Smart Door Lock

Cheng-An Hsieh *
chengan2@andrew.cmu.edu

Ben Chiang *
benchian@andrew.cmu.edu

Jennifer Wang *
hanyiwan@andrew.cmu.edu

1 Abstract

This project presents a smart door lock system integrating face and voice recognition to enhance security and accessibility in home or office environments. The system leverages two microcontroller boards: Arduino Nicla for face detection using MobileNetv1 and Arduino Nano 33 for audio detection. Both models are trained and deployed through Edge Impulse. The face detection component identifies authorized users, while the audio detection module recognizes predefined voice commands. A solenoid lock mechanism was considered but presented power challenges due to the Arduino's limited output capacity of 30mA versus the solenoid's 1.5A requirement. The project emphasizes embedded machine learning, real-time inference, and energy-efficient design, offering a customizable, cost-effective alternative to commercial smart locks.

2 High-level Block Diagram

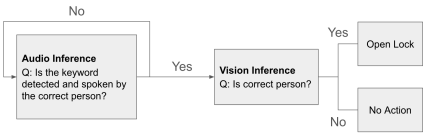

Figure 1: Behavior Block Digram

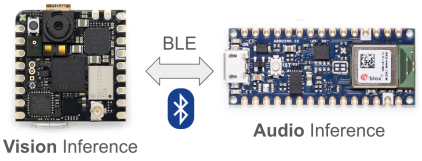

Figure 2: Structure

3 Dataset Source and Method of Collection and Data cleaning

Before gathering the data, we brainstormed and decided to use our team member Jennifer as the target.

3.1 Face images

We collected 191 images and split them into 172 images for training and 19 images for testing. The dataset includes 89 face images of Jennifer and 83 images featuring other individuals or background scenes.

We used our smartphone to collect 97 images, the Arduino Nano 33 BLE Sense to capture 62 images, and the Arduino Nicla to gather 32 images. The image quality varied to enhance the model's robustness. We discarded some images that were excessively blurred due to slow shutter speed.

*These authors contributed equally to this work

Preprint. Under review.

3.2 Audio

The voice command is set to "peanut." We collected 745 samples, including 267 target samples from Jennifer and 478 non-target samples. All 267 target samples featured different tones but the same "peanut" command. The 478 non-target samples included instances of "peanut" spoken by other individuals and various background noises.

We used the Arduino Nano 33 BLE Sense to gather voice data. During data collection, users repeated the command "peanut" for 10 seconds, and the audio recordings were segmented into individual voice commands. Each segment was cropped using a 700ms window, with potential time shifts applied as a form of data augmentation. Additionally, corrupted voice samples were manually reviewed and removed to ensure dataset quality.

4 Feature Extraction

4.1 Image Classification

To reduce computational complexity and memory usage, we use grayscale images for image classification, which is crucial for embedded systems like the Arduino Nicla Vision. The images are resized to 96×96 pixels to ensure uniformity while maintaining sufficient detail for effective recognition.

Features are extracted using the MobileNetV1 architecture, which is lightweight and optimized for embedded environments. The network extracts spatial features through depthwise and pointwise convolutions, making it well-suited for devices with limited resources. Data augmentation techniques such as horizontal flipping, rotation, and random cropping were applied to enhance model generalization.

4.2 Audio Classification

The audio classifier uses Mel Frequency Cepstral Coefficients (MFCC) to extract features from the audio signal. Key parameters include 15 MFCC coefficients per frame, with a frame length and stride of 0.02 seconds, ensuring distinct, non-overlapping segments for analysis. The feature extraction leverages 32 Mel filters applied to the signal, with an FFT length of 512 for high-resolution frequency domain representation. Frequencies between 20 Hz and 4000 Hz are analyzed, covering the typical range of human speech. These parameters collectively ensure accurate and robust feature representation for audio classification.

5 Classifier Architecture and Rationale

5.1 Audio Classifier

The audio classifier is designed using a 1D convolutional neural network (CNN) to extract temporal features from audio signals while ensuring compatibility with the constrained hardware of the Arduino Nano 33 BLE Sense. The model accepts 600 audio features as input, reshapes them into 15 columns, and processes them through two sequential 1D convolutional layers with filters of sizes 16 and 8, each followed by pooling and dropout layers with a 0.25 dropout rate to prevent overfitting. The flattened feature vector is passed to a fully connected output layer with two nodes representing "Jennifer's voice" and "Other." This architecture balances simplicity and performance, effectively extracting temporal patterns while maintaining computational efficiency for real-time inference.

5.2 Face Classifier

The face classifier employs MobileNetV1 with a 96x96 input size and a width multiplier of 0.25. The final dense layer is removed, and no dropout is applied, ensuring efficient feature extraction while minimizing computational overhead. This streamlined configuration supports face recognition tasks in real-time on constrained hardware platforms.

6 Confusion Matrix and Overall Accuracy Metrics

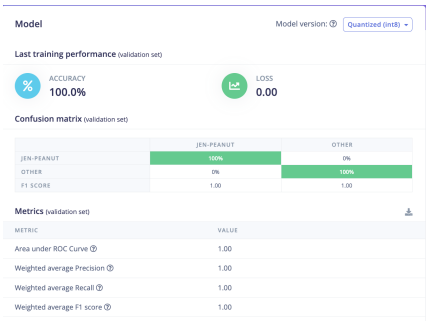

Figure 3: Confusion matrix for audio classification

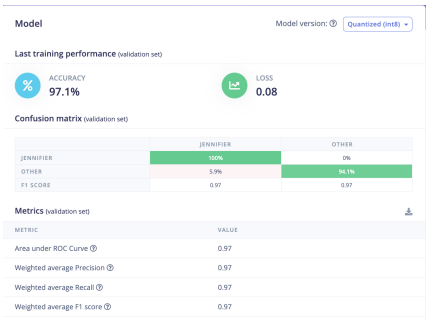

Figure 4: Confusion matrix for face detection

7 Deployment Method: Hardware Target and Software Techniques

Our deployment utilizes two separate boards: the Arduino Nicla Vision and the Arduino Nano 33 BLE. The facial recognition model runs on the Nicla Vision due to its increased RAM and computational power, enabling the deployment of a MobileNet-V1 model alongside BLE communication

The audio inference pipeline is implemented on the Arduino Nano 33 BLE because of its lower compute requirements. This is due to the lower dimensionality of audio data and the additional pre-processing performed by the MFCC block. The two boards communicate via BLE, with the Nicla Vision functioning as the central server and the Arduino Nano as a peripheral.

The Arduino Nano continuously samples audio, performs inference, and broadcasts results via BLE notifications. This setup offloads the tasks of continuous sampling and processing from the central server. The Nicla Vision activates its computationally intensive vision pipeline only upon receiving a notification from the Nano, reducing power consumption. This design is advantageous for battery-powered applications as it conserves the power budget by limiting resource-intensive operations to when they are needed.

8 Significant Challenges and Lessons Learned

Hardware-software library compatibility issues: Our initial plan to use a single Nicla Vision board for both pipelines. However, we ran into library compatibility issues with the M4 cores. The libraries that interface with sensors (microphone and camera) cannot be deployed on the M4 core

RPC library issues: The RPC library does not support custom structs, limiting our ability to deploy audio inference pipeline on the M4 core by sending the audio signal over RPC.

Iterative problem solving: We eventually adopted our design by splitting the pipelines across two boards. This highlights the iterative problem-solving process during embedding development. This pivot ultimately led to a more efficient and simple solution.

Understanding power constraints We ran into issues with the door lock drawing over-budgeted current at 1.5A. Careful evaluation should be done prior to parts ordering and during the system design phase the ensure every part fits within the power budget.

9 Links to Images or Videos

Video Demo Link