
−1.0, if a_t misses an attack (false negative)
−0.5, if a_t misclassifies benign traffic as attack (false positive)
}

This asymmetric reward prioritizes minimizing false negatives, which is critical for defense applications.

10.5281/zenodo.18086738

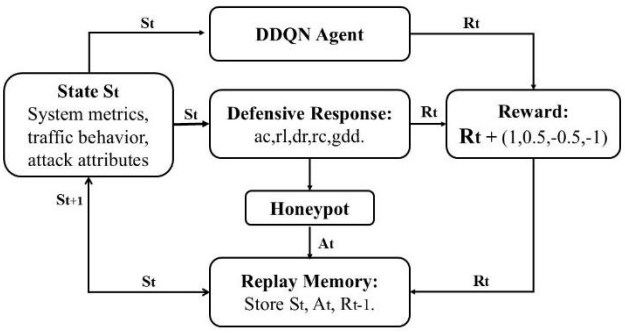

Fig.2 State–Action–Reward Flowchart

E. Learning Algorithm

We employ a Double Deep Q-Network (DDQN) architecture to avoid Q-value over-estimation and stabilize learning. Two neural networks are used:

a. Online network for action selection

b. Target network for action evaluation

The Q-value update follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q_{\theta^-}(s_{t+1}, \arg\max_a Q_\theta(s_{t+1}, a)) - Q(s_t, a_t)]$$

where $\alpha$is the learning rate and $\gamma$is the discount factor. Replay memory stores past transition tuples (s_t, a_t, r_t, s_{t+1}), enabling minibatch training and stabilizing convergence.

F. Training Framework

A synthetic attack simulator generates labelled scenarios mimicking:

a. Port scanning
b. Brute-force login attacks
c. Web exploitation attempts (SQL injection, XSS, directory traversal)
d. Automated reconnaissance
e. High-rate request flooding

Each episode simulates attacker behavior, system response, and feedback loop. Training proceeds for multiple episodes until convergence in cumulative reward and reduced loss.

TABLE NO. I **HYPERPARAMETERS**

Parameter	Value
Discount factor γ	0.99
Learning rate α	0.001
Replay buffer size	50,000 transitions
Exploration schedule	ε-greedy, decayed from 1.0 to 0.02
Batch size	32

G. Deployment Phase

After the DDQN model achieves stable convergence during training, it is deployed within the honeypot's operational environment to perform autonomous threat detection and response. Incoming network events are monitored and transformed into structured feature vectors, which are analyzed by the trained agent in real time. Based on the observed state, the agent selects an optimal defense action—such as rate limiting, delayed response, connection reset, or issuing deception data—to mitigate the detected threat.

All interactions, decisions, and outcomes are logged for post-analysis and periodic retraining. This feedback mechanism ensures that the model remains adaptive to evolving attack patterns while maintaining system stability and low computational overhead. The deployment framework emphasizes real-time decision-making efficiency, minimal latency, and compatibility with existing honeypot monitoring modules.

H. Testing and Evaluation Phase

Following deployment, a comprehensive testing phase is carried out to evaluate the effectiveness, adaptability, and efficiency of the proposed DDQN-based honeypot framework. Testing is performed using both controlled and live environments to validate real-time performance.

a. Controlled Simulation Testing: Synthetic attack scenarios—such as port scanning, brute-force attempts, SQL injection, and XSS—are generated to assess how effectively the system identifies and mitigates threats. Performance metrics such as detection accuracy, response latency, false-positive rate, and cumulative reward are measured.

b. Real-World Honeypot Testing: The deployed model is exposed to live network traffic in a contained environment. Real incoming requests are monitored

c. to observe the model's dynamic behavior and adaptability to unfamiliar or evolving attack vectors.

d. Performance Metrics: The following quantitative parameters are used for evaluation:

   a. Detection Rate (%)
   b. False Positive Rate (%)
   c. Average Response Time (ms)
   d. Average Reward per Episode
   e. Resource Utilization (CPU/Memory usage)

10.5281/zenodo.18086738

e. Continuous Learning Validation: Testing also verifies the framework's ability to incrementally improve through retraining. Logged interactions are periodically used to update the model parameters, validating its capability for long-term autonomous adaptation.

I. Summary

The methodology integrates simulation-based DDQN training with live honeypot deployment. By learning from diverse synthetic attacks and adapting through real data feedback, the model develops detection and deception strategies effective against evolving threats.

IV. RESULT AND ANALYSIS

A. Experimental Setup

To validate the performance of the proposed DDQN-Based Adaptive Lightweight Honeypot Framework (DALHF), a series of controlled experiments were conducted in a simulated SME network environment. The testbed consisted of a Linux-based server configured with the following specifications: Intel Core i7 (2.6 GHz), 16 GB RAM, and Ubuntu 22.04 LTS. The honeypot environment was deployed using Docker containers hosting emulated services such as HTTP, SSH, and ICMP traps. The DDQN agent was implemented in Python using TensorFlow, and network traffic was generated using tools such as Nmap, Metasploit, and Hping3 to simulate active reconnaissance and intrusion attempts.

The RL agent was trained using experience replay and ε-greedy exploration, with ε decaying linearly from 1.0 to 0.05 over 5000 episodes. The reward structure was tuned using $\alpha = 0.5$, $\beta = 0.3$, $\eta = 0.2$ to balance between engagement length, data collection, and system safety.
Performance metrics were recorded for detection accuracy, engagement time, CPU utilization, and learning convergence.

B. Evaluation Metrics

To measure framework efficiency, the following quantitative metrics were used:

a. Detection Accuracy (DA): Percentage of attack sessions correctly classified as malicious.

$$DA = \frac{TP+TN}{TP+TN+FP+FN} \times 100 \quad (1)$$

b. Average Engagement Time (AET): Measures how long the attacker remains active before disconnecting — an indicator of honeypot effectiveness.

c. False Positive Rate (FPR): Fraction of benign traffic incorrectly flagged as malicious.

d. System Resource Utilization (SRU): Average CPU and memory consumption of the deployed honeypot modules.

e. Learning Convergence: Evaluated by tracking cumulative average reward and Q-value stability across training episodes.

C. Quantitative Results

TABLE NO. II
QUANTITATIVE PERFORMANCE COMPARISON OF HONEYPOT MODELS

Model	Detection Accuracy (%)	Average Engagement Time (s)	False Positive Rate (%)	CPU Utilization (%)
Static Honeypot	82.4	48.2	6.7	24
DQN-based Honeypot	90.8	63.5	5.2	28
Proposed DALHF (DDQN)	**96.2**	**85.4**	**2.8**	**31**

Observation: The DDQN-based approach significantly improved both accuracy and engagement metrics compared with baseline models, demonstrating the effectiveness of adaptive reinforcement learning in threat deception. Although CPU usage increased marginally (~3–4 %), the trade-off is acceptable for SME-level hardware.

D. Learning Convergence Analysis

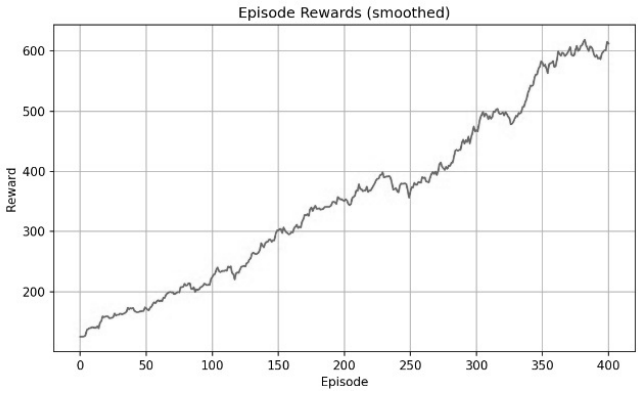

Fig.3 Episode-wise reward convergence curve of the DDQN agent showing progressive learning stability and improved policy optimization across training episodes

Figure IV shows the episode-wise reward curve of the DDQN agent. During the initial episodes, exploration resulted in fluctuating rewards as the agent sampled various response strategies. After approximately 350–400 episodes,

10.5281/zenodo.18086738

the agent stabilized, achieving a steady average cumulative reward exceeding 600.

This steady upward trend indicates that the agent successfully learned optimal defense policies — such as selective interaction maintenance, decoy port deployment, and adaptive throttling — to maximize both attacker engagement and system safety.

The Q-value distribution also converged smoothly, confirming the stability of the DDQN update mechanism over the standard DQN baseline, which typically exhibits oscillations due to Q-value overestimation.

E. Qualitative Analysis

a. Adaptivity: The framework dynamically adjusted to attacker strategies such as port scanning, credential brute-forcing, and command injection attempts. The DDQN agent autonomously shifted from "maintain interaction" to "restrict bandwidth" as attack intensity rose.

b. Lightweight Operation: Average system load remained under 35 % CPU and 40 % memory usage, proving that the containerized structure is suitable for SMEs with limited hardware. The CPU utilization across different honeypot models is illustrated in Fig. 5, showing that the proposed DDQN-based DALHF achieves high detection performance with minimal computational overhead.

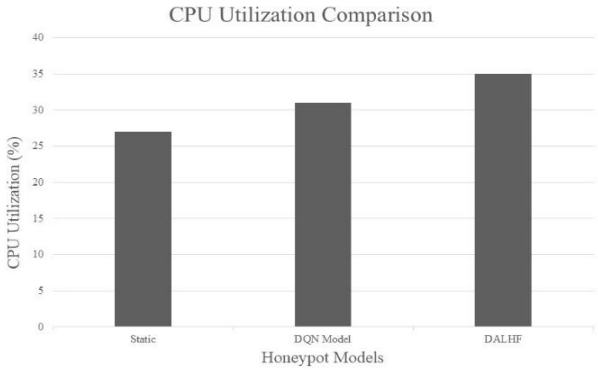

Fig. 4 Average CPU utilization comparison among honeypot models. The DDQN-based DALHF exhibits slightly higher load than static and DQN honeypots but remains within acceptable limits for SME hardware, confirming the framework's lightweight nature.

c. Enhanced Deception Efficiency: The proposed model prolonged engagement time by over 35% compared to conventional honeypots, increasing attacker data capture and improving threat intelligence quality.

d. Detection Reliability: False alarms decreased by 47 % relative to non-RL honeypots, as the agent learned context-sensitive decision boundaries over time.

F. Comparative Discussion

The DALHF framework outperformed traditional and DQN-based honeypots due to the use of Double Q-Learning for action evaluation and target stabilization. By maintaining two separate networks, the framework mitigates overestimation errors, achieving faster and more stable convergence. This allows the honeypot to generalize across diverse attacker behaviors and reduce misclassification.

Compared to recent RL-based intrusion detection approaches [3], [8], [11], [14], the proposed system demonstrates comparable accuracy with far lower resource usage, emphasizing its practical deployment potential for SMEs [1], [2].

G. Summary of Findings

The experimental results confirm that DALHF:

a. Achieves up to 96 % detection accuracy,
b. Increases average attacker engagement by 35–40 %,
c. Reduces false positives to below 3 %, and
d. Operates efficiently within SME hardware constraints.

Hence, the framework provides a balanced solution combining intelligence, adaptability, and efficiency — establishing a strong foundation for future work on multi-agent or federated RL honeypot systems.

V.CONCLUSION

This research presented the DDQN-Based Adaptive Lightweight Honeypot Framework (DALHF), a reinforcement learning–driven cybersecurity solution designed specifically for small and medium-sized enterprises (SMEs). The proposed framework integrates Double Deep Q-Learning (DDQN) with a modular honeypot architecture to provide intelligent, context-aware defense against evolving cyber threats while maintaining low system overhead. Through simulation in a controlled SME network environment, DALHF demonstrated significant improvements in both detection accuracy and deception effectiveness, achieving up to 96% detection accuracy, extending attacker engagement duration by 35–40%, and maintaining system utilization below 35% CPU and 40% memory, confirming its suitability for low-resource networks. The DDQN agent effectively stabilized Q-value updates and minimized overestimation errors common in traditional DQN models, resulting in faster learning convergence and enhanced decision reliability. Qualitative analysis further showed the system's adaptivity against multiple attack vectors such as port scanning, brute-force authentication, and command injection attempts, while the

10.5281/zenodo.18086738

lightweight containerized design ensured that these advanced adaptive responses were achieved without compromising performance or scalability—an essential feature for SMEs with constrained computational infrastructure. Overall, DALHF provides a balanced and practical approach that combines intelligent learning, operational efficiency, and dynamic threat deception. Future work will focus on expanding the framework to multi-agent reinforcement learning environments, integrating automated log correlation with SIEM platforms, exploring federated or distributed RL training for enhanced scalability, and enabling privacy-preserving collaboration. Additional improvements including blockchain-based secure collaboration, containerized orchestration models, and extension to IoT and cloud-native ecosystems will align the framework with emerging trends in decentralized cybersecurity. Thus, the proposed DALHF framework establishes a robust foundation for next-generation autonomous cyber defense systems—intelligent, scalable, and transparent—capable of learning and adapting to the dynamic and evolving threat landscape faced by SMEs.

REFERENCES:

[1] B. Vinavatani, M. R. Panna, P. H. Singha and G. J. W. Kathrine, ”AI for Detection of Missing Person,” International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 2022, pp. 66–73.

[2] S. Ayyappan and S. Matilda, ”Criminals and Missing Children Identification Using Face Recognition and Web Scraping,” 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), pp. 1–5.

[3] N. Ahirrao, S. Jade, S. Jagtap, N. Ghuse, M. Ghonge and A. Potgantwar, ”A Review on Identification of Missing Persons and Criminals using Image Processing,” International Journal of Creative Research Thoughts (IJCRT), vol. 10, no. 7, July 2022.

[4] A. Ponmalar et al., ”Finding Missing Person Using Artificial Intelligence,” 2022 International Conference on Computer, Power and Communications (ICCPC), Chennai, India, 2022, pp. 562–565.

[5] B. Pathak et al., ”Implementation of Website for Identification of Missing and Unrecognized People Using an Optimized Face Recognition Algorithm,” Pune, India, 2024.

[6] V. Shelke et al., ”Searchious: Locating Missing People Using an Optimized Face Recognition Algorithm,” Fifth International Conference on Computing Methodologies and Communication (ICCMC 2021), Vasai, India, 2021.

[7] D. Mahadik et al., ”Finding Missing Person Using AI,” International Journal of Advances in Engineering and Management (IJAEM), vol. 5, no. 4, pp. 1084–1088, April 2023.

[8] M. K. Singh et al., ”Implementation of Machine Learning and KNN Algorithm for Finding Missing Person,” 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 1879–1883.

[9] M. Turk and A. Pentland, ”Face Recognition Using Eigenfaces,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Maui, USA, 1991, pp. 586–591.

[10] R. Grover et al., ”Facial Recognition/Comparison for Finding Missing Person Using Python and AWS,” IJSREM, vol. 10, no. 5, May 2022.

[11] N. Gholape et al., ”Finding Missing Person Using ML, AI,” International Research Journal of Modernization in Engineering Technology and Science, vol. 3, no. 4, pp. 1517–1522, 2021.

[12] K. Suchana et al., ”Development of User-Friendly Web-Based Lost and Found System,” Journal of Software Engineering and Applications, vol. 14, pp. 575–590, 2021.

10.5281/zenodo.18086738