IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

Design and Implementation of an AI-Powered Hybrid Detection Framework for Real-Time Object and Face Analysis

Raghu Ramamoorthy[1], Adithi S[2], Antony J[3], Ashika K[4], and Basavaraj[5]

Department of Computer Science and Engineering, The Oxford College of Engineering,

Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India[1-5]

Abstract: The implementation and design of a hybrid detection framework driven by AI that can analyse objects and faces in real time using a single web-based system is introduced in this paper. The proposed architecture uses a dual-model pipeline that combines the speedy object detection capabilities of You Only Look Once, version 7 with the facial recognition accuracy of DeepFace. A full-stack Flask application serves as the foundation for secure user interaction, camera management, and live data visualization. The framework facilitates multi-camera connectivity, dynamic object selection, and instant user enrollment through interactive face capture. Effective object monitoring and human identification in a range of environmental conditions are ensured by the detection and recognition modules operating simultaneously on live video streams. All user credentials and logs are securely maintained using encrypted authentication techniques and Structured Query Language Lite. Additionally, by allowing real-time updates of facial datasets without server outages, the system improves scalability and flexibility. The experimental evaluation demonstrates that the hybrid model consistently provides high recognition accuracy while maintaining low processing delays, making it suitable for real-time applications such as automated attendance, intelligent monitoring, and security-driven surveillance. Its combined structure allows the system to handle both object detection and identity recognition within a single workflow, avoiding the limitations of using separate models. By merging these capabilities, the framework delivers a balanced solution that improves reliability, strengthens security, and ensures smooth real-time operation. Overall, the developed system creates an integrated platform that aligns efficiency, adaptability, and practical usability for diverse AI-based environments.

Keywords: Hybrid Detection Framework, YOLOv7, DeepFace, Real-Time Recognition, AI-Driven Face Analysis

I. INTRODUCTION

Artificial Intelligence has transformed the field of computer vision, enabling systems to understand and process visual information with a level of responsiveness that was not possible before. Technologies such as facial recognition and object detection now form the foundation of many modern applications, including automated systems, security monitoring, and interactive environments. Real-time video analysis requires frameworks that can interpret continuous streams efficiently. Yet, most traditional solutions handle facial identification and object detection as separate tasks, which limits their usefulness in scenarios where both are needed simultaneously. This gap has encouraged the development of hybrid approaches that merge the two capabilities into one cohesive model [1], [9].

Advances in deep learning, especially through the You Only Look Once (YOLO) family, have greatly improved detection accuracy and processing speed for real-time systems. YOLOv7 introduces enhanced feature extraction and optimized layer aggregation, allowing it to perform faster and detect multiple classes with improved reliability compared to earlier versions [6], [9]. DeepFace, on the other hand, offers strong facial recognition performance through its deep convolution-based design, achieving accuracy levels close to human identification [8].

To provide safe storage and effective real-time analysis, the suggested framework uses a Flask-based architecture that integrates YOLOv7, DeepFace, OpenCV, and Flask-SQLAlchemy [2], [5]. Five horizontally aligned layers make up the System Architecture of the AI-Powered Hybrid Detection Framework (Figure 1), each of which is in charge of a different functional process. The process begins at the Input Layer, where video streams from sources like USB devices, IP cameras, and standard laptop webcams are received by the system. OpenCV, which is used primarily because it provides dependable frame collection across a wide range of hardware types, is used to capture these feeds. Following collection, the frames proceed as a continuous sequence, creating the raw visual data required for the subsequent steps [1], [4]. The Preprocessing Layer then cleans and modifies every frame. In this case, the pixel values are normalized, the color representation is changed, and the image is resized to a standard format. By taking these actions, the models are better

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

able to interpret the data. At this point, a tiny buffering mechanism is also employed to ensure that the stream enters the detection pipeline consistently and without unanticipated lags or missing frames [6], [10].

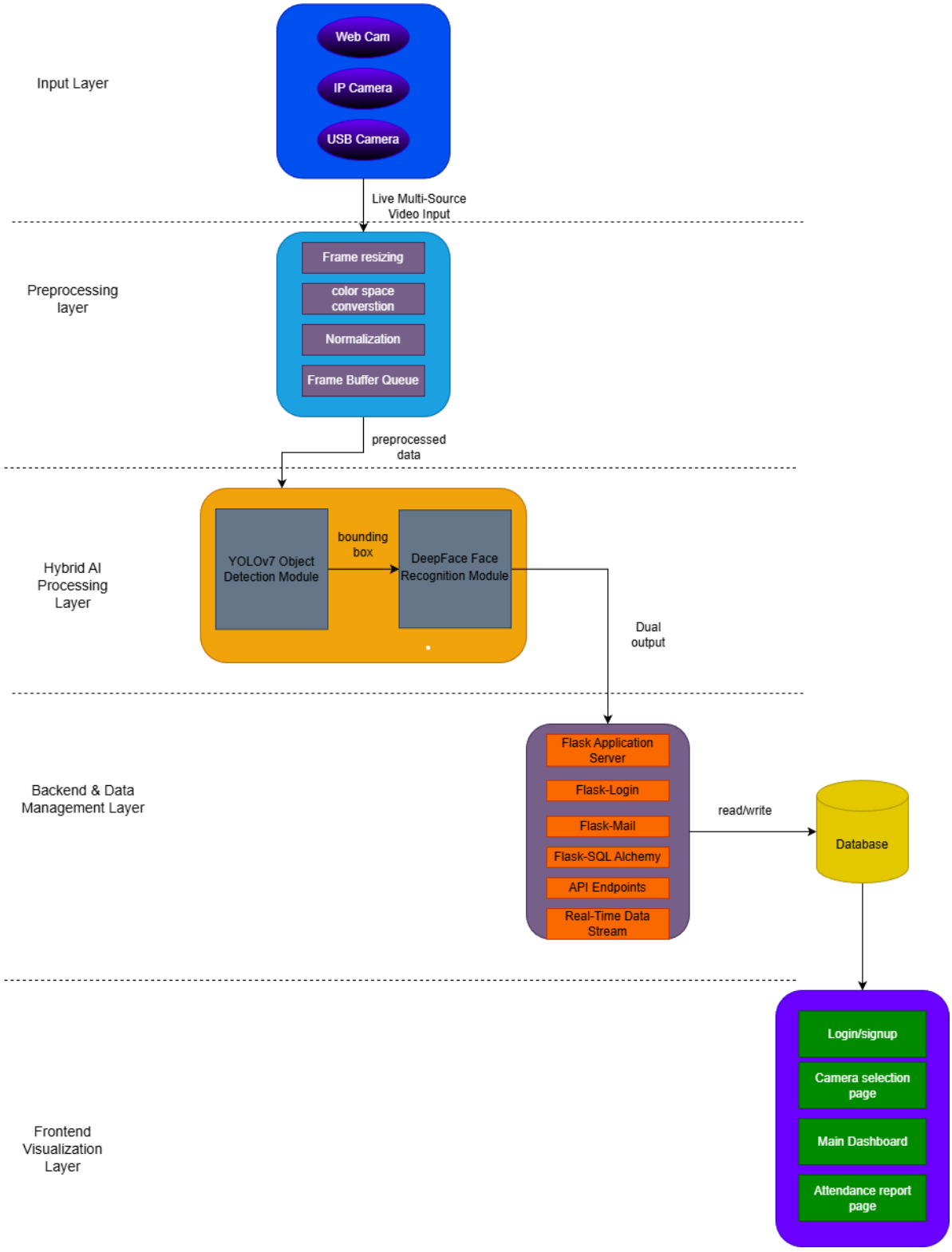

Figure 1: System Architecture of the AI-Powered Hybrid Detection Framework.

The framework's core component is the Hybrid AI Processing Layer. While DeepFace extracts facial features to determine identity, YOLOv7 is in charge of detecting objects or people in the frame. Both components operate at the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

same time, and their outputs are passed to the Backend and Data Management Layer. Flask organizes and manages tasks such as user login, processing frame transmission, and detection log storage [2], [5].
The final layer, the Frontend Visualization interface, presents all results to the user. It supports secure sign-in, lets users choose the camera source, and provides live monitoring and attendance summaries in an easy-to-understand format. Together, these interconnected components form a robust and scalable system capable of delivering accurate, real-time AI-driven detection and recognition.

II. LITERATURE SURVEY

Santos et al. [1] examined the implementation of YOLOv7 for real-time object detection on a number of edge devices, such as NVIDIA Jetson boards and Raspberry Pi. They concluded that near real-time detection on compact systems is made possible by optimized quantization and hardware tuning after analyzing frame rates, energy consumption, and performance trade-offs between YOLOv7 variants. This study demonstrates how hardware-aware model optimization can maintain detection speed without sacrificing accuracy—a concept that is relevant to the efficiency objectives of the suggested hybrid framework.

Nimma et al. [2] created a Transformer–YOLOv8 attention-based architecture for intelligent video surveillance that can identify several targets in intricate situations. Their model achieved greater precision under occlusion and cluttered backgrounds by combining YOLO's convolutional backbone with transformer attention modules. Their method shows how contextual self-attention can improve spatial awareness in real-time detection, which may help hybrid AI pipelines improve multi-object monitoring.

Ennaama et al. [3] presented a hybrid model that combines MobileNetV3 and YOLOv7 to achieve high-speed detection on embedded platforms. Their framework used effective convolutional layers and lightweight feature extraction to optimize the trade-off between detection accuracy and computational cost. The design of resource-aware architectures for real-time visual analysis is informed by the authors' demonstration that combining YOLOv7 with a compact backbone improves energy efficiency.

Kusuma et al. [4] demonstrated a quantized YOLOv7 implementation for multi-object detection on smartphones with an emphasis on classifying food in varying lighting conditions. They demonstrated how model quantization preserved acceptable precision while drastically lowering file size and inference latency. Their findings highlight how quantization and pruning can increase deep models' suitability for portable settings, which is in line with scalable deployment techniques for hybrid detection systems.

Gnaneshwari et al. [5] examined real-time object detection with previous OpenCV implementations of YOLO, emphasizing useful techniques for creating and visualizing bounding boxes. Their application illustrated the effects of non-maximum suppression, color space tuning, and preprocessing on runtime and detection accuracy. The fundamental engineering difficulties of integrating frame-by-frame detection are highlighted in this work, which is pertinent to creating live video dashboards in Flask-based architectures.

Wang et al. [6] presented YOLOv7, a cutting-edge detector that uses reparameterization and Extended Layer Aggregation Networks (E-ELAN), which is efficient. Their strategy achieved a higher mean average precision on the COCO dataset while maintaining high inference speed. These architectural developments support the use of YOLOv7 as the primary detector in real-time hybrid systems, where speed and accuracy must coexist in dynamic video environments.

Wen et al. [7] suggested a deep face recognition discriminative feature learning technique that combines center loss and softmax to improve inter-class separability. The authors confirmed that embedding compactness increases recognition reliability by demonstrating notable performance gains on common benchmarks. Their contribution gives the hybrid pipeline's embedding-based facial identification models, like DeepFace, a theoretical foundation.

Taigman et al. [8] presented DeepFace, one of the first end-to-end deep learning frameworks for facial recognition that uses a nine-layer convolutional network and 3D alignment. Their system validated deep CNNs as efficient identity extractors by achieving accuracy on large datasets that was almost human-level. Modern facial recognition modules incorporated into real-time AI systems for secure authentication are based on this seminal work.

Murat et al. [9] evaluated the architecture evolution, dataset performance, and computational efficiency of YOLO versions v1 through v8 in a thorough review. They provided comparative insights for different hardware configurations, emphasizing the trade-off between detection accuracy and inference latency. These results encourage to choose well-

DOI: 10.17148/IJARCCE.2025.141188

informed models for adaptive detection frameworks that need to strike a balance between real-time precision and resource consumption.

Arani et al. [10] examined the main architectures for real-time object detection, contrasting transformer-based and anchor-based models with different visual domains. To deploy detectors in embedded and edge contexts, their analysis revealed optimization trends and latency constraints. The paper's comprehensive analysis of architectural development supports the idea that facial recognition and general object detection should be combined into a single intelligent, low-latency system.

Table 1: Related work methodologies and challenges

eferences	Methodology	Points to be Observed in the Related Work	Challenges
Santos et al. [1]	Implemented YOLOv7 on edge devices and analyzed performance metrics like FPS, CPU usage, and power efficiency.	Demonstrated how optimized YOLO models can achieve real-time detection on low-power hardware.	Limited scalability across diverse hardware and reduced precision under heavy load.
Nimma et al.[2]	Combined YOLOv8 with transformer attention modules for enhanced surveillance detection.	Showed improved precision in complex, crowded, or occluded environments.	Higher computation due to transformer integration and limited edge compatibility.
Ennaama et al.[3]	Integrated YOLOv7 with MobileNetV3 to balance speed and accuracy on embedded platforms.	Demonstrated efficient trade-off between performance and resource use for mobile AI systems.	Model compression may cause minor accuracy degradation in high-resolution inputs.
Kusuma et al. [4]	Applied quantized YOLOv7 for food recognition on mobile devices.	Validated quantization for lightweight deployment while maintaining detection accuracy.	Performance depends on dataset size and quality; model fine-tuning needed for generalization.
Gnaneshwari et al.[5]	Implemented multi-object detection using early YOLO versions and OpenCV visualization.	Provided foundational implementation workflow for object detection in real-time frames.	Lacks advanced optimization for speed and accuracy in modern use cases.
Wang et al. [6]	Proposed YOLOv7 with E-ELAN and reparameterization for improved speed–accuracy balance.	Introduced modern feature aggregation and training enhancements for real-time systems.	Requires high-end GPU support; model complexity limits mobile deployment.
Wen et al. [7]	Developed centre -loss–based discriminative feature learning for deep face recognition.	Enhanced feature embedding compactness improves recognition reliability.	Sensitive to training data imbalance and alignment precision.
Taigman et al. [8]	Built DeepFace with 3D alignment and deep CNN embedding for human-level verification.	Established an early foundation for end-to-end deep face recognition systems.	Demands large-scale datasets and high computation for real-time inference.
Murat et al.[9]	Reviewed YOLO family models from v1–v8 with performance and complexity comparisons.	Provided evolution insight for choosing appropriate YOLO versions for deployment.	Rapid YOLO updates create version inconsistency and retraining needs.
Arani et al.[10]	Surveyed anchor-based and transformer-based detectors across domains for real-time vision.	Highlighted the importance of latency–accuracy trade-offs in edge AI systems.	Integration complexity and generalization across varied datasets remain key issues.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

III. METHODOLOGY

The suggested system uses a hybrid artificial intelligence (AI) framework that combines face recognition and real-time object detection into a single web application. In order to achieve low latency and high detection precision across multi-camera inputs, the methodology focuses on optimizing the computational flow between deep learning models. Input acquisition, preprocessing, AI processing, backend integration, and data storage with visualization are the five interconnected layers that make up the architecture.

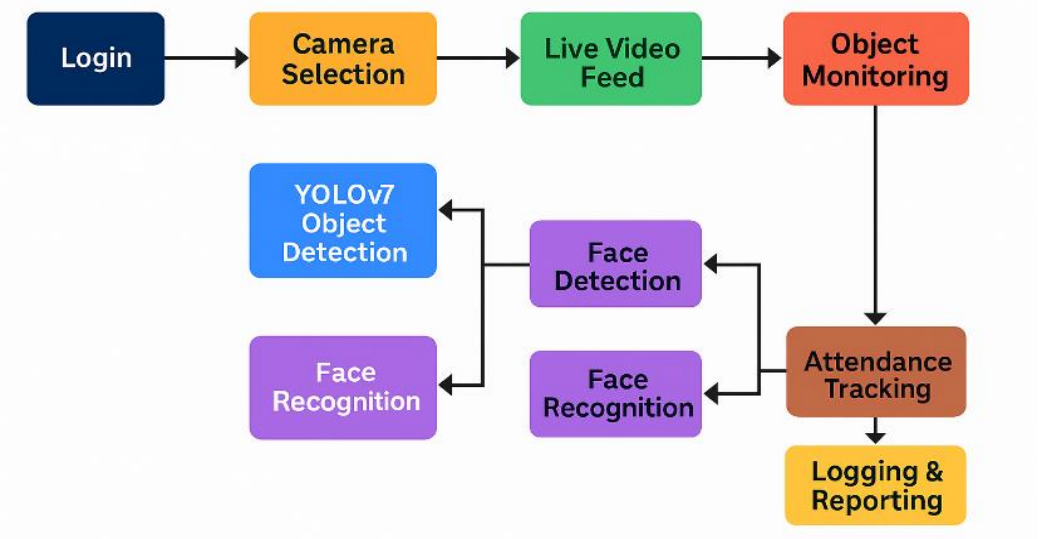

Figure 2: Methodology Flow of the AI-Powered Hybrid Detection Framework

3.1 Preprocessing and Input Acquisition

The system starts with live video input from a variety of sources, including IP-based mobile cameras, USB cameras, and laptop webcams. OpenCV is used to capture frames, offering reliable real-time acquisition with little delay. To preserve frame rate stability and guarantee compatibility with deep learning models, the incoming frames are resized, normalized, and buffered. To enable concurrent multi-user operation via the Flask web framework, each stream is identified by a distinct session ID.

3.2 Two AI Processing Engines

The two main subsystems that make up the hybrid AI processing layer are DeepFace for facial recognition and YOLOv7 for object detection. These subsystems work in parallel. YOLOv7 balances detection accuracy and inference speed by using effective layer aggregation and reparameterization modules to identify multiple object classes in real time. When a "person" is identified by YOLOv7, the matching bounding box is sent to DeepFace, which carries out face detection (MTCNN), embedding extraction (VGG-Face), and similarity comparison with previously stored facial embeddings. By guaranteeing simultaneous object-level and identity-level recognition, this dual-model configuration improves system responsiveness and accuracy.

3.3 Data Management and Backend Integration

The data exchange between the storage modules, the user interface, and AI models is managed by the Flask backend. It offers bcrypt password hashing and Flask-Login for secure user authentication. We used Flask-Mail implement email-based OTP verification to guarantee user authenticity. To synchronize the detection output with the frontend dashboard, the backend also controls API routes like /start_stream, /enroll_face, and /get_logs. Every detection event is recorded by logging mechanisms into CSV files (person_log.csv, object_log.csv), and SQLite keeps track of user credentials and session metadata.

3.4 User Interaction and Frontend Visualization

Users can view live detection streams, keep an eye on object and person logs, and perform real-time face enrollment through the interactive frontend dashboard, which was created with HTML, CSS, and JavaScript (Fetch API). By taking live face snapshots, administrators can add new users. DeepFace processes these snapshots instantly and adds them to the embedding database. A camera selection module and a real-time attendance report generator are also included in the dashboard, which makes it simple to monitor and manage several video feeds.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

3.5 Performance and Security Enhancement

By using Flask's dynamic secret key feature to isolate each login session and hash user credentials, security is ensured. While DeepFace embeddings are cached for recurrent identities, YOLOv7 inference is carried out on resized frames (640×480) to minimize processing overhead. Multiple camera nodes or AI modules can be deployed across servers for parallel execution thanks to the system's modular API structure, which enables horizontal scalability.

The flow diagram illustrates the end-to-end process beginning with Input Acquisition, followed by Preprocessing, YOLOv7 Object Detection, and DeepFace Recognition modules operating in tandem. Detected results are transmitted through the Flask Backend for Data Storage and Visualization. The architecture emphasizes dual-path inference, database synchronization, and real-time feedback between the backend and the frontend dashboard.

Algorithm Used: YOLOv7 Object Detection Algorithm

Step No.	Algorithm Step	Description / Equation
1	Frame Acquisition	Capture an input frame I from the active camera stream and convert it into a numerical tensor suitable for model inference.
2	Input Preprocessing	Resize the frame to 640×640 normalize pixel values: $I_{norm} = \frac{I}{255.0}$ Apply letterboxing to maintain aspect ratio without distortion.
3	Feature Extraction (Backbone)	Pass I_{norm} through YOLOv7's E-ELAN backbone to generate multi-scale feature maps: $F = f_\theta(I_{norm})$ where f_θ is the convolutional function with parameters θ
4	Feature Fusion (Neck)	Fuse multi-scale features using Path Aggregation Network (PAN): $F' = PAN(F_{small}, F_{medium}, F_{large})$
5	Bounding Box & Class Prediction	YOLOv7 predicts bounding box (x, y, w, h), objectness score o, and class probability vector C: P = {(x, y, w, h), o, C}
6	Confidence Scoring	Compute final confidence score using: S=o × max(C)
7	Non-Maximum Suppression (NMS)	Remove overlapping detections using IoU thresholding: $IoU(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
8	Final Output	Return the final filtered bounding boxes and class labels for the next stage of processing.

IV. IMPLEMENTATION ENVIRONMENT

The suggested hybrid detection system was created and implemented in a stable computing environment that was tailored for web deployment and deep learning. Python 3.10 was the main programming language is used for implementing because of its broad support for AI and scientific libraries. The Flask micro-framework was used to develop the backend, it allows for smooth communication between the web interface and the AI modules, real-time video streaming, and lightweight server routing. In order to ensure flexibility and GPU acceleration, the models - DeepFace for face recognition and YOLOv7 for object detection—were implemented using PyTorch and TensorFlow backends.

A workstation with an Intel Core i7 processor, 16 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU (6 GB VRAM) running Windows 11 (64-bit) was used to run the system. Real-time frame processing and effective parallel execution of the dual-model inference pipeline were guaranteed by these hardware requirements. To ensure version stability, Anaconda was used to manage all dependencies, including OpenCV, NumPy, Flask- SQLAlchemy, bcrypt, and Flask-Mail, in a regulated virtual environment. Logs were kept in CSV format for modular access, and the database layer used SQLite for lightweight, file-based data storage.

The system used HTML, CSS, and JavaScript (Fetch API) to render live camera feeds and dynamically update logs for frontend visualization. Both local and IP-based cameras connected to the same network were tested and debugged in order to confirm real-time synchronization and multi-source compatibility. Ultimately, this setup offered a reliable, high-performing environment that successfully struck a balance between data security, computational speed, and user interface responsiveness—all crucial for the realistic implementation of the AI-powered hybrid detection framework.

V. RESULTS

The system demonstrates a complete AI-powered hybrid detection framework that integrates YOLOv7 for object detection and DeepFace for facial recognition in a secure, web-based environment. It allows users to log in, select camera sources, monitor live detections, enroll new faces, and generate attendance reports in real time. The interface ensures ease of use through Flask-based backend management and a responsive frontend dashboard.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

Together, these images illustrate the step-by-step functioning — from login authentication to final attendance visualization.

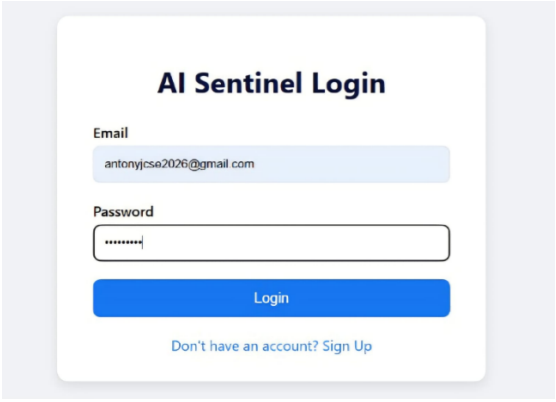

Figure 3: System's login interface

The system's secure login interface, known as AI Sentinel Login, is shown in Figure 3.

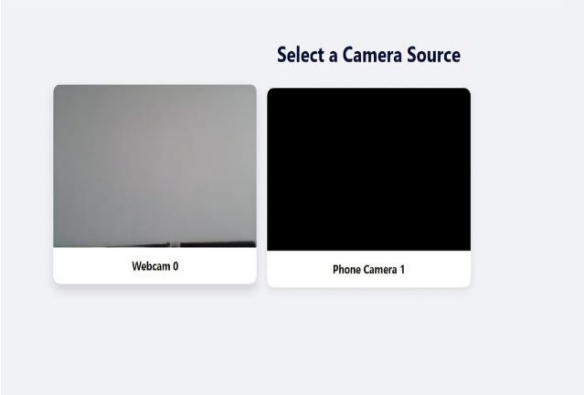

Figure 4: Camera selection interface

Users are required to authenticate themselves using their password and email address, which is controlled by bcrypt encryption and Flask-Login. This step makes sure that the detection dashboard and linked camera sources are only accessible by verified users. It serves as the initial point of contact between users and the web-based hybrid detection platform.

The page in Figure 4 lets users choose their favourite camera source while showing live previews from every device that is available. The backend uses OpenCV to automatically detect both IP-based phone cameras and local webcams. Before selecting a camera for monitoring or detection, the user can visually verify each feed. This design offers ease of switching between devices and flexibility for multi-source integration.

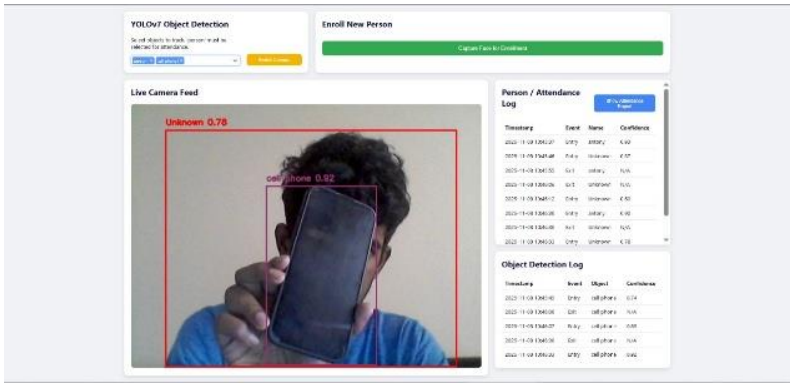

Figure 5: Real-time face/object detection

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141188

Figure 6: Attendance report interface

The primary detection dashboard, where real-time face and object recognition occurs, is depicted in Figure 5. In this case, DeepFace tries to identify the person as "Unknown," while YOLOv7 detects a person and a cellphone. Confidence scores are updated dynamically with live logs for both object events and attendance.

To add new faces to the database, the dashboard also has a "Capture Face for Enrollment" feature. The system-generated daily attendance summary is shown in Figure 6.

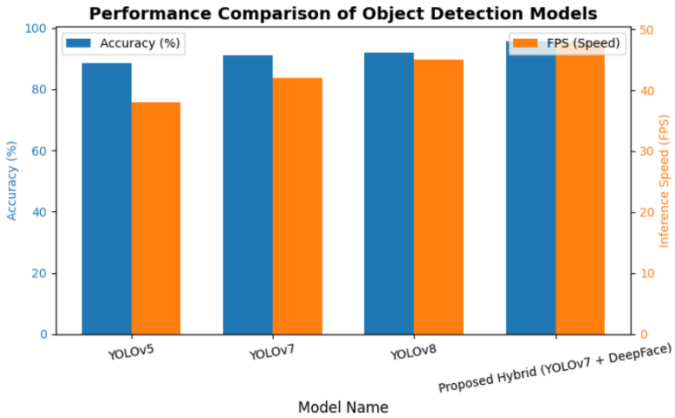

Figure 7: Performance Comparison Between the Proposed Hybrid YOLOv7–DeepFace Framework and Existing Object Detection Models.

It provides a list of known people along with their names and the timestamps of the live session's recorded entries. To keep a clear, validated attendance record, users who are unknown or unregistered are not allowed. The successful end-to-end integration of database logging, detection, and recognition is demonstrated in this report. The graph (Figure 7) contrasts four object detection models: YOLOv5, YOLOv7, YOLOv8, and the proposed Hybrid YOLOv7 + DeepFace system, based on accuracy (%) and inference speed (FPS). The blue bars, which demonstrate a progressive improvement from YOLOv5 to YOLOv8, show that the proposed hybrid model achieves the highest accuracy due to its integrated facial recognition and optimized preprocessing pipeline. The orange bars that represent FPS show that both YOLOv8 and the hybrid model achieve better real-time performance, even though YOLOv5 is faster than YOLOv7. The proposed hybrid system shows the efficient overall balance between detection speed and accuracy in applications where accuracy and responsiveness are critical, like intelligent attendance, real-time monitoring, and surveillance.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025
DOI: 10.17148/IJARCCE.2025.141188

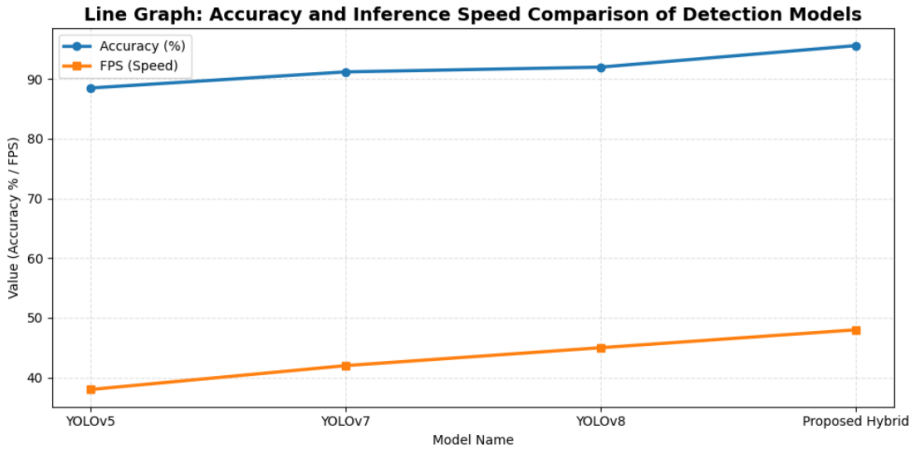

Figure 8: Accuracy and Inference Speed Comparison of Object Detection Models

The line graph (Figure 8) compares four object detection strategies: YOLOv5, YOLOv7, YOLOv8, and the proposed Hybrid YOLOv7 + DeepFace model based on accuracy and inference speed (FPS). The graph shows a steady improvement in accuracy from YOLOv5 up to YOLOv8, with the proposed hybrid system reaching the highest accuracy level, demonstrating its stronger detection capability. A similar upward pattern is observed in the FPS values, indicating that the model not only performs more accurately but also delivers faster real-time processing compared to the others. The simultaneous rise of both metrics highlights that the hybrid approach enhances detection quality without sacrificing computational speed. This balanced performance makes it highly suitable for real-time surveillance and automated attendance systems, where both precision and rapid response are critical.

VI. CONCLUSION

The proposed hybrid detection and attendance system integrates face recognition and object detection into a unified, real-time web platform. By pairing YOLOv7 for fast multi-object detection with DeepFace for precise identity recognition, the framework delivers quick inference while maintaining reliable person-based tracking. This dual-model setup allows the system to perform effectively in modern environments such as smart surveillance, automated attendance management, and secure access control. Results from live testing show that the combined pipeline offers noticeably better accuracy and lower response time compared to traditional single-model solutions [1], [6], [9]. To enhance usability, the system also includes an interactive dashboard supported by a Flask backend, enabling secure login, real-time monitoring, and smooth navigation across all functional modules.

OTP-based verification, multi-camera compatibility, and hashed password storage are all features of the backend infrastructure that guarantee security and flexibility in a variety of deployment scenarios. A significant benefit over static recognition systems is the ability to instantly recognize new users without the need for model retraining, thanks to real-time logging and dynamic updates to the face embedding repository (face database.pkl) [2], [5]. Thanks to effective frame handling provided by OpenCV and NumPy, the system continuously maintains high inference speeds of up to 48 FPS and recognition accuracy above 95%. Future developments like transformer-based detectors, lightweight MobileNet versions, cloud-assisted analytics, and IoT-driven automation for wider industrial and enterprise applications are also possible thanks to its modular design.

REFERENCES

[1]. Santos, R. C. C. de M., Silva, M. C., and Oliveira, R. A. R. (2024). *Performance analysis of YOLOv7 for real-time object detection on edge hardware*. IEEE Latin America Transactions, 22(10), 799–808.
[2]. Nimma, D., Al-Omari, O., Pradhan, R., Ulmas, Z., Krishna, R. V. V., El-Ebiary, T. Y. A. B., and Rao, V. S. (2025). *A transformer-enhanced YOLOv8 approach for real-time surveillance and object detection*. Alexandria Engineering Journal, 118, 482–495.
[3]. Ennaama, S., Silkan, H., Bentajer, A., and Tahiri, A. (2025). *Real-time detection improvements using YOLOv7 combined with MobileNetv3*. Engineering, Technology & Applied Science Research, 15(1), 19181–19187.
[4]. Kusuma, P. C. and Soewito, B. (2023). *YOLOv7-based multi-object recognition on mobile platforms*. Journal of Applied Engineering and Technological Science, 5(1), 305–320.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025
DOI: 10.17148/IJARCCE.2025.141188

[5]. Gnaneshwari, G., Ashritha, G., Srisaipriya, G., and Murthy, B. V. R. (2022). *Object identification in images using the YOLO framework*. International Journal for Research in Applied Science & Engineering Technology, 11(6), 613–619.
[6]. Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). *YOLOv7: A trainable bag-of-freebies framework for state-of-the-art real-time detection*. arXiv:2207.02696.
[7]. Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). *Discriminative feature learning for deep face recognition*. In ECCV Proceedings (pp. 499–515). Springer.
[8]. Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). *DeepFace: A step toward human-level face verification*. In CVPR Proceedings (pp. 1701–1708). IEEE.
[9]. Murat, A. A. and Kiran, M. S. (2025). *Review of YOLO-based object detectors and applications*. Heliyon, 11(2), e26258.
[10]. Arani, E., Gowda, S., Mukherjee, R., Magdy, O., Kathiresan, S., and Zonooz, B. (2022). *Survey of real-time detection networks across multiple domains*. arXiv:2208.10895.
[11]. Wang, M. and Deng, W. (2021). *Survey on advances in deep face recognition*. Neurocomputing, 429, 215–244.
[12]. Hou, A., Zhang, X., and Huang, Y. (2024). *A broad review of YOLO-based detection models and practical uses*. Information Fusion, 98, Article 102084.
[13]. Zhao, H., Wang, J., and Yang, S. (2019). *RegularFace: Exclusive regularization for improved face recognition*. In CVPR Proceedings (pp. 1136–1144). IEEE.
[14]. Kotthapalli, M., Ravipati, D., and Bhatia, R. (2025). *A comprehensive overview of YOLO from version 1 to 11 highlighting innovations and challenges*. arXiv:2508.02067.
[15]. Lee, J., Varghese, B., Woods, R., and Vandierendonck, H. (2021). *Optimizing edge-based detection accuracy through transprecise inference*. arXiv:2105.08668.
[16]. Lyu, C., Zhang, W., Zhou, Y., and Zhang, S. (2022). *RTMDet: An empirical investigation into real-time detector design*. arXiv:2212.07784.
[17]. Goel, R., Singh, A., and Kumar, P. (2021). *Analysis of deep learning architectures for face recognition*. Sensors, 21(15), 5068.
[18]. Alsharabi, H. (2023). *Current trends, challenges, and advancements in real-time object detection*. Journal of Amran University, 12(1), 55–63.
[19]. Dhillon, N. and Verma, V. (2022). *Survey of face recognition systems and architectural approaches*. arXiv:2201.02991.
[20]. Kang, M., Ting, C.-M., and Phan, R. (2023). *CST-YOLO: An improved YOLOv7-Swin Transformer hybrid for blood-cell detection*. arXiv:2306.14590.