*electronics*

*Article*

# Artifical Intelligence-Based Smart Security System Using Internet of Things for Smart Home Applications

**Hakilo Sabit** [1,2]

[1] Department of Electrical and Electronic Engineering, Auckland University of Technology, Auckland 1010, New Zealand; hakilo.sabit@aut.ac.nz
[2] School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand

**Abstract:** This study presents the design and development of an AI-based Smart Security System leveraging IoT technology for smart home applications. This research focuses on exploring and evaluating various artificial intelligence (AI) and Internet of Things (IoT) options, particularly in video processing and smart home security. The system is structured around key components: IoT technology elements, software management of IoT interactions, AI-driven video processing, and user information delivery methods. Each component's selection is based on a comparative analysis of alternative approaches, emphasizing the advantages of the chosen solutions. This study provides an in-depth discussion of the theoretical framework and implementation strategies used to integrate these technologies into the security system. Results from the system's deployment and testing are analyzed, highlighting the system's performance and the challenges faced during integration. This study also addresses how these challenges were mitigated through specific adaptations. Finally, potential future enhancements are suggested to further improve the system, including recommendations on how these upgrades could advance the functionality and effectiveness of AI-based Smart Security Systems in smart home applications.

**Keywords:** AI; smart home security; IoT; face recognition; motion detection; smart home app

check for updates

Academic Editor: Valeri Mladenov

Received: 2 January 2025
Revised: 28 January 2025
Accepted: 1 February 2025
Published: 4 February 2025

**Citation:** Sabit, H. Artifical Intelligence-Based Smart Security System Using Internet of Things for Smart Home Applications. *Electronics* **2025**, *14*, 608. https://doi.org/10.3390/electronics14030608

## 1. Introduction

At the start of the twenty-first century, the world entered the Internet era, fundamentally transforming how people live and work. With the rapid advancement of information technology and the Internet, a new application emerged, known as the Internet of Things (IoT) [1]. A smart home is an application within the IoT environment, consisting of physical devices connected to the Internet. These devices communicate with one another to deliver innovative, intelligent services to users. Over time, advancements in technology have significantly shifted people's expectations regarding Home Automation and how they interact with their homes [2]. The increasing affordability and widespread use of electronic devices and the Internet have played a key role in this transformation. Today's modern Home Automation Systems are a sophisticated blend of Ubiquitous Computing Devices and Wireless Sensor/Actuator Networks. However, the growing demand for 'Convenient Access' has introduced new security challenges within the Home Automation landscape. There is a pressing need for a viable and accessible approach that enables the collection of data and identification of risky behaviors within smart-home environments [3], while ensuring user privacy is preserved. Despite the growing interest in integrating artificial intelligence (AI) and Internet of Things (IoT) technologies in smart home security, there remains a significant research gap in thoroughly exploring and evaluating the range of

AI and IoT options, particularly in the context of video processing. Limited studies have focused on comparing various AI algorithms and IoT configurations to determine their effectiveness and adaptability for real-time security applications in smart homes. This gap underscores the need for a comprehensive analysis of different AI-driven video-processing techniques and IoT frameworks, which could reveal valuable insights for enhancing security, optimizing resource use, and addressing privacy concerns in the smart home domain. To address these challenges, this study proposes to develop innovative strategies that leverage the capabilities of artificial intelligence (AI) and Internet of Things (IoT) technologies. This includes exploring a range of AI algorithms suited for real-time data processing, enhancing IoT connectivity for seamless device integration, and implementing robust privacy-preserving mechanisms. This study presents the development of a smart security system designed to monitor specific areas and notify a designated user when motion is detected, all while maintaining a high level of user privacy. The system is composed of three primary components: a motion sensor, a computational unit capable of performing advanced computations, and a camera. Upon detecting motion, the system activates the camera, which is then utilized by the computational unit to apply artificial intelligence (AI) techniques. The user is subsequently notified of the detected motion and is provided with the option to view the AI-processed camera feed. Figure 1 illustrates the proposed smart security system function routine. A primary functionality of this project is its capacity to alert the user upon motion detection. Following this, the system enables the user to access either the live or recorded camera feed, with the AI model assisting in identifying the subject. Alternatively, the alert can be dismissed if the AI recognizes the individual as non-threatening. This study addresses the issue of home security and its common limitations. Although many households rely on security cameras to safeguard their property, it is impractical to continuously monitor camera footage. This study seeks to overcome this challenge by developing a more intelligent and autonomous security system that does not require constant human oversight and only issues notifications when necessary. Throughout the project, user privacy remained a paramount concern, and critical decisions were made to safeguard privacy, significantly shaping the project's direction from its initial planning stages onward.

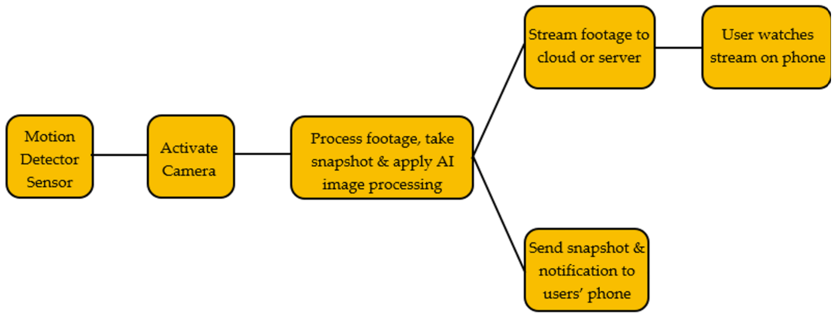

Motion Detector Sensor
Activate Camera
Process footage, take snapshot & apply AI image processing
Stream footage to cloud or server
User watches stream on phone
Send snapshot & notification to users' phone

**Figure 1.** The proposed smart security system function routine.

The objective of the AI system is to assist users in assessing the threat level of a given situation, enabling them to determine whether it is necessary to view the camera feed or conclude that the situation poses no risk. The AI accomplishes this by first applying motion tracking to moving objects or individuals, facilitating the initial detection of movement. In the subsequent stage, the AI employs facial recognition to further evaluate the level of threat. If the AI identifies a known individual, it classifies the situation as non-threatening. However, if the AI does not recognize the individual, it signals a potential security breach, indicating a heightened threat level. In this study, OpenCV will be employed due to its status as an open-source software platform offering a comprehensive computer vision library suitable for commercial applications. OpenCV is widely adopted by major companies, including Google, Microsoft, Intel, and Sony, among others [4]. It supports multiple programming languages, such as C++, Python, Java, and MATLAB, providing flexibility in selecting the platform for AI implementation without compromise. Furthermore, the extensive online documentation and support available for OpenCV significantly enhance the efficiency of troubleshooting and research efforts. The system is designed to provide users with significant control over its functionality. Users can specify the duration for which the camera remains active upon motion detection and have the option to activate the camera remotely via cloud access, thereby eliminating the need for physical intervention. Furthermore, the system supports continuous customization of the AI model, allowing users to adjust parameters such as the motion tracking threshold (i.e., the minimum movement size to be tracked) and to add new faces to the facial recognition database.

The rest of this paper is organized as follows: Section 2 reviews related work, while Section 3 introduces the proposed security model. Section 4 describes the security system application, and Section 5 presents the results. Section 6 discusses the findings, and Section 7 concludes the paper with suggestions for future work.

## 2. Related Work

### *2.1. Face Detection Algorithms and Libraries*

Face detection and recognition are essential tasks in computer vision, with applications spanning security systems, social media, and beyond. Over time, numerous algorithms and frameworks have been developed to address the challenges of detecting and recognizing faces in images and video streams. One widely adopted and lightweight framework is OpenCV, which provides classical methods like the Haar Cascade Classifier and Local Binary Patterns (LBPs) for real-time detection. Researchers, including those in home security applications [5–8], commonly utilize OpenCV for face detection and recognition. However, traditional OpenCV methods often fall short in accuracy compared to modern deep learning-based approaches, especially in complex environments. The Single Shot Multibox Detector (SSD) is a deep learning-based method known for its real-time performance in object detection, including faces, by using a single neural network to predict bounding boxes and class scores in one pass [9–11]. Despite its speed, SSD struggles with faces in extreme poses or occlusions. The Multi-task Cascaded Convolutional Networks (MTCNN), which uses a cascade of three convolutional networks for face detection, bounding box regression, and facial landmark detection, offers high accuracy and robustness in varying orientations, but it is slower and can face difficulties in challenging lighting conditions or occlusions [12–14]. FastMTCNN improves upon this by offering similar accuracy with faster processing, though it is less effective in extreme conditions [15,16]. RetinaFace provides exceptional accuracy and robustness under difficult conditions, but it is computationally intensive and slower [17–19]. MediaPipe is a fast, real-time framework suitable for mobile devices, though it is less accurate for face recognition [20,21]. YOLOv8 stands out for its remarkable speed and accuracy in real-time detection and recognition, though it requires significant computational resources [22–24]. YuNet excels in multi-pose detection and recognition in diverse conditions, but it is computationally demanding and slower on lower-end devices [25–27]. Lastly, CenterFace is optimized for real-time applications, particularly in crowded environments, but its performance can decline for small faces or extreme angles. Summary of the face detection librarires and tools are presented in Table 1.

**Table 1.** Face detection libraries and tools comparison.

| Algorithm | Speed | Accuracy | Robustness to Pose | Resource Efficiency |
|---|---|---|---|---|
| Haar-Cascade | High | Moderate | Low | High |
| HOG + SVM | Moderate | Moderate | Moderate | Moderate |
| SSD | High | High | Moderate | Moderate |
| YOLO | Very High | High | Moderate | Low |
| MTCNN | Moderate | High | High | Moderate |
| Faster R-CNN | Low | Very High | High | Low |

### *2.2. Face Recognition Models*

Facial recognition, a technology used to verify personal identities by pinpointing and measuring facial features from a digital image or a video frame, has become an important technology for various applications. With real-time face recognition, faces can be detected and identified instantly using a camera feed or video stream. There are many robust open-source libraries and tools available today that make it easy to build real-time facial recognition capabilities into applications and systems. Over the years face recognition models have evolved significantly, leveraging advancements in machine learning and deep learning. These models vary significantly in methodology, performance, and application. Traditional models like Eigenfaces and Fisherfaces [28–31] utilize PCA and LDA, respectively, to reduce dimensionality and enhance class separability, making them computationally efficient but sensitive to lighting, pose, and dataset size. The Local Binary Pattern Histogram (LBPH) model employs local feature encoding, offering robustness to lighting variations but limited scalability for complex datasets [32–35]. Modern deep learning-based models, such as DeepFace [36,37] and FaceNet [38–41], leverage CNNs and embedding techniques to achieve state-of-the-art accuracy and robustness across diverse conditions, though they require significant computational resources and extensive training data. Dlib [42–46] combines HOG and CNNs for lightweight applications, while DeepID and ArcFace push the boundaries of accuracy and generalization with advanced loss functions, making them ideal for large-scale biometric and commercial systems. These models balance trade-offs between computational cost, accuracy, and dataset requirements, catering to varied real-world scenarios. The different models comparison is presented in Table 2.

**Table 2.** Face recognition models comparison.

| Model | Accuracy | Robustness | Scalability | Computational Cost |
|---|---|---|---|---|
| Eigenfaces | Moderate | Low | Low | Low |
| Fisherfaces | High | Moderate | Moderate | Moderate |
| LBPH | Moderate | High | Low | Low |
| DeepFace | Very High | Very High | High | High |
| FaceNet | State-of-the-Art | Very High | Very High | High |
| Dlib | High | High | Moderate | Moderate |
| DeepID | Very High | Very High | High | High |
| ArcFace | State-of-the-Art | Very High | Very High | High |

### *2.3. Security Vulnerabilities and Privacy in Smart Home AI*

The increasing reliance on smart home devices on the Internet of Things (IoT) has raised concerns regarding security and privacy, prompting various studies to propose methodologies and frameworks for improving the safety and efficiency of these systems. Murat et al. [47] highlight the importance of securing smart home devices and introduce a comprehensive methodology for IoT security assessments. Their work suggests that deeper firmware analysis and automated fuzzing capabilities could significantly enhance the robustness of these assessments. Similarly, Kim et al. [48] propose a forensic methodology tailored to smart home devices with displays, revealing that these control devices often store substantial amounts of user-related information. They emphasize the need for further research to refine the methodology and broaden its applicability. Shah et al. [49] present a secure smart home framework that utilizes AI algorithms, such as Isolation Forest (IF) and K-Nearest Neighbors (KNN), for anomaly detection, alongside blockchain and IPFS for secure data storage. Their research also addresses the challenges of blockchain latency and mining costs, proposing solutions for efficient and secure smart home operations. Wang et al. [50] focus on the Connected Internet of Things (CIoT) in smart homes, offering an architecture that incorporates blockchain, federated learning, and a Gateway Peer method to enhance security, scalability, and automation. Rahim et al. [51] evaluate logit-boosted CNN models for anomaly detection and face recognition in smart home IoT devices, achieving high accuracy while identifying future research directions, such as improving model generalizability and addressing privacy concerns. Lastly, Asghar et al. [52] analyze existing user-authentication schemes for smart homes, identifying vulnerabilities and proposing an enhanced scheme, with future research focusing on adaptive authentication and scalability. These studies collectively contribute to advancing the security, scalability, and performance of smart home systems.

### *2.4. Privacy Concerns and Mitigation Strategies for Home Security Cameras*

The primary privacy concerns associated with home security cameras include unauthorized access [53], data privacy [54], surveillance overreach [55], and physical security risks.

Unauthorized Access: Cameras are susceptible to hacking, weak passwords, and outdated firmware, which can compromise their security. Mitigation strategies include employing strong, unique passwords, enabling two-factor authentication (2FA), and regularly updating firmware to address vulnerabilities.

Data Privacy: Concerns in this area stem from risks associated with cloud storage, unclear data policies, and potential third-party sharing without user consent. To safeguard data privacy, it is essential to use cameras that feature end-to-end encryption and adhere to transparent privacy policies. Additionally, opting for local storage or trusted cloud providers with robust security measures is recommended.

Surveillance Overreach: Cameras may inadvertently capture areas beyond their intended focus, such as neighboring properties or private moments. This issue can be mitigated by employing privacy-enhancing techniques such as privacy masks, geofencing, or adjustable recording zones. Proper positioning of cameras to restrict their field of view to intended areas further reduces the risk of overreach.

Physical Security Risks: Stolen or tampered cameras can result in the loss of footage and privacy breaches. To address these risks, cameras should be secured in tamper-resistant mounts, and footage should be regularly backed up. Enabling tamper alerts provides an additional layer of protection against unauthorized physical access.

### *2.5. Encryption Standards*

Encryption standards are essential for securing digital data by ensuring confidentiality, integrity, and authenticity. Common standards include AES, known for its strong security and performance with 128, 192, or 256-bit keys; RSA, which uses public and private keys for secure exchanges; and TLS/SSL, ensuring secure internet communications, such as HTTPS. SHA provides data integrity checks, and ECC offers strong encryption with shorter keys suitable for IoT devices. These standards are vital for secure communications, protecting e-commerce transactions, and complying with data protection regulations.

### *2.6. Anonymization Techniques*

Digital image anonymization is crucial for protecting individual privacy by obscuring identifiable information in images. Techniques such as blurring, pixelation, and masking are commonly employed to conceal sensitive features like faces or license plates. Advanced methods include the use of generative models, such as the Realistic Anonymization using Diffusion (RAD) framework, which utilizes Stable Diffusion and ControlNet to produce high-quality synthetic images [56]. Additionally, selective feature anonymization approaches, like the privacy-preserving semi-generative adversarial network (PPSGAN), add noise to class-independent features to maintain data utility while protecting privacy [57]. These techniques balance the need for data utility with the imperative of privacy protection, ensuring compliance with regulations and ethical standards.

### *2.7. Compliance with Regulations Such as ENISA/ETSI*

Home security cameras must adhere to regulations set by ENISA (European Union Agency for Cybersecurity) and ETSI (European Telecommunications Standards Institute) to ensure privacy and data protection [58]. ENISA provides guidelines for securing IoT devices, emphasizing strong authentication, encryption, and secure data storage. ETSI focuses on telecommunications standards, recommending frameworks for secure communication protocols and privacy policies. Compliance with these standards ensures that home security cameras protect user privacy and meet legal requirements.

## 3. The Proposed Security Model

The developed system integrates a Raspberry Pi equipped with a camera and passive infrared (PIR) sensors. The camera remains in low-power mode until the PIR sensors detect motion. Upon activation, the camera initiates recording, and the captured footage is processed using OpenCv's *Dlib* and *face-recognition* libraries to apply both face detection and facial recognition algorithms. This processed footage is then uploaded to the user's Google Drive database. Simultaneously, the system sends the user an email containing a link to the uploaded footage. Additionally, a text message is dispatched to alert the user of the detected disturbance, directing them to check the provided email link. A proof-of-concept application has also been developed to serve as a user interface, enabling the user to interact with the security system by viewing notifications and accessing stored footage.

The system is structured around key components: IoT technologies (PIR Sensor and Camera), AI-driven video processing (Object Detection and Face Recognition), user information delivery methods, and hardware platform. Figure 2 shows the system block diagram.

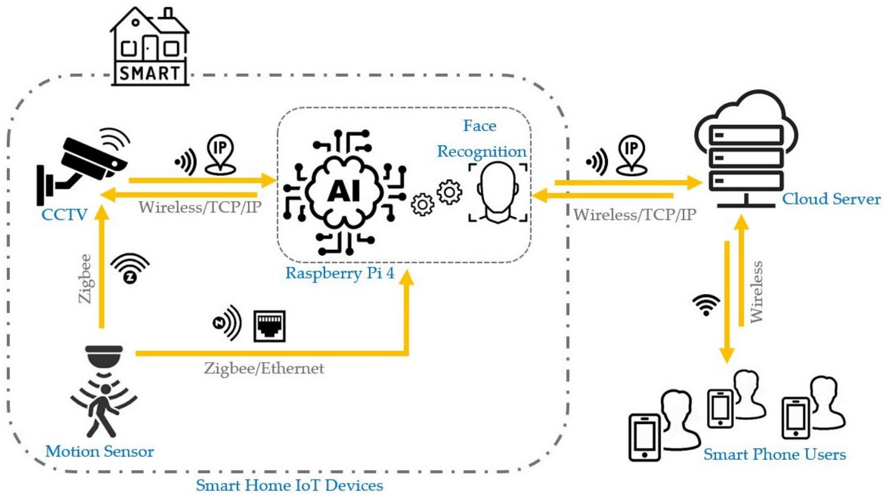


**Figure 2.** The proposed smart security systemblock diagram function routine.

### *3.1. IoT Technology*

Internet of Things (IoT) is the vast array of physical objects equipped with sensors and software that enable them to interact with little human intervention by collecting and exchanging data via a network [59]. IoT technology comprises essential hardware, IP, tools, systems, sensors, and software that enable the creation of smart devices, ranging from medical equipment to industrial machinery. It also includes security features to protect these connected devices from online threats. The IoT working principles are depicted in Figure 3. The IoT technology elements utilized in realizing the presented application are described in the following sections.

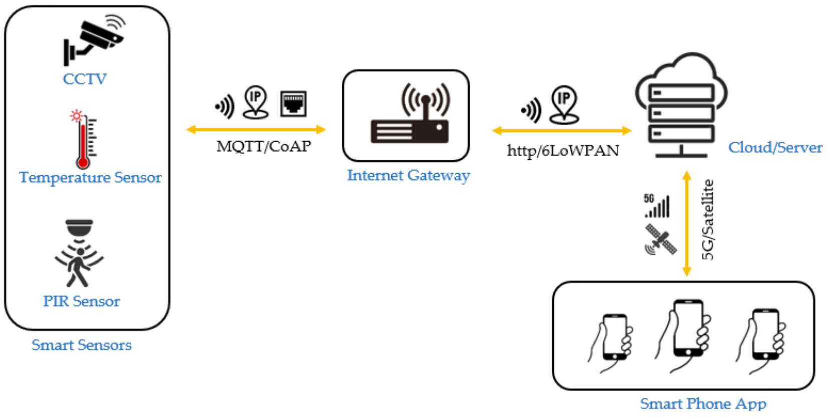


**Figure 3.** Iot technology working.

#### 3.1.1. PIR Sensor

The sensor selected for the security system was the passive infrared sensor (PIR). It works on the principle of the Pyroelectric Effect, as modeled in Equation (1) below [60]. PIR sensors detect infrared radiation emitted by all objects that generate heat. The advantage of PIR sensors lies in their method of detecting differential heat energy. They detect motion by measuring changes in infrared radiation across their two separate sensing elements. When an IR-emitting object crosses the sensing element path, the first element measures it and generates a HIGH signal. When the object crosses the second element's path, the element generates a LOW signal. On detecting a large difference between the signals from the two sensing elements, the sensor outputs a HIGH signal, as shown in Figure 4. This enables the sensor to respond specifically to the heat emitted by a person, making it less susceptible to false signals from non-heat-based movements, such as wind. However, a limitation of PIR sensors is their likelihood of being triggered by animals, a challenge also common to other types of sensors. They are "passive" since they do not emit any heat or energy themselves. It is crucial to note that PIR sensors respond to infrared radiation, not to heat directly.

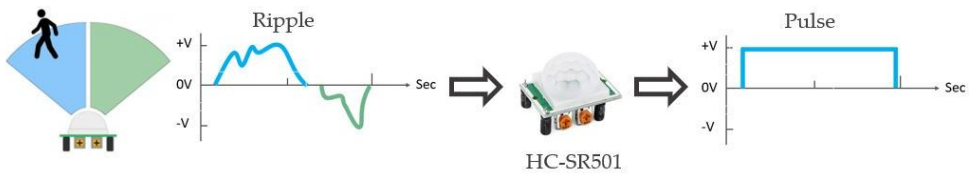


**Figure 4.** The PIR sensor operation.

The voltage response $V(t)$ of a pyroelectric crystal to an infrared (IR) radiation step pulse can be modeled by considering the temperature change in the crystal caused by the absorbed radiation and the resulting polarization change due to the pyroelectric effect. The general expression is given in Equation (1).

$$V(t) = \frac{p\Delta T}{C} \cdot e^{-t/\tau} \tag{1}$$

where we have the following:

- $p$: Pyroelectric coefficient ($C/m^2K$), describing the change in polarization per unit temperature;
- $\Delta T$: Temperature change in the crystal caused by the IR step pulse;
- $C$: Capacitance of the crystal ($F$);
- $t$: Time after the step pulse is applied.

Other sensors such as the Active Infrared and Ultrasonic Sensors (Figure 5) can only be used for quite small areas, as they typically work linearly in such a way that if an intruder walks through a small area such as a doorway, they will interrupt the reflection of a light beam or sound wave, setting off the security system. The PIR sensor, in comparison, can cover large open areas, which matches the typical placement of camera systems where the full range of the camera field of view can be utilized. This makes it the most versatile option for a single security unit system where the sensor, camera, and processing hardware are all part of a combined unit.

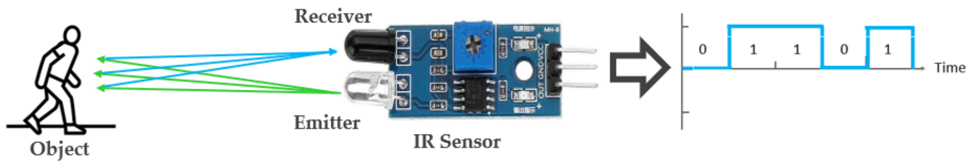


**Figure 5.** The ultrasonic sensor operation.

### 3.1.2. Camera

A camera is used as an edge device that captures valuable footage and provides insight into the security system. The Raspberry Pi High-Quality Camera (HQ) proprietary M12-mount Sony IMX477 Camera was selected for its superior image quality compared to the widely used ESP32-compatible OV2640 camera. This module, featuring an 12.3 MP sensor, supports 1080p video and high-resolution still images, connecting directly to the Raspberry Pi through the Mobile Industry Processor Interface (MIPI) Camera Serial Interface (CSI) port. It is fully compatible with the latest Raspbian OS, making it suitable for applications such as time-lapse photography, video recording, motion detection, and security. The camera's sensor has a native 12.3 MP resolution with a fixed-focus lens, capable of capturing 12.3 megapixels static images and supporting video formats at 2028 × 1080p50, 2028 × 1520p40, and 1332 × 990p120. Thus, the Pi HQ Camera Module was selected. The systems use a client-server architecture depicted in Figure 6: the video feed must go through an AI processing and cloud server to get to the user device. The latency between the surveillance camera being triggered by the PIR sensor and successful message delivery is analyzed in the client-server mode. For delay-sensitive applications, peer-to-peer architecture may be preferred.

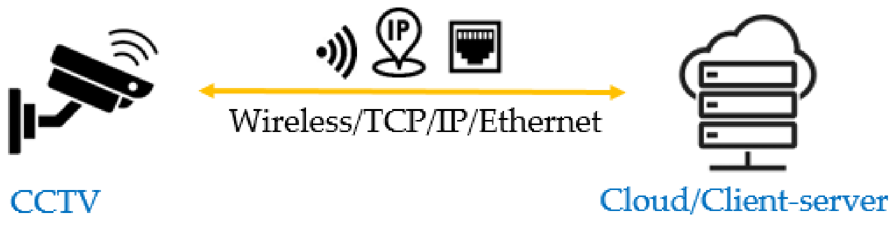


**Figure 6.** Camera in client–server architecture.

### *3.2. AI-Driven Video Processing*

The AI module is implemented on the Raspberry Pi 4B to facilitate face detection and facial recognition. The computer vision model, designed in OpenCV and running on the Raspberry Pi, comprises two distinct models—one dedicated to face detection and the other to facial recognition—operating concurrently in real-time. Given the project's requirements for continuous 24/7 operation, paying special attention to optimizing the code for efficiency ensures consistent performance on the Raspberry Pi. Due to the high computational demands of computer vision, particularly on the Raspberry Pi—a compact device with limited cooling and performance capabilities relative to a full-sized computer, the process here involved initially designing the foundational models, integrating the face detection and facial recognition modules, and finally implementing efficient coding practices to ensure reliable, continuous operation. The two AI models were integrated using a resource-efficient trigger logic, which minimized computational load while ensuring seamless functionality for the final system.

#### Face Detection

Face detection, also known as facial detection, is a computer vision task that identifies human faces in digital images and video. It is conducted by analyzing the visual input to determine whether a person's facial features are present and clearly distinguishing those faces from other objects. Face detection is the first step in face tracking, face analysis, and facial recognition. In the face detection stage, the proposed system pipeline begins with acquiring video frames from a Raspberry Pi camera. The acquired frames are then converted from the original RGB images to grayscale images for the efficiency of the detection model. The images are resized to 500 × 500 pixels for optimization due to limited processing power and consistency of scale. The pixel values are then normalized to reduce computational complexity, and the Gaussian blurring technique is applied to remove image noise for enhanced feature detection. The detection steps output is shown in Figure 7. The Haar cascade algorithm is used to extract facial features. The OpenCV built-in Haar cascade classifier is used as it is already trained on a large dataset of human faces. This algorithm is capable of processing images extremely rapidly and achieving high detection rates [61].

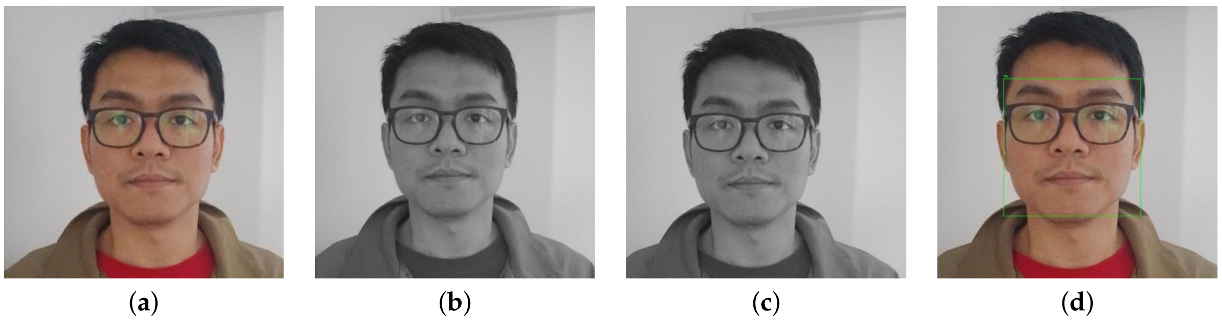


**Figure 7.** Face detetion steps with privacy preserved. (**a**) camera capture; (**b**) grayscale; (**c**) Gaussian blur; (**d**) face detection.

### *3.3. Facial Recognition*

Face recognition can be achieved by employing a deep learning library to train a model capable of identifying and distinguishing facial features. Through this training process, the model learns to recognize distinct characteristics across different faces, enabling accurate identification. Specifically, a deep learning model must be trained to produce a specific classification output for a given input. For example, the model would output an individual's name when presented with an image of their face [62].

To implement facial recognition within this project, a library was required that contained a database of facial signatures for the algorithm to compare against detected faces. However, the primary challenge was the sample size of the library. Larger libraries enhance accuracy by enabling the algorithm to better differentiate between the facial features of various individuals, thus improving the identification of correct identities. These libraries must be trained using extensive datasets of diverse images and faces to achieve the desired level of precision.

Developing and training a custom facial recognition library with such a large dataset was deemed impractical for the scope of this study. Consequently, existing Python libraries 'Dlib' and 'face_recognition', were utilized. The face_recognition library, developed by Adam Geitgey, based on a pre-trained model using a dataset of 3 million images [63] is utilized. Dlib's face recognition system primarily relies on a deep learning-based approach, using a pre-trained model to encode faces into 128-dimensional feature vectors. The face recognition pipeline implementation in this study is summarized below:

Face Detection: Dlib's (HOG) face detection model utilizes Histogram of Oriented Gradients (HoG) features combined with a Support Vector Machine (SVM) classifier. HoG is a popular method for extracting feature descriptors from images, and it is known for being efficient in terms of computational cost, particularly when run on CPUs. This method works well for detecting frontal faces and slightly rotated ones. However, it has limitations, such as reduced accuracy when detecting smaller faces or when faces are occluded. Additionally, it may miss certain facial regions, such as parts of the chin and forehead, during detection [61].
Face Landmark Detection: Dlib's facial landmark detection model uses key facial points, including the right and left eyebrows, eyes, nose, and jaw, to precisely localize and represent important regions of the face. The model utilizes a 68-point landmark system, which allows it to detect these critical facial features with high accuracy. This process plays an essential role in tasks like face alignment and recognition, contributing to robust performance across various conditions such as facial variations and lighting changes [64,65].
Face Encoding: Following face detection, dlib extracts facial features and computes a numerical representation (encoding) for each face. These encodings are used for face recognition. A face encoding is basically a way to represent the face using a set of 128 computer-generated measurements. Two different pictures of the same person would have similar encoding, and two different people would have totally different encoding.
Face Matching: The feature-based recognition is performed by comparing the distance between the input face's facial features and the registered faces. When a registered face meets the matching criteria, the face recognition returns the matching face ID found in the database. The recognition process typically involves calculating the Euclidean distance between the feature vectors of the detected face and known faces stored in a database. If the distance is below a certain threshold, the faces are considered to match. This makes it a powerful tool for both identification (finding a matching face in a database) and verification (confirming if two faces are the same).

The objective of the facial recognition system in this project was to identify two live users and 23 face images. the two live users are referred to as "TTu" and "SPh". To achieve this, a large collection of photographs for each user was initially assembled to create a database. A larger reference dataset enhances accuracy by accounting for variations in facial signatures caused by differing angles, lighting conditions, and facial expressions. During testing, it was observed that the large database significantly affected the system's performance and frame rate. Utilizing the Raspberry Pi 5, the video frame rate has been improved to 30 frames per second, and the sample size of images has been increased to 20 per subject, capturing variations in lighting conditions, facial angles, expressions, and partial occlusion (e.g., wearing glasses). The face database has been expanded to include 25 individuals by incorporating additional open-access images [66].

In this section, faces are detected, compared against the known face database, and, where applicable, the identity of the matched face is appended.

To display this information to the user within the footage, the program overlays a bounding box around the detected face and adds a label indicating the name of the identified individual which is shown in Figure 8.

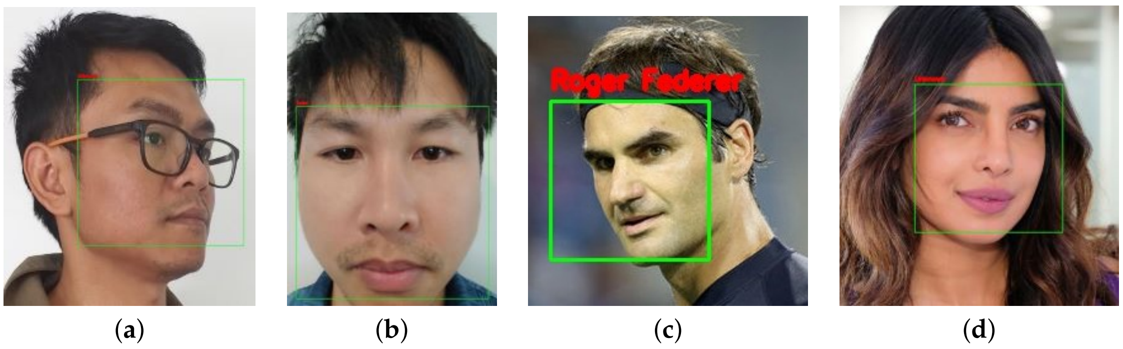


**Figure 8.** Recognized face with an identity label and unrecognized face. (**a**) user "TTu"; (**b**) user "SPh"; (**c**) "R. Federer"; (**d**) "Unknown".

### *3.4. Information Delivery*

In scenarios where a sensor is triggered, leading to the activation of the camera to record and upload footage to Google Drive, it is essential to implement a mechanism to notify the user of the event. To address this, a Python script was developed to send an email containing key details and a link to the uploaded recording on Google Drive.

While email serves as the primary notification method, it relies on the user having an active internet connection. To enhance the reliability of notifications, the Python program also integrates the *Pushbullet* API, which enables the delivery of text messages to the user's mobile device. This ensures that notifications can still be received in the absence of an internet connection, provided the user has access to a cellular network, which is generally more dependable.

Although *Pushbullet* could potentially serve as the sole notification method by including the footage link in the text message, this approach was deemed less secure compared to providing the link within an email. Consequently, the text message notification is limited to prompting the user to check their email for the link.

The initial step in the information delivery process involves uploading the footage, as illustrated in Listing 1. A dedicated Google account was created to facilitate this process, utilizing both Google Drive and Gmail. An API authentication key was obtained from this account to enable file access and uploading. JSON is used to retrieve the recorded file from Raspberry Pi's internal storage and upload it to Google Drive. Subsequently, the metadata file ID is extracted for inclusion in the email message sent via Gmail.

The second stage of this process involves sending an email alert to the user's Gmail account, as depicted in Listing 2. After retrieving the file ID associated with the Google Drive link, Python's SMTP and SSL libraries are employed to send the email and ensure message encryption, respectively.

Initially, the sender and recipient email addresses are defined as simple string variables, allowing for easy modification. Using the retrieved file ID, a unique Google Drive URL for the video is generated. The email is then composed utilizing the MIME library, with a customized message as the body and the URL included as an attachment. Finally, the email is transmitted securely via SSL, ensuring the message is encrypted during delivery.

**Listing 1.** Uploading file to Google Drive.

```
#will be used to upload recorded footage
    headers = {        "Authorization": "Bearer <redacted API key>"
          }
    para = {
        "name": "output.avi"
    }
# File data to upload
    files = {        'data': ('metadata', json.dumps(para), '
        application/json; charset=UTF-8'),
        'file': open("./output.avi", "rb")
    }
   r = requests.post(
        "https://www.googleapis.com/<redacted>",
        headers=headers,
        files=files    )
 # Parse the response for the file ID
    meta = r.json()
    file_id = meta['id']
# Return the ID of the uploaded file
    return file_id
```

**Listing 2.** Send email with file link.

```
# To send the user an email with the link of the file
def send_email(id):
    # CHANGE RECEIVER ADDRESS HERE -------
    to_addrs = 'test.email.18039875@gmail.com'

    # Sender parameters, Body of Email
    sender = 'test.email.18039875@gmail.com'
    sender_password = 'oxqzglbnzfnwktnq'
    from_addr = sender

    # Body of message
    url = 'https://drive.google.com/open?id=' + id
    body = 'Please find your link to the recorded video below, or in
        your Google Drive folder.\n\n'

    # Creating the Message, Subject line, From and To
    msg = email.mime.text.MIMEText(body + url)
    msg['Subject'] = 'MOTION ACTIVATED VIDEO RECORDING'
    msg['From'] = sender
    msg['To'] = to_addrs

    # Gmail uses SSL
    s = smtplib.SMTP_SSL(host='smtp.gmail.com', port=465)
    s.login(sender, sender_password)
    s.sendmail(sender, to_addrs, msg.as_string())
    s.quit()
```

As an alternative notification method that does not require an internet connection, the *Pushbullet* API is utilized to send text message alerts. This system is integrated with facial recognition applied to the recorded footage, allowing the alert to include the name of the individual who triggered the system, if identified. This procedure is illustrated in Listing 3.

**Listing 3.** Send notification alert.

```python
# For sending Notification alert to user's phone
def send_alert(name):
    API_KEY = "reducted"
    pb = Pushbullet(API_KEY)

    if name == "Unknown":
        msg = "Unknown person spotted on the premises. \nPlease see
            your email for recording"
    else:
        msg = name + " was spotted on the camera. \nPlease see your
            email for recording"

    push = pb.push_note("ATTENTION! Motion Activated", msg)
```

To prevent interception or spoofing of Google Drive video footage links shared via email notifications, this project implemented steps such as secure access where the link is available only to a specific individual who can access the link using a given password over HTTP transport. However, future implementation could consider other measures such as two-factor authentication, expiring URL, DomainKeys Identified Mail (DKIM), Sender Policy Framework (SPF), and Domain-based Message Authentication, Reporting, and Conformance (DMARC) to authenticate emails and prevent spoofing as well as Regularly review Google Drive's access logs to detect unauthorized access to shared files or links.

### *3.5. Risk Evaluation of Using Google Drive to Store Sensitive Data*

Using Google Drive to store sensitive data presents several risks, including privacy concerns, unauthorized access, and potential data loss. While Google implements encryption for data in transit and at rest, the fact that encryption keys are controlled by Google raises concerns about data access, including by law enforcement with a warrant. Additionally, weak passwords, phishing attacks, or cyberattacks can compromise account security, granting unauthorized users access to sensitive information. There is also a risk of data loss due to service outages, file deletion, or corruption. Furthermore, compliance with regulatory frameworks such as General Data Protection Regulation (GDPR) [67] or Health Insurance Portability and Accountability Act (HIPAA) [68] may be a concern if Google Drive does not meet specific regional legal requirements for data handling. To mitigate these risks, users should employ strong passwords, enable two-factor authentication, and consider additional encryption tools before storing sensitive data on the platform.

## 4. Security System App

Ideally, the entire notification system and video footage viewing functionality would be integrated into a dedicated application. For this purpose, an application was developed using the JavaScript programming language and the React Native framework. React Native, which is widely utilized in popular applications such as Facebook and Uber Eats, was chosen due to its cross-platform compatibility with Android, iOS, and web platforms. Additionally, React Native offers a range of libraries that facilitate development, including React Navigation, heavily utilized to implement the desired user interface and tab structure, and Expo AV, which supports in-app video playback. The development process also incorporated tools and services such as Expo and Visual Studio Code. Expo proved particularly useful during the development cycle, enabling the application to be run on an Android device via a local area network and updated in real time. This feature significantly enhanced the efficiency of interface design and testing. The application's tabbed interface was implemented using the React Navigation

Bottom Tab Navigator combined with the Native Stack Navigator. The bottom tab navigator defines the names, titles, styles, and icons for each tab, with icons dynamically changing their appearance upon selection. Additionally, a native stack was employed for the in-app video player, allowing a secondary navigation layer within an existing bottom tab. This feature enables greater customization of the video player screen, accessible through the "Locations" tab by selecting a specific location. The "Locations" tab demonstrates two distinct methods for accessing recorded footage as a proof of concept. The first method employs Linking openURL to open the user's Google Drive in an external browser. This approach leverages Google Drive's inherent security, requiring the user to sign in to access the content, ensuring a secure connection. The second method utilizes the aforementioned video stack for in-app playback. In this implementation, a button press navigates to the LiveStreamScreen stack and passes the corresponding video URL as a route parameter. By linking the route parameter to the button, multiple buttons with distinct video URLs can utilize the same LiveStreamScreen function for video playback. In a fully realized implementation, the video URL would automatically update to reflect the latest footage. However, significant security concerns associated with this approach necessitated alternative methods. Listing 4 illustrates the configuration of the video player, which accepts an input URL provided by the pressed button. The player supports full-screen mode and includes external pause and play controls, enhancing user interaction.

**Listing 4.** LiveStreamScreen component.

```
function LiveStreamScreen({navigation , route}) {
    const { videoURL } = route.params;

    const video = React.useRef(null);

    const [status, setStatus] = React.useState({});

    return (
        <View style={styles.container}>
            <Video
                ref={video}
                style={styles.video}
                source={{uri: videoURL}}
                useNativeControls
                resizeMode="contain"
                isLooping
                onPlaybackStatusUpdate={(status) => setStatus(() =>
                    status)}
            />
            <View style={styles.buttons}>
                <Button
                    title={status.isPlaying ? 'Pause' : 'Play'}
                    onPress={() =>
                        status.isPlaying
                        ? video.current.pauseAsync()
                        : video.current.playAsync()
                    }
                />
            </View>
        </View>
    );
}
```

## 5. Results

As outlined earlier, the hardware configuration of the developed system comprises a Raspberry Pi equipped with a camera and a passive infrared (PIR) sensor. The camera remains in a low-power state until the PIR sensor detects motion, at which point the camera is activated to begin recording. The recorded footage is processed using OpenCV to perform facial recognition and object detection. Subsequently, the footage is uploaded to the user's Google Drive database. Simultaneously, the user is notified via email, which includes a link to the uploaded footage, and a text message alerting them of the detected disturbance and prompting them to check their email.

Additionally, a proof-of-concept application has been developed to serve as an interface for user interaction with the security system, allowing the user to receive notifications and access recorded footage.

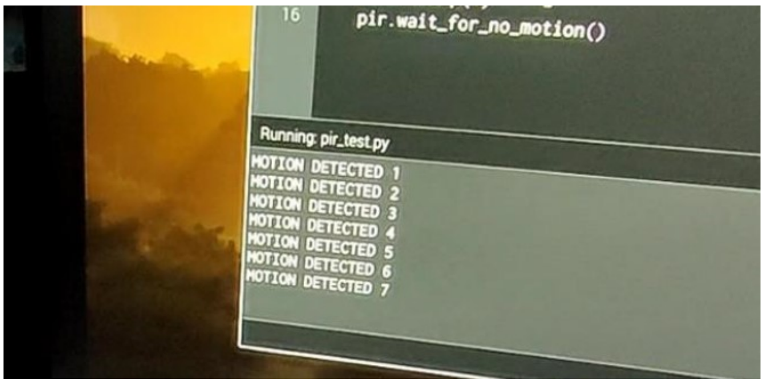
### *5.1. Hardware*

The proposed system is built around the Raspberry Pi 5 with 8 GB of LPDDR4X RAM, running the latest Raspbian OS Bookworm to ensure optimal performance and software compatibility. The system utilizes an M12-mount Sony IMX477 camera module, equipped with a wide-angle lens to achieve an extended field of view. This camera, with its 12.3-megapixel resolution, is capable of capturing high-quality images, making it suitable for tasks such as object detection, facial recognition, and environmental monitoring. Two HC-SR501 Passive Infrared (PIR) sensors are integrated to enhance motion detection capabilities. Each sensor has a horizontal detection angle of approximately 120° and an 8m range. They are strategically positioned to provide overlapping coverage, ensuring comprehensive motion detection in the monitored area. The PIR sensors are connected to the Raspberry Pi's GPIO pins and are used to trigger the camera for event-driven image capture, reducing redundant data collection. For data storage, the system is equipped with a 64 GB SSD card, which provides sufficient capacity for storing high-resolution images and processing logs. The storage solution ensures reliable and fast read/write speeds, essential for handling large image datasets and real-time computational tasks. The Raspberry Pi 5's powerful 2.4 GHz quad-core ARM Cortex-A76 processor and 8GB of RAM enable efficient real-time image processing and machine learning tasks, such as motion analysis and object detection. The camera module is connected via the MIPI CSI-2 interface, ensuring seamless data transfer for high-speed image capture. Power is supplied through a 5 V/5 A USB-C adapter, ensuring stable operation even under high processing loads. This hardware configuration provides a robust platform for applications in environmental monitoring, security, and smart systems research, leveraging the Raspberry Pi 5's performance and the versatility of the integrated sensors and camera.

From a hardware perspective, the primary components subject to testing are the functionality of the camera and the PIR sensor module.

During initial testing, a simple program was implemented to increment a counter each time the PIR sensor was triggered. The sensor was configured in single-trigger mode, where its output remains low until motion is detected, at which point it transitions to a high state for a predefined duration before returning to low, irrespective of continued motion detection.

To evaluate this behavior, a hand was waved back and forth over the sensor, allowing brief pauses between motions to ensure the sensor output returned to the low state. This process was repeated seven times, and the counter correctly incremented with each trigger event, confirming the sensor's proper operation, as illustrated in Figure 9.



```
    pir.wait_for_no_motion()

Running: pir_test.py
MOTION DETECTED 1
MOTION DETECTED 2
MOTION DETECTED 3
MOTION DETECTED 4
MOTION DETECTED 5
MOTION DETECTED 6
MOTION DETECTED 7
```

**Figure 9.** PIR sensor detection test outcome.

For the purposes of this project, the specific mode of the PIR sensor is not critical, as the program is designed to respond to any high signal by initiating recording for a fixed duration, regardless of whether the sensor continues to detect motion. If the sensor remains triggered after the recording period ends, the process will simply repeat.

In scenarios where sensor reliability is further explored, the repeat-trigger mode—where the output remains high as long as motion is detected—could be utilized. This configuration would allow the camera to record continuously but only while motion is actively sensed.

To verify functionality, the sensor was deliberately triggered, successfully initiating the recording and uploading process as expected.

### *5.2. Experimental Procedure*

The face recognition tests were conducted by displaying images on a mobile device and introducing them into the PIR sensor's field of view to activate the camera. The device was held stationary to enable the camera to capture the image and the processor to complete the face recognition process. Upon successful recognition, a text or email alert was generated. This procedure was repeated for all test image samples to ensure a comprehensive evaluation of the system.

### *5.3. AI Processing*

The implemented object detection and facial recognition systems are demonstrated in Figure 10. The facial recognition algorithm identifies individuals by drawing a rectangle around the detected face and labeling it with the corresponding identity.

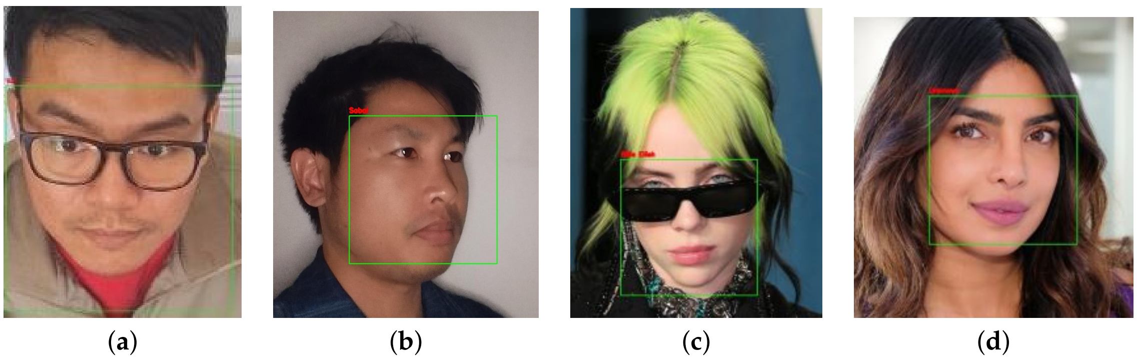


**Figure 10.** Recognized face with an identity label and unrecognized face. (**a**) user "TTu"; (**b**) user "SPh"; (**c**) "B. Elish"; (**d**) "Unknown".

### *5.4. Quantitative Analysis of the Face Recognition System*

To evaluate the performance of a face recognition system, the following metrics and analysis techniques are employed: where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative.

Accuracy: A quantity which measures the proportion of correctly recognized faces out of the total number of test cases calculated as:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2}$$

Precision: The proportion of correctly identified faces (True Positives) out of all instances identified as positive (TP + FP) given as:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3}$$

Recall: Indication of the system's ability to detect all relevant faces given as:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4}$$

F1 Score: A harmonic mean of precision and recall, balancing both metrics given as:

$$F1Score = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{5}$$

False Positive Rate (FPR): The proportion of False Positives among all actual negatives calculated as:

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{6}$$

Sample results and computed metrics according to Equations (2)–(6) are presented in Table 3 and Figure 11, indicating a highly accurate and reliable face recognition system with minimal false positives and strong overall performance.

**Table 3.** Face recognition results.

| Metric | Value |
|---|---|
| True Positives ($TP$) | 236 |
| True Negatives ($TN$) | 30 |
| False Positives ($FP$) | 2 |
| False Negatives ($FN$) | 9 |

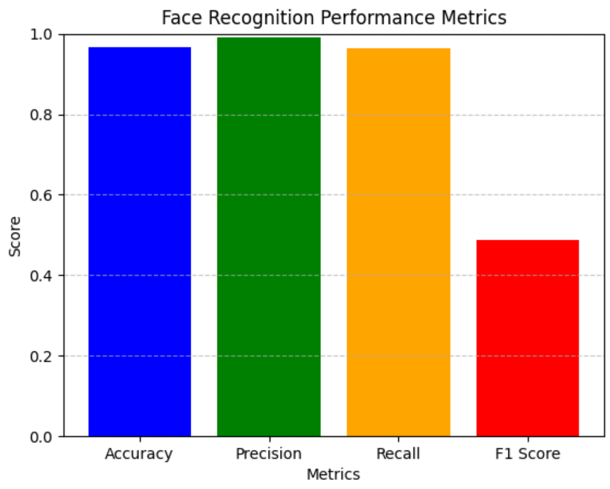
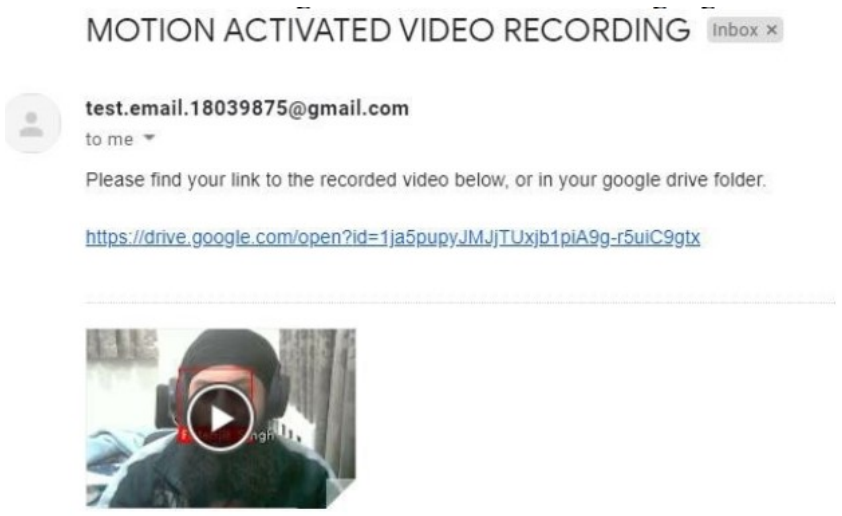


**Figure 11.** Calculated metrics.

### 5.5. Notification and Information Delivery

Once the footage is recorded and processed, it is uploaded to the user's Google Drive in the .avi format. This format was selected due to its superior compressibility compared to .mp4, resulting in smaller file sizes and reduced upload and download times.

Following the upload, an email notification is sent to the user. By default, this email is directed to the Gmail account linked with Google Drive, but it can be easily reconfigured to any preferred email address. An example email, as shown in Figure 12, contains a link to the uploaded footage on Google Drive.


MOTION ACTIVATED VIDEO RECORDING

test.email.18039875@gmail.com

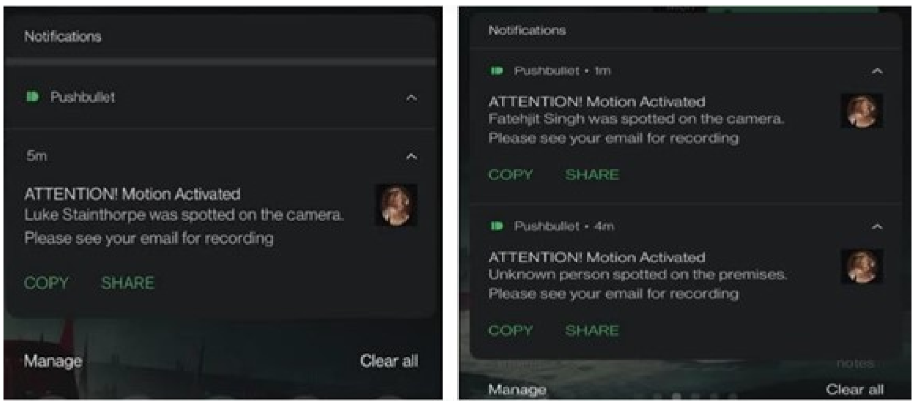
to me

Please find your link to the recorded video below, or in your google drive folder.

https://drive.google.com/open?id=1ja5pupyJMJjTUxjb1piA9g-r5uiC9gtx

**Figure 12.** Example received email with attached link.

Additionally, the user receives a *Pushbullet* notification on their mobile device, alerting them that the security system has been triggered. When facial recognition identifies individuals, the notification includes the detected person's identity, where applicable. Example notifications for two recognized individuals and an unrecognized person are displayed in Figure 13.


Notifications

Pushbullet

5m

ATTENTION! Motion Activated
Luke Stainthorpe was spotted on the camera.
Please see your email for recording

Notifications

Pushbullet • 1m

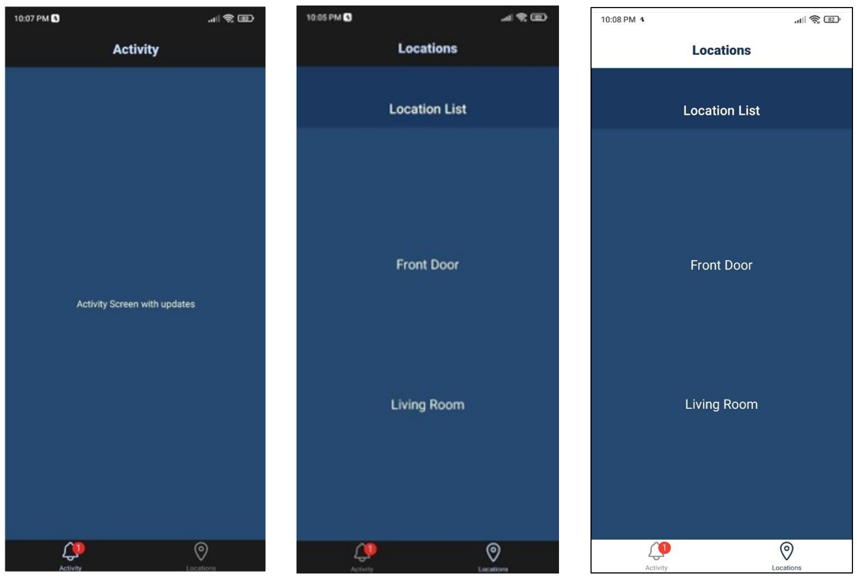
ATTENTION! Motion Activated
Fatehjit Singh was spotted on the camera.
Please see your email for recording

Pushbullet • 4m

ATTENTION! Motion Activated
Unknown person spotted on the premises.
Please see your email for recording

**Figure 13.** Pushbullet notifications for recognized and unrecognized persons.

The last information delivery method created was the proof-of-concept application, developed using the React Native framework, which provides another method for users to interact with the security system, access notifications, and view recorded footage. These notifications would allow users to play the associated footage upon selection. Figure 14a shows the activity tab, which, in a production version, would maintain a log of notifications.

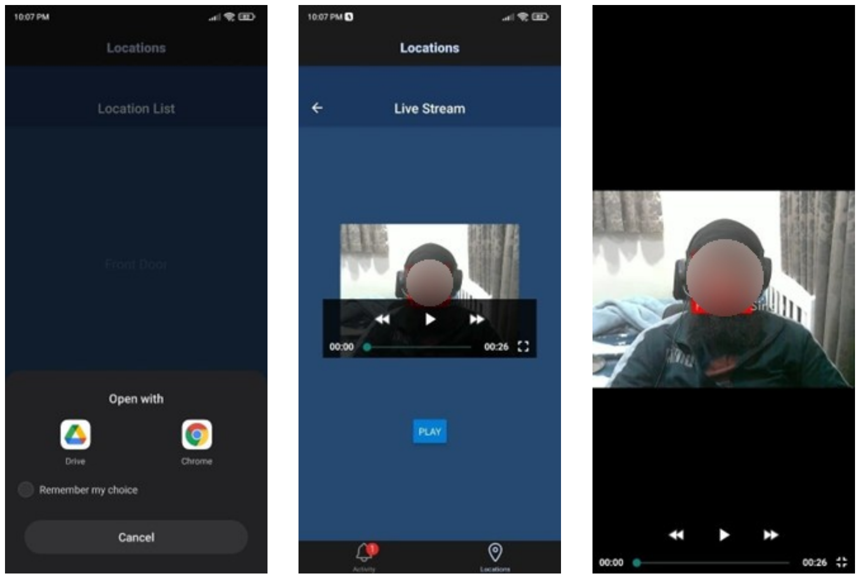


(**a**) (**b**) (**c**)

**Figure 14.** Activity tabs. (**a**) App Activity Screen; (**b**) App Locations Screen; (**c**) Light-Mode screen.

Figure 14b highlights the Locations Tab, which contains buttons for accessing footage from different areas, such as "Front Door" and "Living Room". In an expanded system, these buttons would correspond to additional Raspberry Pi security modules. The interface includes features such as bottom tab icons that change color when selected and support for light and dark mode settings, automatically adjusting header and tab colors, as illustrated in Figure 14b,c.

When the "Front Door" button in the Locations Tab is pressed, the corresponding Google Drive link is opened externally in a browser or the Google Drive app, as shown in Figure 15a. In contrast, pressing the "Living Room" button passes the associated URL to the Live Stream stack screen, where the embedded video player streams the footage, as demonstrated in Figure 15b.



(**a**) (**b**) (**c**)

**Figure 15.** Link and video play buttons. (**a**) Open link button; (**b**) video player button; (**c**) video player fullscreen.

The embedded player in Figure 15c features an external blue pause/play button, which interacts with the player to control playback, dynamically updating its label based on the playback state.

Finally, Figure 15c showcases the player's full-screen functionality, which users can toggle on and off using the icon located at the bottom-right corner of the player.

## 6. Discussion

One concern associated with the use of the PIR sensor in the project's security system is the potential for blind spots in the sensor's field of view when it overlaps with the camera's field of view. This issue is mitigated by incorporating two overlapping PIR sensors, with the monitored area divided into two distinct zones, each assigned to a specific sensor. A centralized control algorithm, implemented on the Raspberry Pi 5, integrates the sensor signals to ensure efficient motion detection. The algorithm dynamically activates or deactivates specific sensors based on the detected activity within the monitored zones. In cases where the sensor's field of view is insufficient, additional PIR sensors could be incorporated with minimal adjustments to the Python software, taking advantage of the Raspberry Pi 5's ample input pins. Despite these concerns, the PIR sensor remains the most suitable option for the security modules in this project, and the system would benefit from the inclusion of additional sensors of varying types to enhance its coverage.

When using artificial intelligence for footage processing, a problem arose with the frame rate of the incoming video. This limitation is mitigated using a more capable processing platform, i.e., Raspberry Pi 5.

The object detection system implemented in this project offers several potential applications for security and smart home systems. In the context of security, the detected location of a moving person can be used to trigger appropriate responses. For example, if an intruder is detected moving toward another zone covered by a separate camera, that camera can be preemptively activated to avoid gaps in footage. In larger areas, the camera system could be mounted on an electric motor that pans to track the intruder's location, keeping them centered in the frame.

Additionally, object detection could be adapted for use in child monitoring systems. In such systems, an emergency response could be triggered if a child approaches a predefined "off-limits" area, such as near electrical outlets or wires.

A proof-of-concept app was developed; however, the goal of completing the app to fully facilitate user interaction with the security system was not achieved. This was primarily due to security concerns regarding video hosting.

The first video playback method, shown in Figure 15c, is secure because it relies on Google Drive's built-in security, requiring the user to sign in to access the video. While this is functional, it is not an ideal solution, as it still requires video playback outside of the app. The ideal approach would involve a built-in video player.

A security issue emerged when using the embedded video player, as the video URL passed to the player must be of .mp4 format. Consequently, the Google Drive links used in the email notification system and the previous video playback method do not work with this player. Although the links can be converted to permanent .mp4 links, doing so bypasses the Google Drive sign-in stage, thereby allowing anyone with the link to access the files, regardless of authentication.

This issue could be resolved if the file hosting service used supports built-in security protocols that require user authentication to access the files. Unfortunately, due to time constraints and limited access to resources, a solution to this problem could not be implemented.

## 7. Conclusions and Future Work

### *7.1. Conclusions*

The project aim of creating a Smart Security System that can detect an intruder and respond intelligently has been achieved with room for future development. As intended, the user is able to access the system footage and information in a secure way using free Google services, and a concept app to repackage this information in a way that promotes ease of use has been created.

### *7.2. Future Work*

As part of the ongoing development of this project, there are several potential avenues for further enhancement. From a hardware perspective, the current project utilizes existing off-the-shelf products, which increases both the cost and physical size of the system. A possible improvement would be to design and fabricate a custom PCB with a microcontroller, which could significantly reduce the product's size by eliminating unnecessary features not required for a security system. This approach would also facilitate the creation of custom housing tailored to the needs of a discrete security system. Additionally, a custom-built product would likely have lower energy consumption, making the inclusion of a battery more feasible, at least as a backup in the event of a power supply disruption. A key constraint in the current project was the desire to perform AI-based video processing locally on the security modules, without relying on an external server. This approach necessitated relatively high processing power to run OpenCV directly on the security modules. An alternative solution could have involved offloading the video processing to an external server. This would enable the security modules to be smaller and less complex, only requiring them to detect an event, record it, and stream the footage to the server. The server could then handle the OpenCV processing, allowing for more powerful video analysis based on its computational capabilities. However, this solution would entail additional hardware requirements for both the security modules and the server, increasing installation and operational costs. Another area of potential improvement is the integration of smaller wireless sensor modules. These modules, which would not use a camera, could send time and location-based alerts upon detection of events. Combining these external sensors with the existing security modules could expand the system's coverage area without the high costs and processing demands of the camera-based system. This would allow for more comprehensive monitoring, with cameras placed in key areas while the wireless sensors cover additional locations. There are also several features that could enhance the user experience and system cohesion within the app. A notification system would allow users to access a log of historical security events in one place. Ideally, these notifications would contain valuable information, such as the date and time of the event, the location where the event occurred, and a snapshot from the camera at the time of the incident. Another possible direction for future development involves exploring the use of Cloud hosting services or personal servers, but one alternative worth considering is the LoRa network. The LoRa network enables long-range communication, allowing data to be transmitted over distances of 5 to 15 km, depending on the environment, without requiring an internet connection. This would be particularly beneficial for users in rural areas, such as farmers, who are more likely to achieve the higher end of this range. Since these users typically work within relatively close proximity to their homes, they would likely remain within the network's range and be able to access their security system without relying on internet connectivity.

**Funding:** This research received no external funding.

**Data Availability Statement:** Included within the paper as well as in [66].

**Acknowledgments:** I sincerely thank my students Thit Tun, Sabai Phuchortham, Luke Stainthorpe, and Fatehjit Singh for their participation in this research project and Auckland University of Technology for the materials used for experiments.

**Conflicts of Interest:** The author declares no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| IoT | Internet of Things |
| AI | Artificial Intelligence |
| PIR | Passive Infrared |

## References

1. Touqeer, H.; Zaman, S.; Amin, R.; Hussain, M.; Al-Turjman, F.; Bilal, M. Smart Home Security: Challenges, Issues and Solutions at Different IoT Layers. *J. Supercomput.* **2021**, *77*, 14053–14089. [CrossRef]
2. Jose, A.C.; Malekian, R.; Ye, N. Improving Home Automation Security: Integrating Device Fingerprinting into Smart Home. *IEEE Access* **2016**, *4*, 5776–5787. [CrossRef]
3. Majumder, A.J.; Izaguirre, J.A. A Smart IoT Security System for Smart-Home Using Motion Detection and Facial Recognition. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 1065–1071. [CrossRef]
4. OpenCV. About OpenCV. Available online: https://opencv.org/about (accessed on 27 November 2024).
5. Prathaban, T.; Thean, W.; Sobirin Mohd Sazali, M. A Vision-Based Home Security System Using OpenCV on Raspberry Pi 3. *AIP Conf. Proc.* **2019**, *2173*, 020013. [CrossRef]
6. Haque, S.; Sarker, B.; Mawla, A.R.; Reza, M.N.; Riya, A.S. Development of a Face Recognition Door Lock System with OpenCV Integration for Securing Residential Properties. In Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 24–28 June 2024; pp. 1–4. [CrossRef]
7. Deshmukh, D.; Nakrani, G.; Bhuyar, M.L.; Shinde, U.B. Face Recognition Using OpenCV Based on IoT for Smart Door. In Proceedings of the International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Jaipur, India, 26–28 February 2019. Available online: https://ssrn.com/abstract=3356332 (accessed on 27 November 2024).
8. Hasan, T.H.; Sallow, A.B. Face Detection and Recognition Using OpenCV. *J. Soft Comput. Data Min.* **2021**, *2*, 86–97. Available online: https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/8791 (accessed on 27 November 2024).
9. Tawsik Jawad, K.M.; Rashid, M.B.; Sakib, N. Targeted Face Recognition and Alarm Generation for Security Surveillance Using Single Shot Multibox Detector (SSD). *Int. J. Comput. Appl.* **2019**, *177*, 8–13. Available online: https://ijcaonline.org/archives/volume177/number22/31027-2019919652/ (accessed on 27 November 2024).
10. Vignesh Baalaji, S.; Sandhya, S.; Sajidha, S.A.; Nisha, V.M.; Vimalapriya, M.D.; Tyagi, A.K. Autonomous Face Mask Detection Using Single Shot Multibox Detector, and ResNet-50 with Identity Retrieval through Face Matching Using Deep Siamese Neural Network. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 11195–11205. [CrossRef] [PubMed]
11. Han, H. A Novel Single Shot-Multibox Detector Based on Multiple Gaussian Mixture Model for Urban Fire Smoke Detection. *Comput. Sci. Inf. Syst.* **2023**, *20*, 1819–1843. [CrossRef]
12. Zhang, L.; Gui, G.; Khattak, A.M.; Wang, M.; Gao, W.; Jia, J. Multi-Task Cascaded Convolutional Networks Based Intelligent Fruit Detection for Designing Automated Robot. *IEEE Access* **2019**, *7*, 56028–56038. [CrossRef]
13. Zou, X.; Zhou, L.; Li, K.; Ouyang, A.; Chen, C. Multi-Task Cascade Deep Convolutional Neural Networks for Large-Scale Commodity Recognition. *Neural Comput. Appl.* **2020**, *32*, 5633–5647. [CrossRef]
14. Zhuang, N.; Yan, Y.; Chen, S.; Wang, H. Multi-Task Learning of Cascaded CNN for Facial Attribute Classification. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 2069–2074. [CrossRef]
15. Zhang, H.; Wang, X.; Zhu, J.; Kuo, C.-C.J. Fast Face Detection on Mobile Devices by Leveraging Global and Local Facial Characteristics. *Signal Process. Image Commun.* **2019**, *78*, 1–8. [CrossRef]
16. Zhang, C.; Xu, X.; Tu, D. Face Detection Using Improved Faster RCNN. *arXiv* **2018**, arXiv:1802.02142. [CrossRef]
17. Liu, X.; Zhang, S.; Hu, J.; Mao, P. ResRetinaFace: An Efficient Face Detection Network Based on RetinaFace and Residual Structure. *J. Electron. Imaging* **2024**, *33*, 043012. [CrossRef]
18. Wibowo, M.E.; Ashari, A.; Subiantoro, A.; Wahyono, W. Human Face Detection and Tracking Using RetinaFace Network for Surveillance Systems. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 13–16 October 2021; pp. 1–5. [CrossRef]

19. Hu, J.; Hou, J.; Chen, Y.; Li, W.H.; Shi, D.H.; Yi, J.; Huang, X.L. Rapid Face Detection in Complex Environments Based on the Improved RetinaFace. In *Proceedings of the ACM Conference*; Association for Computing Machinery: New York, NY, USA, 2023; ISBN 9781450397933. [CrossRef]
20. Thottempudi, P. Face Detection and Recognition Through Live Stream. In *Sustainable Science and Intelligent Technologies for Societal Development*; Mishra, B., Ed.; IGI Global Scientific Publishing: Hershey, PA, USA, 2023; pp. 167–177. [CrossRef]
21. Chauhan, R.; Dhyani, I.; Vaidya, H. A Review on Human Pose Estimation Using Mediapipe. In Proceedings of the 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT), Dehradun, India, 8–9 September 2023; pp. 1–6. [CrossRef]
22. Dewi, C.; Manongga, D.; Hendry; Mailoa, E.; Hartomo, K.D. Deep Learning and YOLOv8 Utilized in an Accurate Face Mask Detection System. *Big Data Cogn. Comput.* **2024**, *8*, 9. [CrossRef]
23. Vemulapalli, N.S.; Paladugula, P.; Prabhat, G.S.; Abhishek, S.; Anjali, T. Face Detection with Landmark Using YOLOv8. In Proceedings of the 3rd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), Patna, India, 21–22 December 2023; pp. 1–5. [CrossRef]
24. Sohan, M.; Sai Ram, T.; Reddy, C.V.R. A Review on YOLOv8 and Its Advancements. In *Data Intelligence and Cognitive Informatics*; Jacob, I.J., Piramuthu, S., Falkowski-Gilski, P., Eds.; Springer: Singapore, 2024; pp. 1–10. [CrossRef]
25. Vasantha, S.V.; Kiranmai, B.; Hussain, M.A.; Hashmi, S.S.; Nelson, L.; Hariharan, S. Face and Object Detection Algorithms for People Counting Applications. In Proceedings of the 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 11–13 December 2023; pp. 1188–1193. [CrossRef]
26. Ansy, S.N.; Bilal, E.A.; Neethu, M.S. Emotion Recognition Through Facial Expressions from Images Using Deep Learning Techniques. In *Data Science and Applications*; Nanda, S.J., Yadav, R.P., Gandomi, A.H., Saraswat, M., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2024; Volume 819, pp. 25–36. [CrossRef]
27. Li, S.Z. Face Detection. In *Handbook of Face Recognition*; Springer: New York, NY, USA, 2005; pp. 47–84. [CrossRef]
28. Ghadekar, P.; Pradhan, M.R.; Swain, D.; Acharya, B. EmoSecure: Enhancing Smart Home Security With FisherFace Emotion Recognition and Biometric Access Control. *IEEE Access* **2024**, *12*, 93133–93144. [CrossRef]
29. Ho, H.T.; Nguyen, L.V.; Le, T.H.T.; Lee, O.J. Face Detection Using Eigenfaces: A Comprehensive Review. *IEEE Access* **2024**, *12*, 118406–118426. [CrossRef]
30. Cardona-Pineda, D.S.; Ceballos-Arias, J.C.; Torres-Marulanda, J.E.; Mejia-Muñoz, M.A.; Boada, A. Face Recognition—Eigenfaces. In *Handbook on Decision Making: Volume 3: Trends and Challenges in Intelligent Decision Support Systems*; Zapata-Cortes, J.A., Sánchez-Ramírez, C., Alor-Hernández, G., García-Alcaraz, J.L., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 373–397. [CrossRef]
31. Pienaar, J.P.; Fisher, R.M.; Hancke, G.P. Smartphone: The Key to Your Connected Smart Home. In Proceedings of the 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, UK, 22–24 July 2015; pp. 999–1004. [CrossRef]
32. Ahonen, T.; Hadid, A.; Pietikäinen, M. Face Recognition with Local Binary Patterns. In *Computer Vision—ECCV 2004*; Pajdla, T., Matas, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3021, pp. 469–481. [CrossRef]
33. Pietikäinen, M. Local Binary Patterns. *Scholarpedia* **2010**, *5*, 9775. [CrossRef]
34. Ahonen, T.; Matas, J.; He, C.; Pietikäinen, M. Rotation Invariant Image Description with Local Binary Pattern Histogram Fourier Features. In *Image Analysis*; Salberg, A.B., Hardeberg, J.Y., Jenssen, R., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5575, pp. 62–71. [CrossRef]
35. Pietikäinen, M.; Hadid, A.; Zhao, G.; Ahonen, T. Local Binary Patterns for Still Images. In *Computer Vision Using Local Binary Patterns*; Computational Imaging and Vision; Springer: London, UK, 2011; Volume 40, pp. 13–30. [CrossRef]
36. Cai, C. Utilizing Multi-Task Cascaded Convolutional Networks and ResNet-50 for Face Identification Tasks. Master's Thesis, University of Northern British Columbia (UNBC), Prince George, BC, Canada, 2021. Available online: https://unbc.arcabc.ca/islandora/object/unbc%3A59248 (accessed on 27 November 2024).
37. Sampaio, E.V.B.; Lévêque, L.; Perreira da Silva, M.; Le Callet, P. Are Facial Expression Recognition Algorithms Reliable in the Context of Interactive Media? A New Metric to Analyse Their Performance. In Proceedings of the EmotionIMX: Considering Emotions in Multimedia Experience (ACM IMX 2022 Workshop), Aveiro, Portugal, 22 June 2022. Available online: https://hal.archives-ouvertes.fr/hal-03789571 (accessed on 27 November 2024).
38. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 815–823. [CrossRef]
39. William, I.; Ignatius Moses Setiadi, D.R.; Rachmawanto, E.H.; Santoso, H.A.; Sari, C.A. Face Recognition Using FaceNet (Survey, Performance Test, and Comparison). In Proceedings of the 2019 Fourth International Conference on Informatics and Computing (ICIC), Semarang, Indonesia, 16–17 October 2019; pp. 1–6. [CrossRef]

40. He, M.; Zhang, J.; Shan, S.; Kan, M.; Chen, X. Deformable Face Net for Pose Invariant Face Recognition. *Pattern Recognit.* **2020**, *100*, 107113. [CrossRef]
41. Cahyono, F.; Wirawan, W.; Rachmadi, R.F. Face Recognition System Using FaceNet Algorithm for Employee Presence. In Proceedings of the 2020 4th International Conference on Vocational Education and Training (ICOVET), Malang, Indonesia, 19 September 2020; pp. 57–62. [CrossRef]
42. King, D.E. Dlib-ml: A Machine Learning Toolkit. *J. Mach. Learn. Res.* **2009**, *10*, 1755–1758. Available online: http://jmlr.org/papers/v10/king09a.html (accessed on 27 November 2024).
43. Boyko, N.; Basystiuk, O.; Shakhovska, N. Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; pp. 478–482. [CrossRef]
44. Xu, M.; Chen, D.; Zhou, G. Real-Time Face Recognition Based on Dlib. In *Innovative Computing*; Springer: Singapore, 2020; p. 177. [CrossRef]
45. Suwarno, S.; Kevin, K. Analysis of Face Recognition Algorithm: Dlib and OpenCV. *J. Inform. Technol. Eng. Sci.* **2020**, *4*, 1. [CrossRef]
46. Mohanty, S.; Hegde, S. V.; Prasad, S.; Manikandan, J. Design of Real-time Drowsiness Detection System using Dlib. In Proceedings of the 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Bangalore, India, 15–16 November 2019; Volume 1–4. [CrossRef]
47. Murat, K.; Topyła, D.; Zdulski, K.; Marzęcki, M.; Bieniasz, J.; Paczesny, D.; Szczypiorski, K. Security Analysis of Low-Budget IoT Smart Home Appliances Embedded Software and Connectivity. *Electronics* **2024**, *13*, 2371. [CrossRef]
48. Kim, S.; Bang, J.; Shon, T. Forensic Analysis for Cybersecurity of Smart Home Environments with Smart Wallpads. *Electronics* **2024**, *13*, 2827. [CrossRef]
49. Shah, K.; Jadav, N.K.; Tanwar, S.; Singh, A.; Ples, C.; Alqahtani, F.; Tolba, A. AI and Blockchain-Assisted Secure Data-Exchange Framework for Smart Home Systems. *Mathematics* **2023**, *11*, 4062. [CrossRef]
50. Wang, Z.; Liu, X.; Shao, X.; Alghamdi, A.; Alrizq, M.; Munir, M.S.; Biswas, S. An Optimized and Scalable Blockchain-Based Distributed Learning Platform for Consumer IoT. *Mathematics* **2023**, *11*, 4844. [CrossRef]
51. Rahim, A.; Zhong, Y.; Ahmad, T.; Ahmad, S.; Pławiak, P.; Hammad, M. Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models. *Sensors* **2023**, *23*, 6979. [CrossRef] [PubMed]
52. Asghar, I.; Khan, M.A.; Ahmad, T.; Ullah, S.; Mansoor ul Hassan, K.; Buriro, A. Fortifying Smart Home Security: A Robust and Efficient User-Authentication Scheme to Counter Node Capture Attacks. *Sensors* **2023**, *23*, 7268. [CrossRef]
53. Ashibani, Y.; Kauling, D.; Mahmoud, Q.H. Design and Implementation of a Contextual-Based Continuous Authentication Framework for Smart Homes. *Appl. Syst. Innov.* **2019**, *2*, 4. [CrossRef]
54. Wang, J.; Amos, B.; Das, A.; Pillai, P.; Sadeh, N.; Satyanarayanan, M. Enabling Live Video Analytics with a Scalable and Privacy-Aware Framework. *ACM Trans. Multimed. Comput. Commun. Appl.* **2018**, *14*, 64. [CrossRef]
55. Cavailaro, A. Privacy in Video Surveillance [In the Spotlight]. *IEEE Signal Process. Mag.* **2007**, *24*, 166–168. [CrossRef]
56. Malm, S.; Rönnbäck, V.; Håkansson, A.; Le, M.; Wojtulewicz, K.; Carlsson, N. RAD: Realistic Anonymization of Images Using Stable Diffusion. In Proceedings of the 23rd Workshop on Privacy in the Electronic Society (WPES '24), New York, NY, USA, 14–18 October 2024; pp. 193–211. [CrossRef]
57. Kim, T.; Yang, J. Selective Feature Anonymization for Privacy-Preserving Image Data Publishing. *Electronics* **2020**, *9*, 874. [CrossRef]
58. Karie, N.M.; Sahri, N.M.; Yang, W.; Valli, C.; Kebande, V.R. A Review of Security Standards and Frameworks for IoT-Based Smart Environments. *IEEE Access* **2021**, *9*, 121975–121995. [CrossRef]
59. Greengard, S. Internet of Things. *Encyclopedia Britannica*, 1 October 2024. Available online: https://www.britannica.com/science/Internet-of-Things (accessed on 31 October 2024).
60. Sudakov, O.; Malenko, A. Realistic Mathematical Model of Passive Infrared Sensor's Signal. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; pp. 757–760. [CrossRef]
61. Viola, P.; Jones, M. Rapid Object Detection Using a Boosted Cascade of Simple Features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, HI, USA, 8–14 December 2001; Volume I, p. I. [CrossRef]
62. Thakurdesai, N.; Raut, N.; Tripathi, A. Face Recognition Using One-Shot Learning. *Int. J. Comput. Appl.* **2018**, *182*, 35–39. [CrossRef]
63. Ageitgey, A. Face_recognition. Available online: https://github.com/ageitgey/face_recognition?tab=MIT-1-ov-file (accessed on 19 November 2024).
64. Kazemi, V.; Sullivan, J. One Millisecond Face Alignment with an Ensemble of Regression Trees. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1867–1874. [CrossRef]

65. Sagonas, C.; Antonakos, E.; Tzimiropoulos, G.; Zafeiriou, S.; Pantic, M. 300 Faces In-The-Wild Challenge: Database and Results. *Image Vis. Comput.* **2016**, *47*, 3–18. [CrossRef]
66. Li, J. LFW Dataset. Available online: https://www.kaggle.com/datasets/jessicali9530/lfw-dataset (accessed on 24 January 2025).
67. Ryngaert, C.; Taylor, M. The GDPR as Global Data Protection Regulation? *AJIL Unbound* **2020**, *114*, 5–9. [CrossRef]
68. Edemekong, P.F.; Annamaraju, P.; Afzal, M.; Hydel, M.J. Health Insurance Portability and Accountability Act (HIPAA) Compliance. In *StatPearls [Internet]*; StatPearls Publishing: Treasure Island, FL, USA, 2025. Available online: https://www.ncbi.nlm.nih.gov/books/NBK500019/ (accessed on 24 November 2024).