ב"ה

Columbia University
Mimoun P Cadosch Delmar
Spring 2015

Real-Time Environmental Sound Recognition for Self-Hosted Security System Using JavaScript

1. Introduction

The purpose of this research project is to build a system that can detect and recognize unusual sounds in the environment for security purposes in a residential setting. This system focuses on three types of sounds: breaking glass, dog barks, and gunshots. These types of sounds are indicative that unusual activity is taking place, and the user ought to be warned about them. The system program is written in JavaScript, so that it can be run from the browser in a mobile phone, tablet, or even a Raspberry Pi, and then placed in a residential setting

The main motivation behind this project is privacy. There exist many products in the market that claim to analyze environmental sound for safety purposes. Most of these devices, however, stream the data to the cloud. This can be considered an intrusion to people's privacy, as these devices can essentially be viewed as microphones in people's homes. The system built here is self-hosted, and performs its tasks while preserving the privacy of its users.

2. System Setup

a) Data sources

The data used in this system to train and test the classifier consists of recordings of dog barks, shattering windows, and gunshots. This data was obtained from youtube thanks to the library *Youtube-dl* and its python wrapper [1]**.** This library enabled us to effectively download the audio of a video by simply providing the video's URL.

In training, for each type of sound (barks, breaking windows, gunshots) we used eight different recordings. Each recording ranges from ten seconds to one minute in length. This length was initially chosen because most videos only had ten to twenty seconds of "pure" sound (e.g. barks), free of any background or interfering noise. The classifier works well when trained with videos of this length, and so this parameter was thus chosen.

b) Features: TED and MFCC

The main feature used to classify sounds are the *Mel Frequency Cepstrum Coefficients (MFCCs)*. MFCCs are widely used for the classification of speech and music signals because of their effective performance [2]. MFCCs are also used to classify non-speech sounds, although they fail to fully reflect the time-dependent features of non-stationary signals such as the environmental noise here considered. The MFCCs were obtained from the Meyda library. This library is written in JavaScript and provides an API for extracting many features in the realm of sound engineering [3].

The other feature used for classification is *energy*, which refers to the time-domain energy of the signal. The documentation in *Meyda* specifies that energy values are calculated as the "infinite integral of the squared signal."

c) Classifier: Multi-Class Support Vector Machines

The machine learning model chosen for classification is the Multi-class Support Vector Machines (SVMs). This supervised model was chosen because of its simplicity, relative robustness and speed. More specifically, the "one versus all" approach is used here, and will be explained in the following lines.

Ordinary SVMs perform binary classification of data, in other words classification for two classes (i.e. positive or negative). Our case, however, consists of three different classes of data: "dogs barking", "breaking glass" and "gunshots." Notice that "ambient noise" is not considered a class for classification purposes, as this is the default sound. Our system tells ambient noise from other classes using energy levels (more details on this in later paragraphs).

In order to classify a particular sound using SVMs given these four classes available, a multi-class SVM, "one versus all" model was chosen. Specifically, we constructed three separate classifiers, each trained to separate one class from the rest. A given test sound is processed by the three classifiers, and the class which classifies the data with greatest margin is chosen [4].

3. System Description

a) Feature extraction

In this paper, we will be using the term *frame* to refer to the sound from which the Meyda library extracts the audio features. For each frame, the *Meyda* library provides a vector with 13 MFCC's frequency coefficients. We will refer to every one of these vectors as v_i. The duration of each sampled frame is 33 ms. This parameter was obtained by tuning the system, and more details on the parameters used are given in the next section.

To simplify our task, rather than considering the MFCCs vector for every single frame, we take the average MFCC values over many frames instead. The number of frames used for averaging the MFCC coefficients is referred to in this paper (and in the code) as *interval size.* Here, the interval size is 150 frames. Experiments show good results for this interval size, as shorter intervals fail to yield correct classifications. The vector obtained from averaging the MFCC values over the interval is referred to as the *average vector,* denoted V_i.

For training our classifiers, we used eight different average vectors $V_1 \ldots V_8$ for each class. Thus, there are 24 vectors in total. Each vector has 26 entries, 13 for the mean value of each coefficient, and an additional 13 for the standard deviation of each coefficient over the interval. Experimentally, adding the standard deviation of the coefficients to V_i resulted in better classification results. In sum, the input to the classifier is a 26-by-24 matrix, denoted here by A. Notice that this process is repeated for each one of the classifiers we build, namely one for each class.

When it comes to testing a given sound, a 26-dimensional vector is extracted from the recorded noise, and classified by the SVM.

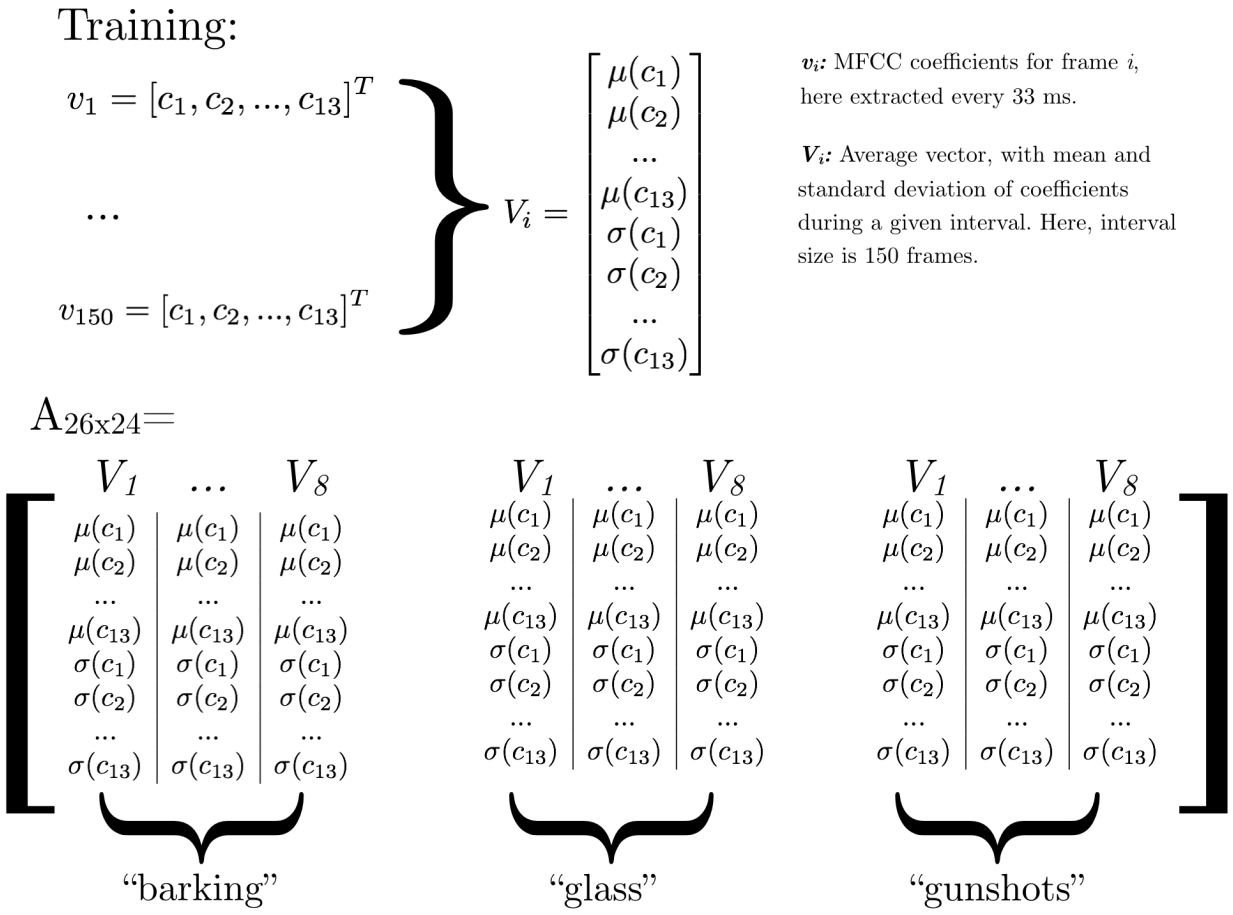

Figure 1: Vectors v_i are extracted from the recording. The mean and standard deviation for each coefficient are computed to make V_i. In the training step, each recording is summarized by a vector V_i and is used to form matrix A, the input to the SVM classifier.

$$labels_{1\times 24} = [(-1)^j, ..., (-1)^j]$$

$$j = \begin{cases} 0 & \text{if } j \text{ is correct class} \\ 1 & \text{if } j \text{ is incorrect class} \end{cases}$$

Figure 2: The labels used to train the data in matrix A in the Multi-class SVM. The vector *labels* has 24 labels, one for each recording. Given that we are using the "one versus all" model, the label is 1 for the class in question (hence $(-1)^0$=1), and -1 for all other classes (hence $(-1)^1$= -1).

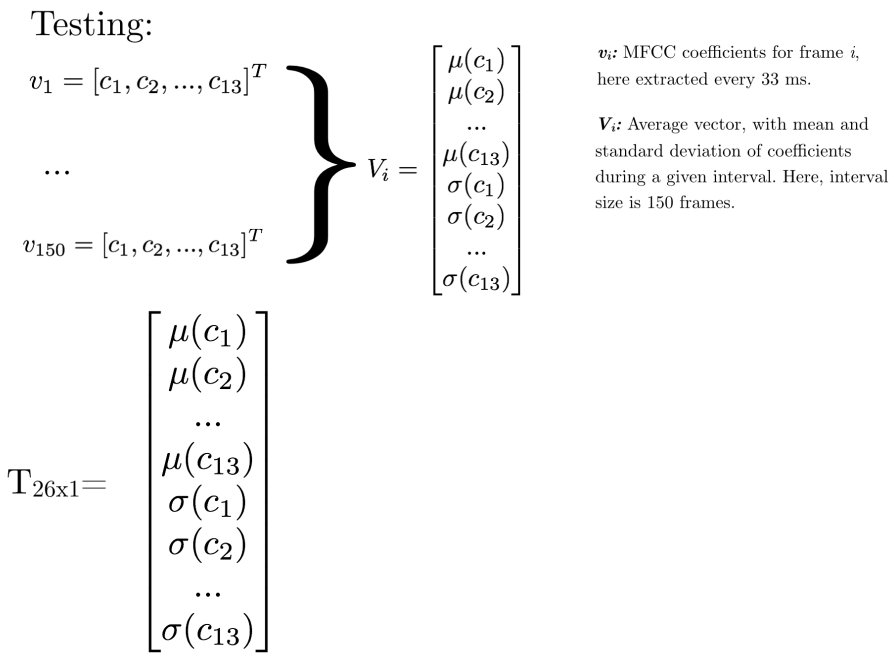

Figure 3: Vector T is the input to the SVM in the prediction step.

b) System flow

This chart illustrates the functioning of the system. The system is not triggered until the level of energy recorded exceeds a certain threshold. Otherwise, the system considers all noise as ambient sound. If the energy threshold is reached, the system starts recording. After 150 frames, the system averages the v_i values to compute V_i. This vector is run by the SVM for classification. The system will only start recording again when the energy threshold is reached again.

The reason I decided to distinguish "ambient noise" from the three other classes of sounds using energy, rather than another SVM, is that ambient noise is very unpredictable, and can contain a wide array of unexpected noises (ambulances, neighbors playing music, ventilation duct, wind, cars outside, etc.). If the system is not trained to assign these unexpected sounds to the "ambient noise" class, it will misclassify them to one of the other three classes.

Using energy levels to differentiate "ambient noise" from other classes of sound has prove a very reliable method, as room energy levels rarely exceed 0.01, while any noise worth warning the user about (dogs barking, gunshots, glass breaking) have levels ranging between 5 and 40.

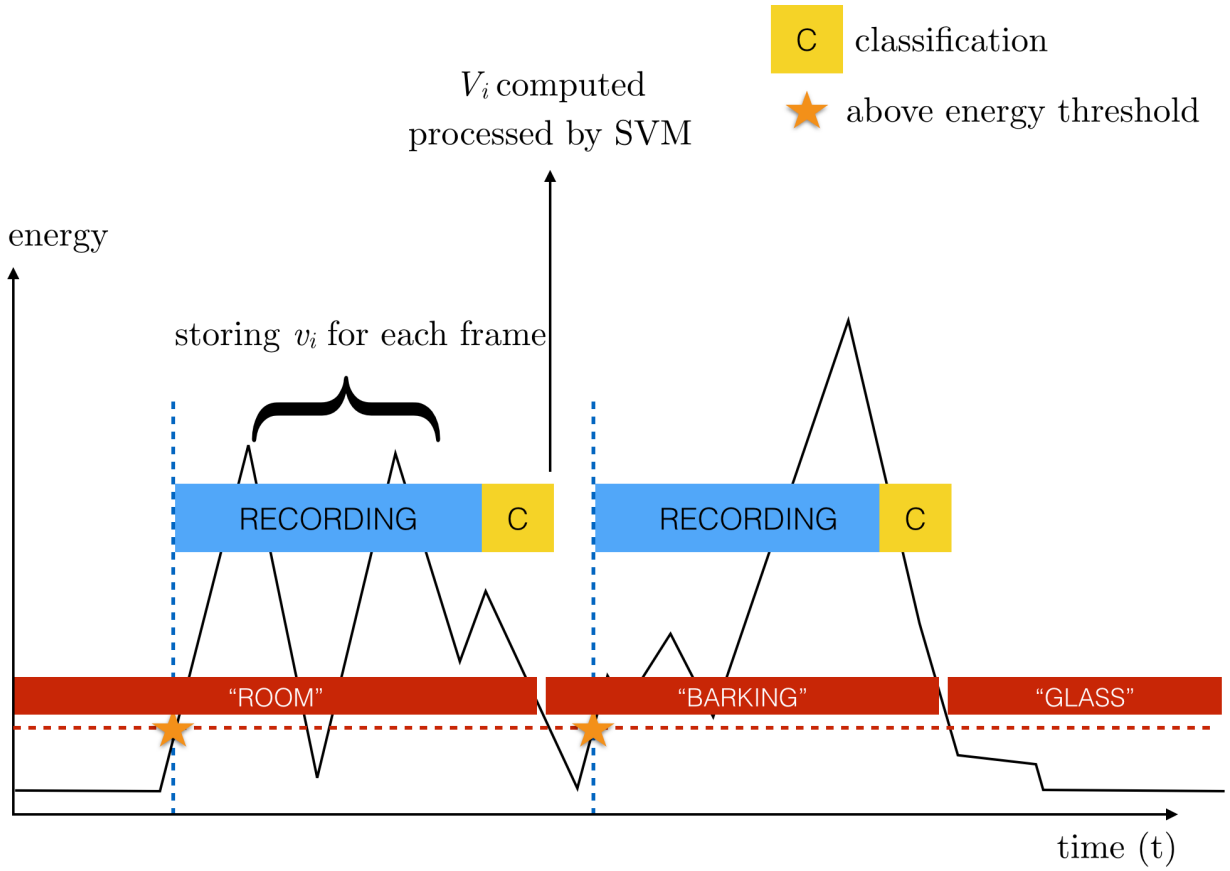

Figure 4: A sample illustration of how the system flows

c) Tuning parameters

There are certain parameters that can be tuned to achieve better classification performance. Specifically, these parameters are the size of the frames used to extract features, and the number of frames used to compute V_i to use as an input to the SVM.

These parameters determine the length of the sound considered for classification. For a simple visualization, this length corresponds to the length of the blue "recording" box in figure 4. This time length, which we can refer to as *interval time*, must be sufficiently long for the features extracted to be truly representative of their class, so that classification is more accurate.

The two parameters mentioned above must be tuned to achieve a desired interval time, as given by the following formula:

$$\text{interval time} = \text{interval size} \times \text{frame length}$$

In our case, we found that an interval time of 5,000 ms (or 5 seconds) yields good classification results. The interval size and frame length were chosen to be 150 and 33 respectively.

4. Results

Experimentally, the system successfully detected and correctly recognized dog barks and shattering glass from the ambient noise. The system was tested with more data downloaded from Youtube the same way the testing data was obtained.

We also found that adding a third SVM to recognize gunshots had a negative effect on the system. Indeed, the third SVM returned very large margin values for most of the data it was tested with, including dog barks and breaking glass. Given that we are using a "one versus all" approach, this SVM led the system to incorrectly label most test data as "gunshots." Thus, the system is more robust if we only keep the first two support vector machines, and discard the third one. I trained the system with a dataset consisting of mixed "guns", and with another dataset consisting exclusively of revolver sounds. Using an exclusive type of "gun" had no substantive effect on the classifier.

5. Use Cases

In this section, we highlight certain use cases where this system can prove useful, especially in residential settings. A use case, as suggested by Professor Henning Schulzrinne, is to use this system in the home of people with hearing disabilities. The system can translate sound information to visual cues, in order to inform the user of relevant events, such as the shattering of glass or the barking of dogs. The sound detected and recognized by the system can be displayed in a tablet, or sent as a message to a mobile phone via email or SMS.
Another use case is for this program to act as a security system. Installed in a residential setting, this system can warn its users of relevant events at night while they sleep, or when they are away from their homes. Recognizing sounds such as the shattering of glass, this system can inform users of events such as a break-in by thieves.

This project was designed with the residential settings in mind, but other settings can also benefit from this system. For instance, our sound detection system can be installed in schools. Given the unfortunate rise in school shootings, this system is an inexpensive and easy-to-install solution that can recognize gunshots and warn the authorities in the event of a shooting. For this application, the classifier should be improved first to classify gunshots. Finally, other settings not covered in this report but worth mentioning as an area of further study are businesses and factories.

6. Advantages Over Existing Systems

Several companies, such as Amazon and start-up Canary, sell devices to ensure home security. The devices in question, however, stream the audio data collected from people's homes to the cloud for processing and decision-making. This architecture presents major vulnerabilities to people's privacy. Canary, for instance, does not disclaim how the audio data is managed and stored once it is recorded by its device. It is not in our scope to speculate, but Canary, as many other companies have recently witnessed, is not immune to attacks by malicious entities. Users without a technical background are exposed to major breaches in their home security without knowing.

Our system, on the other hand, is self-hosted. Specifically, the data is processed in the user's tablet or Raspberry Pi itself, and no data is sent to external servers. Moreover, our system has no storage; the audio data recorded is immediately discarded once the classification has been made. The preservation of people's privacy is the main advantage of our proposed system over the commercial products available. Figure 5 shows the contrasting architectures of our system and its competitors to highlight this advantage.

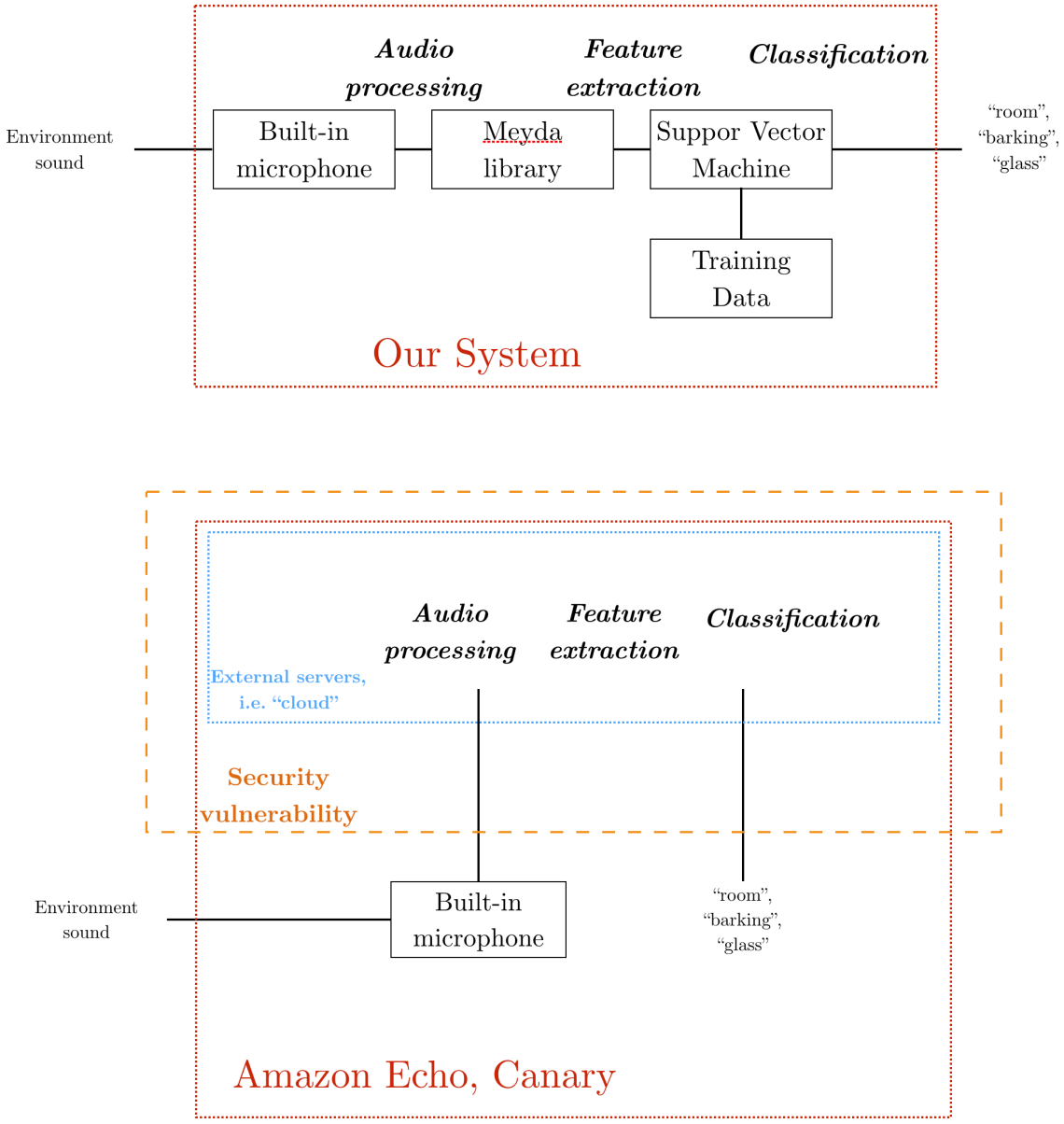

Figure 5: Schematic comparison of our system and commercial products of similar purpose, in order to highglight the main advantage of our system, namely the preservation of users' privacy

7. Conclusion

This report proposes a system for real-time monitoring of the environment by using sound recognition. The system effectively detects the sound of dogs barking and glass shattering. The language used is JavaScript, so this program can run from the browser in a tablet or even in a Raspberry Pi. As a result, users can self-host their own monitoring system, without exposing the privacy of their homes at risk by streaming the data to the cloud.

Bibliography

[1] *Youtube-dl* can be downloaded from the following link:
https://github.com/rg3/youtube-dl/blob/master/README.md#readme
[2] Non-speech Environmental Sound Classification Using Svms With A New Set Of Features, Burak Uzkent, Buket D. Barkana and Hakan Cevikalp
[3] *Meyda* can be downloaded from the following link:
https://github.com/hughrawlinson/meyda
[4] Theoretical proof of Multi-Class Support Vector Machines and the "one versus all" approach can be found here:
https://www.sec.in.tum.de/assets/lehre/ws0910/ml/slideslecture9.pdf