Real-Time Face Detection and Recognition V-Bot for Security: Evaluating Haar Cascade and YOLOv8 Frameworks

Mohammad Sameer, Uday Sai Kiran, Aditya Ippili, and Aila Ramcharan
Dr. Vikram Dhiman

Department of Computer Science and Engineering,
GITAM School of Technology, GITAM (Deemed to be University), Visakhapatnam, India

Abstract. Facial recognition technology is crucial in modern security systems, particularly for authentication and surveillance. This project compares Haar Cascade and YOLOv8 frameworks for real-time face de tection, recognition, along with headcount, gender detection, and pre dicting human mood across static images, videos, and live streams. In this approach, we had created Visual-bot [V-Bot] to detect, recognize, and the results highlight YOLOv8's superior accuracy, real-time efficient, and robust against challenging environments.

1 Introduction

The fastest-evolving technologies are impacting the education, security and law enforce ment sectors, retail, and healthcare. Facial recognition is now capable of identifying and verifying an individual through his facial features. Advanced organizations now expe rience greater efficiency and automated process in the safety solution by recognizing the users more effectively. Therefore, such models have been developed as a result of increasing demands for efficient real-time facial recognition systems that support both machine learning and deep learning approaches. Traditionally, approaches such as the use of Haar Cascade classifiers were core landmarks regarding facial detection and recognition. Work done by Viola and Jones[1] in 2001 established the utilization of Haar Cascade has been preferred primarily because it is fast and simple when detecting faces in static images and video streams. Haar Cascade uses hand-crafted features and relies on a sliding window mechanism, which makes this widely applicable in early-stage security and surveillance systems. However, as effective as it is, Haar Cascade cannot deal with quite complex environments, like changing lighting, occlusions, or multiple faces in a dynamic setting. On the other hand, evolution in deep learning models such as the ”You Only Look Once” (YOLO)[2] framework comes as a giant step toward the future. Even at a substantial increase in the speed of detection with accuracy in facial recognition, the YOLO[3] real-time detection makes great improvements compared to the traditional models. On the contrary, in traditional models, YOLOv8 will look for multiple faces in the very same forward pass of the network. Such extreme environ ments, as shown in the following scenarios–low illumination, complex backgrounds, and overlapping faces–are handled well by YOLO, where others might helplessly struggle. This project attempts to evaluate and compare two models, namely Haar Cascade and YOLOv8, based on their performance along a number of key criteria, including detec tion speed, accuracy, scalability, and robustness in terms of coping with environmental variables. Such a comparative analysis would involve both single and multiple face detection and recognition tasks spanning various data sources, such as static images, video files, and real-time video streams Github [4]

2 Literature Review

Table 1: Summary of Literature Review

Paper Title	Methodology	Advantages	Limitations	Gap
YOLO: Unified, Real-Tim e Object Detection[2]	Single-stage objectdetector that inte grates classification and localization. Uses direct bounding box regression via convolutional layers. Trained end-to-end on full images.	High-speed detectionsuitable for real-time applications. More efficient than multi stage detectors.	Limited ac curacy for small objects. Struggles with precise localization compared to region-bas ed approache s.	Requires im provement s in localization precision and detection of small objects.
YOLOv2: Better, Faster, Stronger[5]	Introduces multi scale training, anchor boxes, and Darknet 19 as the backbone. Employs batch nor malization for better convergence.	Enhances accuracyand speed over YOLOv1. Efficiently balances real-time performance and detection precision.	Difficulty in detecting tiny objects in cluttered scenes. Lim ited accuracy improveme nts in some datasets.	Needs furtheropti mization for small object lo calization and handling occlusions in dense environme nts.

YOLOv3: An Incre mental Im provement [6]	Utilizes FeaturePyramid Networks (FPN) for multi scale detection. Im plements Darknet-53 for feature extrac tion. Adopts logistic regression for bound ing box predictions.	Better small ob ject detection. Higher accuracy than YOLOv2. Supports multi-label classification.	Still less ac curate than heavier models. Faces challenges with very small objects in complex backgroun ds.	Requires refinement s in detecting small ob jects within crowded en vironments. Needs ex ploration of alterna tive network architec tures for im proved speed accuracy trade-off.

Continued on next page
(Continued from previous page)

Paper Title	Methodology	Advantages	Limitations	Gap
Scaled YOLOv4: Scaling Cross Stage Partial Network[3]	Scalable architecture with versions (tiny, standard, large) to accommodate different hardware capabilities. Better feature fusion and multi-scale detec tion. Continued use of advanced data augmentation and training techniques from YOLOv4.	Flexible scalability allows for deploy ment across a variety of devices from edge (tiny) to server grade (large). Better performance across scales, making it adaptable for diverse use cases.	Complexity in choosing and configuring the right model version for specific use cases. Still not the best for small ob ject detection in cluttered scenes.	Further re search on optimizing model ver sions for different de ployment environ ments (e.g., low-power devices). Improved accuracy for objects in densely cluttered or complex scenes.

YOLOv4: Optimal Speed and Accuracy of Object Detection[7]	Combined detection and classification tasks through a joint training ap proach using Ima geNet and COCO datasets.Introduced WordTree, a hierarchical classification system that allows detection of over 9000 object categories.	Detects over 9000 categories in real time, giving it a huge range of objects it can identify.Maintains efficiency by using a hierarchical classifi cation structure.	Dependent on the scope of datasets used for training; accuracy may drop for un seen object variations. Localization of small or thin objects can still be challenging.	Looking into ways to combine different datasets to boost how well the model can classify and detect faces.Improving detection of faces at different scales, especially smaller ones or faces in complex scenes.

Continued on next page

(Continued from previous page)

Paper Title	Methodology	Advantages	Limitations	Gap

Versatile Recogni tion Using Haar-Like Feature and Cascaded Classifier[1]	This recognitionalgorith m uses Haar like features and a cascaded classi fier to make object detection fast and efficient. It combines feature extraction and classification into one streamlined process, boosting ac curacy in recognising objects.	Allows for lowcomputational cost while maintaining high recognition rates, making it efficient for real-time applications.The cascaded structure enhances detection speed by filtering out non-object regions quickly.	The per formance may degrade in complex real-world scenarios with noisy data or significant occlusions, and testing is limited to specific datasets.	Further evaluation of diverse datasets is necessary to establish adaptability to different conditions and enhance generalisa bil ity.
Face De tection in Real Time Live Video Us ing YOLO Algorithm Based on VGG16 Convo lutional Neural Network[8]	Combines the YOLOalgorithm with a VGG16 pre-trained convolutional neural network. Focusing on three stages: creating a labelled ground truth database, feature extrac tion using VGG16 and face detection using YOLOv2. streamlined process, boosting accuracy in recognising objects.	High face detectionspeed and accuracy in real-time live video.Handling var ious face positions, illuminations, and skin colours.	High com putational complex ity requires the use of GPU graphics cards.Limiti ngthe method’s applicability in devices with lower processing power.	Further re search is needed to integrate this method as a preprocess ing step in face recognition systems.Te st its effective ness across different envi ronments and datasets.

3 Problem Identification and Objectives

Facial detection and recognition starts from consumer applications such as security, authentication all the way to human-computer interaction. There are several problems in the different areas :

3.1 Limitations of Haar Cascade

- ∗ Light and Position Responsive Haar Cascade [1]is very sensitive to lighting, and tilted faces reduce the precisions in dynamic environments.
- ∗ High False Positives: Noisy background tends to have high false positives reduced rates, and reliability.
- ∗ Ability to detect and identify many faces at a glance is difficult, limiting its use in crowded settings.

3.2 Challenges with real-time detection and occlusions

• Speed vs. Accuracy[2]: Traditional models are usually fast but not as robust as

- • Deep learning precise models[6] are extremely computationally expensive, af fecting real-time performance.
- • Scaling[7]: Objects of different scales and distances are presented, which raises a challenge for many models.

3.3 Occlusions And Environmental Variations

Occlusions: The classical and modern models break on occluded faces. which means faces that have masks or glasses on them.

Environmental Variations: Depending on the weather and lighting, detection may not be effective due to a complex background.

3.4 Emotion Detection in Security Applications

• Function of Identity Verification: A traditional face recognition system's main function is for identity verification[11]. Identity Verification. The main purpose of a traditional face recognition system is identity verification [11].

Human Behaviour Usage: The emotions experienced by humans can provide important clues concerning human behavior to assist you in identifying potential suspicious behavior. • Emotion recognition is a method of assessing such human behavior that security personnel may use to recognize behaviors that indicate stress, anxiety or aggression, which may assist them in recognizing threats to security.

3.5 Real-Time Application

- • Computational complexity[8]: Deep learning models, require high computa tional resources.
- • The computational power makes them hard to deploy on low-power devices.

3.6 Security And High-Use Cases

- • Demand for High-Performance Systems: Security systems[13] in airports, banks, and public spaces demand real-time, high-accuracy detection, which tradi tional models struggle to provide, particularly in large-scale, multi-person en vironments.

3.7 Objectives

- Create a face detection model that is able to recognize one or many faces in various conditions and is capable of discriminating between them. Further, identify the dominant human emotion in real-time.
- Increase the model's one or many-face identification ability using a reliable identification model. • Upgrade People Counting to improve analytics in real-time.
- Build a simple and user-friendly interface where users can easily choose to work with images, videos, or live streams.
- Create an IoT prototype that implements the facial detection system with IoT system, making it easy to use in different environments and helping in automation, security and access control.

Table 2. Metrics based comparison of Haar Cascade and YOLOv8

Metric	Haar Cascade	YOLOv8
Accuracy (Controlled)	~75%	~96%
Accuracy (Real-time)	~60% (ideal conditions)	~92% (varied conditions)
Frame Rate (GPU)	Not GPU-optimised	35-40 FPS
Frame Rate (CPU)	~20-30 FPS	~12-15 FPS
False Positives	Higher in cluttered scenes	Minimal
Occlusion Handling	Poor	Good
	Pose Variation Handling Struggles with non-frontal poses Robust to various angles	

4 Existing System

One of the earliest techniques in object detection: Haar Cascade, started in the early 2000s as a landmark algorithm in face detection tasks[1]. The cascade constructed with Haar-like features captures changes in intensity across adjacent regions in an image, focusing on contrast patterns between light and dark regions(0s and 1s). The key innovation of the Haar Cascade algorithm was the use of a cascade of classifiers. This allows the system to quickly eliminate non-face regions of the image and direct computational power toward regions likely to contain a face[14]. Each stage of the cascade is increasingly complex, applying more detailed feature extraction to those parts of the image that pass the The fundamental advantage of Haar Cascade is the ease of application along with fast speed. The drawback lies in the condition that there cannot be very wide illumination differences or that face positions have to have considerable variability[15].

4.1 Key Revolutions during the History of Haar Cascade

- 2001 – Viola-Jones Framework[16]: Basic version of the Haar Cascade could detect front faces, which meant applying a cascading structure[16] that in troduced successive classifiers while utilizing the image integral data to get it running hundreds of times faster than general object detection. This allowed real-time detection of faces on fairly average hardware, which was a revolution at the time.
- Popularisation with OpenCV (around the mid-2000s): OpenCV is one of the free software libraries for computer vision developed by Intel[16], which has started using the Haar Cascade algorithm very early; this made the method perhaps the most widely used among researchers and programmers.
- Optimizations and Variations(2000s): Different versions of the Haar Cascade followed with time with which improvements were tried in an attempt to achieve real conditions. Performance adaptations included non-frontal pose handling, scaling for varying face sizes, and more noise-tolerant and partially occlusion-tolerant classifiers[17].
- Haar Cascade remains the popular tool from the 2000s, yet it is much more of a compromise than when it was invented. Being pretty good compared to others, at least at its time, and do have some issues: complex lighting[18], varying orientations, partially occluded faces. Yet, Haar Cascade still remains valid in those areas with severely limited computational resources, including resource-constrained environments.
- Current Usage: Haar Cascade is a pretty simple, fast and efficient solution for applications such as face detection in controlled environments. It's still used in OpenCV and used in lightweight applications, but its supremacy has been surpassed by deep learning-based[19] detectors such as CNNs[20] and more complex real-time detection systems such as YOLO[19].

Haar feature-based cascade classifiers are very popular and effective object detec tion method, as conceived in the seminal 2001 work of Paul Viola[1] and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features." The approach is machine learning-based in which a cascade function is trained on many positive images (positive images, containing the object of interest, for exam ple, faces and negative images, not containing the object). After the classifier is trained, it can be used for objects detection on new images.
For face detection, the Haar Cascade algorithm is devised for the recognition of facial features by positive samples (face images) and negative samples (non-face images). The system relies upon the extraction of simple rectangular features that are similar to Haar-like features, we take the difference between the sum of pixel intensities in a white rectangle and a black rectangle. It focuses on picking up intensity differences across specific areas. This approach works really well on facial areas, like distinguishing between the eyes and cheek regions, because it captures those unique spatial differences.

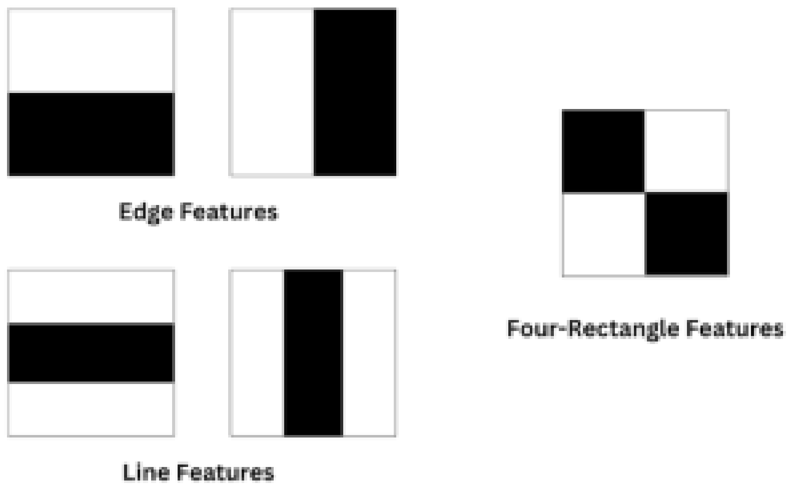

Fig. 1. Haar-like features of Haar Cascade

Once the features are pulled from the image, the classifier has to calculate tons of possible features across the whole thing, which makes the process pretty complex. For example, even on a small 24x24 pixel window, there are over 160,000 potential features. To handle this, the algorithm uses something called an integral image, which speeds up the calculations by organizing the image data more efficiently. Even though it generates a massive number of features, most of them don't actually help with detecting facesthey're either irrelevant or repetitive. To pick out the best ones, the algorithm uses Adaboost, a machine learning[21] method that zeroes in on the features with the lowest error rates. It does this by assigning weights to each training image; images that get misclassified have their weights increased, so the algorithm focuses on improving those in the next round. In the end, it combines all the selected weak classifiers into one strong classifier, where each individual one might be weak but together they make a solid face detector.

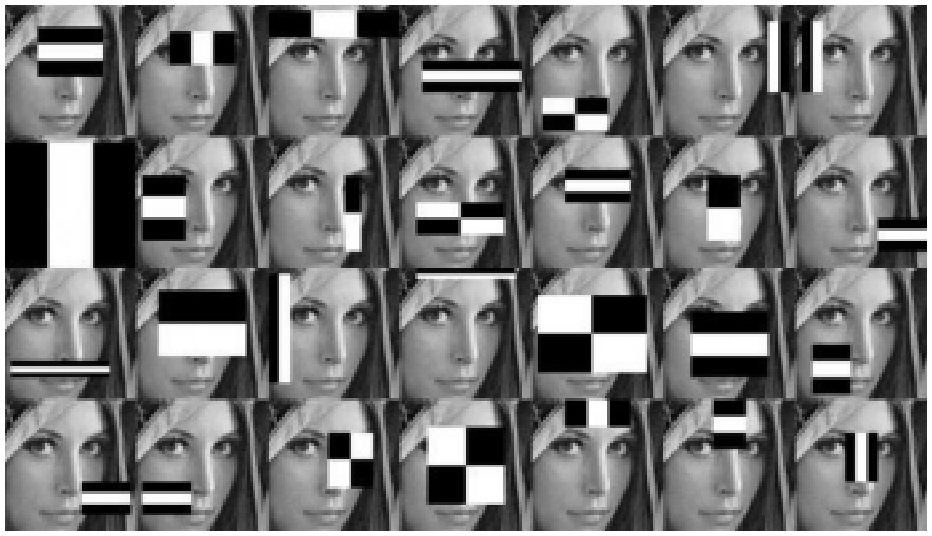
Fig. 2. Haar-like features being used to detect a face

To make detection faster, the Haar Cascade algorithm uses multiple classifiers known as Cascade of Classifiers, where the features are organized into stages. A few features are used at each stage and only the windows that pass the current stage are processed further.This hierarchical approach allows the algorithm to quickly discard non-face regions in an image, reducing the overall computation time.

The original detector described by Viola and Jones[1] comprised more than 6,000 features divided across 38 stages, with the initial stages containing significantly fewer features to quickly eliminate non-face regions. For example, the first stage contained only one feature, while later stages contained up to 50. On average, only 10 features out of the full set of 6,000 were evaluated for each window, leading to both high accuracy and efficient real-time detection.

The strength of Haar Cascade lies in its efficiency and ability to run on less powerful hardware. However, it is prone to errors in complex environments, such as poorly lit areas or when faces are partially obscured.

5 Proposed System

5.1 YOLOv8 Model for Face Recognition

From 2016 onward, the architecture called ‘You Only Look Once’ (YOLO)[2] became a versatile alternative to the popular R-CNN based region-extraction algorithms. YOLO significantly changed the way we look at object detection models. It successfully achieved one-stage object detectors and was based on linear regression as opposed to earlier two-stage approaches. Fast R-CNN and other earlier models were two-stage detectors. They first had to generate candidate bounding boxes[22] and then classifier the objects on those boxes. This process required a lot of computation. But YOLO offered a much better alternative by formulating object detection as a simple linear regression problem rather than having multiple complexities. Thus, the prediction of all the bounding boxes and class probabilities is done in a single forward pass.Over the years, the algorithms under YOLO have seen many advancements. With regard to accuracy, YOLOv2 [5] incorporated approaches such as batch normalisation and the use of anchor boxes, whereas the development of YOLOv3[7] was aimed at enhancing the perception of small objects. YOLOv4[3], however focused on enhancing speed and efficiency to a level where it can go into real-time[7] applications. The newer model of 'You Only Look Once' is an improvement of this trend as it applies sophis ticated convolutional neural network techniques making it quicker and better than its predecessors in the application of face detection in difficult situations.

YOLO is a modern deep-learning model, its application is real-time[7] object detec tion of different objects. In comparison to the previous methods that need many stages of operations within the detection, like the region proposal and the classification, the network carries out the processing in an imaging system within one linear stage, which is a great time operation within the system. This Project is implemented using YOLOv8 as it presents improvement over previous versions like YOLOR, YOLOX, YOLV5, YOLV6, YOLV7[9][23] in terms of speed, accuracy and user-friendly operations.

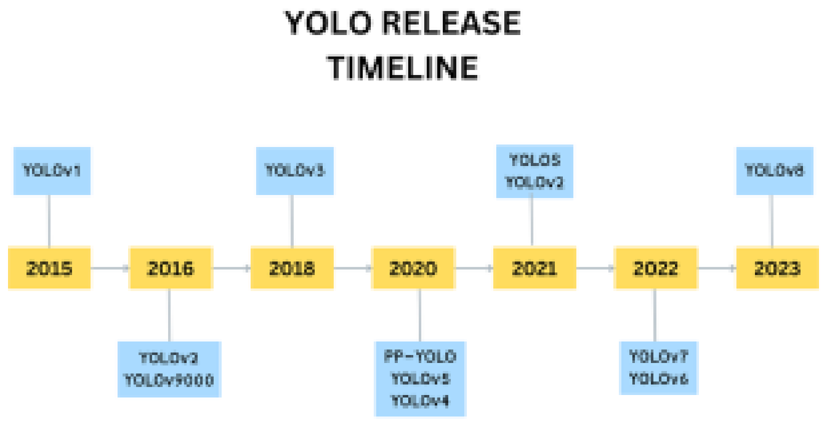

Fig. 3. Timeline of released stable models of YOLO

YOLOv8 uses the multi-scale[3] object detection technique, this lets the model to detect objects of various sizes in a given image. There are some other features that are an improvement on the previous versions and they also include the usage of the ELU activation function. The Exponential Linear Unit (ELU) is a function designed to speed up the reach of densenet neural networks as it solves the problem of vanishing gradient helping to achieve faster convergence. GIoU loss was integrated into YOLOv8 [22]architecture. GIoU loss, or Generalized Intersection over Union loss, incorporates more features than traditional Intersection over Union measurements that is IoU which focuses only on the edges of the objects and gives bounding boxes [22]without consid ering their forms improving object detection.

When it comes to average precision (AP), thresholded at IoU 0.5, the YOLOv8 algorithm displays AP improvement by 1.2 [23]percentage over YOLOv7, which is quite a significant enhancement. This has been achieved despite reducing the model's weight file by 80.6MB[24] for improved effectiveness and easy management in resource constrained settings. YOLO is a cutting-edge real-time[7] object detection system, which has undergone a lot of refinements from when it was first brought to light in 2016 by Joseph Redmon et al. One of the reasons for the Progressive Growing GAN's success is that it separated not only region generation and classification, but also trained the main convolutional layers in an all-at-once manner. In this case, the input image is partitioned into a grid and the bounding boxes with class probabilities for each cell of the grid are predicted at the same time. This type of design provides high precision and allows for real time object detection in the case of YOLO. Most of the earlier detec tion systems, like the R-CNN system for example, consisted of deep machine learning systems that had convoluted architectures and multi-stage representations. Quite to the contrary, YOLO does not offer different stages for generating region proposals and making classifications upon those regions but considers object detection as a regression problem alone.

One of the biggest changes that came with an upgrade to YOLOv8 is that the new design does not rely on anchors. The previous YOLO versions used the approach of defining anchors beforehand. Typed anchors[25] always needed to be tuned and were often the cause of unsatisfactory results for specialized variants of datasets and object types. The anchors are thus completely done away with in YOLOv8, worryingly making the model leaner and more efficacious. With no anchor boxes[10], the architecture is much simplified and the model is better able to learn generalizations for the objects in the images such as faces, irrespective of the scales or placement of the images. Such features extend great relevance in the real-life scenarios such as in face detection[8] systems where one needs to look at the faces which can be turned or far away from the camera

Another notable leap in technology, as far as YOLOv8 is concerned, is using a detection head that is decoupled from the other parts of the system is another critical and crucial innovation. It is meant to ensure that the classification of objects and the regression and localization bounding boxes[22] are done as separate tasks. Almost all previous structures of YOLO have accomplished these 2 contradictory goals in one step. However, due to the advanced design of the parts' interfacing, it was possible to implement such a model as in YOLOv8, where each of the listed operations can be performed in isolation. This configuration has enhanced efficiency such that the model is now able to perform a better discrimination of classes for the objects (in this instance, faces) as well as localization of the objects within the image better. Thanks to the classification and localization divisions, enlargement of precision is reached by

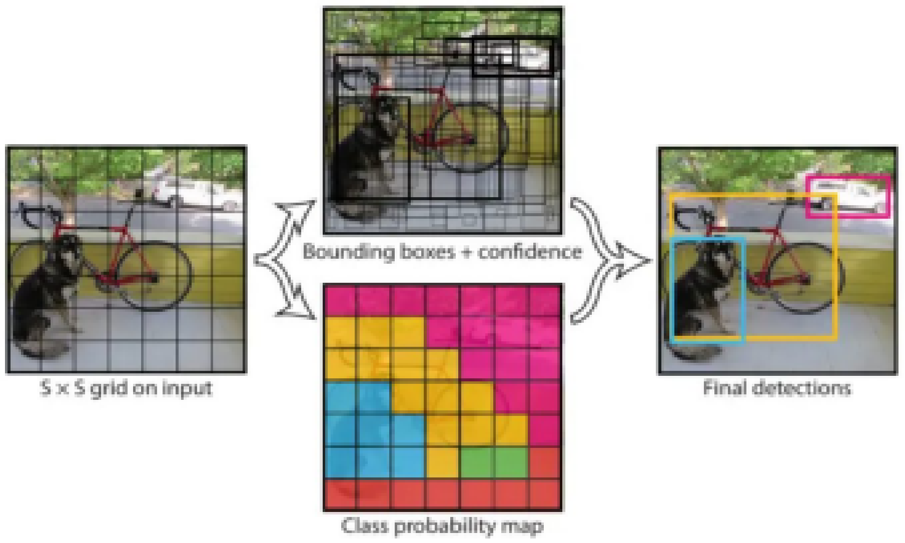

Fig. 4. YOLO predicting bounding boxes and class probabilities from a single pass

YOLOv8, which is very practical for such[8] tasks, as some real-time face detection systems[8] , where speed and precision are critical.

The detection pipeline of YOLOv8 consists of three key units: the backbone, neck, and head. The backbone utilizes a CSP (Cross Stage Partial) network that helps in not only extracting but also distinguishing low-level and high-level features that play an important role in faces detection. The neck makes use of a feature pyramid network (FPN) structure to integrate the features of large and small faces, thus improving the efficiency of face detection[8] in the model. The head, which is the last part, performs separate processing of class predictions and bounding box[22] regression because of a decoupled design. The accuracy and speed of the model are enhanced by this separation, which further increases its efficacy in real-time detecting situations.

6 System Architecture

System architecture involves the various components working together and communi cating effectively to meet the project's goals. The following are the key components:

6.1 Face Detection and Recognition Module

• Input Sources: Loads images, processes videos or captures live feed as the input, from which the frames are captured and fed as input to the face detection model.

- Face Detection : The proposed system outlined in this document includes models (yolov8, Haar cascades) that are used for face detection from input (image / video/webcam). This part of the system works to process the frames and detects bounding boxes for each detected face. The responding part of the system processes the frames and detects the bounding boxes for each of the detected faces.

- Face Recognition: Once faces have been detected, the recognition module compares them with an already-existing database to recognize individual face.
- 1. Emotion Detection: Observing the faces and checking for the primary emotion.

  2. Total People Count: The counting of faces detected per frame for (the) total number of people counted[26].

6.2 Database/Storage/Cloud

- A Database for Recognized Individuals: To store data regarding recognized individuals[17][15] (e.g., name, ID, logs of access).
- For unrecognized faces, the system can store images and metadata for further analysis[20] [19] and manual review.

6.3 Notification and Access Control

- Notification System: Triggers notifications[20][22] to inform when a known individual is detected.
- Access Control: Uses actuators as a prototype to simulate access control. If the detected person is recognized[19], the system allows entry by controlling the motor[27].
- Can trigger alerts if a person’s emotion is flagged[28] as suspicious (security purposes in restricted areas)

6.4 User Interface (UI)

- The system includes a graphical user interface (GUI) built using Tkinter[?] to load images and videos or initiate live detection[23].
- Displays the results of face detection [9]and recognition to the user in real-time.

6.5 Security Features

• Logging and Alerts: Logs access details[20] and can initiate auto security mea sures for unrecognized faces[17] (e.g., sending alerts, manual review).

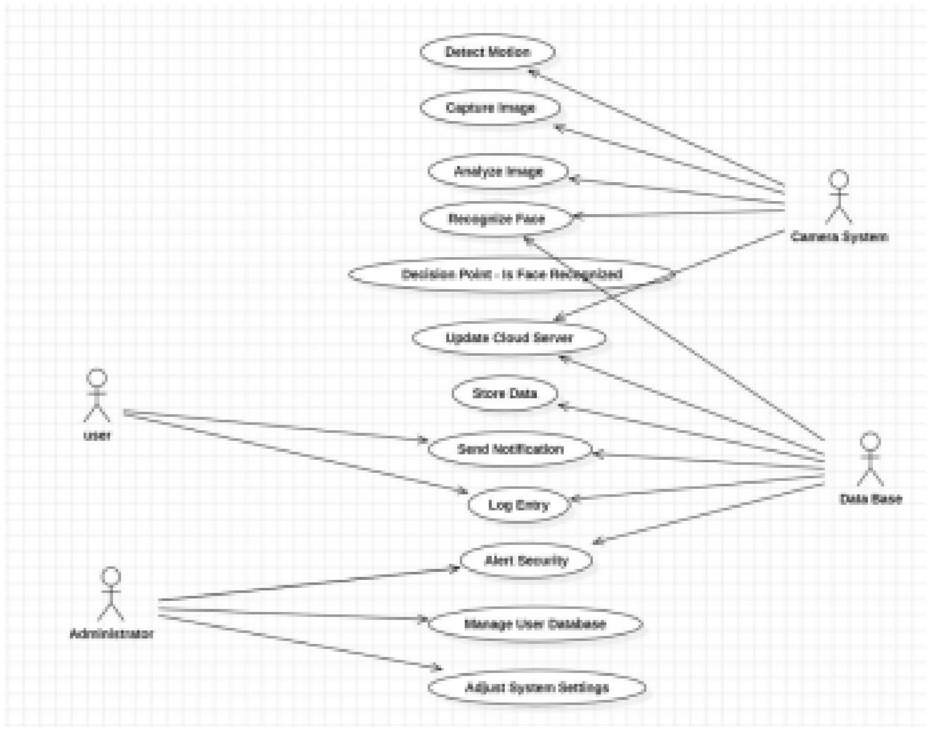

Fig. 5. Use Case Diagram

Actors and Components

1. User: Receives notifications or alerts.
2. Camera System: Detects motion, captures and sends images for analysis.
3. Recognition Software: Processes images for face detection and recognition.
4. Database: Stores data, manages logs, and sends notifications or alerts.
5. Administrator: Manages system settings, reviews unrecognized images, and updates the database.

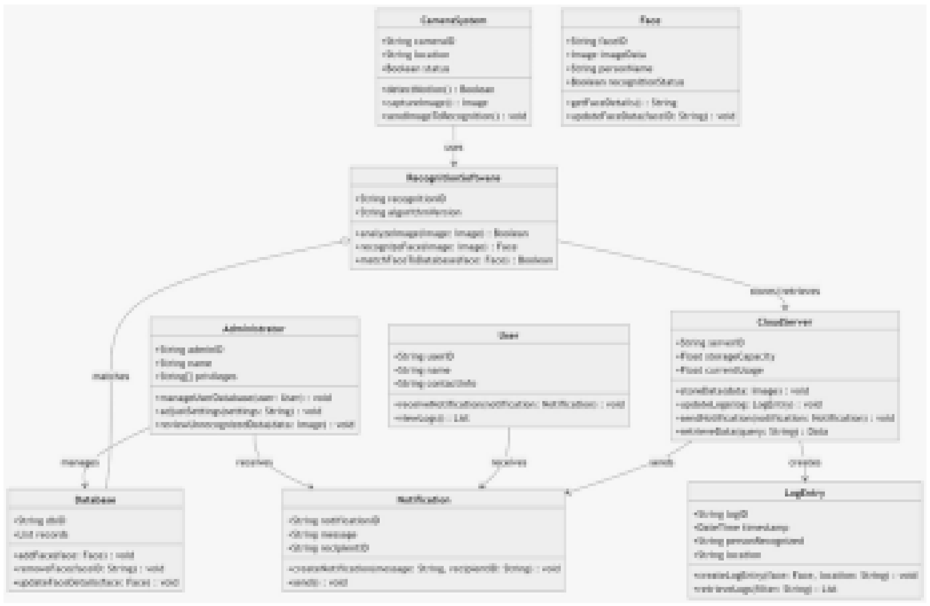

Fig. 6. Class Diagram

Relationships

1. Camera System ”uses” Recognition Software for analyzing and recogniz ing faces (analyzeImage, recognizeFace).

2. Recognition Software has an ”association” with Database to match de tected faces.

3. Recognition Software interacts with Cloud Server to store and retrieve data.

4. Cloud Server ”sends” Notification to User and Administrator.

5.Notifications are “obtained” by the User and Administrator as well as access to Login records.

6.Administrator has a “management” relationship with the Database in handling user data (i.e. adding or deleting faces).

7.LogEntry objects are constructed by Cloud Server if a face was recognized or unrecognized.

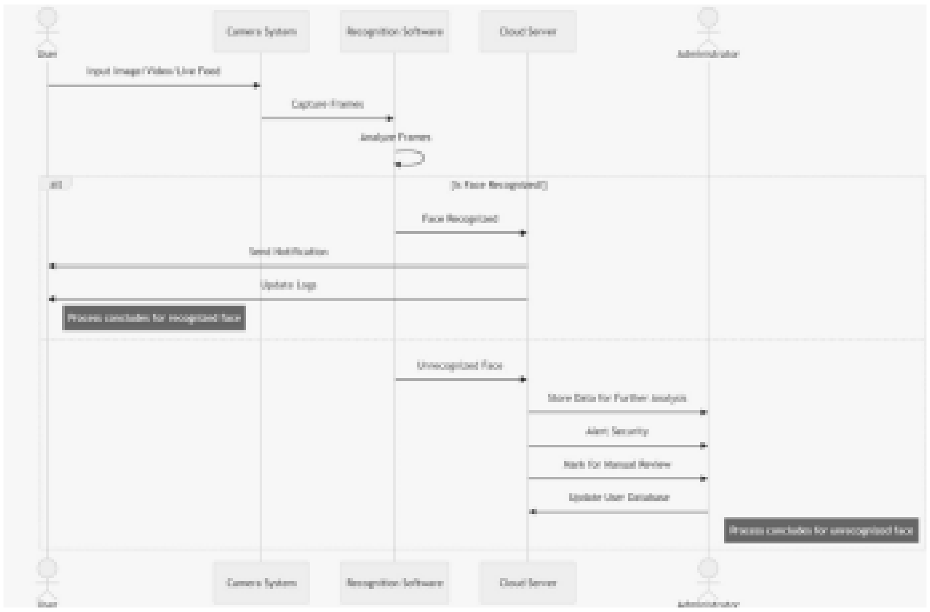

Fig. 7. Sequence Diagram

Actors:

1. User (on the left): Receives notifications and alerts.
2. Camera System: Input, captures images.
3. Recognition Software: Analyzes and matches faces.
4. Cloud Server: Manages data storage and sends alerts.
5. Administrator (on the right): Reviews unrecognized entries and manages the system.

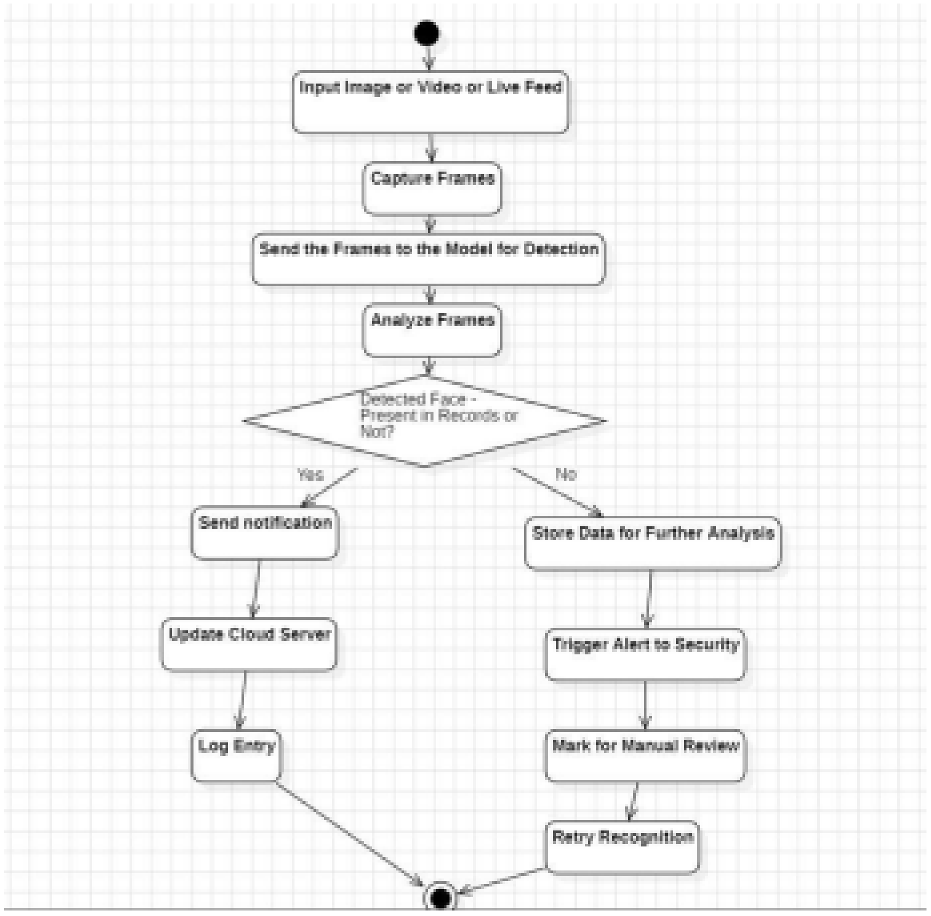
Input Image or Video or Live Feed
Capture Frames
Send the Frames to the Model for Detection
Analyze Frames
Detected Face - Present in Records or Not?
Yes
No
Send notification
Update Cloud Server
Log Entry
Store Data for Further Analysis
Trigger Alert to Security
Mark for Manual Review
Retry Recognition

Fig. 8. Activity Diagram

Flow:

- Start: Input → Capture Frames → Model Detection → Analyze Frames → Decision Point - Is Face Detected?
- If Yes: Continue to Recognize Face → Decision Point - Is Face Recognized? • If Yes:
  - ∗ Send Notification
  - ∗ Update Cloud Server
  - ∗ Log Entry
  - ∗ End
- If No:
  - ∗ Store Data for Further Analysis
  - ∗ Trigger Alert to Security
  - ∗ Mark for Manual Review
  - ∗ Retry Recognition
  - ∗ End

• If No: Return to Input → End

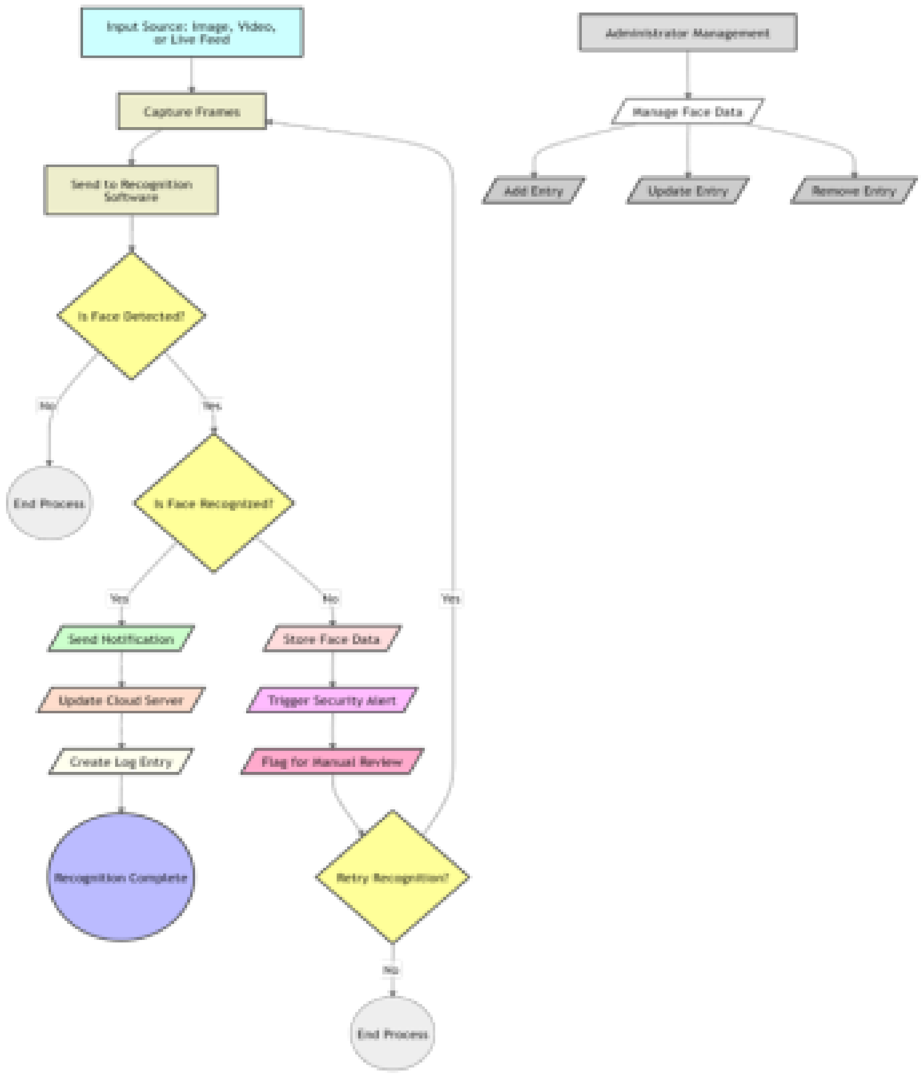

Fig. 9. Data Flow Diagram

7 Tools

The project utilises a range of tools and technologies, such as programming languages, machine learning models, libraries, hardware, and communication protocols.

7.1 Programming Languages

– Python: Python is the main language used in this project for building face detection and recognition algorithms[20], as well as for developing the GUI, thanks to its wide range of libraries and support.

7.2 Machine Learning Models

– YOLOv8 (You Only Look Once): YOLO is a state-of-the-art contemporary deep learning model for object detection[2] that works incredibly well with detecting objects and, in this case, detecting faces. It is known for speed and, therefore, accuracy, which makes it great for real-time processing[7].–

- Haar Cascades: Haar Cascade is a conventional machine learning algorithm[1] that detects faces by using specific features called Haar features. It is typically not used outside of its baseline purpose in reference[16] to observing newer models.

7.3 Libraries and Frameworks

– • OpenCV: Employed to execute computer vision tasks[18], such as modifying images, detecting faces, and drawing bounding boxes.• Ultralytics YOLO: Supports YOLO models[23] for implementation in Python and simplifies the use of YOLOv8 with the project[9]. • Tkinter: Supplies a simple user interface, allowing the user to load images, load video, and run detection process on a live feed of a webcam • NumPy: Simplifies cooperation and data handling, and is the de facto method for image processing

7.4 Hardware and IoT Devices

Microcontroller (Arduino/Raspberry Pi/NodeMCU): Responsible for controlling the servo motor [19], and hardware-controlled access.

Servo Motor: Serves as door lock/unlock system based on the recognition[29] results.

Webcam or External camera: Supplies live feed to the model[22] for the detection purposes.

7.5 Databases

SQLite/MySQL: Designed to store data[20][17], including facial identities, timestamps, and activity logs.

File system: Utilized to store media files[15] and saved unrecognized face images[29].

7.6 Communication Protocols

MQTT: A lightweight messaging communication protocol[19], it is widely used in resource-constrained environments to send control signals to the servo motor[27].

HTTP/REST API: Facilitates synchronization between the face recognition system[22] and the microcontroller.

7.7 Development Environment

IDE/Text Editor: Any IDE or text editor for development[15] can be utilized, including PyCharm[17], Visual Studio Code, or Jupyter Notebooks.

Version Control (Git/GitHub): Used for version control[17] and collaboration with others for version control.

8 Implementation

1. YOLOv8 Model: YOLOv8 is used to detect faces in real-time, with both speed and accuracy[9]. The model can detect faces quickly in difficult settings with multiple faces.
2. Face Recognition Library: This models are employed to encode faces, and hence match encoded images of face profiles to identify an individual[7]. The library is run alongside a live feed & takes the live feed, then contrasts the live encoding with stored, earlier encodings, and is then able to say if they match.
3. Logging utilities: Records the recognized faces, timestamping them in the process. This allows for a record keeping mechanism for audits or manual review, if needed[10].
4. OpenCV: This opens the camera to capture video and processes it frame by frame, displaying it back in real-time [23]. OpenCV is also quite useful since it feeds the detection and recognition pipelines with live footage.
5. Pickle utilities: Once the face encodings are stored in a pickle file, they can be retrieved faster and more efficiently.

8.1 Flow of Execution

Dataset Creation

1. Input: The input for the faces for encodings[1] can be done using static images inside a folder with the person's name, or a webcam live-feed.
2. Detect Faces: The Haar Cascade algorithm is used to detect faces [16] in frames to create the dataset.
3. Save Detected Faces: Store the detected faces[18] in a local directory, resiz ing them for consistency.
4. Encode Faces: Use the face recognition library to generate[30] encodings for each face.
5. Label Encodings: Associate each encoding with the corresponding person's name.
6. Store Encodings: Save these encodings in a serialized format (e.g., Pickle) for future use.

Live Recognition

1. Input Feed: Capture live video feed from the webcam.
2. Face Detection: Use YOLOv8 to detect faces[7] in real-time with high accu racy.
3. Recognition: Compare detected faces with stored encodings to identify indi viduals.
4. Logging: Record recognized faces with timestamps in a CSV file for trace ability.
5. Display Results: Show detected faces and recognition[9] results in real-time on the GUI or video feed.

8.2 Dataset Creation

Directory Structure

We will create a dedicated directory to house the datasets of images.

• Every subdirectory will correspond to one individual, allowing for images to be associated with each name.

Haar Cascade for Initial Detection

- *Haar-like features will be used to detect face regions [18].*
- *Detected images will be standard resized to fixed resolution(i.e. 200x200) and saved.*

Encoding and Labeling

- *Each detected face will be encoded with the face recognition library, and each will be labelled with a name for later use in recognition [30]. Storing the encodings:*

Storing the encodings

- *Pickle will be used to serialize the encodings for efficient reading and stored with a .pkl file.*

8.3 Face Detection with YOLOv8

8.4 Face Recognition

The face recognition model assists in the accurate identification of the faces. The detected faces are then encoded and these encodings are compared with the pre stored encodings. Thus ensuring the satisfactory face recognition of individuals depending on the trained dataset.

8.5 Emotion Detection

- *Emotion Detection: The model classifies expressions into seven key categories: neutral, angry, sad, happy, fear, disgust, and surprise.*
- *Detection Pipeline: The preprocessed image is passed through the emo tion detection pipeline, and the emotion with the highest probability match is selected as the detected emotion.*
- *Visual Representation: Each detected face is surrounded by a bounding box whose color corresponds to the detected emotion, providing an intuitive visual representation.*
- Emotion Colors:
  - ∗ Neutral: Green (0, 255, 0)
  - ∗ Angry: Red (0, 0, 255)
  - ∗ Sad: Blue (255, 0, 0)
  - ∗ Happy: Yellow (0, 255, 255)
  - ∗ Fear: Purple (128, 0, 128)
  - ∗ Disgust: Dark Green (0, 128, 0)
  - ∗ Surprise: Orange (255, 165, 0)
- Bounding Box: The detection box is drawn with the corresponding emotion color:

8.6 Logging Recognized Faces

A robust logging system records recognized faces with timestamps.This ensures traceability and helps in maintaining records for future reference.

9 Conclusion

Implementing face detection and recognition of the selected models by multiple ap proaches in this capstone provides evidence of how amply flexible and efficient different machine learning models can be. The main objectives were addressing two well-known algorithms-YOLOv8 and Haar Cascades-for detecting and recognizing faces against their varied scenarios: single, multiple face detection, and single, and multiple face recognition.

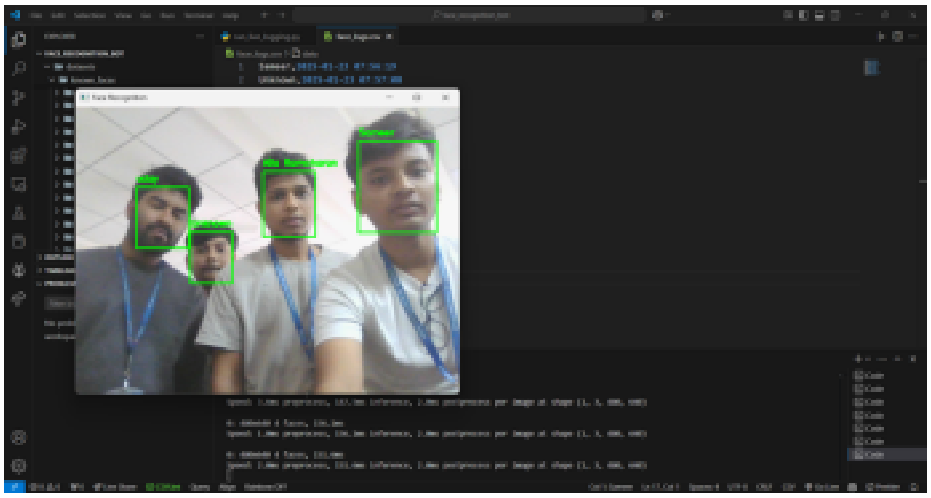
Fig. 10. Face Recognition Bot working on live feed

The comparison of YOLOv8 and Haar Cascade illustrates the different strengths and weaknesses of the models. YOLOv8 has all the rates and robustness of modern deep-learning-based approaches to detect multiple faces at the same time using varying lighting and occlusions[7] [9] [23]. This performance is attributed to the model’s ability to learn very complicated patterns in data via convolutional neural networks, making it a good candidate for real-time applications. In contrast, Haar Cascades’ detection speed was faster, owing to its lightweight nature; however, this system exposed less consistent performance when the environment[1] [16][30] was complex scenarios are not traditional approaches for accuracy in connection to the changed modern applications.

By integrating IoT, the face recognition system gets enriched by a huge increment in utility. The model will showcase a highly efficient and automated entry management system, where the controlling mechanism of access is by actuators[19] [27]. Security aspects may improve in real applications when the system recognizes the known person on entry and notifies the security service. For unknown faces, backing up the data for future analysis ensures ongoing learning and improvement of the capture model [17], [15]. This powers a strong framework for security applications by integrating impediments like automated notifications and manual review processes.

This project incorporates established features of the integration of clear face detec tion models like YOLOv8 with IoT-based automation. It does a comparison between YOLOv8 and HaarCascades, explaining the progress that has been achieved in deep learning for computer vision, and underlining how traditional methods may still have their worth in resource-constrained[20], [29]. environments[1], [20]. Together with the strengths of each model and IoT for real-time decision-making, this project offers a stepping stone towards subsequent intelligent developments in security systems.

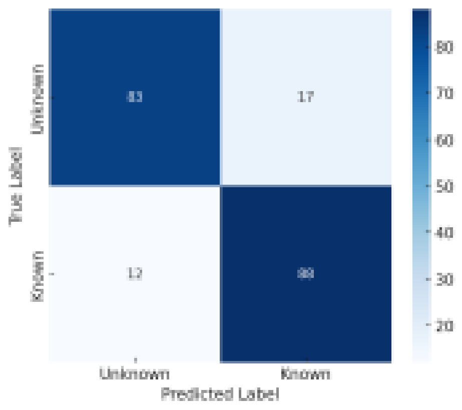

Fig. 11. Prediction and True Label

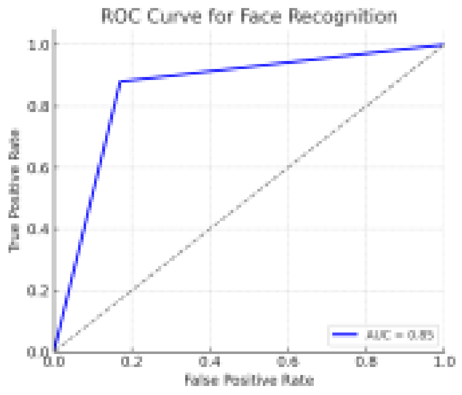

Fig. 12. ROC Curve

References

1. J. Nishimura and T. Kuroda, “Versatile recognition using haar-like feature and cascaded classifier,” IEEE Sensors Journal, vol. 10, no. 5, pp. 942–951, May 2010.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
3. C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-yolov4: Scaling cross stage partial network,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 13 024–13 033.
4. L. Meng, W. Yuan, M. Li, B. Zhang, and L. Zhang, “Facial detection algorithm based on improved yolov5s,” in 2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), vol. 6, 2024, pp. 1605–1609.
5. X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, X. Hu, D. Yu, Y. Ma, and O. Yoshie, “Pp-yolov2: A practical object detector,” arXiv, 2021.
6. J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv, 2018.
7. A. Bochkovskiy, C.-Y. Wang, and H. yuan Liao, “Yolov4: Optimal speed and accuracy of object detection,” arXiv, 2020.
8. H. Aung, A. V. Bobkov, and N. L. Tun, “Face detection in real time live video using yolo algorithm based on vgg16 convolutional neural network,” in International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 2021, pp. 697–702.
9. S. Tariyal, R. Chauhan, Y. Bijalwan, R. Rawat, and R. Gupta, “A comparative study of mtcnn, viola-jones, ssd and yolo face detection algorithms,” in International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), 2024, pp. 1–7.
10. M. Shi and Y. Gao, “Lightweight real-time face detection method based on improved yolov4,” in International Conference on Computer Information Science and Artificial Intelligence (CISAI), 2021, pp. 273–277.
11. D. Shukla, R. Kumari, and A. Bhargavi, “Human face detection and emotion recognition using opencv through ai,” in 2024 12th International Conference on Internet of Everything, Microwave, Embedded, Communication and Networks (IEMECON), 2024, pp. 1–5.
12. H. Al Fatta, A. Dahlan, and L. D. Farida, “Systematic literature review on realtime emotion detection using ai approach,” in 2024 7th International Conference on Information and Communications Technology (ICOIACT), 2024, pp. 222–227.
13. A. Anish, S. R, A. H. Malini, and T. Archana, “Enhancing surveillance systems with yolo algorithm for real-time object detection and tracking,” in 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), 2023, pp. 1254–1257.
14. O. Sahin and S. Ozer, “Yolodrone: Improved yolo architecture for object detection in drone images,” in 2021 44th International Conference on Telecommunications and Signal Processing (TSP), 2021, pp. 361–365.
15. P. J. Lu and J. H. Chuang, “Fusion of multi-intensity image for deep learning-based human and face detection,” IEEE Access, vol. 10, pp. 8816–8823, 2022.
16. J. Nishimura and T. Kuroda, “Multiaxial haar-like feature and compact cascaded classifier for versatile recognition,” IEEE Sensors Journal, vol. 10, no. 11, pp. 1786–1795, 2010.

17. M. M. A. P. et al., “Navigating the yolo landscape: A comparative study of object detection models for emotion recognition,” IEEE Access, vol. 12, pp. 109 427–109 442, 2024.
18. C. Kim, S. I. Choi, M. Turk, and C. H. Choi, “A new biased discriminant analysis using composite vectors for eye detection,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 4, pp. 1095–1106, 2012.
19. I. V. S. L. Haritha, M. Harshini, S. Patil, and J. Philip, “Real-time object detection using yolo algorithm,” in 6th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2022, pp. 1465–1468.
20. H. J. Mun and M. H. Lee, “Design for visitor authentication based on face recognition technology using cctv,” IEEE Access, vol. 10, pp. 124 604–124 618, 2022.
21. Q. Chen, N. D. Georganas, and E. M. Petriu, “Hand gesture recognition using haar-like features and a stochastic context-free grammar,” IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 8, pp. 1562–1571, 2008.
22. A. Anish, S. R, A. H. Malini, and T. Archana, “Enhancing surveillance systems with yolo algorithm for real-time object detection and tracking,” in 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), 2023, pp. 1254–1257.
23. A. A. S. Chan, M. F. L. Abdullah, S. M. Mustam, F. A. Poad, and A. Joret, “Face detection with yolov7: A comparative study of yolo-based face detection models,” in International Conference on Green Energy, Computing and Sustainable Technology (GECOST), 2024, pp. 105–109.
24. A. A. Sufian Chan, M. Abdullah, S. M. Mustam, F. A. Poad, and A. Joret, “Face detection with yolov7: A comparative study of yolo-based face detection models,” in 2024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), 2024, pp. 105–109.
25. M. Shi and Y. Gao, “Lightweight real-time face detection method based on improved yolov4,” in 2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI), 2021, pp. 273–277.
26. T. Khajontantichaikun, S. Jaiyen, S. Yamsaengsung, P. Mongkolnam, and U. Ninrutsirikun, “Emotion detection of thai elderly facial expressions using hybrid object detection,” in 2022 26th International Computer Science and Engineering Conference (ICSEC), 2022, pp. 219–223.
27. O. Sahin and S. Ozer, “Yolodrone: Improved yolo architecture for object detection in drone images,” in 44th International Conference on Telecommunications and Signal Processing (TSP), 2021, pp. 361–365.
28. A. M. Cadayona, N. M. S. Cerilla, D. M. M. Jurilla, A. K. D. Balan, and J. C. d. Goma, “Emotional state classification: An additional step in emotion classification through face detection,” in 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), 2019, pp. 667–671.
29. T. Wu and X. Fan, “A novel gesture recognition model under sports scenarios based
on kalman filtering and yolov5 algorithm,” IEEE Access, vol. 12, pp. 64 886–64 896, 2024.
30. L. Goldmann, U. J. Monich, and T. Sikora, “Components and their topology for robust face detection in the presence of partial occlusions,” IEEE Transactions on Information Forensics and Security, vol. 2, no. 3, pp. 559–569, 2007.