

Comparative Analysis of Deep Learning Models for an Automated Video Surveillance System

Amrita Biswas¹, Abhinav Sharma², Pritam Kalita³

^{1, 2, 3} Department of Computer Science and Engineering, Sikkim Manipal Institute of Technology, Majhitar,
Sikkim Manipal University, Gangtok, Sikkim, India

Email: ¹amrita.a@smit.smu.edu.in, ²abhinav_202100063@smit.smu.edu.in, ³pritam_202000074@smit.smu.edu.in

Abstract

Intelligent Video Surveillance (IVS) systems play an important role in ensuring public safety and security. Traditional surveillance methods depend mostly on manual monitoring. As a result, the process can be cumbersome and prone to error. With the advent of deep learning and image processing techniques, surveillance systems have become more accurate, autonomous, and context-aware. This survey paper provides a comprehensive review based on experiments of deep-learning based methods used for human detection, face detection, feature extraction and recognition. We have proposed an automated Video Surveillance system that will detect people and identify them from CCTV footage. Various models have been compared and benchmarked each for object detection (YOLO, SSD, Faster R-CNN, and Mask R-CNN), face detection (HOG, MTCNN, Haar Cascade, and RetinaFace), and face recognition (ResNet34, FaceNet, ArcFace, and InceptionResnetV1-VGGFace2). This paper also highlights current challenges and outlines future directions for building robust surveillance systems.

Keywords: Intelligent Surveillance System, Object detection, Face detection, Face recognition, YOLO, HOG, Resnet-34.

1. Introduction

In recent years, the demand for enhanced security in both public and private domains have risen. Conventional surveillance approaches work but autonomous human detection is an issue. Deep learning methodologies are often used in surveillance to address these issues and has inspired a lot of research in this field. [1, 2].

In this work a three-stage process for real-time human identification from CCTV footage has been proposed. It has three stages namely- human detection, face detection, and finally face recognition [3, 4, 5]. The CCTV footage video is the input. It is preprocessed and human detection carried out using YOLO [6, 7]. The model provides output bounding boxes for candidate individuals. The next stage carries out localization of face using the HOG [8] detector. ResNet-34 [9] model has been used to extract facial features that are matched against a stored database using Euclidean distance to determine the identity of the individual. The humans detected are logged with timestamps. The bounding box and time stamps are viewed on a web-based dashboard designed for viewing in real time. It also helps with the offline analysis.

Background

Nowadays, security cameras are widely used in public places, shopping malls, schools, and offices. As the number of cameras increases, there is also an increasing need for smarter systems. So that the systems can automatically monitor and understand activities independently of the human operators. In conventional surveillance systems tracking and identifying humans is a cumbersome process. is time-consuming and requires a lot of effort. Security staff often spend hours reviewing video footage to find one specific person [1, 2].

In large areas such as college campuses, malls, or public events, manual identification becomes extremely difficult. During emergencies, delays in identifying or tracking individuals can cause serious issues.

Deep learning-based surveillance systems are used to automatically detect and identify people in real time. These systems help to reduce the workload for security personnel. They also improve the speed and accuracy of surveillance operations.

Our work focuses on the key parts of such systems: detecting people, detecting faces, and recognizing

individuals from CCTV footage. In this paper we review advanced deep learning models used for such tasks and compare their speed and accuracy of performance.

2. Proposed Work

Human Detection-Model Evaluation Strategy

Human detection is the first step in the proposed system.

The objective of the human detection algorithm is to detect human objects in real time. It is implemented by training and evaluating multiple deep learning-based object detection models. The goal was to identify the most effective model in terms of both detection accuracy and speed.

Several object detection models were implemented and assessed for human detection. The models were primarily evaluated based on two key metrics: inference speed and confidence score. Inference speed was measured in terms of Frames Per Second (FPS), indicating the model's real-time processing capability. Confidence score was evaluated using the bounding box confidence score. This score provides a robust measure of how accurately the model localizes objects in a frame. By analyzing both Frames Per Second (FPS) and bounding box confidence scores, a balanced evaluation of speed and detection quality was achieved. This helped identify the most suitable model for deployment in the intelligent surveillance system.

We tested several variants of YOLO [6, 7] (You Only Look Once). YOLOv5n, was tested as it is a lightweight and fast model suitable when there are constraints in resources; YOLOv8n, YOLOv8m, and YOLOv8x, which are more advanced iterations of YOLO [6, 7] were also tested. They offered a speed-accuracy trade-off; and the more recent YOLOv11n and YOLOv11s were tested. They provided better accuracy in real-time, making them ideal for modern detection tasks.

Besides YOLO [6, 7], SSD [10] (Single Shot Multibox Detector) using the SSD300 architecture with a VGG16 [11] backbone was also implemented. SSD [10] offers good efficiency and solid detection performance but often results with smaller objects are not satisfactory. For more accurate detection, especially in complex scenes, Faster R-CNN [12] with a ResNet50 [13] backbone was used. As a two-stage detector, Faster R-CNN [12,21] provides high detection accuracy and reliable bounding box predictions. Lastly, Mask R-CNN [14], an extension of Faster R-CNN [12] that includes an

additional segmentation mask branch was explored. Mask R-CNN [14] with a ResNet50 [13] backbone, was also implemented. Though it was slower it offers better detection accuracy. [15, 16, 17, 18].

Customized training of YOLO [6, 7] models was also conducted to fine-tune performance for our specific human detection task. The models trained were yolo11s.pt and yolo11n.pt, with training epochs ranging from 10 to 80. Image input sizes of 416 and 640 were used. The performance of each configuration was evaluated using the mAP@50 metric, which was recorded across training epochs. Important parameters like learning rate, image resolution, and epoch count were carefully tuned to obtain optimal performance considering detection accuracy and training time. All custom training was conducted in the Google Colab environment, cloud-based GPUs were used for quick computation.

For human detection, three distinct datasets were used independently during experimentation. Human dataset was sourced from Kaggle [19], which provides a large collection of labeled human images, consisting of 17,754 images (train set: 13,754 images and validation set: 4,000 images). Another was the COCO dataset [20], from which only human-labeled images were extracted, consisting of 1,07,478 images (train set: 64,115 images, validation set: 2,693 images and test set: 40,670 images). Additionally, a custom CCTV dataset was used, which was collected from inhouse CCTV footage, comprising of manually human annotated images, specific to surveillance environments, consisting of 1,01,310 images (train set: 70,917 images, validation set: 15,196 images and test set: 15,197 images). All datasets were annotated in YOLO [6, 7] format to maintain consistency across training procedures. Each dataset was treated separately to evaluate model performance under different data distributions. The dataset sizes varied, with a typical 90%–10% split between training and validation sets. Before training, images were preprocessed and resized to fixed dimensions of 416×416 or 640×640 depending on the input requirements of the respective models.

Face Detection-Model Evaluation Strategy

After a human is identified in the video stream, face detection is applied to identify and isolate the facial region of the person. The face detection model scans the detected human region to locate facial features.

Once the face is detected, face recognition is performed to identify or verify the person. The face detection models were evaluated based on two primary criteria: inference speed and detection reliability. Inference speed is determined by measuring how many frames the model could process per second under consistent hardware conditions. We evaluated the model's consistency in detecting faces under varying illumination conditions, poses, and image resolutions qualitatively. Also both CPU and GPU performances were considered to understand how well each model scaled with hardware. This helped select models that met the speed requirements needed for real-time systems besides detection accuracy.

Several face detection models were implemented. Datasets used included CCTV footage from inhouse camera system and the WIDER FACE [22] validation dataset, consisting of 3,226 images. Traditional methods like HOG [8] and Haar Cascade [23] were tested on CPU and displayed reasonable performance. Haar Cascade [23] outperformed HOG [8] based approach in terms of processing speed. Deep learning-based models such as MTCNN [24] showed improved accuracy and faster performance when run on GPU compared to CPU. Among the models tested, RetinaFace [25] provided the highest accuracy but had significantly slower performance on CPU. So it is less ideal for real-time use unless GPU support is available. Models like Haar Cascade [23] and GPU-accelerated MTCNN [24] offered the best balance between speed and detection reliability for real-time face detection in surveillance systems.

These findings highlighted the trade-offs between accuracy and speed across both traditional and deep learning-based face detection models. Classical methods offered faster inference on CPU but at the cost of lower detection accuracy, while deep learning models provided improved accuracy with greater computational demands. The results reinforced the importance of selecting models based on hardware availability, as real-time surveillance systems require a careful balance between detection performance and processing efficiency.

Face Recognition-Model Evaluation Strategy

Face recognition is the process of identifying or verifying a person's identity based on facial features. The system first extracts facial features such as the distance between the eyes, shape of the nose, jawline,

and other distinguishing landmarks. These features are then converted into a numerical vector (known as a face embedding), which is compared against a database of known faces. Matching is typically done using a similarity metric like cosine similarity or Euclidean distance. If the distance between the input face and a stored face is below a certain threshold, the system recognizes the person as a known individual. All face recognition models were assessed using a consistent set of evaluation metrics to determine their suitability for real-time surveillance deployment. Accuracy on the Labeled Faces in the Wild (LFW) [27] test set, consisting of 13,233 images was used to gauge the model's ability to correctly identify and verify faces under unconstrained conditions. The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) [28] curve was calculated to measure the overall discriminative power of each model. Inference time, recorded in milliseconds per image, was considered important for real-time performance, since the aim was to integrate the mode into live surveillance feeds. Also, the embedding size produced by each model was analyzed, as it impacts both memory usage and matching speed during face comparison tasks. These metrics were used to select the most efficient and accurate model.

Several architectures were implemented and evaluated. FaceNet [29], using a lightweight convolutional backbone, was implemented via a widely used deep learning library. It learns compact face embeddings using a triplet loss function and offers a good balance between recognition accuracy and processing speed. ArcFace [30], based on a deeper residual network, produced highly distinctive face representations using an angular margin loss and consistently achieved the best recognition accuracy in our evaluations. Inception ResNetV1 - VGGFace2 [31], trained on a large-scale face dataset, provided strong performance in both accuracy and generalization, performing comparably to ArcFace [30] in most scenarios. A ResNet-based model from the dlib library [32] was also included to benchmark traditional methods. While it was less accurate than the deep learning models, it provided a simple and efficient end-to-end pipeline for face detection and recognition. These models were compared to identify the optimal balance between computational load and recognition accuracy use in real-time surveillance environments.

3. Methodology

System Architecture

The steps followed are shown in Fig 1.

- Live or recorded video footage is received from CCTV cameras for analysis.
- Frames are extracted from the video and resized to a consistent resolution to prepare them for detection models.
- Each frame is passed through a human detection model (e.g., YOLO [7, 8]) to detect the presence of human figures.
- Bounding boxes are extracted around each detected human to isolate individual regions of interest.
- Each human bounding box is further processed using a face detection model to locate the face region within the person's bounding box.
- Detected face regions are cropped from the image and prepared for recognition.
- Each cropped face image is passed through a face recognition model that generates a face embedding (a unique numerical representation of the face).
- The generated face embedding is compared with stored embedding in a face database using cosine similarity or Euclidean distance to determine a match.
- If a match is found, the face is labeled as a verified face and associated with the corresponding identity.
- If no match is found, the face is labeled as an unverified face.
- The timestamp of the unverified face detection is stored in a database for recordkeeping.
- The unverified face image is also stored in a separate database of unknown individuals.
- The face data collected is sent to a web application dashboard.
- The web dashboard displays logs, timestamps, and analytics.
- The process stops for the current frame and resumes for the next frame in the CCTV footage.

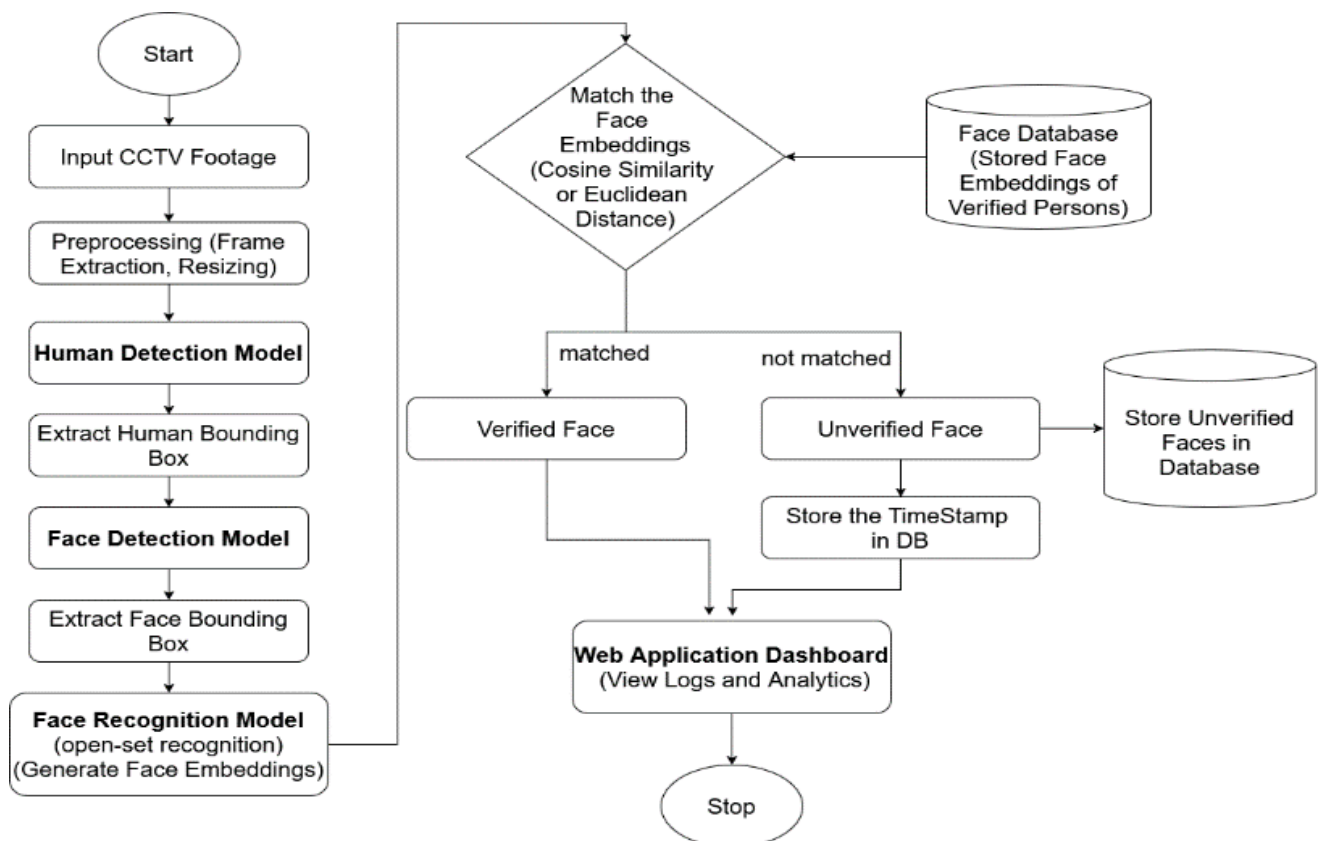


Fig 1. System Diagram

4. Training And Testing

Evaluation of Object Detection Models for Human Detection

To evaluate the various object detection algorithms suitable for human detection several pre-trained models were first compared. The models tested included variants of YOLO [6, 7] (You Only Look Once), SSD [10] (Single Shot Detector), and two-stage detectors like Faster R-CNN [12] and Mask R-CNN [14]. Each model was evaluated based on inference speed (frames per second) and bounding box confidence score. The bounding box confidence score reflects the model's certainty that a detected region contains a human figure within each frame. This score projects how reliably a model identifies and localizes objects. Analyzing both the confidence scores and the Frames per Second (FPS), helped attain a trade-off between detection accuracy and processing speed.

In YOLO [6, 7] object detection models, the predicted bounding box as two components. The first component, is known as the objectness score $P(\text{object})$ and represents the model's confidence that any object, regardless of class, is present within the predicted bounding box. This score ranges from 0 to 1 and is predicted using a sigmoid activation function. During training, it is learned as a binary classification task distinguishing between boxes that contain objects and those that do not. The second component is the class probability $P(\text{class}|\text{object})$, which is the model's predicted probability that the detected object belongs to a particular class, given that an object is present in the box. This probability also ranges from 0 to 1 and is computed using a softmax activation for mutually exclusive classes or sigmoid for multi-label scenarios. The final class-specific confidence score for each bounding box is computed by multiplying the objectness score by the conditional class probability.

$$\text{Confidence} = P(\text{object}) \times P(\text{class} | \text{object})$$

This combined score reflects both the likelihood that the bounding box contains an object and the likelihood that the object belongs to a specific class, providing a comprehensive measure of confidence for detection.

Frames Per Second (FPS) has been used to measure inference speed. It is the measure of number of frames the model can process in one second. FPS is calculated as the reciprocal of the time taken to process a single frame, expressed in seconds. If the time required to

infer one frame is known, the FPS can be determined by dividing 1 by this time.

$$\text{FPS} = \frac{1}{\text{Time taken to process a single frame}}$$

Higher FPS values indicate faster inference speeds.

For comparison of detection performance, the models were tested in two scenarios: one with few people (Fig 2) and one with more people.



Fig 2. Detection in case of Less People. (Top-left) YOLO; (Top-right) Faster R-CNN; (Bottom-left) SSD; (Bottom-right) Mask-RCNN.

Fig 3 is a snapshot of the models' performance when multiple people are present simultaneously in the frame. The evaluation highlights each model's capability to detect overlapping and densely packed human figures, while maintaining speed and accuracy.

From these tests, it is observed that in some scenes the Single Shot Detector (SSD [10]) failed to detect objects (person), when the objects appeared very small or were captured in poor lighting conditions. In comparison, both YOLO [6, 7] and Faster R-CNN [12] and Masked R-CNN [14] exhibited better performance. The results have been shown in Table I.

The YOLO11n and YOLO11s object detection models were also explored for person identification. They were tested on a Human dataset sourced from Kaggle [19]. The dataset comprises a mix of training and validation images in varying sizes and quantities. Key performance metrics such as mAP@50 and training time are shown in Table II which helped to assess the impact of training epochs, image sizes, and total images on detection accuracy and efficiency.



Fig 3. Detection in case of More People. (Top-left) YOLO; (Top-right) Faster R-CNN; (Bottom-left) SSD; (Bottom-right) Mask-RCNN.

The YOLO11n and YOLO11s object detection models were also explored for person identification. They were tested on a Human dataset sourced from Kaggle [19]. The dataset comprises a mix of training and validation images in varying sizes and quantities. Key performance metrics such as mAP@50 and training time are shown in Table II which helped to assess the impact of training

epochs, image sizes, and total images on detection accuracy and efficiency.

The YOLO11n model is also evaluated on the widely used COCO human dataset [20] to benchmark its performance under standard object detection conditions. The results have been shown in Table III.

YOLO11n's performance has also been evaluated on a in house college CCTV dataset, consisting of 1,01,310 images. The results obtained have been shown in Table IV.

The YOLOv11n pre-trained model has also been compared with a custom-trained model in terms of inference speed (FPS) and confidence scores (Bounding box confidence score) as shown in Fig 4 and results shown in Table V. While the pre-trained model demonstrates reliable speed, the custom-trained version shows improvement in accuracy, showing the effectiveness of dataset-specific training in enhancing model performance for targeted surveillance applications.

Table I. Evaluation of object detection models

Model	Type	Inference Speed (FPS)	Average Bounding Box Confidence Score	Remarks
YOLOv5n	One-stage	64.16	71%	Fastest among all tested models
YOLOv8n	One-stage	60.05	72%	Very fast with slightly better accuracy than YOLOv5n
YOLOv8m	One-stage	31.76	76%	Balanced trade-off between speed and accuracy
YOLOv8x	One-stage	12.75	79%	High accuracy but slower inference
Faster R-CNN	Two-stage	4.14	86%	High accuracy; suitable for applications where speed is secondary
Mask R-CNN	Two-stage	3.69	86%	Includes segmentation; high accuracy but slow
SSD300 with VGG16	One-stage	11.09	70%	Good speed but poor performance on small-sized objects

Table II. YOLO training results on Kaggle Human Dataset

Model	Epochs	Image Size	No. of Images	mAP@50	Training Time
yolo11s	10	640	13,754 (12,378 train/ 1,376 val)	0.57	1h
yolo11s	20	416	17,643 (15,878 train/ 1,765 val)	0.59	1.33h
yolo11s	40	416	17,643 (15,878 train/ 1,765 val)	0.58	2.67h
yolo11s	60	416	2,754 (2,478 train / 276 val)	0.53	0.83h
yolo11s	80	640	2,754 (2,478 train / 276 val)	0.55	1.5h

Model	Epochs	Image Size	No. of Images	mAP@50	Training Time
yolo11n	15	416	17,643 (15,878 train / 1,765 val)	0.56	1h
yolo11n	40	416	17,643 (15,878 train / 1,765 val)	0.57	3h

Table III. YOLO training results on COCO Human Dataset

Model	Epochs	Image Size	Images (Train/Val)	mAP@50	Training Time
yolo11n	5	640	66,808 (64,115 / 2,693)	0.729	1.830h

Table IV. YOLO training results on inhouse developed dataset.

Model	Epochs	Image Size	Images (Train/Val)	mAP@50	Training Time
yolo11n	1	640	86,113 (70,917/ 15,196)	0.958	0.595h
yolo11n	5	640	86,113 (70,917/ 15,196)	0.972	3.038h

Table V. Comparison of YOLO pre-trained model with custom-trained model

Model	Average FPS	Average Confidence Score
YOLOv11n (pre-trained)	56.91	61%
YOLOv11n (custom-trained)	58.39	78%

Pre-trained models show a lower confidence score when compared with custom-trained YOLOv11n models as shown in Table V.



Figure 4. Real-time detection performance comparison. (Left) YOLO pre-trained model; (Right) YOLO custom-trained model

This indicates that the custom training has significantly improved the model's ability to recognize humans in the specific context and lighting conditions of the target environment.

B. Evaluation of models for Face Detection

To evaluate the effectiveness of various face detection algorithms, a comparative analysis was conducted using several pre-trained models on actual CCTV footage. Models such as HOG [8], Haar Cascade [23], MTCNN [24] (on both CPU and GPU), and RetinaFace [25] were tested using different hardware configurations. Key performance metrics including inference speed (FPS), and time taken per frame (s) were recorded to assess each model's efficiency. The

comparison highlights the strengths and limitations of each method in terms of speed and computational requirements, providing guidance on their suitability for real-time or high-accuracy face detection applications. Results have been shown in Table VI.

Another comparative analysis was made on the WIDER FACE [22] validation dataset, consisting of 3,226 images, to evaluate the inference time and accuracy. Results have been shown in Table VII.

In this evaluation, the HOG [8] - based detector, running on CPU, exhibited the slowest average inference time of 0.5298 seconds per image and the lowest accuracy at 14.54%. The Haar Cascade [23] classifier, also on CPU, demonstrated faster performance with an average inference time of 0.1886 seconds per image and improved accuracy of 21.93%. The MTCNN [24] model was tested on both CPU and GPU, showing significant gains when executed on GPU—achieving an average time of 0.1477 seconds per image and an accuracy of 44.85%, compared to 0.3215 seconds and 44.73% on CPU. RetinaFace [25], although the most computationally intensive with an average inference time of 2.6423 seconds per image on CPU, achieved the highest detection accuracy of 57.26%. These results highlight a clear trade-off between speed and accuracy, with GPU-accelerated MTCNN [24] and RetinaFace offering substantially better detection performance on the WIDER FACE [22] validation set. Accuracy in this evaluation was calculated using the formula:

$$\text{Accuracy} = \left(\frac{\text{Total detections}}{\text{Total ground truth}} \right) * 100$$

Figure 5 shows the results on in house CCTV footage. The focus is on how well each model maintains detection accuracy, speed, and stability while processing live video streams or camera feeds.



Figure 5. Real-time detection performance comparison. (Top-left) RetinaFace; (Top-right) Haar Cascade; (Bottom-left) HOG; (Bottom-right) MTCNN.

In this test case it has been observed that traditional face detection methods such as HOG [8] and Haar Cascade [23] often encounter challenges in detecting small (less than 30×30 pixels) or non-frontal faces, particularly in low-resolution or angled views. MTCNN

[24] offers improved performance with small faces but can be less consistent when handling non-frontal orientations. RetinaFace [25] has shown the ability to detect both small and non-frontal faces more effectively in several evaluations. These observations suggest that different methods vary in their strengths and limitations depending on the specific detection scenario, emphasizing the importance of model selection based on the target application environment.

C. Evaluation of Face Recognition Models

To evaluate the effectiveness of various face recognition algorithms, a comparative analysis was conducted using several pre-trained models. This subsection compares the performance of several pre-trained face recognition models evaluated on the Labeled Faces in the Wild (LFW) [27] dataset, consisting of 13,233 images. Key metrics such as accuracy, Area Under the Curve (AUC) [28], inference time, and embedding size are presented to provide a comprehensive overview of each model's effectiveness. The comparison highlights trade-offs between recognition accuracy and computational efficiency. Results have been shown in Table VIII.

Among the evaluated pre-trained face recognition models, ArcFace [30], ResNet34 [9] (dlib), and InceptionResNetV1 - VGGFace2 [31] achieve the highest accuracy of 0.9471, marginally outperforming FaceNet [29]'s 0.9402.

Table VI. Evaluation of face detection models (on CCTV footage)

Model	Hardware	Inference Speed (FPS)	Time taken per Frame (s)
HOG	CPU	4.83	0.1992
Haar Cascade	CPU	15.16	0.0562
MTCNN	CPU	10.02	0.0893
MTCNN	GPU	16.78	0.0491
RetinaFace	CPU	0.36	2.7845

Table VII. Evaluation of face detection models (on WIDER FACE validation dataset)

Model	Hardware	Avg Time/ Image (s)	Total Time (s)	Accuracy (%)
HOG	CPU	0.5298	1709.13	14.54
Haar Cascade	CPU	0.1886	608.39	21.93
MTCNN	GPU	0.1477	476.41	44.85
MTCNN	CPU	0.3215	1037.13	44.73
RetinaFace	CPU	2.6423	8524.09	57.26

Table VIII. Comparison of Face Recognition Models on Labeled Faces in the Wild (LFW)

Model	Accuracy	AUC	Inference Time (ms)	Embedding Size
FaceNet (InceptionResnetV1)	0.9402	0.9932	104.89	128D
ArcFace (ResNet100)	0.9471	0.9897	869.27	512D
ResNet34 (dlib library)	0.9471	0.9897	855.58	512D
VGGFace2 (InceptionResnetV1)	0.9471	0.9897	881.07	512D

In terms of overall discrimination ability, FaceNet [29] leads with the highest AUC of 0.9932, reflecting superior separation between genuine and impostor pairs. FaceNet [29] also demonstrates significantly faster inference times (~105 ms) compared to the substantially slower ArcFace [1], ResNet34 [9], and InceptionResNetV1 - VGGFace2 [31] models (~850–880 ms), highlighting its suitability for real-time applications. Furthermore, FaceNet [29] employs a more compact 128-dimensional embedding, optimizing memory and computation, while the others use larger 512-dimensional embeddings that may enhance feature representation at the expense of increased resource consumption. These results underscore a trade-off where FaceNet [29] offers an efficient balance between speed, compactness, and accuracy, whereas the other models provide marginally better accuracy with higher computational cost.

Evaluation of the face recognition models involved key metrics such as accuracy, AUC (Area Under the ROC Curve) [28], and inference time. Accuracy quantifies the proportion of image pairs correctly classified as either belonging to the same person (genuine) or different individuals (impostor).

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of pairs}}$$

In this face recognition systems, Euclidean distance is used to compare feature embeddings extracted from facial images. Given two embedding vectors, the Euclidean distance is calculated as:

$$d(\vec{e}_1, \vec{e}_2) = \sqrt{\sum_{i=1}^n (e_{1i} - e_{2i})^2}$$

where n is the dimensionality of the embedding space. A smaller distance indicates higher similarity between faces, and a predefined threshold θ is used to classify pairs: if $<\theta$, the pair is predicted as the same person; otherwise, they are predicted as different. For threshold-independent evaluation, Area Under the

ROC Curve (AUC) [28] is used. AUC is computed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various thresholds:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

The AUC is then approximated using the trapezoidal rule over the ROC points:

$$\text{AUC} = \sum_{i=1}^{n-1} (\text{FPR}_{i+1} - \text{FPR}_i) \cdot \frac{\text{TPR}_{i+1} - \text{TPR}_i}{2}$$

AUC serves as a threshold-independent performance metric that reflects the model's ability to discriminate between matching and non-matching pairs. An AUC [28] of 1.0 indicates perfect separability, while 0.5 suggests random performance. Inference time, represents the average time the model needs to process a pair of images, offering insight into the computational efficiency of the recognition system. Collectively, these metrics provide a comprehensive assessment of model accuracy, robustness, and practical applicability.

5. Results

This system demonstrates a comprehensive intelligent video surveillance system that integrates real-time human detection and face recognition on video streams using advanced deep learning techniques. User control features are implemented through the Flask backend to toggle detection modules, set ROI, rename or delete faces, and interact with matched face records. Snapshot of the user interface has been shown in Fig 6.

The architecture was designed for both performance and usability, resulting in several key achievements outlined below:

A. Human Detection and Face Recognition

The system integrates YOLO [6, 7] to accurately and efficiently detect humans in video streams. Detected faces are then encoded and matched against a known

face database using the face_recognition library (HOG [8] and Resnet-34 [9]). To speed up processing, especially with large datasets, multiprocessing techniques were employed, enabling parallel face recognition tasks without slowing down the system.

B. Video Playback Controls:

Interactive video playback functionalities were implemented to improve the user's control and experience. These include standard options like pause and play, along with advanced features such as frame skipping, timestamp seeking, and fast forward/rewind. These tools help in detailed scene analysis.

C. Region of Interest (ROI) Selection:

A dynamic Region of Interest (ROI) selection feature was introduced, allowing users to define specific areas of the video frame for processing. This reduces computational load by ignoring irrelevant parts of the frame, leading to faster processing and more efficient resource utilization.

D. Auto Face Recognition Workflow:

The system automates face registration by detecting and saving new, unidentified faces as unknown_X.jpg, which are dynamically added to the face database. For each recognized face, the interface displays three image views: a cropped face image, a full-body crop of the detected person, and the corresponding reference image from the database. This enhances clarity and helps with easy visual verification.

E. Efficient UI and Database Interaction:

A user-friendly web interface was developed to support seamless interaction with the face database. Features include a powerful search functionality for known faces, and direct in-browser operations for renaming or deleting images. Cached face encodings improve performance by reducing initial load time, and the system also supports auto-refreshing of the face database to reflect updates in real time.

F. Matched Face Management:

The system maintains a detailed record of both known and unknown matched faces. It displays the total number of matches and allows users to filter results based on known or unknown status. Additionally, it includes the ability to clear matched face logs, helping keep the data organized and relevant.

G. Performance Optimization:

To ensure scalability and responsiveness, performance was enhanced through multiprocessing and caching strategies. These improvements enable the system to manage large volumes of facial data without lag. For devices lacking GPU acceleration, optional frame skipping was added to maintain smooth operation and reduce processing demand.

These features make the application suitable for real-time surveillance, post-event face search, and video review workflows. The system's flexibility and user-driven control options ensure adaptability across a range of practical deployment environments.

6. Conclusion

This survey critically evaluated state-of-the-art models for human detection, face detection, and face recognition within the context of an intelligent video surveillance system. Our analysis reveals that YOLO [6, 7] - based architectures offer the best balance between accuracy and real-time performance for human detection, making them suitable for edge devices and medium-resource deployments. For scenarios demanding maximum precision without real-time constraints, two-stage models like Faster R-CNN [12] and Mask R-CNN [14] remain highly effective despite their latency. In the face detection domain, MTCNN [24] emerged as the most reliable and efficient solution when GPU resources are available, while Haar Cascade [23] and HOG [8] remains a viable option in CPU-bound environments with limited computational capacity. For face recognition, FaceNet [29] stands out for its superior accuracy, inference speed, and compact 128D embeddings - ideal for scalable, low-latency systems. Although models like ArcFace [30] and InceptionResNetV1 - VGGFace2 [31] offer marginally higher accuracy, their high computational demands make them more appropriate for GPU-enabled or offline processing tasks. Overall, the findings underscore the importance of selecting models not only based on performance metrics but also considering hardware constraints and deployment environments.

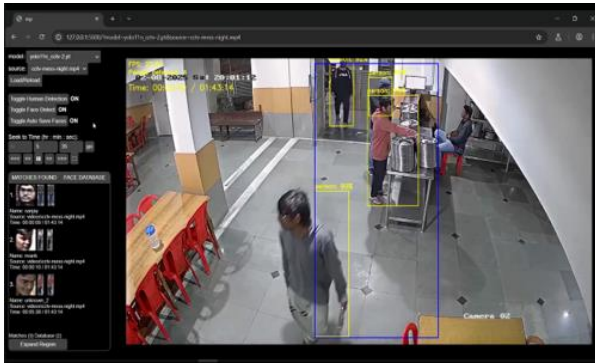


Fig 6. Snapshot of the user interface

References

- [1] P. Manogna, K. Manasa, and M. S. Haeshitha, "Intelligent Video Surveillance System Using AI," *Int. J. Sci. Res. Eng. Trends*, vol. 11, pp. 2443–2446, 2025.
- [2] S. W. Ibrahim, "A Comprehensive Review on Intelligent Surveillance Systems," *Commun. Sci. Technol.*, vol. 1, pp. 7–14, 2016.
- [3] B. Amirgaliyev, M. Mussabek, T. Rakhimzhanova, and A. Zhumadillayeva, "A Review of Machine Learning and Deep Learning Methods for Person Detection, Tracking and Identification, and Face Recognition with Applications," *Sensors*, vol. 25, p. 1410, 2025.
- [4] N. Azatbekuly, A. Mukhanbet, and S. D. Bekele, "Development of an Intelligent Video Surveillance System Based on YOLO Algorithm," in *Proc. IEEE 4th Int. Conf. Smart Inf. Syst. Technol. (SIST)*, 2024.
- [5] J. Xu, "A Deep Learning Approach to Building an Intelligent Video Surveillance System," *Multimed. Tools Appl.*, vol. 80, pp. 5495–5515, 2021.
- [6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 779–788.
- [7] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines*, vol. 11, p. 677, 2023.
- [8] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Diego, CA, USA, June 2005, pp. 886–893.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778.
- [10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 21–37.
- [11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556*, 2015.
- [12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 1137–1149, 2017.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778.
- [14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2961–2969.
- [15] R. Raj, "Real-Time Human Detection for Intelligent Video Surveillance: An Empirical Research and In-Depth Review of Its Applications," 2023.
- [16] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, "Comparative Analysis of Deep Learning Image Detection Algorithms," *J. Big Data*, vol. 8, p. 66, 2021.
- [17] Q. Shan, Q. Yang, J. Ding, Y. Hou, et al., "Comparison of Different Object Detection and Classification Models," *Appl. Comput. Eng.*, vol. 134, pp. 89–101, 2025.
- [18] John and D. Meva, "A Comparative Study of Various Object Detection Algorithms and Performance Analysis," *Int. J. Comput. Sci. Eng.*, vol. 8, pp. 158–163, 2020.
- [19] F. Elmenshawii, "Human Dataset," *Kaggle*, 2021.
- [20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Eur. Conf. Comput. Vis. (ECCV)*, Zurich, Switzerland, Sept. 2014.
- [21] S. Sumit, S. Bisht, S. Joshi, and U. Rana, "Comprehensive Review of R-CNN and Its Variant Architectures," *Int. Res. J. Adv. Eng. Hub*, vol. 2, pp. 959–966, 2024.
- [22] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 2016.

- [23] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *Int. J. Comput. Vis.*, vol. 57, pp. 137–154, 2004.
- [24] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multi-Task Cascaded Convolutional Networks," *IEEE Signal Process. Lett.*, vol. 23, pp. 1499–1503, 2016.
- [25] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single-Stage Dense Face Localisation in the Wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Seattle, WA, USA, June 2020, pp. 1–12.
- [26] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, pp. 273–297, 1995.
- [27] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," *University of Massachusetts, Amherst, Tech. Rep. 07-49*, Oct. 2007.
- [28] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognit. Lett.*, vol. 27, pp. 861–874, 2006.
- [29] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, June 2015, pp. 815–823.
- [30] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, June 2019, pp. 4690–4699.
- [31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI)*, San Francisco, CA, USA, Feb. 2017, pp. 4278–4284.
- [32] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.