2023-2024 Fall Semester
COMP413
Internet of Things
Final Project Report

Instructor
Abdulkadir KÖSE

Subject
An IoT Based Gunshot Detection System

Submission Date
14.01.2024

Submitted by
Ahmet SEZGİN
Mert ERDEM
Beyzanur YÜCE - 2011051071
Zehra MOĞULKOÇ - 110510223

Group 9

1. INTRODUCTION

The issue of public safety continues to escalate in numerous cities globally. As indicated by [1], current statistics reveal that emergency services remain uninformed about approximately 80% of gunshot incidents. Consequently, the development of gunshot detection technology emerges as a critically important area of study and innovation. In response to the surge in armed violence, our IoT project introduces a robust solution: utilizing IoT sensors and devices capable of capturing audio data from diverse locations, we have developed a deep-learning-based sound classification system specifically designed for the precise detection of gunshot sounds within ambient environmental noise. This integration aims to significantly enhance existing security frameworks by rapidly identifying firearm-related incidents.

In this report, we highlight the strategic utilization of the Orange Pi, a versatile and cost-effective IoT platform, to facilitate our gunshot detection system. The Orange Pi platform demonstrates proficiency in handling audio recording, processing, and data transmission tasks, aligning seamlessly with the requirements of our project. Its incorporation within our system highlights its crucial function in initiating the early phases of our gunshot detection process, enabling a smooth flow of audio data essential for subsequent machine learning analysis.

Our initiative introduces a crucial component: a sophisticated Convolutional Neural Network (CNN) rooted in deep learning, which is thoroughly designed to identify and differentiate gunshot sounds among varying environmental noise. This CNN model holds a central position within our project, playing a pivotal role in strengthening current security frameworks by swiftly and precisely recognizing incidents involving firearms.

This approach facilitates fast, low-cost and effective responses to incidents that often evade immediate detection. Instantaneous transmission of location data upon detecting a single gunshot enables rapid medical aid and intervention to prevent further escalation. For a clearer understanding, a comprehensive overview of the system design is given below:

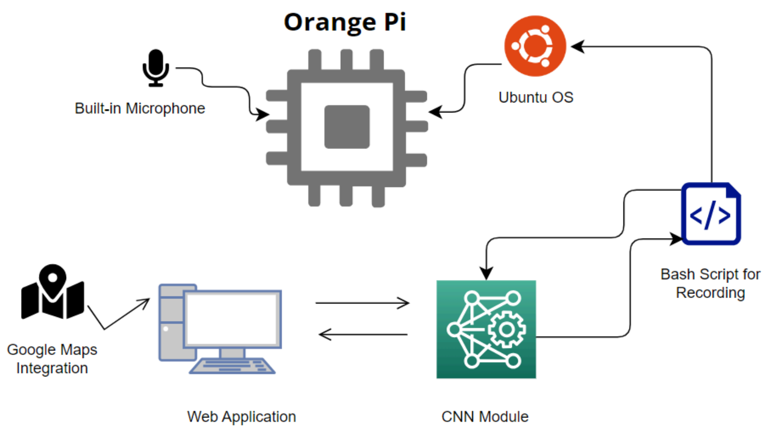


Fig. 1: Overview of the system design

Moreover, the deployment of this innovative technology directly aligns with two Sustainable Development Goals (SDGs). Firstly, it aligns with SDG 11 (Sustainable Cities and Communities) by fostering safer urban environments through advanced security measures. Additionally, in alignment with SDG 3 (Good Health and Well-being), this model promises rapid responses to firearm-related incidents, potentially saving lives and contributing to public health and safety.

2. SYSTEM MODEL

2.1. Hardware Design:

Our hardware mainly consists of a board named “Orange Pi PC”. The biggest advantage of this device is it’s relatively cheap compared to similar equipment such as Raspberry Pi. Another massive advantage of this product is the fact that it’s already equipped with a built-in microphone out of the box. The built-in microphone allows us to test our system while still being relatively scalable. In addition to the board, we have also utilized lots of peripheral devices for the sake of setup and configuration of the board.

The board we are using is equipped with Ubuntu OS. Thus, we can use the **alsa** to record audio using our board. We can execute recording actions automatically by using bash script. Here is a script that we used to automatically record audio samples:

```
#!/bin/bash

# Configuration
RECORD_DURATION=5  # sample duration
FILENAME="sample.wav"

while true; do
   # This line records audio for the specified duration (-d).
   arecord -D hw:0,0 -f S16_LE -r 44100 -c 2 -d "$RECORD_DURATION" "$FILENAME"

   # Sleeping for the sake of energy optimization
   sleep 15
done
```

Once the recording is done, it is automatically sent to a PC where the ML algorithms are automatically run. Our initial intention was to use a tinyML system directly on board so that we can have an edge computing system where the communication is as efficient as it could be. However, due to hardware related technical limitations, we weren't able to fully implement the tinyML system on the board. We believe that this is something that is easily achievable on a hardware device such as Raspberry Pi since we have everything - including the tflite model- ready and tested.

In our system, everything works automatically after the initialization step in order to simulate a real world environment scenario. To accomplish this automation, we have set up multiple scripts all working with each other seamlessly. Here is a step-by-step outline of our system:

- Firstly, the audio is recorded via the script mentioned above.
- Then, the audio sample is sent in order to be processed by the tflite model we have created.
- Next, the tflite model determines if the audio it received is a gunshot or not.
- If the gunshot is detected, audio sample is sent to our web application along with the coordinates of the node
- Web application zooms into the coordinates provided in the interactive map and plays the audio sample.

2.2. Software Design:

2.2.1. Machine Learning Model:

Many prior studies [2]-[3] have demonstrated the effectiveness of convolutional neural networks (CNNs) in the realm of sound classification. The foundation of our work lies in the CNN model and augmentation of the UrbanSound8K dataset, fully expanded to accommodate a dedicated subset of 1674 gunshot audio files, providing a balanced binary classification with 7929 non-gunshot files. These modifications resulted in a comprehensive dataset of 9603 audio files, enriched and tailored specifically for the task of gunshot identification. This augmentation strategy fortified our detection model's efficacy by empowering it to discern between gunshot sounds and diverse ambient noises encountered in real-world scenarios.

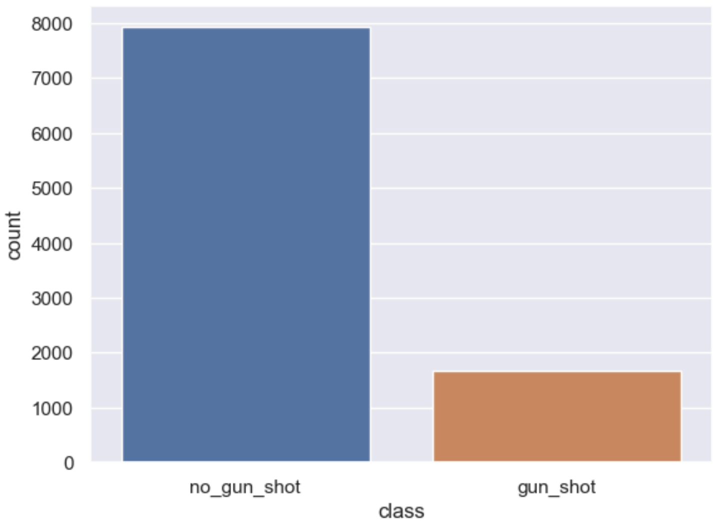


Fig. 2: Illustration of the distribution of classes within the dataset.

2.2.2. Feature Extraction:

Our feature extraction methodology centers around the extract_features function, meticulously crafted to facilitate efficient and effective audio analysis. Within this framework, we compute Mel-frequency cepstral coefficients (MFCCs), utilizing specific parameters (n_mfcc=40, n_mels=40, n_fft=1103) tailored to our application's requirements. Subsequent aggregation of these MFCCs, achieved by averaging across time frames, culminates in the generation of a representative feature vector. This refined approach amplifies our model's proficiency in pattern recognition, enabling accurate gunshot identification amidst a myriad of ambient noise scenarios. The distinctiveness between gunshot and non-gunshot audio manifestations is elucidated in the subsequent section.

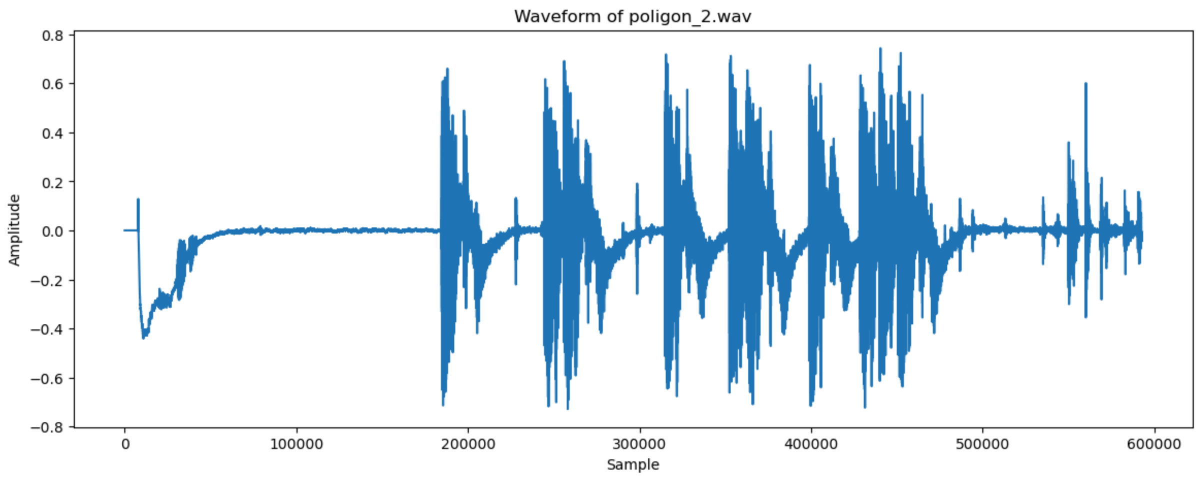


Fig. 3: Visualization of gunshot audio.

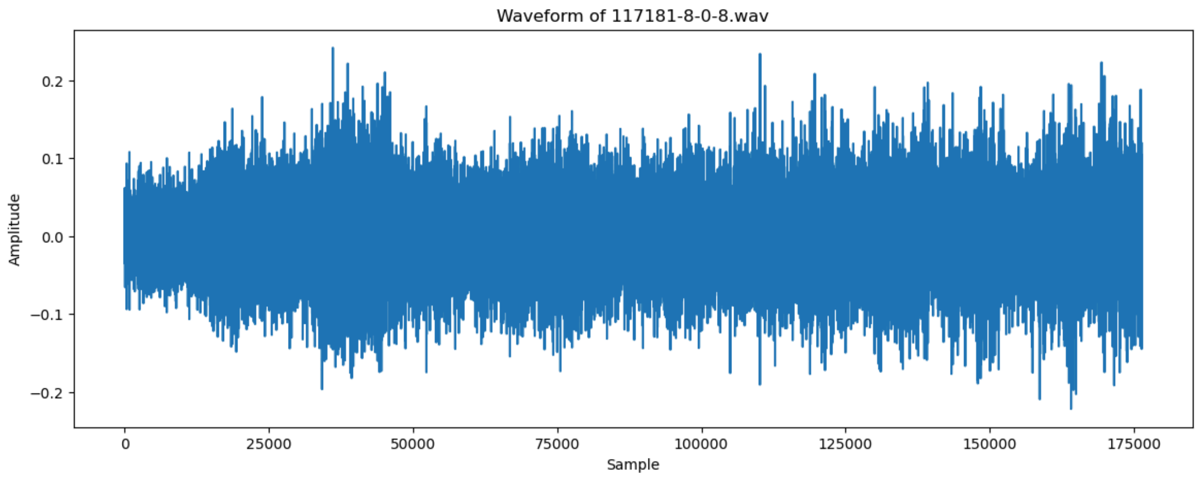


Fig. 4: Visualization of ambient audio.

2.2.3. CNN Model:

The model architecture we have constructed operates through a sequence of convolutional layers, strategically engineered to extract intricate spatial features from audio data. This setup includes multiple Conv2D layers, each initiating with 16 filters and progressively increasing the filter count to 32, 64, and finally 128. Employing a 2x2 kernel size and 'same' padding strategy, these convolutional layers efficiently extract essential features from the input audio spectrograms. Following each convolutional layer, a MaxPooling2D layer reduces spatial dimensions, enhancing computational efficiency while Dropout layers, interspersed between convolutions, prevent overfitting by deactivating a fraction of neurons randomly during training, sustaining model generalization. Culminating the architecture is the GlobalAveragePooling2D layer, conducting global average pooling on spatial dimensions, effectively consolidating features and compacting the model's parameters. The final layer, a Dense layer with a sigmoid activation function, is ideal for binary classification tasks, contingent upon appropriate configuration of 'num_labels' for desired classification outcomes.

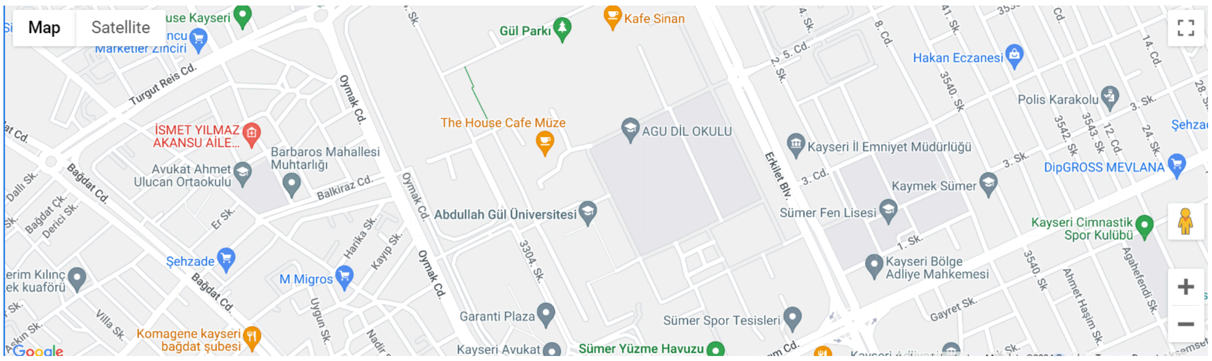
To fortify our detection model's efficacy and precision, we augmented the widely recognized Urban8K dataset by infusing an additional 900 gunshot sound samples. This augmentation technique facilitated the establishment of a binary dataset, empowering the model to adeptly discern between gunshot sounds and the diverse array of ambient noises encountered in real-world scenarios.

Key highlights of our approach encompass the integration of 900 supplementary gunshot samples into the Urban8K dataset, significantly refining the model's performance. Our inclusively designed CNN architecture is purpose-built for high-precision gunshot detection tasks, configured explicitly for binary classification, demonstrating exceptional acuity in distinguishing gunshot sounds amidst a spectrum of environmental noises.

2.2.4. Web Application:

We wanted to simulate the process of notifying authorities in a real-world scenario. For this purpose, we created a web application. The main feature of this web application is that it receives audio and location data via an API call. Our web app has an interactive map provided by Google services built in. Our web app can also play the audio files it receives.

Once the web application receives audio and location data(which happens when the TFLite model determines there is a gunshot), location of the node as well as the audio sample that is assumed to be the gunshot noise is displayed in the web application. Then, the web application automatically zooms into the coordinates of the node and plays the audio sample.



We believe that this web application serves the rudimentary purpose of simulating an interface between the authorities and our gunshot detection node.

2.2.5. Integration of TinyML for Edge Deployment:

To enhance the efficiency of our Gunshot Detection IoT system for edge deployment, we converted our trained Keras model into a compact TensorFlow Lite (TFLite) format optimized for devices with limited computational resources. Initially, the pre-trained model weights were loaded, followed by conversion using the TensorFlow Lite Converter, resulting in a lightweight TFLite model. Despite its compatibility for edge nodes, due to challenges encountered with TensorBoard installation on our board, the model was implemented on a remote device. This approach ensures efficient and responsive gunshot detection capabilities, emphasizing adaptability while prioritizing resource conservation in diverse IoT environments.

3. RESULTS AND DISCUSSION

Upon rigorous evaluation of our implemented Gunshot Detection IoT system, the following performance metrics were observed:

-**Training Accuracy:** Achieved a commendable accuracy rate of 91.57% during the training phase, underscoring the model's capability to effectively learn and discern intricate patterns within the dataset.

-**Test Accuracy:** Demonstrated a significant validation prowess with a test accuracy of 86.60%, affirming the system's reliability and precision in real-world scenarios.

In addition, we have also went to the shooting range “Combat Poligon” in order to test our ML algorithm on real world data. We recorded audio samples of gunshots in the shooting range. We then tested our machine learning algorithm using the data we acquired. Our algorithm was able to correctly determine the sounds we recorded in the shooting range as gunshots.

In conclusion, the IoT project developed serves as a pivotal advancement in security measures, offering rapid detection of armed attacks with heightened efficiency and cost-effectiveness relative to traditional notification-based or video detection systems. By harnessing the capabilities of sensor technology and real-time voice recording, the system swiftly pinpoints incident locations and dispatches vital information directly to security forces, enabling prompt and decisive actions. Moreover, the versatility of this system is noteworthy, as it can be seamlessly implemented in densely populated areas, high-risk zones, or even personalized settings such as vehicles or residences based on specific customer requirements. This adaptability amplifies its applicability across diverse operational landscapes, ensuring enhanced situational awareness and risk mitigation. By capitalizing on IoT capabilities, this project exemplifies the transformative potential of innovative technologies in refining security protocols, safeguarding individuals, and protecting assets across multifaceted environments.

4. FUTURE WORK

In our ongoing pursuit of enhancing system capabilities, we have delineated several future endeavors aimed at fostering comprehensive integration and sustainability. A primary focal point entails optimizing energy utilization by transitioning the system into a passive state when not actively detecting gunshots. This strategic shift aims to circumvent continuous sensor monitoring, thereby conserving energy resources and bolstering operational efficiency.

Furthermore, our strategic roadmap encompasses the incorporation of LoRa (Long Range) technology, a pivotal advancement that facilitates expansive deployment throughout urban landscapes. By leveraging LoRa's extensive range and low-power capabilities, the system can achieve ubiquitous coverage across metropolitan areas, ensuring seamless integration and connectivity across diverse locales.

Additionally, our future initiatives include refining localization capabilities to augment sensor precision and accuracy in pinpointing incident locations. By enhancing localization algorithms and methodologies, we aspire to refine the system's geospatial capabilities, thereby facilitating more precise and reliable location-based services.

Collectively, these strategic enhancements aim to cultivate an environmentally sustainable, energy-efficient, and universally accessible system infrastructure. By embracing these future-oriented initiatives, we aspire to elevate system performance, scalability, and adaptability, ensuring optimal functionality, resilience, and value proposition across various operational contexts and environmental settings.

Furthermore, we aim to expand the capabilities of the project by incorporating additional audio phrase detections commonly associated with crime scenes, such as phrases like 'Help,' 'Catch,' 'Gun,' 'Ambulance,' and more. By broadening the scope in this manner, we anticipate enhancing the model's accuracy significantly, enabling it to identify a wider range of critical situations more effectively. This expansion will not only increase the project's applicability in real-world scenarios but also provide security forces with more comprehensive insights and immediate alerts during emergencies. Through rigorous testing under practical conditions, we aspire to refine the system further, ensuring its robustness, reliability, and responsiveness in diverse environments and challenging situations.

5. SOURCE CODE

The complete source code is accessible via the following links:
https://github.com/ZehraMogulkoc/Gunshot_Detection

Extracting Features of Audio Files

In [185…

```python
import struct

class WavFileHelper():

    def read_file_properties(self, filename):
        wave_file = open(filename,"rb")

        riff = wave_file.read(12)
        fmt = wave_file.read(36)

        num_channels_string = fmt[10:12]
        num_channels = struct.unpack('<H', num_channels_string)[0]

        sample_rate_string = fmt[12:16]
        sample_rate = struct.unpack("<I",sample_rate_string)[0]

        bit_depth_string = fmt[22:24]
        bit_depth = struct.unpack("<H",bit_depth_string)[0]

        return (num_channels, sample_rate, bit_depth)
```

In [186…

```python
# Load various imports
import pandas as pd
import os
import librosa
import librosa.display

wavfilehelper = WavFileHelper()

audiodata = []
for index, row in metadata.iterrows():
    file_name = os.path.join(audio_dir, 'fold'+str(row["fold"])+'\\', str(row["file_name"]))
    data = wavfilehelper.read_file_properties(file_name)
    audiodata.append(data)

# Convert into a Panda dataframe
audiodf = pd.DataFrame(audiodata, columns=['num_channels','sample_rate','bit_depth'])
```

In [210…

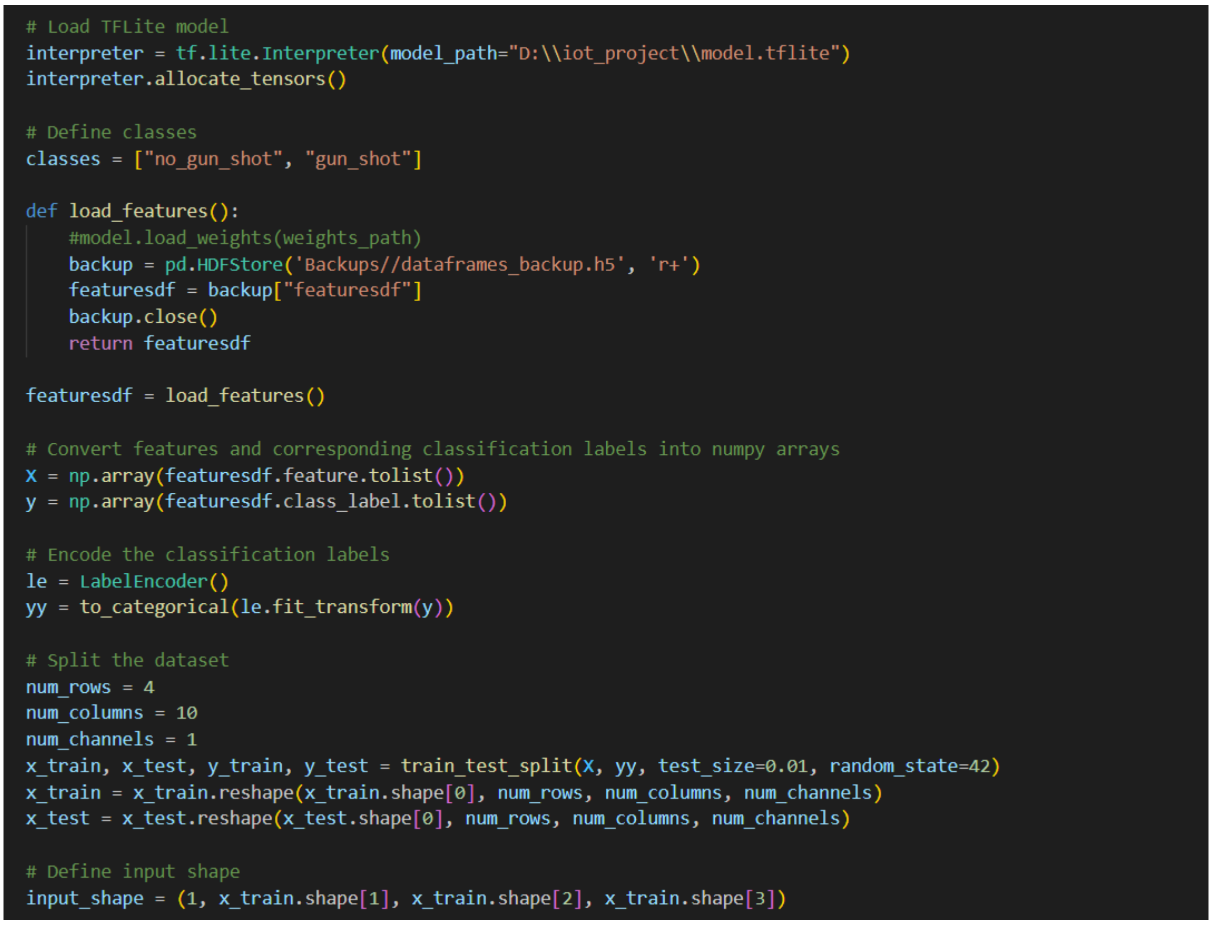
```python
num_rows = 4
num_columns = 10
num_channels = 1
```

In [211…

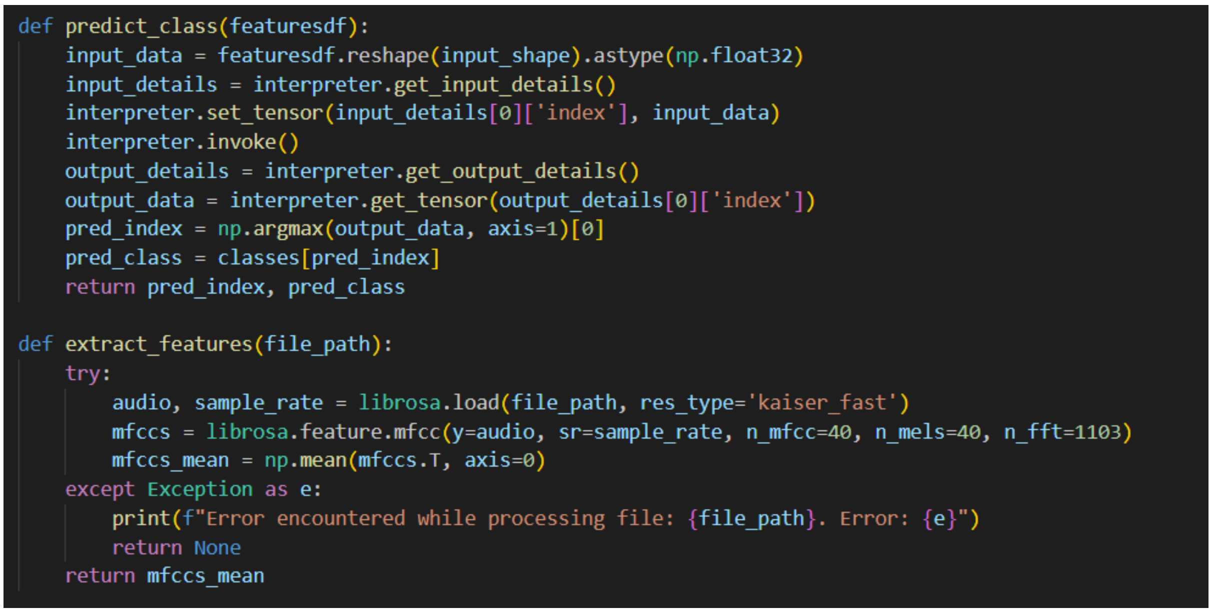
```python
x_train = x_train.reshape(x_train.shape[0], num_rows, num_columns, num_channels)
x_test = x_test.reshape(x_test.shape[0], num_rows, num_columns, num_channels)

num_labels = yy.shape[1]
filter_size = 2
```

In [212…

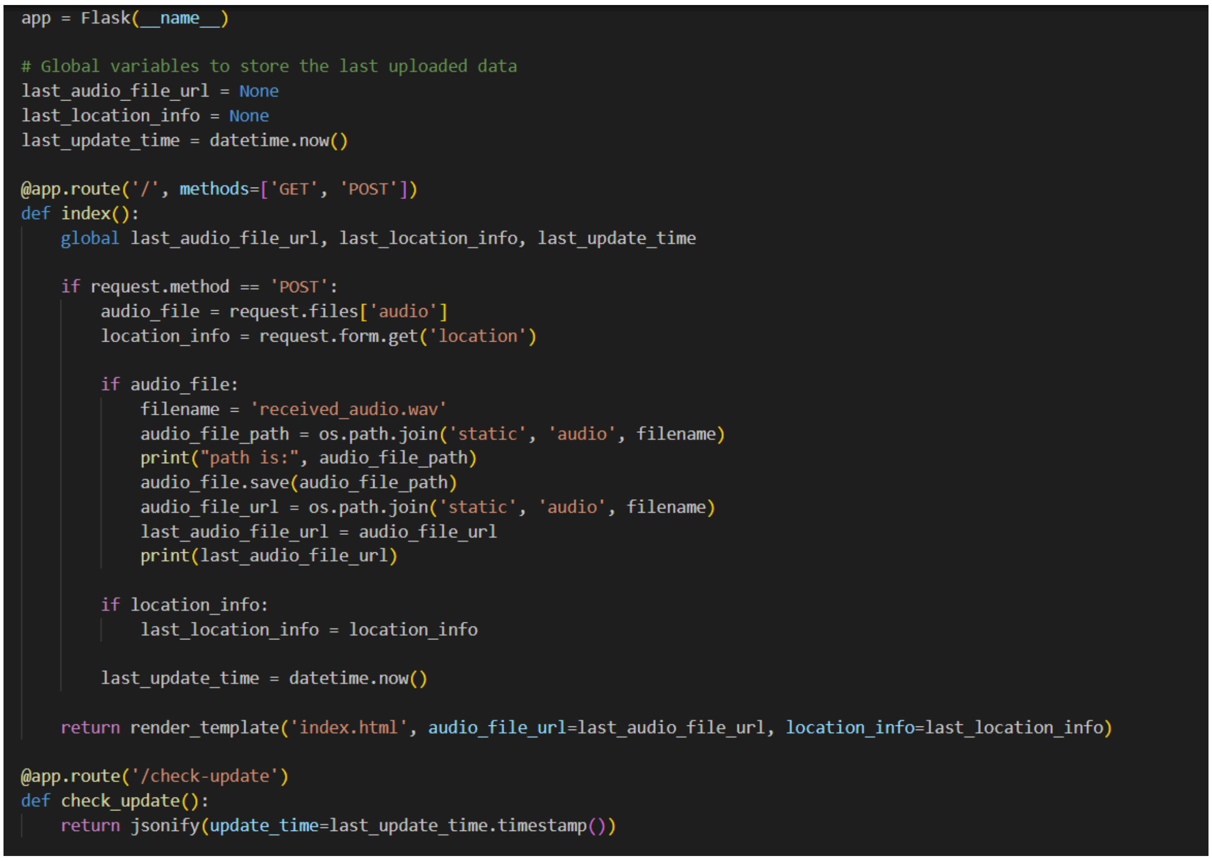
```python
# Construct model
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, padding="same", input_shape=(num_rows, num_columns, num_channels), activation='r
model.add(MaxPooling2D(pool_size=1))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=2, padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=1))
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=2, padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=1))
model.add(Dropout(0.2))

model.add(Conv2D(filters=128, kernel_size=2, padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=1))
model.add(Dropout(0.2))
model.add(GlobalAveragePooling2D())

model.add(Dense(num_labels, activation='sigmoid'))
```



```python
# Load TFLite model
interpreter = tf.lite.Interpreter(model_path="D:\\iot_project\\model.tflite")
interpreter.allocate_tensors()

# Define classes
classes = ["no_gun_shot", "gun_shot"]

def load_features():
    #model.load_weights(weights_path)
    backup = pd.HDFStore('Backups//dataframes_backup.h5', 'r+')
    featuresdf = backup["featuresdf"]
    backup.close()
    return featuresdf

featuresdf = load_features()

# Convert features and corresponding classification labels into numpy arrays
X = np.array(featuresdf.feature.tolist())
y = np.array(featuresdf.class_label.tolist())

# Encode the classification labels
le = LabelEncoder()
yy = to_categorical(le.fit_transform(y))

# Split the dataset
num_rows = 4
num_columns = 10
num_channels = 1
x_train, x_test, y_train, y_test = train_test_split(X, yy, test_size=0.01, random_state=42)
x_train = x_train.reshape(x_train.shape[0], num_rows, num_columns, num_channels)
x_test = x_test.reshape(x_test.shape[0], num_rows, num_columns, num_channels)

# Define input shape
input_shape = (1, x_train.shape[1], x_train.shape[2], x_train.shape[3])
```



```python
def predict_class(featuresdf):
    input_data = featuresdf.reshape(input_shape).astype(np.float32)
    input_details = interpreter.get_input_details()
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_details = interpreter.get_output_details()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    pred_index = np.argmax(output_data, axis=1)[0]
    pred_class = classes[pred_index]
    return pred_index, pred_class

def extract_features(file_path):
    try:
        audio, sample_rate = librosa.load(file_path, res_type='kaiser_fast')
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40, n_mels=40, n_fft=1103)
        mfccs_mean = np.mean(mfccs.T, axis=0)
    except Exception as e:
        print(f"Error encountered while processing file: {file_path}. Error: {e}")
        return None
    return mfccs_mean
```



```python
app = Flask(__name__)

# Global variables to store the last uploaded data
last_audio_file_url = None
last_location_info = None
last_update_time = datetime.now()

@app.route('/', methods=['GET', 'POST'])
def index():
    global last_audio_file_url, last_location_info, last_update_time

    if request.method == 'POST':
        audio_file = request.files['audio']
        location_info = request.form.get('location')

        if audio_file:
            filename = 'received_audio.wav'
            audio_file_path = os.path.join('static', 'audio', filename)
            print("path is:", audio_file_path)
            audio_file.save(audio_file_path)
            audio_file_url = os.path.join('static', 'audio', filename)
            last_audio_file_url = audio_file_url
            print(last_audio_file_url)

        if location_info:
            last_location_info = location_info

        last_update_time = datetime.now()

    return render_template('index.html', audio_file_url=last_audio_file_url, location_info=last_location_info)

@app.route('/check-update')
def check_update():
    return jsonify(update_time=last_update_time.timestamp())
```

6. SPECIAL THANKS

Special thanks to Combat Poligon for allowing us to record gunshot audio samples for our project.

REFERENCES

[1] Irvin-Erickson, Yasemin, Bing Bai, Annie Gurvis, and Edward Mohr. ”The effect of gun violence on local economies.” Washington, DC: Urban Institute (2016).

[2] A. Chaturvedi, S. A. Yadav, H. M. Salman, H. R. Goyal, H. Gebregziabher and A. K. Rao, "Classification of Sound using Convolutional Neural Networks," 2022 5th International Conference on Contemporary Computing and Informatics (IC3I), Uttar Pradesh, India, 2022, pp. 1015-1019, doi: 10.1109/IC3I56241.2022.10072823.

[3] Takahashi, Naoya, Michael Gygli, Beat Pfister, and Luc Van Gool. ”Deep convolutional neural networks and data augmentation for acoustic event detection.” arXiv preprint arXiv:1604.07160 (2016).

[4] A. Morehead, L. Ogden, G. Magee, R. Hosler, B. White and G. Mohler, "Low Cost Gunshot Detection using Deep Learning on the Raspberry Pi," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 3038-3044, doi: 10.1109/BigData47090.2019.9006456.