

Article

Towards an Indoor Gunshot Detection and Notification System Using Deep Learning

Tareq Khan 

School of Engineering, Eastern Michigan University, Ypsilanti, MI 48197, USA; tareq.khan@emich.edu

Abstract: Gun violence and mass shootings kill and injure people, create psychological trauma, damage properties, and cause economic loss. The loss from gun violence can be reduced if we can detect the gunshot early and notify the police as soon as possible. In this project, a novel gunshot detector device is developed that automatically detects indoor gunshot sound and sends the gunshot location to the nearby police station in real time using the Internet. The users of the device and the emergency responders also receive smartphone notifications whenever the shooting happens. This will help the emergency responders to quickly arrive at the crime scene, thus the shooter can be caught, injured people can be taken to the hospital quickly, and lives can be saved. The gunshot detector is an electronic device that can be placed in schools, shopping malls, offices, etc. The device also records the gunshot sounds for post-crime scene analysis. A deep learning model, based on a convolutional neural network (CNN), is trained to classify the gunshot sound from other sounds with 98% accuracy. A prototype of the gunshot detector device, the central server for the emergency responder's station, and smartphone apps have been developed and tested successfully.

Keywords: convolutional neural network; Mel Frequency Cepstral Coefficients (MFCC); Jetson Nano; smartphone app; SQL database; server; Bluetooth; Wi-Fi; gunshot; push notification



Citation: Khan, T. Towards an Indoor Gunshot Detection and Notification System Using Deep Learning. *Appl. Syst. Innov.* **2023**, *6*, 94. <https://doi.org/10.3390/asi6050094>

Academic Editor: Christos Douligeris

Received: 14 August 2023

Revised: 1 October 2023

Accepted: 13 October 2023

Published: 19 October 2023



Copyright: © 2023 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

From January 2023 to September 2023 in the USA, the total number of deaths due to gun violence was 31,394—1273 were children between the ages of 0 and 17 and the total number of injuries was 27,408 [1]. It is estimated that 31% of public mass shootings occur in the USA, although the USA accounts for only 5% of the world's population [2]. One way to reduce the loss from gun violence is to detect the incident early and notify the police as soon as possible. In this project, a novel gunshot detector device is developed that automatically detects the indoor gunshot sound and then sends the gunshot location to the nearby police or emergency responder's station within a second using the Internet. The users of the device and the emergency responders also receive smartphone notifications as soon as the shooting happens. Additionally, the device captures gunshot sounds along with timestamps, providing valuable data for post-crime scene analysis. A deep-learning-based gunshot sound classification algorithm is developed and implemented in the gunshot detector device. A central server for the emergency responder or police station and smartphone apps is also developed in this project. The overall operation of the proposed system is shown in Figure 1. The needs and significances of the proposed system are mentioned below:

- The United States sees the most school shootings in the world [3]. Shootings inside other indoor places such as in homes, shopping malls, clubs, and places of worship are also becoming widespread around the world [4,5]. The proposed device can be attached to the walls or ceilings of these places—similar to smoke detectors—and they can notify the police as soon as a gunshot is fired. The proposed system will help to stop the shooter early and the injured people can be taken to the hospital quickly, thus more lives can be saved.

- Individuals affected by gun violence, including witnesses, bystanders, and neighbors, may undergo feelings of stress, depression, anxiety, and post-traumatic stress disorder (PTSD). More than 5% of America's children have witnessed a shooting and it causes them psychological distress [6]. The proposed device can lead the police to the crime scene as soon as possible and can give peace of mind.
- Solely in 2010, emergency rooms received 36,000 firearm assault victims, with 25,000 requiring hospital admission, resulting in a staggering USD 630 million in medical expenses. The overall economic impact of gun violence on the American economy is estimated to be at least USD 229 billion annually [6]. The proposed device will bring the police quickly and reduce medical costs and property damage.

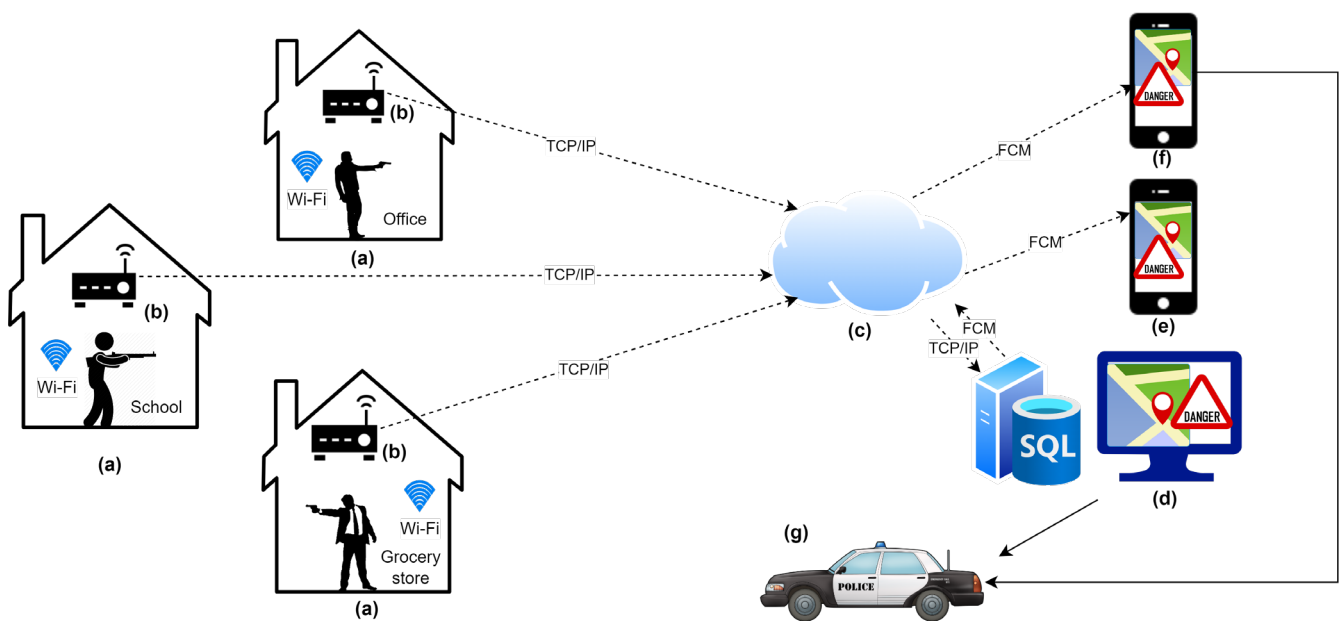


Figure 1. Crime scene (a) where the shooting happened. The gunshot detector device (b) is connected to the Wi-Fi of the building. It detects gunshot sounds and sends data to the central server through the Internet (c) using a Transmission Control Protocol/Internet Protocol (TCP/IP) protocol. The crime scene location is marked on the map (d), event data are saved in the Structured Query Language (SQL) database, and the software sends notifications using Firebase Cloud Messaging (FCM) to the user's smartphone app (e) and the emergency responder's smartphone app (f). The emergency responder's car (g) is dispatched.

The rest of this paper is organized as follows. The related works and comparison with other works are discussed in Section 2. In Section 3, materials and methods are discussed for the custom deep learning model for gunshot detection, and prototype system architecture consisting of the device, central server, and smartphone apps. Results of the deep learning model training, validation, and testing as well as the performance of the prototype system are discussed in Section 4. In Section 5, the discussion and future works are presented. Finally, Section 6 presents the conclusion.

2. Related Work

Some cities utilize companies such as SoundThinking (formerly ShotSpotter) (Fremont, CA, USA) [7] to detect and localize gunshots on a large scale. In this system, sensor modules are installed around the city in outdoor places—it is not used for indoor crimes. Moreover, these systems are extremely expensive to run and maintain, and gunshot detection involves both automated and manual human analysis. This system costs up to USD 90,000 annually per square mile of coverage. This system is installed by the city authority and not by individuals or institutions for personal use. To reap the benefits of this system, a user needs

to move to one of the cities where this system is implemented, which is an overhead. In this paper, the proposed gunshot detection device is mainly targeted for indoor gunshot detection and any user can place it in the rooms of a building. The hardware cost of the proposed device is approximately USD 300.

In ref. [8], the method for identifying shotgun blasts relies on the detection of the distinctive muzzle blast signature. The authors developed a specialized filter tailored to recognize shotgun muzzle blasts from the digitized audio signals. The work in ref. [9] proposes a gunshot noise detection method using Zero-Phase Technique. The works in ref. [8,9] process the signals in the time domain instead of using state-of-the-art machine learning techniques. The accuracy of their approaches was not reported and no hardware implantation results are presented.

In ref. [10], the authors extracted 11 features from each signal—from a dataset of both gunshot and non-gunshot instances—and used them as input to a neural network for classification. This study employs a neural network with a default MATLAB implementation, featuring one hidden layer containing 10 neurons. The authors report a precision of 69.3%. However, most deep learning approaches with many hidden layers produce better accuracy and precision.

The authors in ref. [11] use a semi-supervised Non-negative Matrix Factorization (NMF) approach, composed of training and separation stages, to detect gunshots. The result shows that the maximum true positive (TP) was 50% when signal-to-noise ratios (SNR) were 5 dB. No hardware implantation results are presented.

The authors in ref. [12] implement two parallel Gaussian Mixture Modelling (GMM) classifiers for discriminating screams from noise and gunshots from noise. Different audio features are used to train the classifiers. The authors report a precision of 93% to detect events when the SNR is 10 dB. Embedded system implantation and notification systems are not included in the work.

In ref. [13], the authors propose a gunshot event recognition system based on audio and visual features fed into a support vector machine (SVM) classifier. The authors developed a semantic gunshot scene description from video sequences by incorporating gunshot sounds, human emotion, and human activity analysis. The maximum precision reported for gunshots is 73.46%.

The work in ref. [14] uses two Artificial Neural Networks (ANN) to detect muzzle blasts and shockwaves from the gunshot sound. A gunshot is recognized if both the muzzle blast and shockwave are identified. A band-pass filter is used to remove undesirable frequencies from the gunshot sound. Then, spatial and frequency domain features are extracted and fed to the ANNs. Each ANN contains only one hidden layer. The system implements an array of four omnidirectional microphones, connected to a commercial data acquisition (DAQ) recording system. MATLAB is then used to analyze and classify the signal. The authors report a 99% accuracy in classifying M16 gunshots from background noise.

In the work of ref. [15], convolutional neural network (CNN) models such as VGG16, InceptionV3, and ResNet18 are trained with transfer learning for gunshot detection. Mel Frequency Cepstral Coefficient (MFCC) features are generated from the audio signals and then fed into the CNN models. An accuracy over 99% is reported for the ResNet18 model for their dataset. The hardware implantation results and notification systems are not presented in the paper.

The work in ref. [16] uses a custom CNN model for gunshot classification. Spectrograms are generated from audio signals and fed to the CNN model. The proposed model reports an accuracy of over 99% for their custom dataset. The model is implemented in a low-cost hardware system consisting of a USB microphone, Raspberry Pi board, and a short message service (SMS) modem. When a gunshot is detected, the system sends an SMS alert message to a fixed list of phone numbers. However, the system does not include custom user and device configuration using a smartphone app and plotting the location on a map.

A comparison with published related works is shown in Table 1. Here, we see that a CNN-based deep learning model is trained with the largest dataset in the proposed work, and it has high accuracy and precision. It should be noted that achieving high accuracy on a large dataset indicates that the model is generalized well without overfitting. The proposed system is implemented in a Jetson Nano-based embedded system which contains GPU and TensorRT engine for fast inferencing, whereas the work in [16] uses Raspberry Pi that neither has GPU nor TensorRT support, resulting in slower inferencing. As the proposed device is connected to Wi-Fi, it can obtain the date and time and record the gunshot sounds with timestamps for post-crime analysis. In this work, a complete gunshot detection system is implemented consisting of a gunshot detector device; a central server having an SQL database, plotting on a map, data searching, and push notification sending capabilities; and two smartphone apps—for the user and the emergency responders. Users and devices can be configured using smartphone apps considering possible many-to-many relationships. As soon as the gunshot event happens, the smartphone apps receive real-time push notifications with location information plotted in Google Maps.

Table 1. Comparison with other works.

	M. Hrabina, et al. [10]	J. Morillas et al. [11]	G. Valenzise et al. [12]	C. Chen et al. [13]	C. Galangue et al. [14]	J. Bajzik et al. [15]	A. Morehead et al. [16]	Proposed
Classifier	ANN	NMF	GMM	SVM	ANN	CNN	CNN	CNN
Dataset size	11,004	215	-	459 videos	917	7000	<90,000	670,000
Accuracy	-	-	-	-	99%	99%	99%	98%
Precision	69.3%	-	93%	73.46%	-	-	-	98%
Embedded system implementation	no	no	no	no	no	no	yes	yes
Record gunshot with timestamp	no	no	no	no	yes	no	no	yes
Plot on map	no	no	no	no	no	no	no	yes
User and device configuration	no	no	no	no	no	no	no	yes
Database implementation	no	no	no	no	no	no	no	yes
Smartphone notification	no	no	no	no	no	no	yes (using SMS)	yes

3. Materials and Methods

In this work, a deep-learning-based classifier is designed for the detection of gunshot sounds. Subsequently, the system depicted in Figure 1 is constructed.

3.1. A Deep Learning Model for Gunshot Detection

3.1.1. Dataset Generation

This project has generated a substantial dataset consisting of 670,000 sound samples, each with a duration of one second. This dataset comprises two distinct classes: ‘gunshot’ and ‘other’. The ‘other’ sounds are any sound other than gunshot sounds, i.e., they are non-gunshot sounds. To classify a sound as a ‘gunshot’ or ‘other’ class, the deep learning model needs to be trained with examples of both classes of sounds. In the developed dataset, each class has 335,000 samples. The gunshot sounds were collected from different online sources such as the BGG dataset [17,18], the Free Firearm Sound Effects Library [19], the Gunshot audio dataset [20,21], the Gunshot Audio Forensics Dataset [22], the Gunshot/Gunfire Audio Dataset [23,24], and gunshot sounds from the Urbansound8k Dataset [25,26]. The ‘other’ sounds were also collected from online sources such as the Urbansound8k Dataset [25,26], the ESC-50 Dataset [27,28], the FSD50K dataset [29,30], and the snoring dataset [31,32].

The collected gunshot WAV audio files had a collection of gunshots from different guns such as AK-12, AK-47, IMI Desert Eagle, M4, M16, M249, MG-42, MP5, and Zastava

M92. These audio files had different durations. Moreover, these files often contained silence, human talking, environmental sounds, bullet shell falling and trailing sound after a gunshot, etc., for more than one second. These sounds must be removed from the gunshot class samples to make a high-quality dataset and to increase the accuracy of the trained model. To do that, all these files were first joined using WavePad Sound Editor [33]. Then, using the silence threshold function of the sound editor software, silences are removed. However, it was found that this method is not fully accurate and some silences were still there. Then, the joined file was split into equal-sized one-second duration files. As the joined file was not evenly divisible by one second, the last split file was deleted. We define a valid one-second gunshot sound file that contains at least one starting of a gunshot sound at any place in the one-second duration. To automatically remove silent sound files, a Python code was written that reads the maximum absolute amplitude of each WAV file, compares it with a threshold, and removes the files if it is below the threshold. However, this time-domain approach is still not so accurate to filter out all the silences. To remove the unwanted sounds, all the one-second gunshot sound files, approximately 20,000 samples, were manually heard by humans and filtered. This manual effort is necessary to make the dataset a high-quality dataset and to reduce false alarms from the trained model. As our goal is to develop a gunshot detector embedded system hardware with a microphone, we would like to make the dataset that is generated from the same microphone that will be used in the embedded system. To do that the one-second gunshot sounds were merged into a single file, played from a computer, recorded in another computer using the same microphone that will be used in the embedded system, and then split the recorded file into one-second files.

Data augmentation in deep learning for sound is a technique where we create new training examples by slightly modifying the existing ones. This helps the model learn more effectively from the data it has. For sound, it means changing aspects such as pitch and speed; or adding background noise to audio recordings. By doing this, we provide the model with a broader range of examples, making it better at recognizing different variations in sound in real-world scenarios, ultimately improving its performance in tasks such as speech recognition or sound classification. In this project, the audio data augmentation [34] library is used to generate more sound samples. For each sample, 20 additional augmented samples were generated by: slightly shifting the signal to the left and right in the time axis, changing the tone, and the speed. The empty places were filled with silence in those files.

The collected 'other' files contain non-gunshot sounds such as silence, mild noise, clock ticking, the door opening and closing, toilet flushing, the siren of an emergency vehicle, rain, streetcar, people talking, baby crying, animal voices, washing machine, and vacuum cleaner. This dataset contains possible false alarms for gunshots such as fireworks, can opening, door knocking, glass breaking, clapping, drums, and thunderstorms [25–32]. Gunshot sounds from some of these datasets were identified from the dataset metadata and automatically removed using Python scripts. These audio files had different durations. To make one-second files: all these files were first joined, and then the joined file was split into equal-sized one-second duration files. The last split file was deleted as the joined file was not evenly divisible by one second.

3.1.2. Normalization and Feature Extraction

The audio files are then normalized with the min–max normalization method using the Pydub library [35]. The stereo audio samples are made single channel audio by taking only the left channel data. Then, feature extraction is performed by converting the time-domain audio signal to the Mel Frequency Cepstral Coefficients (MFCCs) [36,37]. The main principle behind MFCC is to condense essential information into a concise set of coefficients, inspired by the human ear's auditory perception. To compute the MFCC, the time-domain audio signal is first partitioned into frames lasting 20–40 milliseconds each. For each frame, the power spectrum is determined. Subsequently, triangular-shaped Mel filter banks are computed and applied to the power spectra, generating a spectrogram. Notably,

the human ear exhibits superior sensitivity to subtle pitch changes in lower frequencies (below 1 kHz) compared to higher frequencies. To account for this sensitivity discrepancy, the first ten filters in the Mel filter bank are linearly spaced approximately 100, 200, ..., and 1000 Hz. Beyond 1 kHz, these filters are distributed according to the logarithmic Mel scale. Following this, the logarithm of all filter bank energies is calculated, and their discrete cosine transform (DCT) is performed to decorrelate the filter bank coefficients. This process effectively captures the salient characteristics of the sound, making it suitable for subsequent sound classification tasks.

In this project, the sound sample is segmented into frames lasting 30 milliseconds each. To extract meaningful features, the MFCCs are computed using the SpeechPy [38] library. For this purpose, 32 filters are employed in the filter bank, and the Fast Fourier Transform (FFT) is applied with 512 points. The resulting MFCC representation consists of 32 cepstral coefficients. In Figure 2, we can observe examples of a gunshot sample and a non-gunshot sample presented both in the time domain and their corresponding MFCC representations. Remarkably, after applying MFCC calculations, the one-dimensional time-domain sound signal is transformed into a two-dimensional signal with a size of 32×32 , effectively resembling an image. Leveraging this MFCC-based image representation, we can employ image classification deep learning architectures, such as convolutional neural networks (CNNs), to classify the sample images. Consequently, a dataset comprising 670,000 MFCC images is assembled from the sound samples, serving as the input for the deep learning network during the classification process. The MFCC data and their associated class labels are then randomly shuffled maintaining the correspondence between data and label.

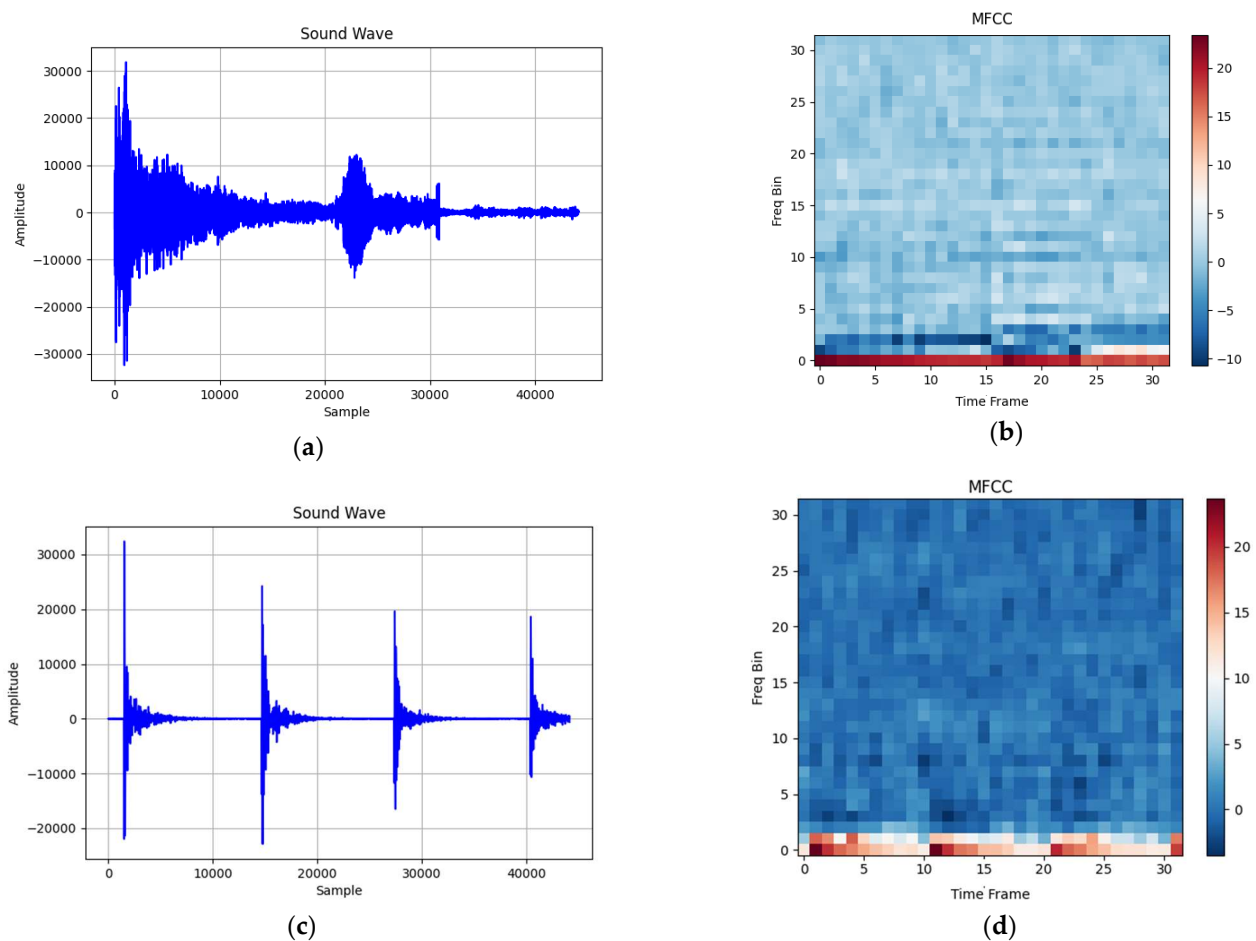


Figure 2. (a) Time-domain gunshot sound; (b) MFCC of the gunshot sound in (a); (c) time-domain non-gunshot sound of door knocking; (d) MFCC of the non-gunshot sound in (c).

3.1.3. Convolutional Neural Network Architecture

Several iterations are made to find a model that can fit the dataset well and also has sufficient capacity (i.e., parameters) to avoid overfitting. To find the right model capacity, first training is performed using a model with large parameters. The batch size is set as large as possible until memory issues are reported. Then, gradually the model capacity is reduced until fitting becomes difficult. The learning rate and learning rate decay parameters were reduced when the validation loss was not decreasing for a long-time during training. Finally, a deep learning model, as shown in Figure 3, is used to classify the sound as gunshot or as other. In this project, we convert the sounds to images as discussed in Section 3.1.2. and shown in Figure 2. The one-dimensional time-domain sound samples are converted to two-dimensional frequency domain MFCC image representation. These MFCC images are used as input of the deep-learning-based classifier to classify the MFCC images, which are actually representations of sounds. The different layers and the optimizer of the model are briefly described below.

- **MFCC Input Image:** The MFCC image is structured as a tensor with dimensions of (32, 32, 1). To ensure compatibility with deep learning models, the pixel data type undergoes conversion to a floating-point format. For the purpose of pixel value normalization, we calculate and store the mean and standard deviation of the training dataset in a separate file. Subsequently, each pixel value in all dataset images is normalized by subtracting the mean and dividing by the standard deviation.
- **The Convolutional Layer:** A 2-D convolutional layer applies sliding convolutional filters to the input data. It conducts convolution by sliding these filters along both the vertical and horizontal axes, computing the dot product between the weights and input, and then adding a bias term [39]. The proposed model incorporates two convolutional layers, each utilizing 3×3 filters. These filters are initialized with random values and function as learnable network parameters. For example, in Figure 3, the conv2d layer comprises 2 filters, each sized 3×3 with padding, resulting in 2 output layers having the same height and width as the input layer. Similarly, the conv2d_1 layer includes 2 filters, each sized 3×3 with padding.
- **The Activation Layer:** Non-linear activation functions, specifically the rectified linear unit (ReLU) [40], are applied after the convolutional and dense layers (except the last dense layer). The ReLU layer performs a threshold operation on each element, setting any value less than zero to zero. This activation function introduces non-linearity into the network, enabling it to capture intricate patterns and enhance its classification performance.
- **The Max Pooling Layer:** This layer conducts down-sampling by partitioning. Here, the input is divided into 2×2 size rectangular regions and the maximum number from each region is sampled [41].
- **The Flatten Layer:** The spatial dimensions of the input are collapsed here, and it converts it into a one-dimensional vector. It transforms the input tensor having dimensions of (8, 8, 1) into a 64 size single-dimensional vector.
- **The Dense Layer:** The dot product between the input and a weight matrix is computed in this layer. Following this, a bias vector is added to the result [42,43]. Random values are used to initialize both the weight matrix and bias, and they serve as learnable parameters of the model. This layer is also known as the fully connected (FC) layer.
- **Loss Function and Optimizer:** The last fully connected layer, dense_2, integrates the extracted features for image classification. Consequently, the last dense layer output size is set to one for binary classification. Then a score function, Sigmoid [44], is used. The agreement between predicted scores and the ground truth labels are quantified by the loss function. The job of the optimizer is to find the global minima by varying the network parameters. Global minima can be achieved when the loss function reaches its minimum value. In the proposed model, the binary_crossentropy loss is computed, and the RMSprop [45] optimizer is utilized.

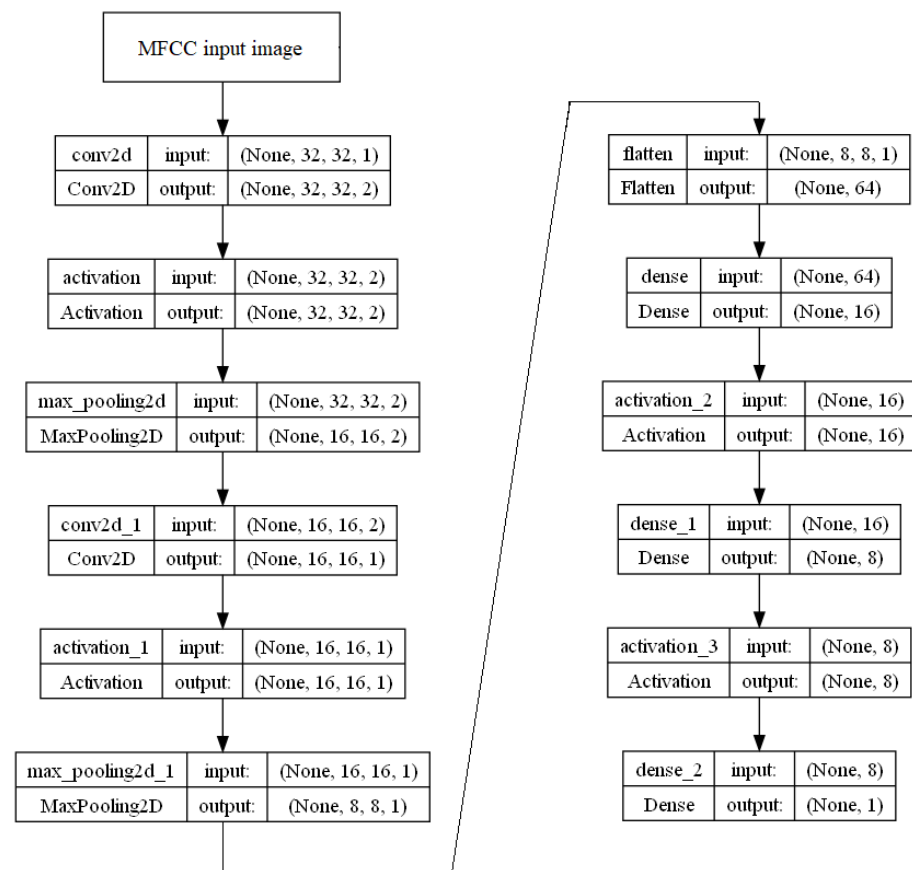


Figure 3. The architecture for the convolutional neural network.

3.1.4. Training the Deep Learning Model

The dataset of 670,000 sound samples was converted to MFCC images, and it was then divided into three distinct subsets: 70% of images (i.e., 469,000) were allocated for training, 15% of images (i.e., 100,500) for validation, and 15% of images (i.e., 100,500) were reserved for testing. The latter set was withheld until after the model had undergone training and validation, allowing for a final assessment of model accuracy using previously unseen test samples. The mean and the standard deviation of the training dataset images are calculated and saved in the norm.npy file. Then, from all three data subsets, the mean was subtracted and then divided by the standard deviation to normalize the dataset.

The deep learning architecture, as illustrated in Figure 3, was implemented using the Python programming language with the Keras library. Keras, a high-level neural networks application programming interface (API) built upon TensorFlow [46], was employed for its versatility and user-friendly interface. Model training was carried out on a desktop computer featuring a 12th Gen Intel Core i7 processor (6 Cores) clocked at 2.10 GHz, 32 GB of RAM, and an NVIDIA GeForce RTX3070 graphics processing unit (GPU).

After training the CNN model, an H5 model is generated that can be used for inferencing. To reduce the inference time of the H5 model on the Jetson Nano, NVIDIA-TensorRT [47,48] is used to convert the model to a TRT engine. TensorRT encompasses a deep learning inference optimizer and runtime, optimizing deep learning inference applications for minimal latency and enhanced throughput. It offers INT8 and FP16 optimizations, wherein reduced precision substantially diminishes inference latency.

3.2. Prototype System Architecture

The gunshot detection system architecture comprising the gunshot detector device, the central server for the emergency responder's station, and smartphone apps for users and emergency responders—as shown in Figure 1—is designed and developed. The user

places the device in a room and then uses the smartphone app to configure the Wi-Fi of the device, and also to update the user and the device information to the central server. Emergency responders use a smartphone app to update their information on the server. Once the configuration is performed, the user and the emergency responders are ready to receive smartphone notifications to any place in the world as long as there is Internet coverage. A concise overview of the various system modules is provided below.

3.2.1. Gunshot Detector Device

The gunshot detector device listens to the sounds in the environment and classifies it as gunshot or other. If a gunshot is detected—it sends data to the central server through the Internet using Transmission Control Protocol/Internet Protocol (TCP/IP) protocol and also saves the gunshot sound files in the device locally. The device is configured with the developed smartphone app. A brief description of the device's hardware and firmware follows.

Hardware

The block diagram of the hardware unit of the gunshot detector device is shown in Figure 4. The primary processing unit employed is the NVIDIA® Jetson Nano™ Developer Kit [49], a single-board computer known for its compact size and energy efficiency. This embedded platform excels in running neural network models effectively, such as image classification, object detection, segmentation, and more. The Jetson Nano™ Developer Kit is equipped with a powerful Quad-core ARM A57 microprocessor running at 1.43 GHz, and 4 GB of RAM. Additionally, it has a 128-core Maxwell graphics processing unit (GPU), a micro SD card slot, USB ports, general purpose input/output (GPIO), and various integrated hardware peripherals. An omnidirectional microphone with a built-in sound card [50] is interfaced with the Jetson Nano using a USB. To connect with a smartphone using Bluetooth and to access the Internet wirelessly, a wireless Network Interface Card (NIC) supporting both Bluetooth and Wi-Fi [51] is connected to the M.2 socket of the Jetson Nano. An LED to indicate the program is running—referred to as the heartbeat LED—is interfaced with a GPIO pin of the Jetson Nano. For the power supply, a 110 V AC to 5 V 4A DC adapter is used. To keep the microprocessor cool, a cooling fan is employed with pulse width modulation (PWM)-based speed control, positioned above the microprocessor. The photograph of the gunshot detector device prototype is shown in Section 4.

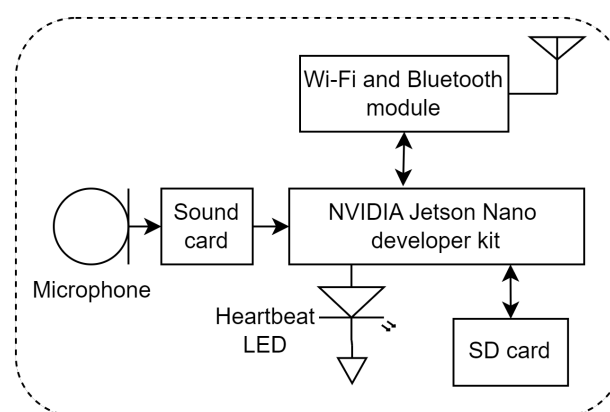


Figure 4. Block diagram of the gunshot detector device hardware.

Firmware

The Jetson Nano board is equipped with a 64 GB SD card, hosting Bionic Beaver, which is a specialized version of the Ubuntu 18.04 operating system. The application software is developed using the Python language, and all required packages, including JetPack 4.6.3, are installed on the system. Three Python programs—to configure Wi-Fi, detect gunshots,

and access the recorded sounds—run in parallel in separate threads after the system boots. They are briefly described below.

Configure Wi-Fi: The purpose of this program is to configure the Wi-Fi connection of the device using the user's smartphone. After the booting, this program enables the Bluetooth advertisement [52] of the Jetson Nano, so that the device is visible to the user's smartphone when scanning for nearby Bluetooth devices. Here, the Jetson Nano works as a Bluetooth server, and the smartphone as a Bluetooth client. The program then waits for a Bluetooth connection from the client using a socket [53]. A timeout is used that will close the socket, disable Bluetooth advertising, and terminate the program if there is no connection request within 30 min after the boot. This shortening of the advertisement duration by timeout will prevent unwanted access to the device using Bluetooth. Once the smartphone connects with the device, the Bluetooth advertising is disabled and it waits to receive commands from the smartphone. The smartphone needs to know the Wi-Fi service set identifier (SSIDs) that are nearby to the device. When the smartphone sends a command to the device requesting the list of nearby SSIDs, the device generates the list using the nmcli tool for Linux devices [54] and sends the list to the smartphone. In the smartphone, the user can choose the desired Wi-Fi SSID the device should connect and enter the password. The smartphone then sends a command, which includes the SSID and password, to the device requesting to connect. Once the device receives the command for Wi-Fi connection, the device tries to connect with the requested SSID and then replies with the connected SSID and its' local IP address. After the Wi-Fi configuration is performed, the smartphone sends a performed command, and the device then closes the socket connection, enables advertising, and waits for a new Bluetooth connection up to the timeout.

Detect Gunshot: A flowchart of the gunshot detection firmware is shown in Figure 5. First, it captures a one-second sound having a sampling rate of 44,100 Hz [55]. To classify the sound, it is normalized using the min–max normalization method [35] and then the MFCC of the sound signal is calculated [38]. Then, reading from the norm.npy file, the mean is subtracted and then divided by the standard deviation. The signal is then classified using the generated TRT engine, as discussed in Section 3.1.4., as gunshot or other. The engine outputs the gunshot probability of the captured sound, thus a probability greater than or equal to 0.5 is classified as gunshot sound. The flag `isGunshot` is set to `True` if the sound is classified as a gunshot, and `False` otherwise. The heartbeat LED is turned on at the beginning of the classification, and it is turned off after the classification.

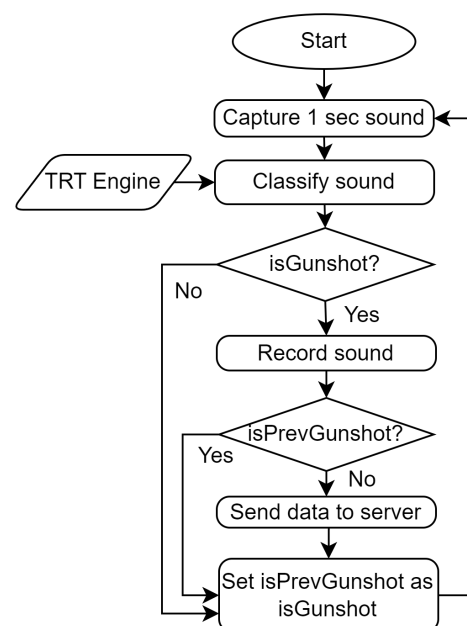


Figure 5. Flowchart of the gunshot detection firmware implemented in the microcontroller.

If a gunshot is detected, then the sound is saved as a WAV file [56] in the `rec_gs` folder in the SD card with the current date and time as the filename. As the device is connected to Wi-Fi, it can provide the correct date and time information [57]. To avoid continuous notifications when several gunshots are fired one after the other within a short time, the program sends only one notification to the server for the first captured sound and does not send any notification for the successive gunshot sounds until a non-gunshot sound is captured. The last detected sound status is saved in the `isPrevGunshot` flag. If `isGunshot` is True and `isPrevGunshot` is False, then the program tries to connect with the central server with TCP/IP protocol using socket [58] having a timeout of 5 s. In Python, a socket is similar to a communication endpoint that allows to send and receive data over a TCP/IP network, which is a set of rules for transmitting data between devices on the Internet or a local network. Once the IP address and port number of the target computer are specified, the socket can be used to establish a connection and exchange data. After connecting with the server, the device sends a data string containing the serial number of the device and the current date and time. The program reads the Bluetooth's media access control (MAC) address [52] of the Jetson Nano and it is used as the serial number of the device. Then, `isPrevGunshot` flag is updated with the `isGunshot` flag value and the process repeats.

Server for Accessing Recorded Sounds: To access and play the recorded gunshot sounds for post-crime analysis, a Hypertext Transfer Protocol (HTTP) server [59] runs in the device at port 8000 having the working directory as the `rec_gs` folder, where the gunshot sounds are saved. Thus, these files can be accessed and played by the user's smartphone using the local Internet Protocol (IP) of the device and the port number as long as the smartphone and the device are connected to the same Wi-Fi network. The smartphone obtains the local IP of the device when its Wi-Fi is configured.

3.2.2. Central Server Software

The central server, developed with Visual C# and Microsoft SQL Server [60], contains functionalities for plotting the gunshot event on the map, generating alerts, sending push notifications to smartphones, and querying the database using a graphical user interface (GUI). This server can be hosted on a computer in any institution, such as a school or office, where emergency responders can monitor the events.

SQL Database

The software implements an SQL database. The database tables, their fields, and their relationships are shown in Figure 6. In Figure 6, the primary key of each table is marked using a key sign on the left side of the field name; and the lines indicate relationships between a primary key field at the left and a foreign key field at the right.

The `user_tbl` table contains the customer or the user information such as name, address, email, and phone. The user's smartphone's Android ID [61] is used as the unique UserID. One user can have several smartphones and each smartphone app is treated as a separate user. To send notifications to the user's smartphone in the event of a gunshot, the Firebase Cloud Messaging (FCM) registration token [62] is stored in this table in the `FCMID` field. A unique Android ID and a unique FCM registration token are generated for each user when the person installs the smartphone app. The `device_tbl` table contains the device information. As discussed in Section 3.2.1. in Firmware the Bluetooth MAC address is used as the unique device serial number and it is stored in the `DeviceSN` field. The location information of the device such as latitude and longitude, address, floor, and room are stored in this table so that emergency responders can quickly go to the place where the gunshot is detected. Users can assign a nickname to the device and it is stored in the `Name` field. The local IP of the device is stored in the `IP` field. The `user_device_tbl` connects the users and the devices. One user can have multiple devices installed, such as in several rooms in a building. One device can have multiple users, such as each family member in a home. Thus, there could be a many-to-many relationship between users and devices. Each row of this table connects a user with a device using the UserID and DeviceSN namely. The

ID field of this table is an autoincrement primary key field. The er_tbl table contains a list of emergency responder's information such as ERID, FCMID, name, address, email, and phone. Similar to the user table, ERID stores the Android ID, and the FCMID stores the FCM registration token. The event_data_tbl table contains information on each gunshot event such as the serial number of the device where the gunshot is detected, date, time, and location information of the device. This table keeps track of all the gunshot events and can be used for querying data.

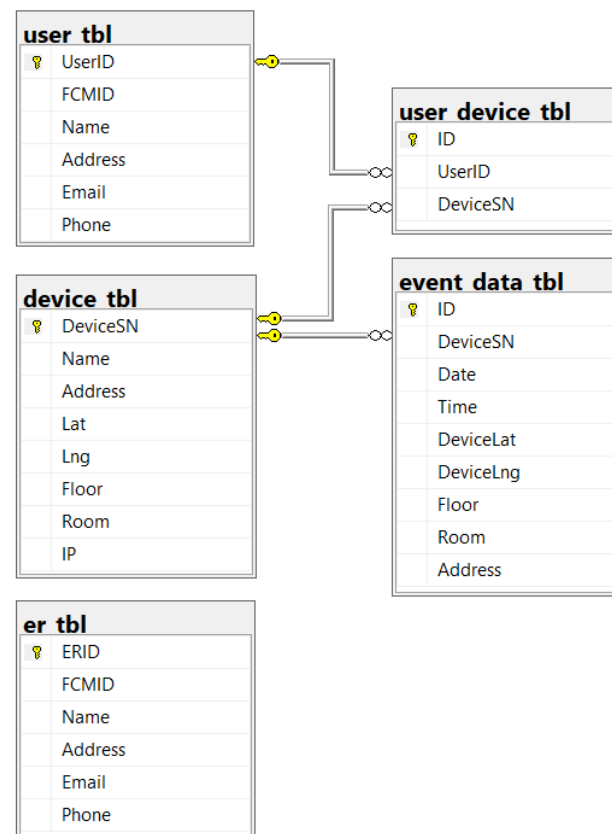


Figure 6. Tables, fields, and relationships of the database. The primary key of each table is marked using a key sign on the left side of the field name.

Data Processing in TCP Server

The central server implements a Transmission Control Protocol (TCP) server [63] and listens at port 8050. Connecting the gunshot detector devices or smartphones to this server necessitates a stable public IP address and an accessible port. The router's public IP, provided by the Internet service provider (ISP), typically remains constant and serves as the fixed public IP. To facilitate the transmission of incoming data packets from the Internet to our custom TCP server port, we set a static local IP for the server computer and configure port forwarding [64] on the router. Additionally, the port number is opened in the Firewall settings [65]. The TCP server receives user and device configuration data, emergency responder configuration data from smartphones; and gunshot notification data from gunshot detector devices. The first byte of the data indicates whether it is user and device configuration data, emergency responder configuration data, or gunshot notification data. The handling of these three kinds of data is briefly described below.

User and device configuration data: The user and device configuration data string contains: each field value of the user_tbl; the total number of devices; and each field value of the device_tbl for each device. Each field value is separated by a vertical line character, |, instead of a comma because a comma could be part of the 'address' field of a user. When the data arrive at the server: the data are parsed, saved in variables, and stored in the

database tables. If the UserID already exists in the user_tbl, then that user information is edited by updating the row with the data; otherwise, a new user is added by inserting a new row in the table. SQL queries [66] are executed from the software by connecting to the database to accomplish these tasks. Then, for each device listed in the data, the DeviceID is checked in the device_tbl. If the DeviceID already exists in the device_tbl, then the device information is updated with the data; otherwise new device data are added to the table. After that, the user_device_tbl is updated to assign the devices to the user. First, all the rows containing the UserID of the user are deleted. Then, for each device listed in the data, the UserID and the DeviceSN are inserted as rows in the table. In this way, the assignment of devices with the user is maintained whenever the user adds, edits, or removes a device.

Emergency responder configuration data: This data string contains each field value of the er_tbl. Once the data arrive at the server: the data are parsed, saved in variables, and stored in the database table. The emergency responder's information is updated if the ERID already exists in the er_tbl, and a new emergency responder is added if the ERID does not exist in that table.

Gunshot notification data: These data arrive at the server from the gunshot detector device when a gunshot event is detected. It contains the DeviceSN, and the event date and time. After the data arrive at the server: the location information of the device is queried from the device_tbl using the DeviceSN; a new row is inserted in the event_data_tbl to save the event information in the database; plotted on the map with a marker [67]; gunshot detection message is displayed; a warning sound is generated; FCM push notifications [68] are sent to the smartphones of the users of that device and all the emergency responders. To send the push notifications to each user assigned to the device: the FCM registration tokens for each user of the device are gathered from the user_device_tbl and user_tbl using a multiple table query. Each push notification contains the DeviceSN, the location information of the device, and the event date and time.

Searching Gunshot Events

The software implements a GUI where the user can choose a range of dates and times, a rectangular area on the map, or both, to search for gunshot events. An SQL query is made based on the chosen criteria and the result data are retrieved from the database. Then, the gunshot events from the result data are plotted on the map and the associated location and user information are displayed.

3.2.3. Smartphone App

Two smartphone apps are developed for the Android platform: for users and emergency responders. These apps contain a settings window where the user's or emergency responder's information, as shown in user_tbl and er_tbl namely in Figure 6, can be entered. The UserID and the ERID, which are the unique Android IDs [61] of the smartphones, and the FCMID, which is the FCM registration token [62], are assigned automatically without manual input.

The main difference between these two apps is that the user app contains options for configuring their devices, whereas the emergency responder app does not contain options for device configuration as they are not users of any device. The setting window contains a custom list view that shows the list of devices the user has. New devices can be added and existing devices can be edited or removed from here. The properties of these devices, as shown in device_tbl in Figure 6, can be updated by selecting the device. To make the device location input process easier: the smartphone can be placed near the device, and the GPS location and address information can be retrieved automatically using the GeoLocation [69] library. The Wi-Fi configuration of the device, as discussed in Section 3.2.1 in Firmware, is implemented with GUI in the app. It contains a window where nearby Bluetooth devices can be searched and connections can be made. The device must be paired with the smartphone before connection. After connecting: the Bluetooth MAC address is assigned as the DeviceSN, the list of available Wi-Fi SSIDs is retrieved from the device and

shown in the app, and the user can choose the desired SSID and provide a password—as discussed in Section 3.2.1 in Firmware. When the app leaves the setting window: the smartphone connects with the central server using the Internet as a client and sends the configuration data using a socket that updates the database in the server.

Once the Wi-Fi of the device is configured, the smartphone app obtains the local IP of the device. Using the local IP and the HTTP server port of the device, the gunshot sounds recorded in the device can be accessed and played from the smartphone.

The first screen of these apps contains a list of gunshot events showing the device name and serial number; its location; and the date and time of the event. These apps are registered in the FCM [70] dashboard for receiving push notifications. In this application, there is a background service called FirebaseMessaging [71]. When this service receives a push notification message from Firebase Cloud Messaging (FCM), it triggers a callback function. Subsequently, the app performs several actions, including adding the received message to a list, saving the list to a file, generating a smartphone notification, and updating the list view on the screen. If the user clicks on any item in the list view, the application opens Google Maps, setting the destination to the gunshot detector device's current location. This feature enables the user or an emergency responder to navigate to the site promptly.

4. Results

4.1. Gunshot Detection Deep Learning Model Results

The CNN model, as discussed in Section 3.1.3, is trained and validated simultaneously until the validation loss is smaller than or equal to 0.05, or for 5000 epochs—whichever is reached first. The training and validation batch size is set to 2048. The learning rate and the learning rate decay are set to 1×10^{-6} and 1×10^{-7} , respectively. Graphs illustrating the trends of loss vs. epochs and accuracy vs. epochs for both the training and validation datasets are presented in Figure 7a and 7b, respectively. These plots vividly depict a consistent decline in loss and a corresponding increase in accuracy as the number of epochs progresses. Upon reaching the 1501 epoch mark, the model achieved a validation loss of 0.05 and stopped after 1 h 4 min, and 22 s of training. Remarkably, both the training and validation datasets demonstrated an accuracy of approximately 0.98 after the 1501 epoch training phase.

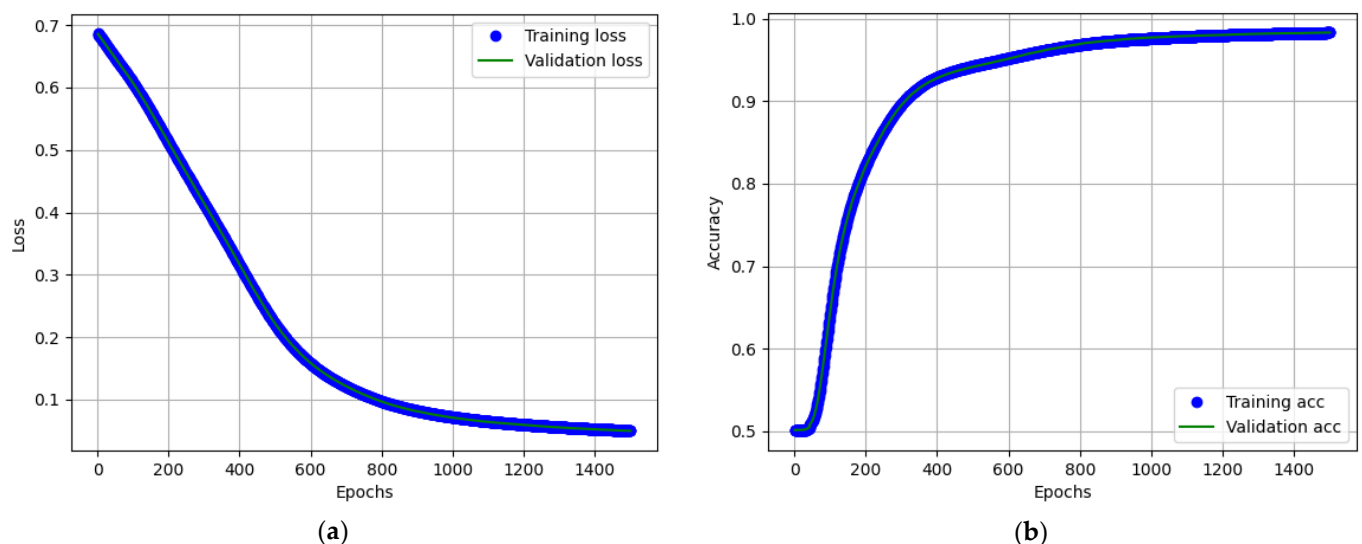


Figure 7. (a) Loss vs. epochs for training and validation datasets; (b) accuracy vs. epochs for training and validation datasets.

After completing the training and validation phases, the model, comprising a total of 1224 learned parameters (including filters, weights, and biases), was saved in an H5 file. The model's disk size amounted to 62.66 kB. Subsequently, the model underwent

testing using an unseen test set containing 100,500 samples. During this testing phase, the model achieved a loss of 0.0505 and an accuracy of 0.98. Table 2 provides an overview of the loss and accuracy values for the training, validation, and test datasets, highlighting that the model exhibits similar accuracy across all sets, indicating its robust generalization. Figure 8 illustrates the confusion matrix for the test dataset, while Table 3 presents the precision, recall, and f1-scores for the test dataset. The proposed deep learning model achieves an accuracy of 98% and 2% of the data are misclassified. The reason for this 2% misclassification could be that some sounds might be affected by variations in recording conditions, or different speakers, making it challenging for the model to generalize across diverse inputs. The choice of hyperparameters and architecture might not be perfectly suited for some ambiguous sounds, potentially leading to misclassification. The inherent ambiguity in certain sound patterns or overlapping acoustic features can pose difficulties for any model, limiting its accuracy to less than 100%. Thus, achieving 100% accuracy with CNNs is difficult for a large dataset due to these inherent challenges in the data and modelling.

Table 2. The loss and accuracy of the training, validation, and test datasets.

	Training	Validation	Test
Loss	0.0504	0.0500	0.0505
Accuracy	0.9831	0.9832	0.9828

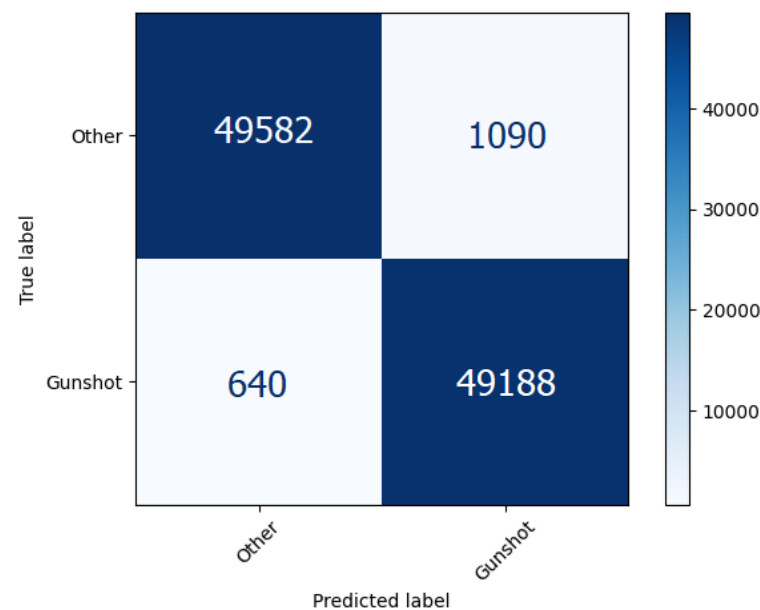


Figure 8. Confusion matrix of the test dataset.

Table 3. The precision, recall, and f1-scores of the test dataset.

	Precision	Recall	f1-Score
Gunshot	0.98	0.99	0.98
Other	0.99	0.98	0.98

4.2. Prototype System Results

A prototype of the proposed system comprising the gunshot detector device, the central server for the emergency responder's station, and smartphone apps has been developed and tested successfully. A photograph of the gunshot detector device, labeling different parts, is shown in Figure 9a. The device enclosed in casing [72], having a dimension of

approximately $15.5 \times 12.3 \times 4$ cm, is shown in Figure 9b. The device is programmed according to the discussion in Section 3.2.1. in Firmware part and is configured to run the programs automatically on boot. On the Jetson Nano device, the average pre-processing time of one recorded sound which includes MFCC generation and normalization is 29 ms, and the inference time by the deep learning H5 model is 212 ms. However, after converting the H5 model to the TRT engine, as discussed in Section 3.1.4, the inference time by the TRT engine is reduced to only 3.9 ms, making the inferencing 54 times faster. The power consumption of different parts and the entire device is measured using the jetson-stats library [73] and shown in Table 4.

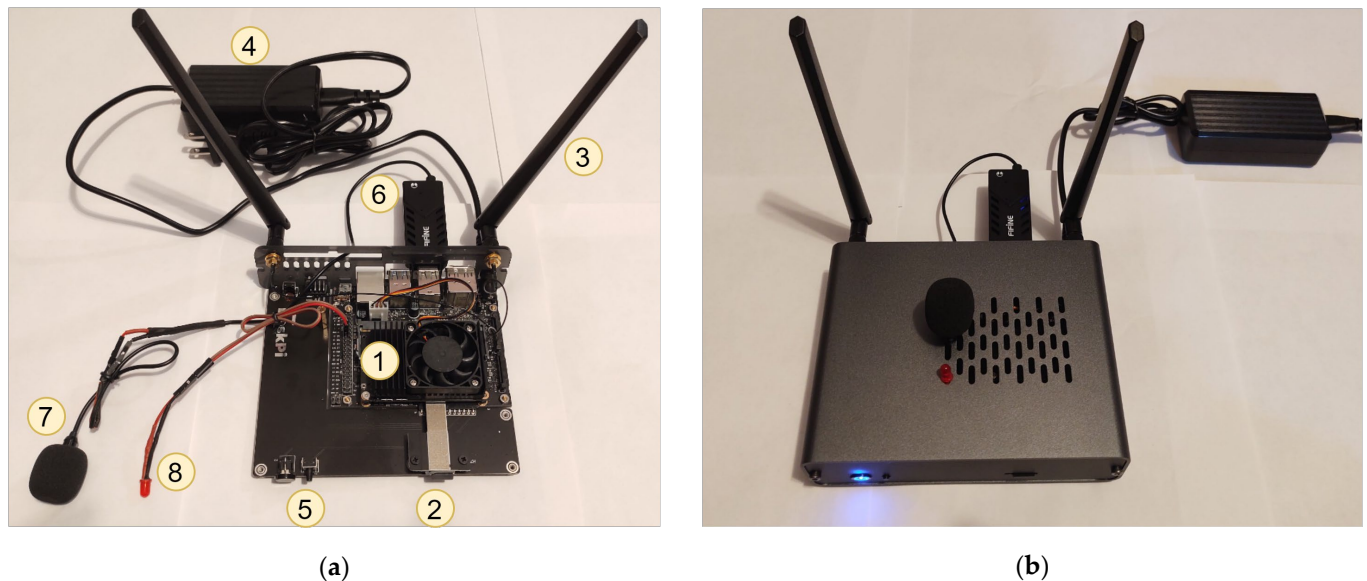


Figure 9. (a) Photograph of the gunshot detector device: (1) Jetson Nano with a wireless module and cooling fan; (2) SD card; (3) antenna; (4) DC adapter; (5) power and reset switches; (6) sound card; (7) microphone; (8) heartbeat LED. (b) Gunshot detector device enclosed in casing.

Table 4. Power consumption of the gunshot detector device.

Hardware Part	Power
Jetson Nano's CPU	854 mW
Jetson Nano's GPU	40 mW
Entire Device	2.7 W

After the device is powered up, the heartbeat LED starts to blink indicating the program is running and listening for sounds. The central server, as discussed in Section 3.2.2, was running on an Internet-connected computer. The system is then configured using the smartphone app—as discussed in Section 3.2.3. Some screenshots of the smartphone app for configuring the emergency responder, the user, and their device are shown in Figure 10. Using the app, a user and a device are added, and the Wi-Fi of the device is configured successfully. In the central server, the user and device information got updated as expected. Using the smartphone app designed for emergency responders, an emergency responder was also added to the system.

8:59

Settings

Config ER

ER ID: 07ed791e0cd77d33

Name: Bill James

Address: 505 W Michigan Ave, Ypsilanti, MI 48197

Email: bill@emergency.com

Phone: (734) 000-0000

Status: Connected with server.

(a)

6:00

Settings

Config User

User ID: 0207b1e42c61ae3e

Name: Tareq Khan

Address: 7863 Hampton Ct, Ypsilanti, MI 48197

Email: tareq.khan@emich.edu

Phone: (734) 756-0827

Config Device

Gym_GSD, S/N: 60:F2:62:B4:6F:94

Status: Connected with server.

(b)

6:01

Device Properties

Name: Gym_GSD S/N: 60:F2:62:B4:6F:94

Address: 7863 Hampton Ct, Ypsilanti Charter Twp, MI 48197, USA

Latitude: 42.1750144 Longitude: -83.6189435

Floor: 2 Room: Gym

IP: 192.168.1.34

(c)

6:02

Find Device using Bluetooth

MAC: 2A:2E:BA:84:32:46

GunshotDetector_01, MAC: 60:F2:62:B4:6F:94

[TV] Samsung 6 Series (55), MAC: C0:97:27:00:8C:5F

MAC: 7C:ED:21:8C:9A:2B

Connecting to GunshotDetector_01...

(d)

6:06

Setup Device WiFi

Tareq

Gobblue07

Tareq-5G

MyWifi

ZeusPalace

SSID: Tareq-5G

Password:

WiFi Connected: Tareq-5G, IP: 192.168.1.34

(e)

Figure 10. Screenshots of the smartphone apps: (a) emergency responder configuration in the emergency responder app; (b) user configuration and list of devices in the user app; (c) device properties configuration and buttons for its Wi-Fi setup and playing recorded gunshot sounds; (d) searching and connecting with the device using Bluetooth; (e) setup Wi-Fi SSID of the device.

The gunshot detector system was tested inside a lab environment, by playing recorded sounds near the device rather, than performing actual shootings to avoid the destruction of properties. During testing, different sounds other than gunshots were played and they were successfully detected as others. Then, gunshot sounds were played, sometimes mixing with other environmental noise and background sounds, and the device successfully detected the sound as a gunshot and notified the central server within a second. Upon receiving the notification data from the device, the central server: successfully marked the location of the gunshot event on the map, displayed the assigned user and device information in the event log, saved the event data in the database, generated warning sounds, and sent smartphone notifications to the assigned user and all the emergency responders. Some screenshots of the central server software and smartphone app after a gunshot event are shown in Figures 11 and 12 namely. The system was also tested with multiple emergency responders, multiple users, and devices with many-to-many relationships, and notifications were sent successfully as expected.

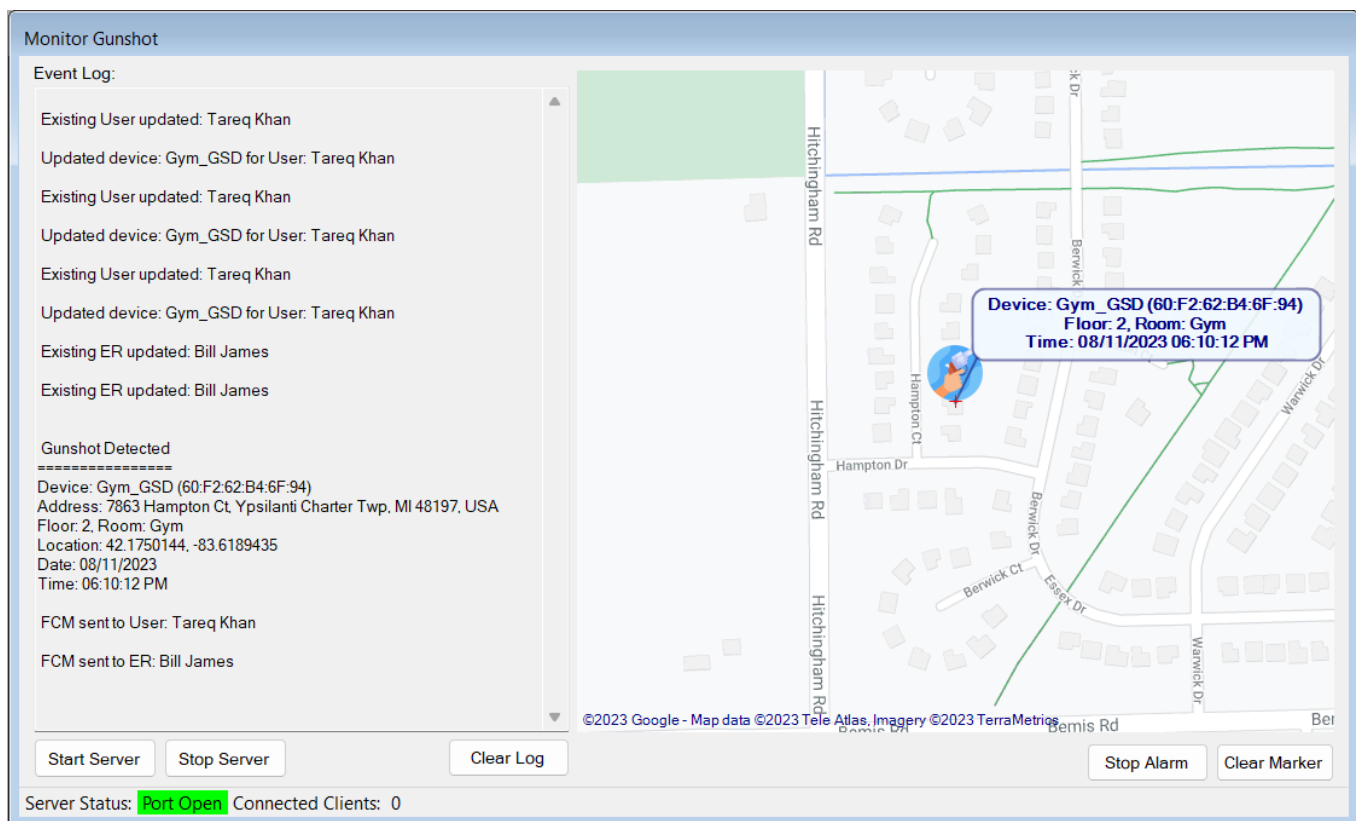


Figure 11. Screenshot of the central server software marking gunshot event on map (right) and displaying associated information in the event log (left).

In the central server, gunshot events can be successfully searched using a range of dates and times, a rectangular area on the map, or both. A screenshot of the searching gunshot event is shown in Figure 13.

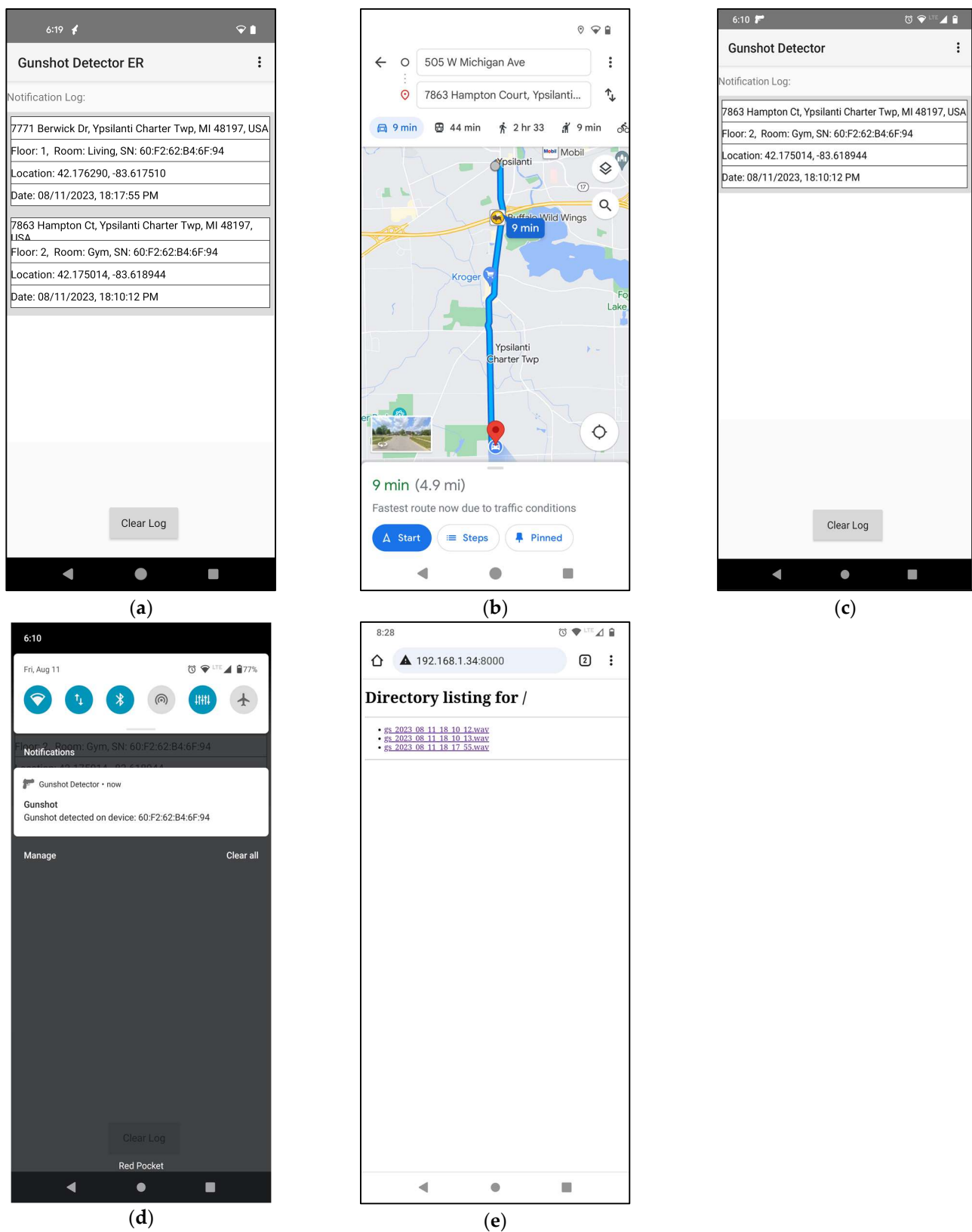


Figure 12. Screenshots of the smartphone apps: (a) emergency responder app's list view showing a list of gunshot events with location, date, and time; (b) clicking on the list item shows the direction to the gunshot event location on map; (c) user app's list view showing a list of gunshot events with location, date, and time; (d) smartphone notification when gunshot is detected; (e) accessing recorded gunshot sounds on the device from user's smartphone app.

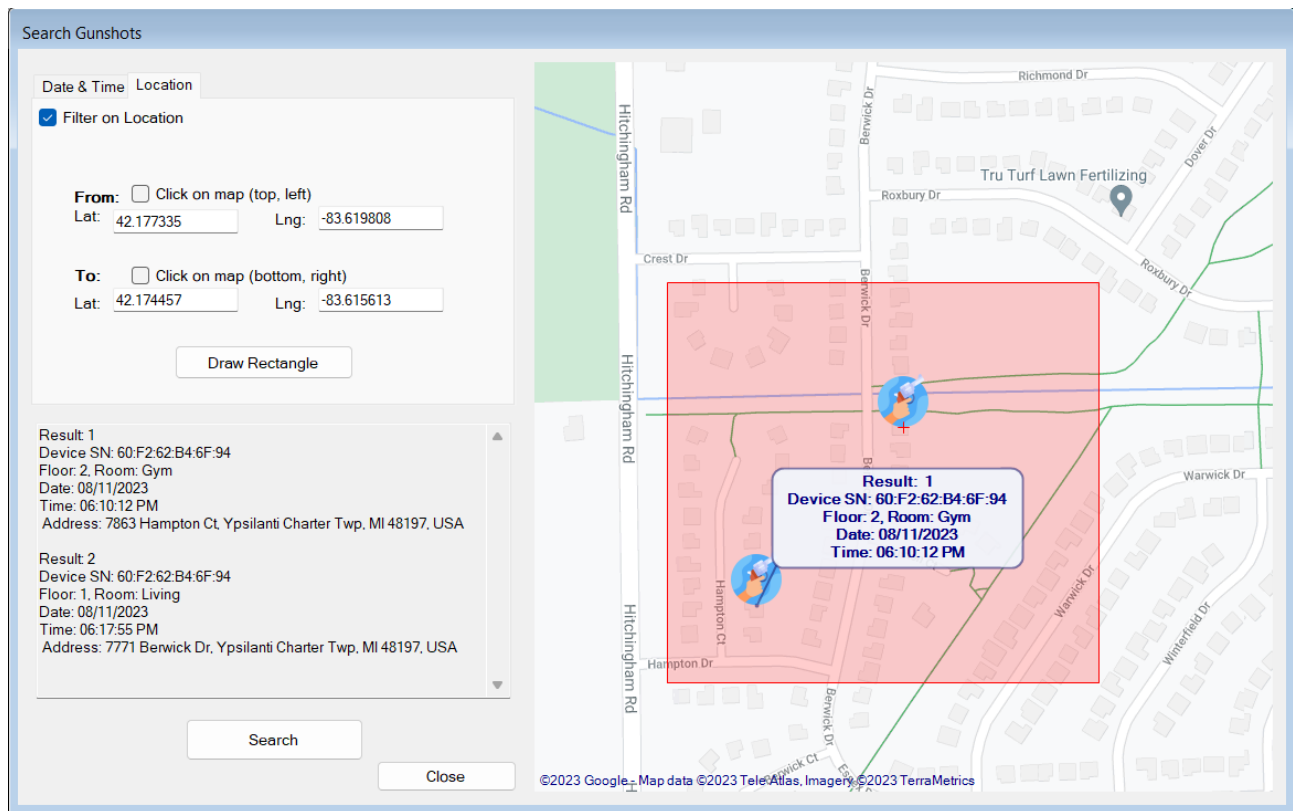


Figure 13. Screenshot of central server software for searching gunshot events based on a chosen rectangular area as shown in red, date and time, or both.

5. Discussion and Future Work

The proposed gunshot detection system is targeted for indoor use such as inside schools, grocery stores, and offices. Thus, possible false alarm sounds from outside, such as fireworks, may have less effect on this system. As discussed in Section 3.1.1, the other (i.e., the non-gunshot) dataset class contains possible false alarm sounds for gunshots such as fireworks, can opening, door knock, glass breaking, clapping, drum, and thunderstorms [25–32]. Thus, the deep learning model is already trained to classify these sounds as other sounds. To further protect the system from false alarms generated by fireworks, the probability threshold level of the classifier can be automatically increased when fireworks generally happen—for instance, on 4 July and the night of 31 December in the USA.

The proposed gunshot detection system can also be used in homes. However, the gunshot sounds generated by the people in the home from mobile, computer games, or movies can produce false alarms. To solve this problem, an app can be developed that will be installed on those mobiles and computers. The app will run in the background, read the sounds generated by the mobile or the computer, classify it as gunshot or other according to the proposed deep learning model, and then notify the gunshot detector device using Wi-Fi if a gunshot sound is detected in the mobile or in the computer. If the gunshot detector device detects a gunshot sound through its microphone and it also receives such notification from this app, it will then recognize the gunshot as a false alarm. We plan to implement this solution in the future.

A mischievous act could be that someone intentionally plays a gunshot sound using a mobile device near the detector to create a false alarm. This mischievous act can be encountered by interfacing an infrared sensor with the gunshot detector device. Infrared radiation, which includes wavelengths beyond the visible spectrum, is emitted when a gunshot is fired [74]. This phenomenon is primarily associated with the intense heat generated during the firing process. When a firearm is discharged, the rapid combustion of

gunpowder within the cartridge produces extremely high temperatures. This intense heat causes the surrounding air and the firearm's components, including the barrel, to heat up significantly. As a result, these hot objects emit infrared radiation, which can be detected by infrared sensors. Thus, if the gunshot detector device detects only sound without any significant increase in infrared radiation near it, then it will be considered a false alarm. We plan to implement a sensor fusion approach by interfacing an infrared sensor with the device in the future. To further increase accuracy and lower false alarms, we plan to implement object detection (such as guns and people) from camera images [75]. Moreover, the proposed sound classification method can be used to detect situations such as crying, glass breaking, and drone arrivals [76].

Data transmission from the device to the central server using sockets may introduce security vulnerabilities. The issue of handling security is planned to be implemented in future work. The detected gunshot sounds, saved as WAV files inside the device, are not transmitted to the central server. These files can only be accessed by the user's smartphone when he/she is on the same Wi-Fi network as the device. Thus, there is no privacy concern.

Due to safety reasons, the proposed gunshot detection system was tested by playing recorded gunshot sounds instead of actual shooting with firearms. We plan to test the system with actual guns inside a shooting range in the future. Moreover, if a silencer is used on a gun, then the generated sound will be different than the traditional gunshot sound. To detect these types of gunshot sounds: we need to collect samples of gunshot sounds with silencers, add them to the gunshot class dataset, and then retrain the model. In the future, we plan to make a new gunshot sound dataset recorded inside a shooting range with the same microphone used in the device. After training the model with this dataset, the system will be tested with actual shooting with different types of firearms—with and without silencers—inside a shooting range. We plan to test the system in the shooting range with different distances and measure its performance considering environmental noise in the future.

If there are multiple gunshot detector devices, such as in different classrooms of a school, and a gunshot is detected by more than one device, then the central server can prioritize the location of the device that detected the largest sound volume. The device can measure the maximum volume of the sound and send it to the server when a gunshot is detected. We plan to implement this feature in the future.

The proposed device needs a Wi-Fi connection to send data to the central server when a gunshot is detected. As Wi-Fi is generally available indoors, the device will work indoors only and will not work outdoors. However, the device can be used outdoors by interfacing with a cellular modem, which will give Internet access to the device outdoors. In the outdoors, the power supply might be a challenge. We plan to explore the possibility of outdoor usage of the device in the future.

6. Conclusions

In this project, a novel gunshot detector device is developed that automatically detects the indoor gunshot sound and then sends the gunshot location to the nearby law enforcement or emergency responder's station within a second using the Internet. The users of the device and the emergency responders also receive smartphone notifications as soon as the shooting happens. A deep-learning-based gunshot sound classifier is trained, validated, and tested using a large dataset. A prototype of the gunshot detector device, the central server for the emergency responder's station, and smartphone apps for users and emergency responders have been developed and tested successfully.

Funding: This research was funded by the Faculty Research Fellowship (FRF) award and Supplemented Research Support award of Eastern Michigan University.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Gun Violence Archive. Available online: <https://www.gunviolencearchive.org/> (accessed on 22 September 2023).
- Lankford, A. Public Mass Shooters and Firearms: A Cross-National Study of 171 Countries. *Violence Vict.* **2016**, *31*, 187–199. [CrossRef] [PubMed]
- Number of Mass Shootings in the United States between 1982 and April 2023. Available online: <https://www.statista.com/statistics/811487/number-of-mass-shootings-in-the-us/> (accessed on 27 July 2023).
- 49 Killed in Mass Shooting at Two Mosques in Christchurch, New Zealand. Available online: <https://www.cnn.com/2019/03/14/asia/christchurch-mosque-shooting-intl/index.html> (accessed on 27 July 2023).
- How Frequently Do Church Shootings Occur? Available online: <https://lifewayresearch.com/2020/02/12/how-frequently-do-church-shootings-occur/> (accessed on 27 July 2023).
- Effects of Gun Violence. Available online: <https://www.bradyunited.org/issue/effects-of-gun-violence> (accessed on 27 July 2023).
- SoundThinking. Available online: <https://www.soundthinking.com/law-enforcement/gunshot-detection-technology> (accessed on 31 July 2023).
- Samireddy, S.R.; Carletta, J.; Lee, K. An embeddable algorithm for gunshot detection. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 68–71. [CrossRef]
- Thanhikam, W. Gunshot noise detection using zero phase technique. In Proceedings of the 2015 Asian Conference on Defence Technology (ACDT), Hua Hin, Thailand, 23–25 April 2015; pp. 183–186. [CrossRef]
- MHrabina; Sigmund, M. Gunshot recognition using low level features in the time domain. In Proceedings of the 2018 28th International Conference Radioelektronika (RADIOELEKTRONIKA), Prague, Czech Republic, 19–20 April 2018; pp. 1–5. [CrossRef]
- Lopez-Morillas, J.; Canadas-Quesada, F.J.; Vera-Candeas, P.; Ruiz-Reyes, N.; Mata-Campos, R.; Montiel-Zafra, V. Gunshot detection and localization based on Non-negative Matrix Factorization and SRP-Phat. In Proceedings of the 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), Rio de Janeiro, Brazil, 10–13 July 2016; pp. 1–5. [CrossRef]
- Valenzise, G.; Gerosa, L.; Tagliasacchi, M.; Antonacci, F.; Sarti, A. Scream and gunshot detection and localization for audio-surveillance systems. In Proceedings of the 2007 IEEE Conference on Advanced Video and Signal Based Surveillance, London, UK, 5–7 September 2007; pp. 21–26. [CrossRef]
- Chen, C.; Abdallah, A.; Wolf, W. Audiovisual Gunshot Event Recognition. In Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, 8–11 October 2006; pp. 4807–4812. [CrossRef]
- Galangque, C.M.J.; Guirinaldo, S.A. Gunshot Classification and Localization System using Artificial Neural Network (ANN). In Proceedings of the 2019 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 18 July 2019; pp. 1–5. [CrossRef]
- Bajzik, J.; Prinosil, J.; Koniar, D. Gunshot Detection Using Convolutional Neural Networks. In Proceedings of the 2020 24th International Conference Electronics, Palanga, Lithuania, 15–17 June 2020; pp. 1–5. [CrossRef]
- Morehead, A.; Ogden, L.; Magee, G.; Hosler, R.; White, B.; Mohler, G. Low Cost Gunshot Detection using Deep Learning on the Raspberry Pi. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3038–3044. [CrossRef]
- Park, J.; Cho, Y.; Sim, G.; Lee, H.; Choo, J. Enemy Spotted: In-game Gun Sound Dataset for Gunshot Classification and Localization. In Proceedings of the 2022 IEEE Conference on Games (CoG), Beijing, China, 21–24 August 2022; pp. 56–63. [CrossRef]
- BGG Dataset (PUBG Gun Sound Dataset). Available online: <https://github.com/junwoopark92/BG-Gun-Sound-Dataset> (accessed on 1 August 2023).
- Jaszczak, B.; Nelson, B. Free Firearm Sound Effects Library. Available online: <https://opengameart.org/content/the-free-firearm-sound-library> (accessed on 1 August 2023).
- Tuncer, T.; Dogan, S.; Akbal, E.; Aydemir, E. An automated gunshot audio classification method based on finger pattern feature generator and iterative relief feature selector. *ADYU Mühendislik Bilim. Derg.* **2021**, *8*, 225–243.
- Gunshot Audio Dataset. Available online: <https://www.kaggle.com/datasets/emrahaydemr/gunshot-audio-dataset> (accessed on 1 August 2023).
- Gunshot Audio Forensics Dataset. Available online: <http://cadreforensics.com/audio/> (accessed on 1 August 2023).
- Kabealo, R.; Wyatt, S.; Aravamudan, A.; Zhang, X.; Acaron, D.N.; Dao, M.P.; Elliott, D.; Smith, A.O.; Otero, C.E.; Otero, L.D.; et al. A multi-firearm, multi-orientation audio dataset of gunshots. *Data Brief* **2023**, *48*, 109091. [CrossRef] [PubMed]
- Gunshot/Gunfire Audio Dataset. Available online: <https://zenodo.org/record/7004819#.Y8WJfHbMK3A> (accessed on 1 August 2023).
- Salamon, J.; Jacoby, C.; Bello, J.P. A dataset and taxonomy for urban sound research. In Proceedings of the 22nd ACM International Conference on Multimedia, New York, NY, USA, 3–7 November 2014; pp. 1041–1044.
- Urbansound8k Dataset. Available online: <https://urbansounddataset.weebly.com/urbansound8k.html> (accessed on 1 August 2023).
- Piczak, K.J. ESC: Dataset for Environmental Sound Classification. In Proceedings of the 23rd Annual ACM Conference on Multimedia, Brisbane, Australia, 26–30 October 2015; pp. 1015–1018.

28. ESC-50: Dataset for Environmental Sound Classification. Available online: <https://github.com/karolpiczak/ESC-50> (accessed on 1 August 2023).
29. Fonseca, E.; Favory, X.; Pons, J.; Font, F.; Serra, X. FSD50K: An Open Dataset of Human-Labeled Sound Events. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 829–852. [CrossRef]
30. FSD50K Dataset. Available online: <https://zenodo.org/record/4060432> (accessed on 1 August 2023).
31. Khan, T. A deep learning model for snoring detection and vibration notification using a smart wearable gadget. *Electronics* **2019**, *8*, 987. [CrossRef]
32. Snoring Dataset. Available online: <https://www.kaggle.com/datasets/tareqkhanemu/snoring> (accessed on 1 August 2023).
33. WavePad Audio Editing Software. Available online: <https://www.nch.com.au/wavepad/index.html> (accessed on 1 August 2023).
34. Pydiogment Library. Available online: <https://github.com/SuperKogito/pydiogment/> (accessed on 1 August 2023).
35. Pydub Library. Available online: <https://github.com/jiaaro/pydub> (accessed on 1 August 2023).
36. Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [CrossRef]
37. Fayek, H. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's in-between. Available online: <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html> (accessed on 2 August 2023).
38. SpeechPy. Available online: <https://speechpy.readthedocs.io/en/latest/intro/introductions.html> (accessed on 2 August 2023).
39. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
40. Nair, V.; Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
41. Nagi, J.; Ducatelle, F.; Di Caro, G.A.; Cireşan, D.; Meier, U.; Giusti, A.; Nagi, F.; Schmidhuber, J.; Gambardella, L.M. Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition. In Proceedings of the IEEE International Conference on Signal and Image Processing Applications (ICSIPA2011), Kuala Lumpur, Malaysia, 16–18 November 2011.
42. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 1026–1034.
44. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
45. A Look at Gradient Descent and RMSprop Optimizers. Available online: <https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b> (accessed on 2 August 2023).
46. Keras: The Python Deep Learning Library. Available online: <https://keras.io> (accessed on 9 August 2023).
47. NVIDIA-TensorRT. Available online: <https://developer.nvidia.com/tensorrt> (accessed on 4 August 2023).
48. Accelerating Inference in TensorFlow with TensorRT User Guide. Available online: <https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html#worflow-with-savedmodel> (accessed on 4 August 2023).
49. Jetson Nano Developer Kit. Available online: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit> (accessed on 3 August 2023).
50. USB Lavalier Lapel Microphon. Available online: <https://www.amazon.com/Lavalier-Microphone-Cardioid-Condenser-K053/dp/B077VNGVL2> (accessed on 3 August 2023).
51. Wireless NIC Module for Jetson Nano. Available online: <https://www.amazon.com/Wireless-AC8265-Wireless-Developer-Support-Bluetooth/dp/B07V9B5C6M/> (accessed on 3 August 2023).
52. Bluetooth Device Configure. Available online: <https://manpages.ubuntu.com/manpages/trusty/man8/hciconfig.8.html> (accessed on 3 August 2023).
53. PyBluez. Available online: <https://pybluez.readthedocs.io/en/latest/> (accessed on 3 August 2023).
54. Wi-Fi Wrapper Library. Available online: <https://pypi.org/project/wifi-wrapper/> (accessed on 3 August 2023).
55. Play and Record Sound with Python. Available online: <https://python-sounddevice.readthedocs.io/en/0.4.6/> (accessed on 4 August 2023).
56. Python Module for Reading and Writing WAV Files. Available online: <https://pypi.org/project/wavio/> (accessed on 4 August 2023).
57. Date and Time Library. Available online: <https://docs.python.org/3/library/datetime.html> (accessed on 4 August 2023).
58. Socket—Low-Level Networking Interface. Available online: <https://docs.python.org/3/library/socket.html> (accessed on 3 August 2023).
59. HTTP Server. Available online: <https://docs.python.org/3/library/http.server.html> (accessed on 7 August 2023).
60. SQL Server 2022 Express. Available online: <https://www.microsoft.com/en-us/sql-server/sql-server-downloads> (accessed on 7 August 2023).
61. Android Identifiers. Available online: <https://developer.android.com/training/articles/user-data-ids> (accessed on 8 August 2023).
62. FCM Registration Token. Available online: <https://firebase.google.com/docs/cloud-messaging/manage-tokens#ensuring-registration-token-freshness> (accessed on 8 August 2023).
63. C# TCP Server. Available online: <https://www.codeproject.com/articles/488668/csharp-tcp-server> (accessed on 8 August 2023).

64. How to Port Forward. Available online: <https://www.noip.com/support/knowledgebase/general-port-forwarding-guide/> (accessed on 8 August 2023).
65. How Do I Open a Port on Windows Firewall? Available online: <https://www.howtogeek.com/394735/how-do-i-open-a-port-on-windows-firewall/> (accessed on 8 August 2023).
66. Thompson, B. C# Database Connection: How to Connect SQL Server. Available online: <https://www.guru99.com/c-sharp-access-database.html> (accessed on 8 August 2023).
67. GMap.NET—Maps for Windows. Available online: <https://github.com/judero01col/GMap.NET> (accessed on 8 August 2023).
68. FcmSharp. Available online: <https://github.com/bytcfish/FcmSharp> (accessed on 8 August 2023).
69. GeoLocation. Available online: <https://www.b4x.com/android/forum/threads/geolocation.99710/#content> (accessed on 8 August 2023).
70. Firebase Cloud Messaging. Available online: <https://firebase.google.com/docs/cloud-messaging> (accessed on 8 August 2023).
71. FirebaseNotifications—Push Messages/Firebase Cloud Messaging (FCM). Available online: <https://www.b4x.com/android/forum/threads/b4x-firebase-push-notifications-2023.148715/> (accessed on 8 August 2023).
72. GeeekPi Nano Case. Available online: <https://www.amazon.com/GeeekPi-Support-Developer-Powerful-Computer/dp/B098J4JMLG/> (accessed on 10 August 2023).
73. Jetson-Stats. Available online: https://mnext.it/jetson_stats/ (accessed on 10 August 2023).
74. Kerampran, C.; Gajewski, T.; Sielicki, P.W. Temperature Measurement of a Bullet in Flight. *Sensors* **2020**, *20*, 7016. [CrossRef] [PubMed]
75. Khan, M.U.; Misbah, M.; Kaleem, Z.; Deng, Y.; Jamalipour, A. GAANet: Ghost Auto Anchor Network for Detecting Varying Size Drones in Dark. In Proceedings of the 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), Florence, Italy, 20–23 June 2023; pp. 1–5. [CrossRef]
76. Anwar, M.Z.; Kaleem, Z.; Jamalipour, A. Machine Learning Inspired Sound-Based Amateur Drone Detection for Public Safety Applications. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2526–2534. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.