Information Technology Course
Module Individual Project
Supervised by Dr. Andreas Pech

AI-driven energy-efficient approach for human detection using an ultrasonic sensor system

Anurag De
1400450
anurag.de@stud.fra-uas.de

Abstract: **In modern office environments, energy consumption due to lighting makes up a big part of the total energy consumption. For this reason, the productive use of automatic lights, combined with effective human occupation detection, can lead to greater energy savings and is environmentally friendly. In this paper, a two-stage energy-efficient hybrid human detection system is proposed, utilizing a Red Pitaya board and an SRF02 ultrasonic sensor at its core. The system uses both signal processing and more computationally intensive deep learning techniques to detect humans in its monitoring zone successfully. In its two-stage detection framework, the first stage detects general activity using signal processing, and then the more computationally intensive convolutional neural network (CNN) is utilized in the second stage for effective human classification. A comprehensive graphical user interface (GUI) is provided to the user, which allows them to control the system, monitor it in real-time, and adjust the settings. The CNN model, which is integrated with the system, achieved a high accuracy of 95.79% on a separate validation dataset from a simulated real-world office environment.**

Keywords—Red Pitaya, Convolutional Neural Network, Short Time Fourier Transform, Spectrogram, Graphical User Interface, Signal Processing, Peak Detection, Ultrasonic Sensor.

I. Introduction

As we see an increase in demand for highly intelligent environments, whether it is smart homes or automatic building management systems, accurate and cost-effective methods of detecting human presence are vital. Detection of human occupancy can be a highly effective strategy for reducing building energy consumption. Instead of relying on manual switches, an automatic occupancy detection sensor system can automatically turn the lights on or off. If the person is not present, the system turns the light off. Studies have shown that up to 30% of energy can be saved by utilizing motion sensors to regulate lights [1]. Ultrasonic sensor systems can offer numerous benefits for this purpose. They are inexpensive, reliable, and even work in the dark. Ultrasonic sensors utilize a principle similar to echolocation, which bats and dolphins utilize. Distance is measured by emitting a high-frequency sound wave, far above the range of human hearing, and then calculating the time it takes for the echo of the sound to bounce back. The ultrasonic frequency range begins at 20 Kilohertz and extends up to a few hundred Kilohertz. Since the speed of sound is constant (343 meters per second) in air, the distance can be calculated by multiplying the time by the speed of sound and then dividing it by 2. As they are durable and affordable, they make them suitable for large-scale deployments, such as in large office environments. Traditional sensors, such as passive infrared sensors, although inexpensive, are unable to detect static individuals properly. Camera-based systems are highly accurate, but they also raise privacy concerns. However, ultrasonic sensors can address these concerns, as they are accurate in detecting simple indoor activities [2] and also preserve privacy.

In our proposed system, as shown in Figure 1, an ultrasonic sensor system consisting of a Red Pitaya STEMlab board and an SRF02 ultrasonic distance sensor is used for the detection of human occupancy in a simulated office environment. The user will be able to control settings through a graphical user interface (GUI). The Red Pitaya STEMlab 125-14 is a versatile, multi-purpose open-source System-on-Chip, which contains a dual-core ARM Cortex-A9 processor and an FPGA (Field Programmable Gate Array) [3]. The device can be controlled by the Ubuntu operating system on board. The SRF02 is an inexpensive ultrasonic rangefinder. It is capable of measuring distances from 16 cm to 6 m [4]. It operates on a standard 5 V supply. The devices are connected via the I2C bus.

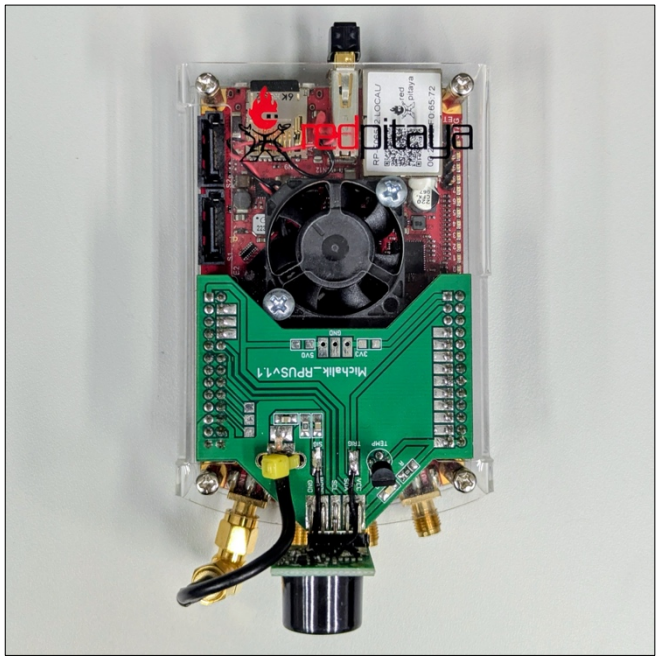

Figure 1: Red Pitaya board with SRF02 ultrasonic distance sensor

The human detection application runs on a laptop, which is connected to the sensor system via Ethernet. It operates on a client-server model, where the Red Pitaya system acts as the server and the application acts as the client. A C program running inside the Red Pitaya board utilizes the ultrasonic sensor to capture 25000 ADC (Analog-to-Digital-Converter) data samples efficiently. This data is sent via UDP (User Datagram Protocol) connection to the Python application for further analysis. For the promotion of energy efficiency, the detection system works on two stages. In the initial stage (Activity Detection), the application uses a light-weight peak detection to detect sudden changes in the environment, for example, sudden movements inside the sensor's monitoring zone. If any sudden changes are detected, the application then turns the LED7 of the Red Pitaya board on. The application then moves on to the next stage (CNN Classify). At the beginning of building this project, numerous ADC data samples, simulating different scenarios of an office environment, were collected as 1D signals and converted to 2D spectrograms for capturing the signal's frequency over time. The processed data was then used to train a 2D CNN classification model using the PyTorch framework. In the second stage, we use the custom pre-trained model for accurate human classification using machine learning. The user also has options to tune numerous settings of this application. In the later stage, also, if human presence is detected continuously, the LED7 stays on, as shown in Figure 2.

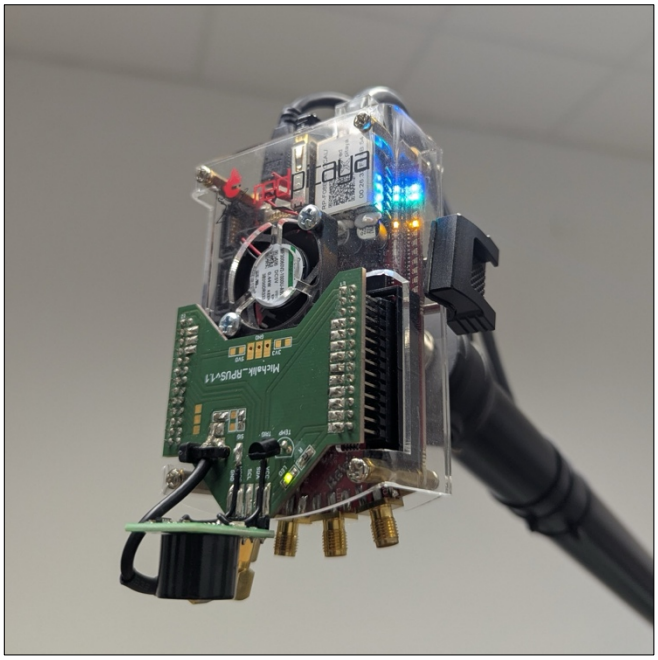

Figure 2: LED7 switched on in the Red Pitaya board

Once the human presence is not detected, then the LED turns off after a certain period of time, and then the application moves back to the initial stage again.

II. Methods

A. Data acquisition strategies

In order to simulate the working of the sensor system in an office environment, ADC data samples have been collected in an indoor setting to cover a range of common office activities and conditions. The sensor system was kept at a fixed height of 230 cm on a metal stand. The end goal was to create a robust and representative dataset that helps in the distinction of humans and non-humans in an environment. The collected data can be primarily divided into two categories: Human and non-human.

- The data for the human category were collected from three different human subjects. The subjects were in different stationary positions, like standing, sitting on a chair, or at an office table, in various positions. The influence of clothing was also considered by collecting data for each subject with and without a jacket. A total of 60000 data signal samples were taken for this category. Figure 3 shows one such scenario, where the human subject is sitting at the desk.
- The data for the non-human category was acquired to represent common office objects and environmental states, such that a wide variety of static office-related scenarios are covered. Various furniture items, like a chair, an office table, and drawers, were used. Objects like laptops, bags, books, etc., were placed on them to simulate realistic non-human conditions. Figure 4 shows one such scenario. A total of 60000 data signal samples were also taken for this category to ensure that the model is trained on an equal number of examples of both categories.

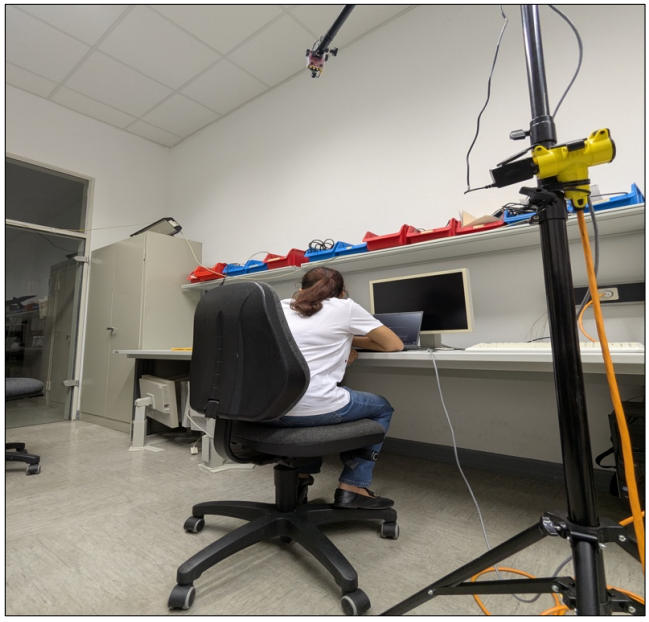

Figure 3: Data collected while the human subject is sitting at the desk

This data was then used to train the 2D CNN model using PyTorch. To test the model's performance, an independent validation dataset was also created to evaluate the model's capability in unseen scenarios. Data from a new human subject was also collected, whose data was not used in training. This approach ensures that our validation set represents a truly unseen population. A total of 12000 data signal samples were taken with a 1:1 ratio of human and non-human categories. For human and non-human categories, similar scenarios to training data acquisition were used.

Figure 4: Data collected with a bag and a jacket on top of a chair

B. Pre-processing of data

The primary goal of data pre-processing is to convert the raw signal data collected from different scenarios to a format suitable for training the deep learning model. The data acquired using data acquisition software saves the signal data in CSV format. All the rows correspond to an individual signal from the sensor system. The first 5500 data points are removed as they contain the sending pulse. Machine learning models require fixed-size input, so a fixed length of 19517 data points is taken. If the size is larger, it is truncated to match this length. Also, shorter signals are padded with zeroes to do the same. The one-dimensional time-series signals are then converted to two-dimensional spectrograms using STFT(Short Time Fourier Transform) using torch.stft python module. FFT(Fast Fourier Transform) window size of 256 and Hop length of 128 were used, resulting in 50% overlapping between consecutive windows. To reduce spectral leakage, a Hann window was also applied before applying the FFT. Spectrograms help us by giving details about the frequency content of a signal through time. The final spectrogram is obtained by taking the absolute value (or magnitude) after STFT for each frequency component. The final processed spectrogram data was saved as a NumPy array. Another array containing labels was also created, which contained the respective labels("1" for Human and "0" for Non-human). Each spectrogram has 129 Frequency bins and 153 Time steps. Figure 5 visually represents spectrograms from both human and nonhuman data. The color intensity indicates magnitude in decibels(dB).

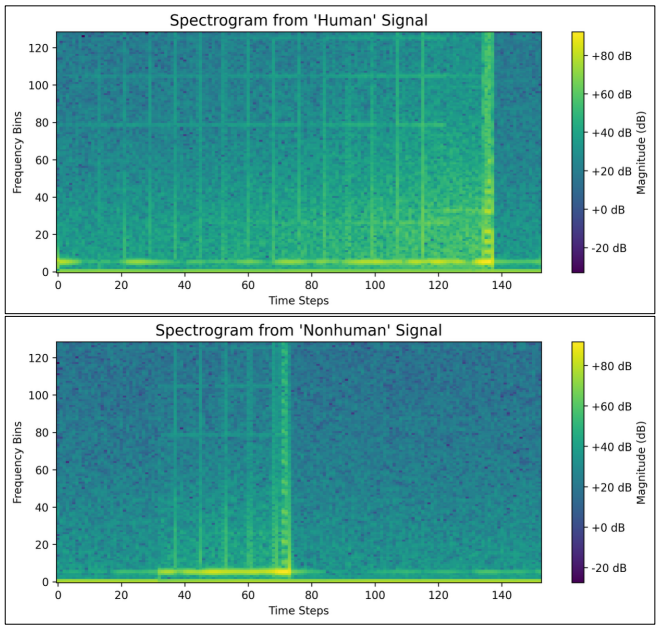

Figure 5: Spectrogram plots for human and nonhuman signal samples

C. Machine Learning

The pre-processed data was used to train a 2D Convolutional Neural Network model using PyTorch. PyTorch was chosen for this project as it is simple, powerful, and flexible. The model SignalCNN2D is used by our application for the classification of human and non-human. The model architecture network consists of the following layers:

- The input layer accepts a 129x153 single-channel spectrogram.
- A 2D convolutional layer with one input channel and 16 output channels. It is followed by batch normalization, a ReLU activation function, and a Max Pooling layer.
- A second 2D convolutional layer with 16 input channels and 32 output channels. It is also followed by batch normalization, a ReLU activation function, and a Max Pooling layer.
- The output from the convolutional blocks is flattened and passed to a Linear layer, mapping the flattened features to 128 units, followed by ReLU activation. Then the Dropout layer is used for the prevention of overfitting with a dropout rate of 0.5. Finally, another Linear layer maps 128 units to 2 outputs corresponding to our two output classes, human and non-human.

The model design prioritizes resource efficiency and simplicity. It is lightweight and promotes fast training and inference, hence making it very useful for the human detection application. The model has 4,986,018 total trainable parameters. Figures 6 and 7 show the high-level representation and detailed model architecture of the SignallCNN2D model.

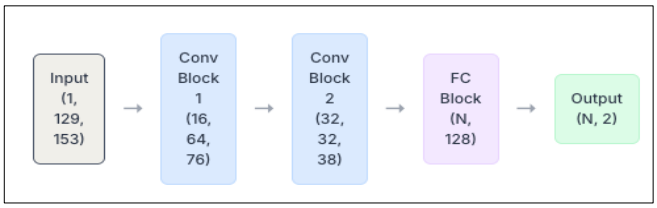

Figure 6: High-level representation of the architecture of SignalCNN2D

SignalCNN2D Model Architecture

Layer Group	Layer (Type)	Details	Output Shape (Batch Size, C, H, W)	Parameters
Input	-	1-channel Spectrogram	(N, 1, 129, 153)	0
conv_block1	Conv2d	16 filters, kernel 3x3, stride 1, padding 1	(N, 16, 129, 153)	160
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 16, 129, 153)	32
	ReLU	-	(N, 16, 129, 153)	0
	MaxPool2d	Kernel 2x2, stride 2	(N, 16, 64, 76)	0
conv_block2	Conv2d	32 filters, kernel 3x3, stride 1, padding 1	(N, 32, 64, 76)	4,640
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 32, 64, 76)	64
	ReLU	-	(N, 32, 64, 76)	0
	MaxPool2d	Kernel 2x2, stride 2	(N, 32, 32, 38)	0
-	Flatten	-	(N, 38912)	0
fc_block	Linear	In: 38912, Out: 128	(N, 128)	4,980,864
	ReLU	-	(N, 128)	0
	Dropout	p = 0.5	(N, 128)	0
Output	Linear	In: 128, Out: 2 (human/non-human)	(N, 2)	258
			Total Trainable Parameters:	4,986,018

Figure 7: Detailed Model Architecture of SignalCNN2D

D. Project architecture and workflow

Figure 8 illustrates the high-level operational workflow of the whole system.

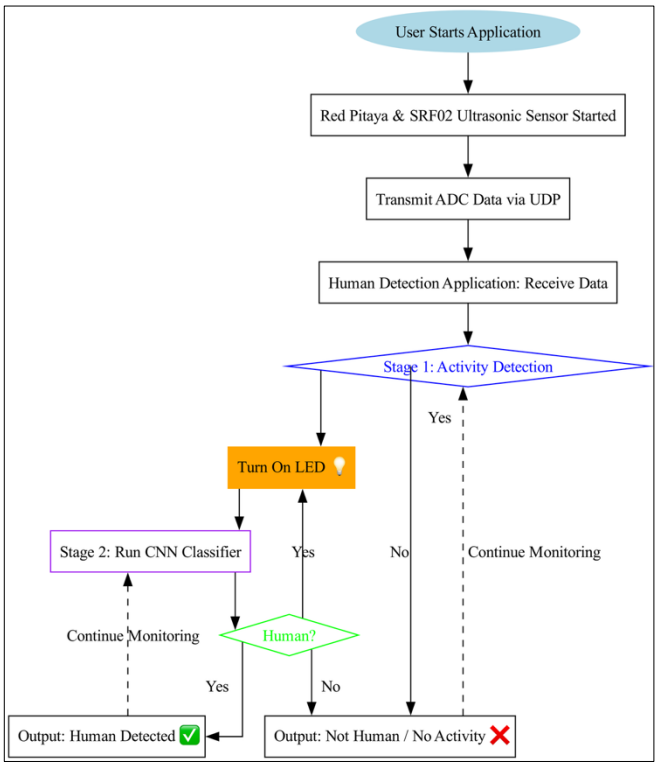

Figure 8: Flowchart to show the operational workflow of this project

When the user starts the sensor system from the human detection Python application, an SSH connection is established to the Red Pitaya OS, and the compiled data acquisition program in C is executed. After a brief pause, the program in Red Pitaya is initialized, and the application, as a UDP client, performs a UDP handshake with the server, which in this case is the compiled program running inside the sensor and waiting for a UDP command from the client application. After the handshake completes successfully, the system enters a continuous data acquisition and processing loop. A UDP message is sent to the program running inside the Red Pitaya OS, requesting the most recently acquired sensor data packet. In response, the program sends a data packet containing a 17-field metadata header and 25000 raw ADC samples inside. The data packet is received in the Python application and processed further. It is plotted in the GUI of the application, and for the signal to be processed by the pre-trained 2D CNN model, a specific slice of the signal (19517 samples) is taken by removing the first 5500 samples. This request-response cycle continues, and a steady stream of data is created between the sensor system and the Python application for real-time analysis and classification.

The application uses a two-stage detection cycle to first detect general activity using general signal processing techniques and then classify the activity using deep learning. This saves computational resources by only using a neural network when necessary. The worker thread inside the application continuously handles the raw signal containing 25000 samples received from the sensor system. The application processes two signals every second. In the "Activity Detection" mode, the most prominent peak in the signal is identified using scipy.signal module of Python, and its index position is tracked for a short time period of two seconds (that means four consecutive signals). This is done to prevent the detection of a random shift in peaks due to noise. If the position of the peak shifts by more than a pre-defined threshold of 2000 index points (this can be changed by the user in settings), this is considered "Activity" and the LED turns on. The LED is turned on inside the Red Pitaya OS by the application, by writing a hexadecimal value directly to the hardware memory register, activating the GPIO(General purpose Input/Output) pin connected to LED7.

Once any significant activity is detected, the application moves to "CNN Classify" mode. In this mode, a slice of the received signal is processed by converting it to a spectrogram using Short-Time Fourier Transform and feeding it to a PyTorch 2D CNN classification model for inference. The model then analyzes it and gives a probability score for human presence as output. If the probability crosses a certain threshold (the Default is 0.99, which can be changed by the user in settings), the LED is turned on and the status is updated in the application interface. There is also a debounce logic implemented to prevent single false positives. The application requires a certain number of consecutive predictions (Default is 3, can be changed by the user) to be above the threshold probability for it to confirm human presence. With no confirmed human presence, the "CNN Classify" mode lasts for four times the time-out period value from the time it is activated.

E. Graphical User Interface (GUI)

This section describes the Graphical User Interface(GUI) of the Python application. It is based on the PyQt6 framework and is designed for real-time monitoring and control of the system. Figure 9 shows the initial starting window of the application.

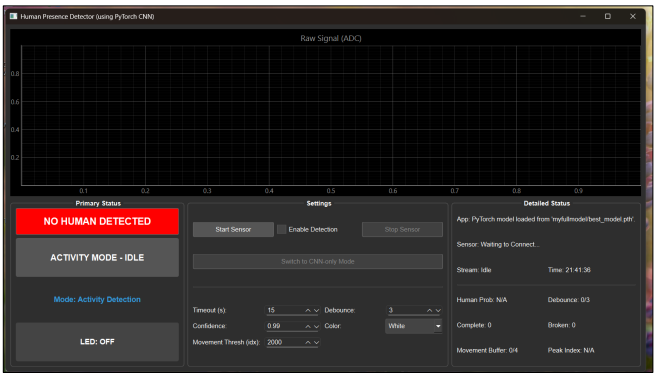

Figure 9: Initial user interface of the application before starting the application

The whole detection system, described in section D, starts once the "Start Sensor" button is pressed. Once the data stream is stable from the sensor, pressing the "Enable Detection" button enables the two-stage detection cycle. Pressing the "Stop Sensor" button stops the whole human detection process. The "Detailed Status" section gives the user a comprehensive, real-time overview of the system's performance and current status. The "Settings" section allows the user to fine-tune the parameters of the detection system. The "Primary Status" section visually shows a quick summary of the most critical system information using color-coded indicators. The graph in the top section of the application displays the real-time plot of the signal.

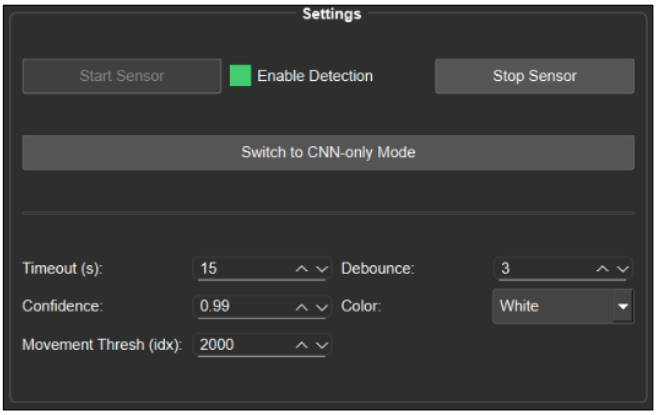

Figure 10: Settings section of the application

In the "Settings" section, numerous tuning options are given to the user, as shown in Figure 10.

- Timeout (s) allows the user to set the time duration, in seconds, of how long the human or activity detection statuses remain active before timing out and reverting to the idle state, as well as how long the LED stays on. Default is 15 seconds.
- Confidence allows the user to set the minimum probability score from the CNN model over which it is considered positive human detection. Default is 0.99.
- Debounce counter allows the user to set a consecutive number of positive human detections before human presence is confirmed. This is done to prevent false positives. Default is 3.
- Movement Threshold (index) allows the user to set the value of how many index points a signal's peak should move to be flagged as significant movement. Default is 2000.
- Color allows the user to select the graph line color of the signal.
- The user is also given the option to select the detection mode – either the more computationally efficient "Activity+CNN detection" mode or only the CNN detection mode. The latter mode only utilizes the CNN model for continuous human detection.

The "Primary Status" section has the following options,

- Human detection status button displays the status of human presence - "HUMAN DETECTED" in green, as shown in Figure 11, or "NO HUMAN DETECTED" in red.
- Activity Indicator label indicates whether activity is detected or not – "ACTIVITY MODE – IDLE" in dark grey or "ACTIVITY DETECTED (Movement)" in blue.
- Mode label depicts which detection mode is currently active: "Mode: Activity Detection" or "Mode: CNN Classify".
- LED indicator shows whether the LED is on or off in the GUI, as shown in Figure 11.

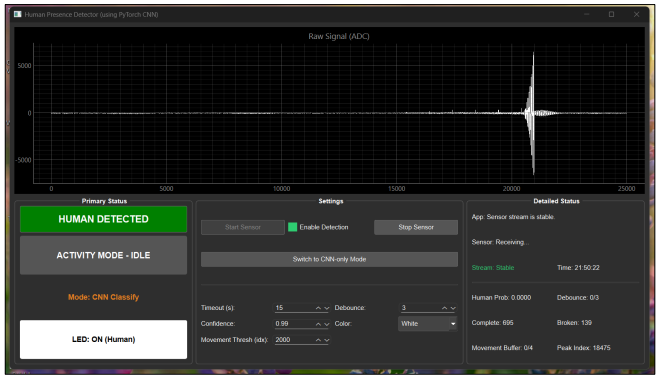

Figure 11: GUI of the application when a human is detected

The "Detailed Status" section gives the user a real-time overview of the application. It has the following options:

- App gives the user high-level messages about the application status.
- Sensor gives the user information about the status of the connection to the Red Pitaya sensor system.
- Stream gives the user information about the stability of the incoming data stream from the sensor.
- Time gives the user information about the current system time where the application is running.
- Human probability gives the user information about real-time probability output from the CNN model.
- Debounce gives the user information about the status of the debounce counter.
- Movement Buffer gives the user information about how many recent peak locations have been stored.

- Peak Index gives the user information about the numerical index of the peak of the last analyzed signal.
- Complete / Broken Signals gives the user information about the running count of the total number of successfully received packets and missed/corrupted packets.

III. Results

The SignalCNN2D model used in the application was evaluated by two distinct methodologies in order to assess its performance.

- By using a stratified 80/20 hold-out split. 80% of the data was used for training, and 20% was used as a validation (hold-out) set. 96000 samples were used for training and 24000 samples for validation.
- By using a separate pre-defined validation dataset of 12000 samples. 120000 samples were used for training the model.

In the first way(stratified 80/20 hold-out split), the model was trained for 15 epochs. It achieved a peak validation accuracy of **99.61%** at epoch 12, so it was saved. The total training time was approximately 23 minutes and 13 seconds. The training loss consistently decreased over epochs, and the validation accuracy increased rapidly and plateaued near the peak. This shows that the model did not overfit.

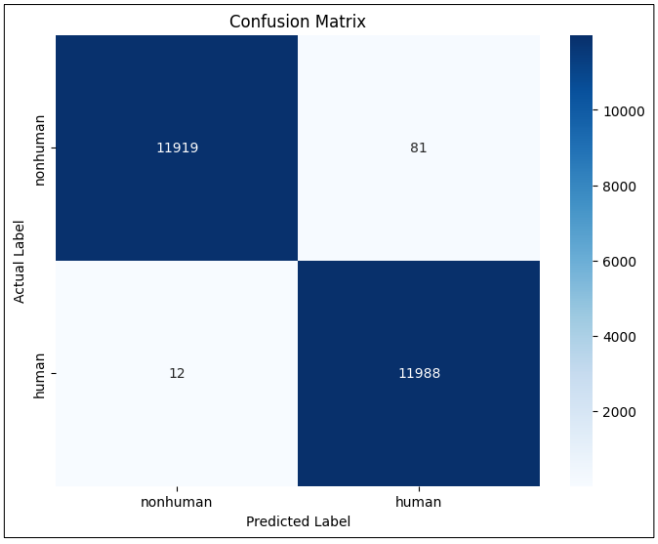

Figure 12: Confusion matrix for model trained in the first way (hold-out split)

Figure 12 confirms the model's very high accuracy. The performance metrics were nearly perfect as shown in Figure 13.

```
==================================================
Classification Report
==================================================
              precision    recall  f1-score   support

    nonhuman       1.00      0.99      1.00     12000
       human       0.99      1.00      1.00     12000

    accuracy                           1.00     24000
   macro avg       1.00      1.00      1.00     24000
weighted avg       1.00      1.00      1.00     24000
```

Figure 13: Classification report for model trained in the first way (hold-out split)

In the second way(validation by a separate dataset – discussed earlier in section A), the model was trained with an early-stopping rule in place with 50 epochs. However, the training process was stopped early in 12 epochs as the performance did not improve after five consecutive epochs. The model achieved peak validation accuracy of **95.79%** after seven epochs. The early stopping prevented the possibility of further overfitting. The total training time was 29 minutes and 18 seconds.

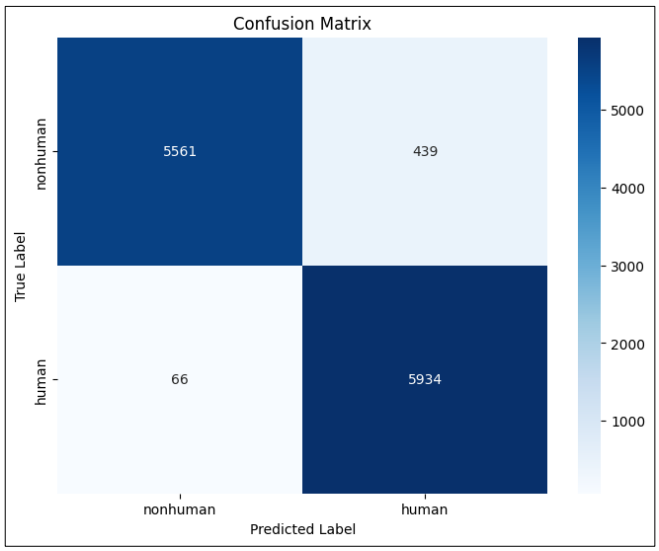

Figure 14: Confusion matrix for model trained in the second way (pre-defined set)

Figure 14 shows that although the model accuracy is very high, the model incorrectly predicted non-human samples as human more than the earlier model. This is also confirmed by the lower precision value of 0.93 in the classification report, as shown in Figure 15.

```
--- Classification Report ---
              precision    recall  f1-score   support

    nonhuman       0.99      0.93      0.96      6000
       human       0.93      0.99      0.96      6000

    accuracy                           0.96     12000
   macro avg       0.96      0.96      0.96     12000
weighted avg       0.96      0.96      0.96     12000
```

Figure 15: Classification report for model trained in the second way (pre-defined set)

The model was tested on a pre-defined validation dataset mimicking potential real-world office environment scenarios, especially with different human subjects as discussed earlier in Section A. However, the model demonstrated high recall for humans (0.99). The latter model was chosen for the application as it performed well with "out-of-distribution" data, demonstrating its generalization capabilities.

The application performance for human detection was also satisfactory. On a few trials of a human subject entering the monitored zone, the system was able to detect human presence every time and turn the LED on in the Red Pitaya board. Also, when non-human objects, for example, a chair, were pushed into the monitored zone, it successfully detected activity and

also started the LED. However, when the objects were classified as non-human by the system, the LED was successfully turned off.

IV. Discussion

The two-stage human detection application performs well in most scenarios in a simulated test environment. It provides a great balance between computational efficiency and high accuracy, thereby promoting energy efficiency. However, the system was not tested on completely new and unseen environments. The model sometimes also exhibited some confusion while detecting non-human objects with complex surfaces (similar to humans), like, for example, a chair with a bag, a cluttered basket, etc. Exploring other machine learning/deep learning techniques, which are fast and non-complex for improving the precision of non-human detection, would also improve the system's accuracy in general.

V. References

[1] P. Chaudhari, Y. Xiao, M. M.-C. Cheng, and T. Li, "Fundamentals, Algorithms, and Technologies of Occupancy Detection for Smart Buildings Using IoT Sensors," *Sensors*, vol. 24, no. 7, pp. 2123–2123, Mar. 2024, doi: https://doi.org/10.3390/s24072123.

[2] A. Ghosh, A. Chakraborty, D. Chakraborty, M. Saha, and S. Saha, "UltraSense: A non-intrusive approach for human activity identification using heterogeneous ultrasonic sensor grid for smart home environment," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 1, pp. 1-22, Jan. 2023, doi: 10.1007/s12652-019-01260-y.

[3] Red Pitaya. STEMlab 125-14 User Manual. [Online]. Available: https://redpitaya.readthedocs.io/en/latest/developer_guide/125-14/top.html. (Accessed: Sep. 24, 2025).

[4] Robot Electronics. SRF02 Ultrasonic Range Finder Technical Specification. [Online]. Available: https://www.robot-electronics.co.uk/htm/srf02tech.htm. (Accessed: Sep. 24, 2025).