ECE 6380-AI Hardware Project Report

Sentinel-AI Voice & Motion Activated Multimodal Edge Security System

Instructor: Professor Mircea R. Stan

Group Name : MST_3
Team Members:

Mughesh Kumar NR
Md Nasim Afroj Taj
Uttam Kumar Saha

Report Content

1. Introduction:

Sentinel-AI is an edge-based security system that uses both visual and audio inputs to monitor an environment in an intelligent and privacy conscious way. The system runs entirely on local hardware and performs real time processing without relying on cloud services. By combining face detection, face recognition, and event-based activation, Sentinel-AI demonstrates how lightweight AI models can be deployed effectively on resource-constrained edge devices such as the NVIDIA Jetson Nano. The project focuses on building a functional prototype that highlights the feasibility of multimodal edge intelligence for security applications. are important.

2. Motivation:

Surveillance systems are commonly used for security in homes, offices, and other indoor spaces, but many existing solutions are not very efficient. Traditional CCTV systems record video continuously, even when nothing important happens. This leads to unnecessary power usage and large amounts of stored video data that are rarely useful. Managing and reviewing this data can become difficult over time, especially for systems that are expected to run for long periods.

Many newer surveillance systems rely on cloud services to process and store data. While this allows more advanced features, it also creates new problems. These systems depend heavily on a stable internet connection, and delays in network communication can affect real-time response. In addition, sending video and audio data to external servers raises privacy concerns, particularly in private spaces such as homes, labs, or offices where sensitive information may be present.

Another issue with many existing systems is that they often depend on only one type of sensor. Vision-only systems can produce false alerts due to lighting changes or background movement, while audio-only systems may miss visual confirmation of events. This lack of context can reduce the reliability of security monitoring.

These limitations motivated the design of Sentinel AI as an event driven system that runs entirely on local hardware. By processing data at the edge and activating only when relevant events are detected, the system aims to reduce power consumption, limit unnecessary data storage, and improve privacy. It will use both audio and visual inputs and also helps provide better results for decision making, making the system more practical for real-world indoor security applications.

3. Key Design Principles:

The design of Sentinel AI is guided by a focus on efficiency, privacy, and practical edge deployment. Instead of operating continuously, the system follows an event-driven approach, activating only when relevant audio or visual signals are detected, which significantly reduces power consumption and unnecessary data processing. All computation is performed locally on the edge device, eliminating the need for cloud connectivity and ensuring that sensitive audio and video data remain private. To improve reliability and contextual understanding, Sentinel-AI combines both audio and visual inputs rather than relying on a single sensing modality. Additionally, the system is designed to run on low-power edge hardware such as the NVIDIA Jetson Nano, emphasizing lightweight models and efficient processing suitable for real-time operation in resource constrained environments.

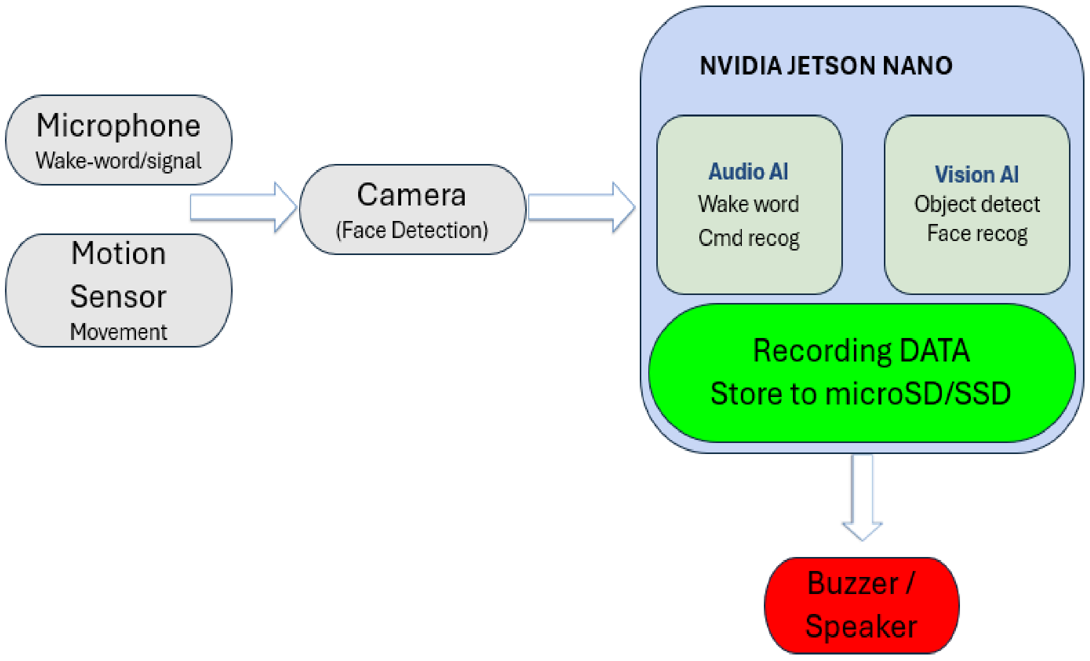
4. Related Work :

Traditional CCTV systems are widely used for security, but they usually record video continuously. This results in high power consumption and large storage requirements, even when no important activity is present. These systems also lack intelligence and cannot selectively respond to events [1].Many modern surveillance systems use cloud-based platforms, such as commercial smart cameras, to perform motion and sound detection. While these systems offer useful features, they rely on cloud connectivity and often transmit audio and video data to external servers. This raises privacy concerns and can introduce latency, especially in environments with limited or unstable internet access [4]. Recent edge AI systems focus on real-time object detection using deep learning models like YOLO. These models can achieve good accuracy, but they often require continuous processing and relatively large datasets, which makes them less efficient for low-power edge devices [2], [3].Face detection and recognition are also commonly used in security applications. Approaches such as MTCNN provide efficient face detection using lightweight cascaded networks [1], and tools like face-api.js make it easier to deploy these methods on edge or browser-based platforms [5]. However, many existing systems rely only on vision and do not integrate multiple sensing modalities.

5. Hardware & Platform

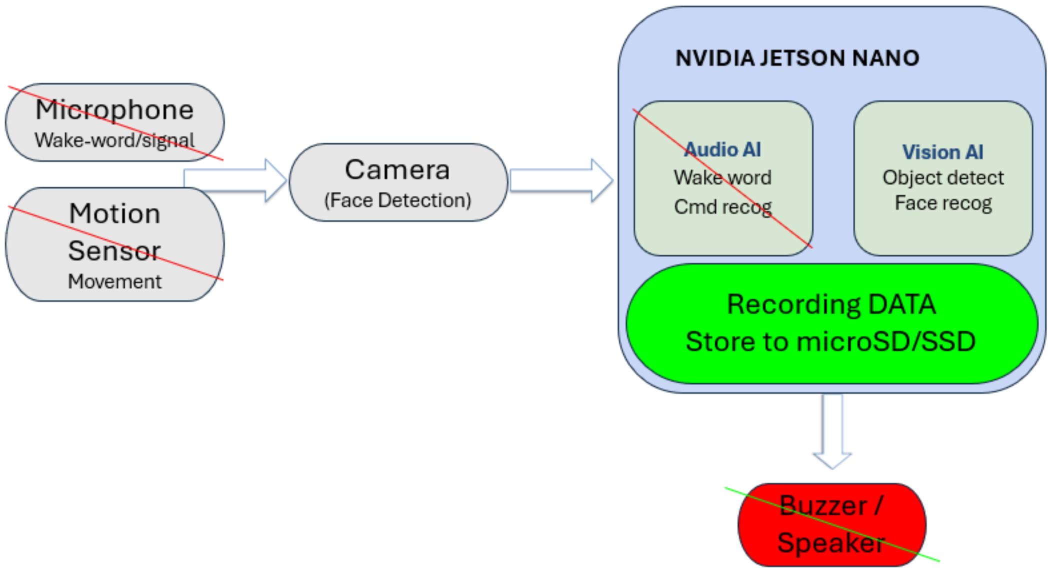
In this project, we use the NVIDIA Jetson Nano as the main edge computing platform for the Sentinel AI system. The Jetson Nano provides an ARM based CPU along with a CUDA capable GPU, which allows us to run face detection and recognition tasks locally in real time. We choose this platform because it is low-power, affordable, and well suited for edge AI applications. We use a USB camera to capture live video, which is directly connected to the Jetson Nano for local processing. All video frames are processed on the device itself, and no data is sent to the cloud, helping us maintain user privacy. For storage we use local memory such as a microSD card to save system outputs and detected events. On the software side, we use a lightweight setup based on JavaScript and HTML, along with face-api.js for face detection and recognition and WebRTC for accessing the camera stream. Overall, the hardware and platform choices allow us to build a simple, efficient, and privacy-preserving edge security system.

6. Workflow:



Picture -1 : Our proposed design workflow

At first, our system takes input from the microphone or the motion sensor. When motion or sound is detected, a signal is sent to the Jetson Nano. The system then checks whether the detected sound matches a predefined wake word stored in the dataset, or whether motion activation has occurred. If either condition is satisfied, the camera is turned on. Once the camera is active, the system starts detecting faces and checks whether the detected face matches the stored dataset. If a match is found, the system announces the name of the detected person with their name and sends a command to open the door. If no match is found, the buzzer is activated as an alert. All events and detection results are stored locally on the microSD card for future security and monitoring purposes.

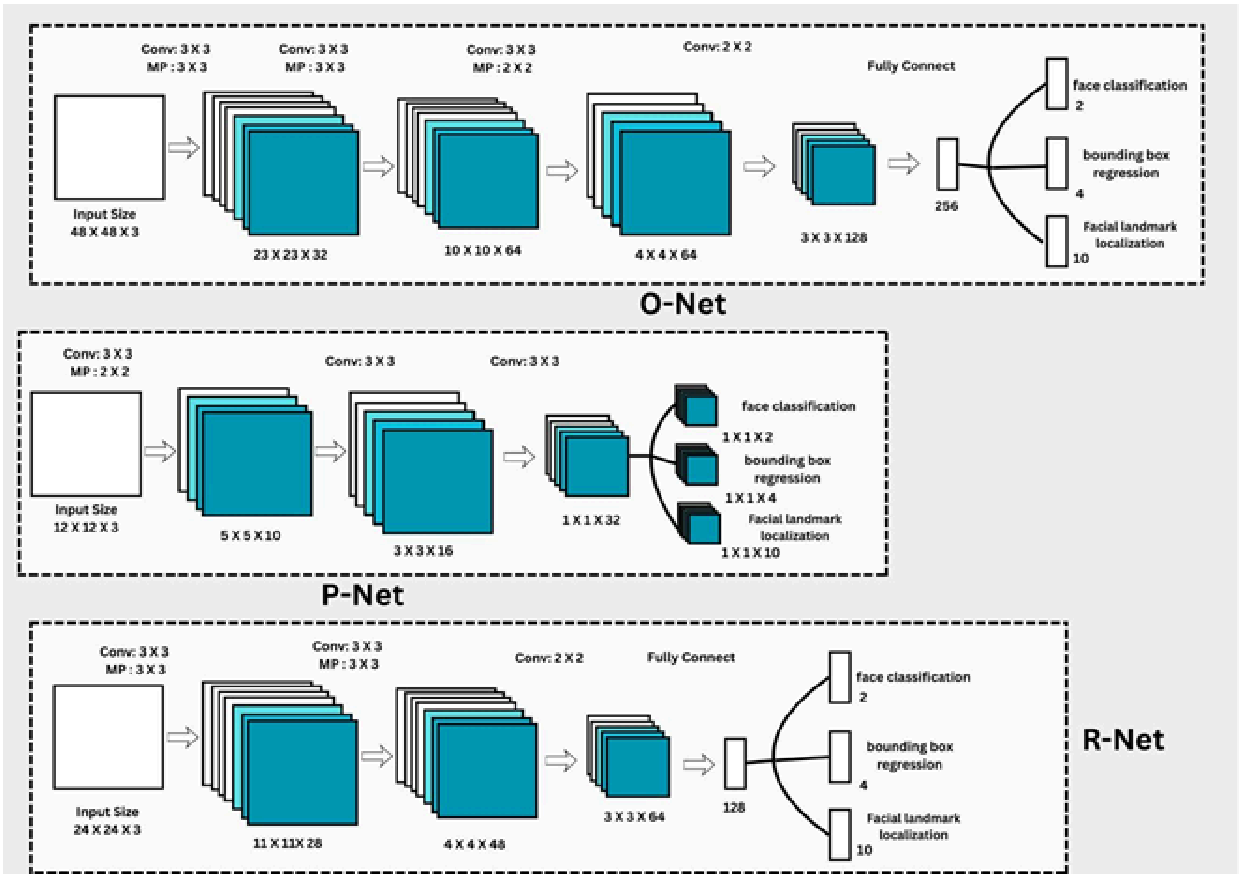


Picture -2: Our current setup workflow

Due to the unavailability of the microphone, motion sensor, and buzzer, we were unable to demonstrate the physical implementation. However, we implemented the corresponding functionality in our code.

7.Algorithm:

MTCNN (Multi-task Cascaded Convolutional Neural Network) is a deep learning algorithm that we use for face detection and basic facial feature recognition. The main purpose of MTCNN in our system is to find human faces in an image and locate important facial points such as the eyes, nose, and mouth. One advantage of using MTCNN is that it is lightweight and does not require a large amount of training data. Even with a small number of images, MTCNN can still give good detection results, which makes it suitable for edge devices**.**



Picture- 3: Algorithm flowchart

MTCNN works using three small neural networks that run one after another: P-Net, R-Net, and O-Net. The first stage, called P-Net (Proposal Network), scans the input image and finds possible face regions. At this stage, the goal is not to be perfect but to quickly find areas that might contain a face. This helps remove many non-face regions early and speeds up the overall process. The second stage is R-Net (Refine Network). R-Net takes the candidate face regions from P-Net and checks them again. It removes incorrect detections and improves the accuracy of the face bounding boxes. Compared to P-Net, R-Net performs a more careful check to make sure the detected regions actually contain a face.The final stage is O-Net (Output Network). This network produces the final and most accurate result. O-Net confirms whether a face is present, adjusts the bounding box more precisely, and identifies key facial landmarks. These final outputs are then used for face recognition in our system.

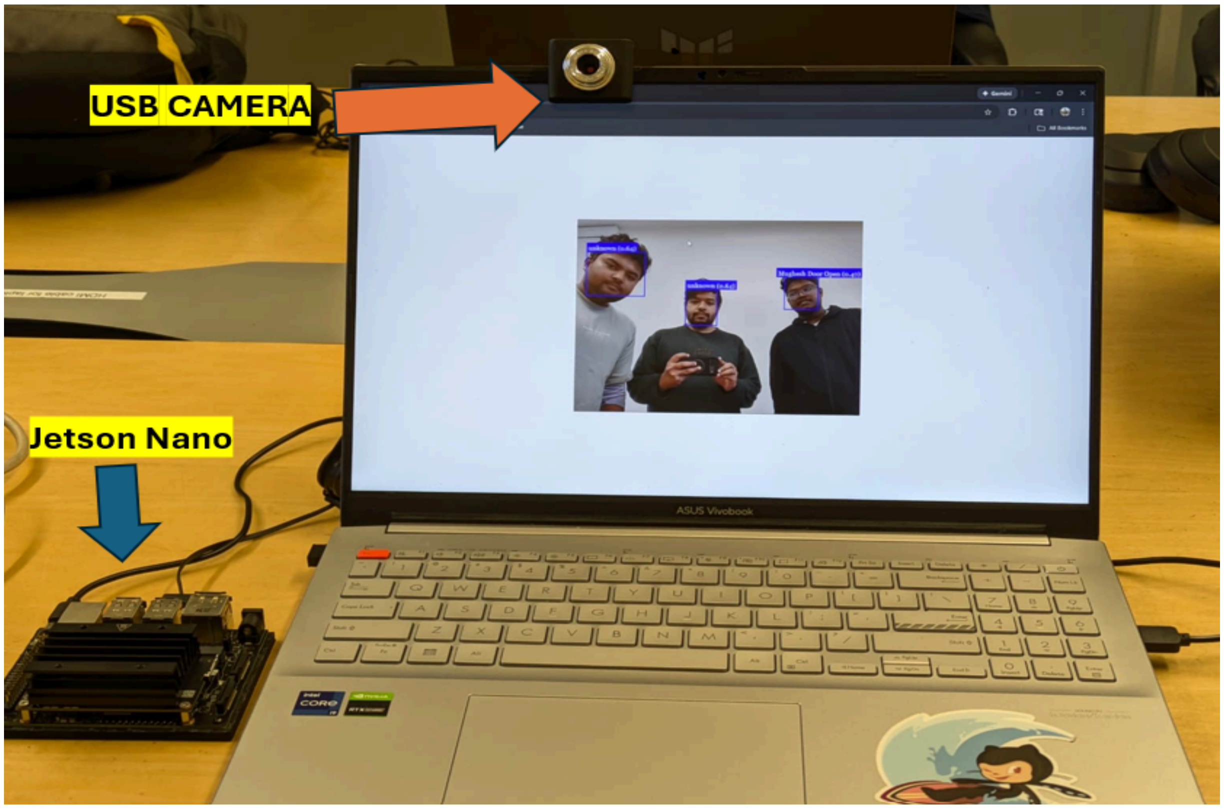
Because the detection process is divided into three steps, MTCNN is both fast and accurate. Each network focuses on a specific task, which reduces unnecessary computation. This cascaded structure makes MTCNN suitable for real-time face detection and recognition on low-power platforms like the Jetson Nano.

8.Dataset & Training Setup:

For our system, we use a very small and simple dataset to support face recognition. Instead of training a large model from scratch, we store only a few labeled face images for each user. In our implementation, we use around four images per person, captured under slightly different angles and lighting conditions. These images are saved locally on the device and are used to generate face descriptors during runtime. The system does not require a separate training phase in the traditional sense; instead, it compares the detected face features with the stored descriptors to determine whether a match exists. This approach reduces computation and memory usage, which is important for edge devices. Adding a new user is also straightforward, as it only requires placing a few images into the dataset folder. Overall, this lightweight dataset and setup allow the system to perform face recognition efficiently while keeping storage and processing requirements low.

9.Results:

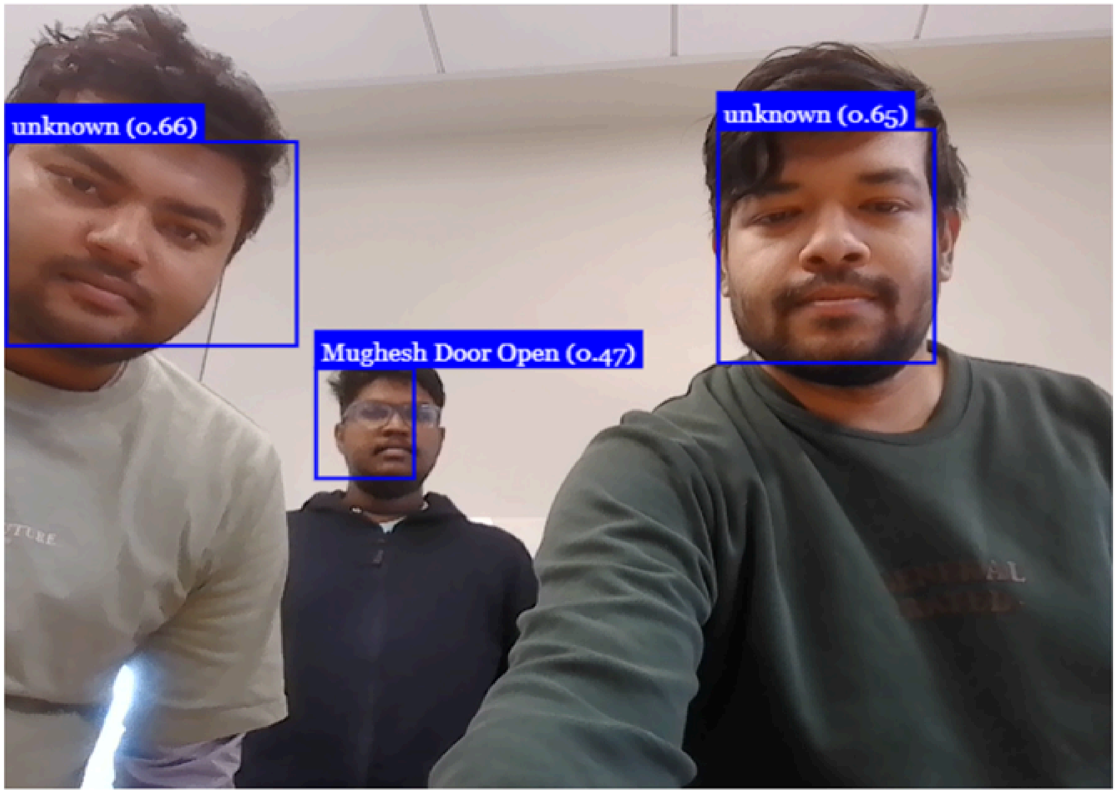
Picture 4 shows our setup, where we can see the Jetson Nano and the camera module, and a laptop monitor is used to display the output. During testing of the Sentinel-AI system on the NVIDIA Jetson Nano, we see that the system is able to detect faces in real time using the USB camera. We observe that bounding boxes appear smoothly around detected faces, and there is no significant delay between camera activation and face detection. Under good lighting conditions, we see that the system can correctly recognize known users from the stored dataset.



Picture -4: picture of our set up

We use Mughesh's (one of our teammate) face as the dataset and train our system to recognize his face. In Picture 5, we can see that the system accurately detects his face and gives the command **Mughesh, Door Open** with accuracy percentage.

We also observe that in the same picture, Uttam Kumar Saha (me) and another team member Taj are present, but since our faces were not included in the training dataset, the system identifies them as **UNKNOWN**. When we appear separately, the system correctly detects our faces as unknown. When we appear together, the system still shows **UNKNOWN**, and it is also able to correctly detect situations where one face is known and the other is unknown. Overall we can say our system works perfectly.



Picture-5 : Our final result where it is able to detect the Mughesh face with command Door Open other face is unknown because it's not in our data set.

10. Challenges & Limitations:

One of the main challenges we faced was related to lighting conditions. We noticed that the system works well under good lighting, but the accuracy of face detection and recognition decreases in low-light environments or when shadows are present on the face. Since we use a small dataset, changes in lighting and face angle sometimes cause the system to fail to recognize a known person.

Another limitation is the small size of the training dataset. We only use a few images per person, which makes the system lightweight and easy to manage, but it also affects recognition accuracy. If the face is too far from the camera or appears at a different angle than the training images, the system may classify it as unknown.

We also faced hardware limitations during the project. Due to the unavailability of the microphone, motion sensor, and buzzer, we could not demonstrate these components in the physical setup. Although the logic for these features is implemented in our code, we were unable to test them in real hardware conditions. Additionally, the system is not fully optimized for GPU acceleration on the Jetson Nano, which limits performance compared to a more optimized implementation.

Overall, these challenges show that while our system works as a prototype, there is still room for improvement before it can be used in real-world scenarios.

11.Future Work:

In the future, we plan to integrate the missing hardware components such as the microphone, motion sensor, and buzzer into the physical setup. This will allow the system to fully demonstrate the event driven workflow, where camera activated only when sound or motion is detected. Implementing these components will make the system more practical and closer to a real world security solution.

We also plan to improve the accuracy of face recognition by increasing the size of the dataset and including images captured under different lighting conditions and face angles. This would help the system perform better in real environments. In addition, we can optimize the vision pipeline using GPU acceleration and TensorRT on the Jetson Nano to achieve faster and more stable performance.

Another possible improvement is expanding the system to support more users and additional security features. For example, we can add notification support or integrate mobile alerts for unknown detections. Overall, these future improvements will help make Sentinel AI more robust, efficient, and suitable for real world deployment.

12.Conclusion:

In this project, we designed and implemented Sentinel-AI, a lightweight and privacy-preserving edge-based security system. The system uses an event-driven approach and combines vision-based face detection and recognition to reduce unnecessary processing and storage. All computation is performed locally on the Jetson Nano, which helps protect user privacy and removes the need for cloud connectivity.

Through testing we observed that the system can able to detect faces in real time and recognize trained users under good lighting conditions. Although the current implementation has limitations, such as a small dataset and incomplete hardware integration, the results demonstrate that practical edge AI security systems can be built using low-cost hardware and lightweight models. Overall we can say our system works perfectly.

13.References

[1] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks,” *arXiv preprint arXiv:1604.02878*, 2016.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv preprint arXiv:1506.02640*, 2015.

[3] R. Khanam and M. Hussain, “What is YOLOv5: A deep look into the internal features of YOLOv5,” *arXiv preprint arXiv:2407.20892*, 2024.

[4] Google Nest, “Learn how Nest cameras detect sound and motion,” Google Nest Help Documentation, accessed 2025.

[5] V. Mühler, “face-api.js: JavaScript face recognition API for the browser and Node.js,” GitHub repository, accessed 2025. [Online]. Available: https://github.com/justadudewhohacks/face-api.js