Short Sound Recognition With A Simple Audio Fingerprinting System

Manuele Favero & Lavinia Verzotto

Digital Signal Processing, a.y. 2023/2024

1 Introduction

A collaborative venture between Ogenus SRL and CSC (Centro di Sonologia Computazionale) led to the creation of TIS (Transmission of Information Sound) code, an application that aim to facilitate the transport of information through a sound system. If the QR code need the use of a photo camera to scan an image, TIScode automatically activate an app by waking up the smartphone with an initial 1 second sound and then the info core sound is transmitted and decoded to create a database-pointer to access the most different type of information (links, images, videos, pdfs).
In this project we studied if an audio fingerprinting system can be a suitable solution for TIS code recognition. The recognition module of the system is subjected to combination of TIScodes with different ambient sounds and audio degradation. Furthermore we introduced a novelty that is usually neglected or too simply approximated: the frequency response of smartphone microphones. We calculated a general representation of the frequency responses of seven different highly commercialized phones through experiments in an anechoic chamber. Finally we combined together all these variables to test a simple representation of an audio fingerprinting system for the task of TIScode recognition and classification.

2 Dataset

2.1 TIS codes

TIS (Transmission of Information Sound) codes are jingle-like short sound of about 5 seconds. For this specific project we needed a great variety of sounds to investigate the best and worst cases when dealing with real application. We decided to generate them with MusicGen [1]. MusicGen is a Transformer-based model capable of generating high-quality music samples conditioned on text descriptions or audio prompts and also unconditionally. The MusicGen model can be decomposed into three distinct stages:

1. The text descriptions are passed through a frozen text encoder model to obtain a sequence of hidden-state representations
2. The MusicGen decoder is then trained to predict discrete audio tokens, or audio codes, conditioned on these hidden-states
3. These audio tokens are then decoded using an audio compression model, such as EnCodec, to recover the audio waveform

We generated 100 different sound messages using the unconditionally generation of MusicGen. We only set the maximum number of created tokens to 256 that lead to the generation of about 5 second (generally from 4.96 to 5.12 seconds) length sound with very random features.

2.2 Anechoic Chamber Misurations

We set up our experimental instrumentation as in *Table 1* in an anechoic room. We set the speaker at the height of 80cm, the smartphones of *Table 2* were at the same height as the speaker thanks to a tripod and they were at 1m of distance from the speakers. We generated a sweep of 10 seconds from 20Hz to 20Khz with Isotope Rx, in order to be able to work with a complete spectrum with all frequencies. The generated signals were recorded with the smartphones: our aim was to calculate the frequency response of the smartphone's microphones having the original signal and the registered signals.
At the end of the experiment we had 3 registrations for each smartphone: one in linear scale, one in logarithmic and one in the Mel scale for a total of 3 original signals and 21 recorded signals. Here in *Figure* **??** we can see the spectrograms of the three original signals:

Instrument	**Description**
Anechoic Room	CSC, Via Gradenigo 6/A (PD)
Sound Generation Software	Isotope Rx
Generated Sounds	Sweep in Linear, Log and Mel scale
Sound Length	10 seconds
Frequencies Band	20Hz-20KHz
Speaker	KRK Classic 5
Speaker Distance	1 m
Height	80 cm

Table 1: Experimental kit for the Anechoic Chamber Misurations

Phone	**Model**
Smartphone 1	OnePlus 8T
Smartphone 2	IPhone 13
Smartphone 3	Samsung Galaxy A54
Smartphone 4	Honor 9X
Smartphone 5	Motorola Moto E7
Smartphone 6	Realme 9 Pro
Smartphone 7	Xiaomi Redmi A03

Table 2: Smartphones List

2.3 Noise Datasets

The TIScode should be reproduced in environments where noise is present; for this purpose, we have selected two datasets to simulate real-life situations that could arise.

- Hospital Noise Dataset [2]: in bustling local hospitals, various types of audio noise can be heard, such as children crying in the background, doors opening and closing, conversations in the reception area and waiting room, as well as sounds from equipment movement and operation. This dataset consists of recordings of hospital ambient noise collected from different areas (such as corridors and waiting rooms) of a busy hospital;
- Ambient Sound Dataset [3]: this dataset is a collection of audio files (WAV) with different kind of ambient noise that can be found in situations of the everyday life: wind, white noise, water, thunderstorm, rain, construction, etc.

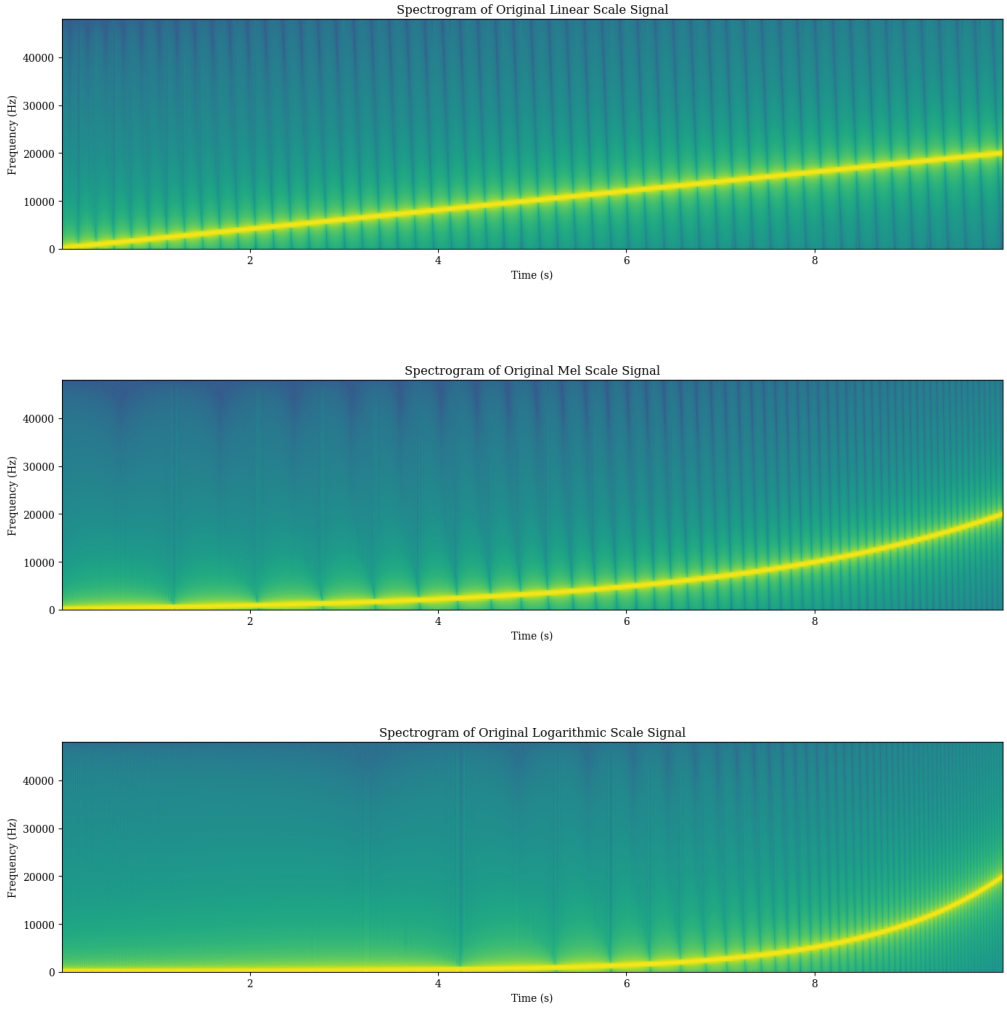

Figure 1: Spectrograms of original sweep signals

3 System Setup

3.1 Frequency Response Calculation

We started by loading two audio signals, one is the original signal (x) and the other one is the same signal recorded in the anechoic chamber (y). These signals are stored in variables x and y, along with their corresponding sampling rates. If either of the signals is stereo, it is converted to mono by averaging the two channels, ensuring that further processing is simpler and more consistent. Since the sampling rates of the two audio files may differ, we checked for any discrepancies and resampled the y signal to match the sampling rate of x. This step is crucial because having consistent sampling rates is necessary for accurate comparison and processing of the signals. Afterward, the signals are adjusted to ensure they are of the same length, this is done by truncating the longer signal to match the length of the shorter one. This alignment is necessary because operations like Fourier Transforms require signals of equal length to produce meaningful results.
The core of this part of the project involves transforming these time-domain signals into the frequency domain using the Fast Fourier Transform (FFT). This transformation provides a frequency-based representation of the signals, denoted as X for the original signal and Y for the recorded signal. With the signals in the frequency domain, the frequency response $H(f)$ is calculated by taking the ratio of $Y(f)$ to $X(f)$. This response essentially characterizes how the original signal x was altered to become the recorded signal y during the recording process.

The next step involves calculating an inverse filter, $H_{inv}(f)$, which is designed to reverse the effects captured in the frequency response $H(f)$. To avoid numerical issues such as division by zero, a small threshold is set, and only values above this threshold are inverted. This inverse filter is then applied to the frequency-domain representation of the recorded signal $Y(f)$. By multiplying $Y(f)$ with $H_{inv}(f)$, we attempted to reconstruct a signal that resembles the original signal x as closely as possible. The resulting frequency-domain signal is then transformed back into the time domain using the inverse FFT. Since numerical errors can introduce small imaginary components during this transformation, only the real part of the signal is retained. Then, we saved this filtered signal as a new audio file, $y_{filtered}$, which represents the cleaned version of the recorded signal.
Finally, we computed metrics like Mean Squared Error (MSE) and Signal-to-Noise Ratio (SNR) to quantitatively assess the accuracy of the filtered signal compared to the original. These metrics help to evaluate how well the filtering process has restored the original signal.

3.2 Audio Fingerprinting System

Audio recognition algorithms had their turning point with the advent of Shazam, which for the first time introduced the so called "audio fingerprinting" [4].
The first step in building similar algorithms is to create fingerprints of tracks - identifying some kind of feature of a song that makes it distinct from other songs. The key to fingerprinting is to look at music in terms of the frequencies of sound that combine to make it, rather than as a string of notes or lyrics over time [5]. The fingerprint can take into account many different features depending on the system: usually Frequency Domain Features (as peaks, distances between peaks), Spectral Features (as Chromagaram, Mel-Frequency Cepstral Coefficients (MFCCs)), Temporal Features (as amplitude variations over time) but also Energy, Harmonic and rhythmic features. Additionally, this kind of fingerprint should be immune to corruption: much like the human ear's ability to filter out background noise and compression artifacts, the fingerprint should be recognisable even in the poorest of conditions.
The method is based on performing FFT (Fast Fourier Transform) on short snippets of the sounds, such as half a second in length, and then have an entire set of key frequencies over time. Our simple fingerprinting system analyze the TIScodes to calculate and store the constellation that characterize the sound with the preeminent magnitude peaks and their position in time. This will allow us to match the fingerprints of the user's recording to any point of the song. With the feature we learned, we can build a constellation of points that characterize the song.
The points in the constellation map are combinatorially associated - each point is paired with several other points to form pairs of frequencies, stored with the difference in time between them. If, for example, two frequencies are converted to 10-bit integers (from 0 to 1024 by placing the exact frequency into a bin) and the time difference between them stored as a 12 bit integer, the pair of points produces a single 32-bit integer hash. This produces many more candidate fingerprints for a song than simply using the constellations, and it is also extremely efficient for the computer to match hashes in the phone recording with hashes stored in the song database [6].
This system works better the longer the recording and the reference songs are, so we wanted to analyze if it is a suitable solution for 5 seconds sound messages. The idea is that smartphones start recording right after the waking-up signal and before the transmission of the infocore there will be a 1 second of silence, useful for allowing slower phones to start recording and eventually use the silence as a recording of environmental noise that can be analyzed and subtracted from the registration to do a first cleaning to the message.
This setup allowed for the recording of the full 5 seconds of TIScode, avoiding decoding issues associated with shorter signals. Finally, the simplest method for matching is to create the hashes for the corrupted audio and see which TIScode in the database has the most matching hashes.

4 Results

4.1 Frequency Response

First, we compared the original signal with a recorded signal, both of which belonged to the same scale (Linear). The frequency response, $H(f)$, derived from these two signals worked very

effectively, as we can see from *Figure 2* where the spectrogram of the original and the filtered signal are very similar. This was evident from the good Signal-to-Noise Ratio (SNR) and low Mean Squared Error (MSE) values of the filtered signal, $y_{filtered}$. SNR is a measure that compares the level of a desired signal to the level of background noise and quantifies how much a signal stands out from the noise (so the higher it is, the better), MSE is a measure of the average squared difference between the estimated values and the actual (true) values (so a good results should have MSE near 0). Subsequently, we saved the previously obtained $H(f)$ and applied it to a new pair of signals from a different scale (Mel). However, this approach did not yield satisfactory results. The SNR was significantly lower, and the MSE was higher, indicating that the filtering process was not as effective when applied to signals from a different scale. As we can see from *Figure 3* the frequency responses of the three scales (Linear, Logarithmic, Mel) are not completely different from each other but are quite similar. However, they are not similar enough to serve as an effective filter for accurately reconstructing the same sound. One reason for this is that the microphones, being of not very high quality, can produce slightly different frequency responses even when recording the same signal.
We have considered other reasons why this approach might not have worked:

- The original frequency response $H(f)$ was calculated based on the Linear scale signals, which likely have different spectral characteristics compared to the Mel scale signals. This means that $H(f)$ is tailored to correct specific distortions present in the Linear scale, and these distortions may not be the same in the Mel scale signals
- The phase relationship between the signals in the Linear scale might differ significantly from that in the Mel scale. If $H(f)$ captures a phase response specific to the Linear scale, applying it to the Mel scale signals could lead to phase mismatches, resulting in poor filtering performance
- If the recording or processing equipment introduced non-linear distortions, these might vary between different scales. $H(f)$ derived from the Linear scale would not account for these non-linearities in the Mel scale, leading to ineffective filtering.

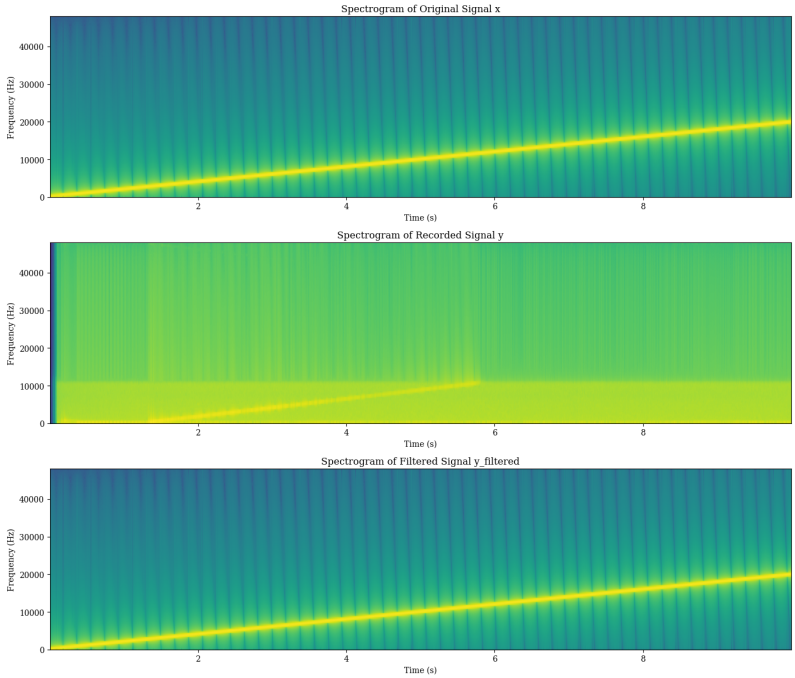

Figure 2: Spectrogram of the Original, Recorded and Filtered signal

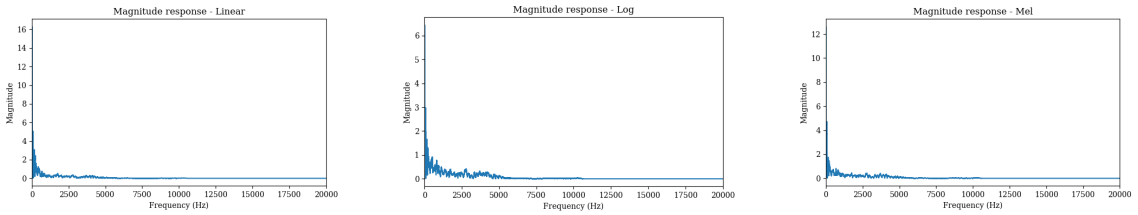

Figure 3: Magnitude of H for Linear, Logarithmic, and Mel Scale of OnePlus 8T

4.2 Sound Recognition Test

We built the sound recognition part of our project in this way:

1. Select one of the two noise dataset
2. Set a the reduction in dB for the TIScode (0, -3, -6, -9)
3. Random choose a TIScode and a noise sample
4. Overlay the two audio samples
5. Try to find a match in the database
6. Repeat points 3,4 and 5 for 500 times

With this algorithm we built histograms of *Figure 4, 6, 7, 8*. Histograms represent the correct matches referred to the two different datasets and the different attenuation values. We, in fact, decided to perform tests with higher and higher attenuation of the TIScode sound to verify how far we could go to go below the 50% threshold. During the process we also calculated the average SNR (without considering the frequency responses) to have a clearer idea of the noise the system can bear (*Table 3*). We can see how the TIScode overlayed with a noise with the same intensity is read correctly in, around, 67% of the cases without any noise correction, audio enhancement or filtering. We focused on the worst cases, but obviously better result can be obtained by diminishing the noise intensity as we can see in *Table 5*. For Samsung Galaxy A54 we also performed test with noise attenuated by 3 and 6 dB (in the histograms for simplicity noise attenuation is displayed as +3/+6, it's not a TIScode attenuation like in the other cases, but the number still indicated the differences in intensity of the two signals). Simulations gave an 87.4% correct matches rate for 3dB noise attenuation and 93.2% for the 6dB one, meaning that the system works quite well when there is not so much noise, taking into account that the SNR for +3dB is -43.27dB on average and -40.16 for 6dB.
Future and in-deep works can surely analyze and include more sophisticated audio fingerprinting systems and up-to-date noise reduction techniques as in [7][8][9]. Further information can be obtained by calculating and testing very recent and very high level smartphones in order to see if microphones capsule are also of higher quality. Among the smartphones we examined, accounting for a 1.5-2% gap due to fluctuations in random simulations, none of them stands out significantly."
The tested phones are representative of the most commonly used models, making them suitable for assessing worst-case scenarios. This includes lower-end models like the Redmi A03 and Moto E7. However, an analysis of higher-end models could also be beneficial.

	Hospital Dataset				**Ambient Dataset**			
TIScode attenuation	**0 dB**	**-3 dB**	**-6 dB**	**-9 dB**	**0 dB**	**-3 dB**	**-6 dB**	**-9 dB**
Average SNR	-52.78dB	-58.79dB	-64.77dB	-70.49dB	-50.77dB	-54.93dB	-62.03dB	-69.39dB

Table 3: Average SNR without considering the frequency response

	Hospital Dataset				Ambient Dataset			
	0 dB	**-3 dB**	**-6 dB**	**-9 dB**	**0 dB**	**-3 dB**	**-6 dB**	**-9 dB**
Without H	67.4%	62.2%	53.4%	40.8%	82.0%	76.6%	69.8%	61.6%
Samsung Galaxy A54	67.2%	60.6%	51.2%	41.6%	83.2%	77.4%	73.8%	64.6%
OnePlus 8T	64.6%	56.0%	46.0%	40.0%	84.4%	75.8%	71.4%	66.8%
iPhone 13	65.2%	58.4%	48.4%	41.0%	85.2%	77.8%	69.8%	63.8%
Honor 9X	72%	61.2%	54.4%	39.2%	82.2%	76.4%	72.6%	62.8%
Motorola Moto E7	67.8%	56.0%	49.4%	41.0%	82.8%	76.0%	67.4%	62.8%
Realme 9 Pro	65.2%	59.0%	48.8%	40.8%	83.6%	79.4%	71.6%	64.0%
Xiaomi Redmi A03	65.6%	59.6%	50.4%	42.8%	82.6%	76.6%	70.4%	63.6%

Table 4: Correct Matches Percentages for Smartphones in Hospital and Ambient Datasets

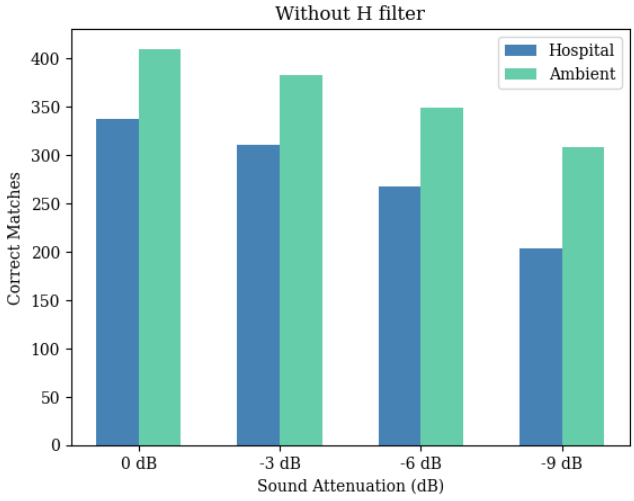

Figure 4: Tests on the system subjected only with the noise of the two datasets

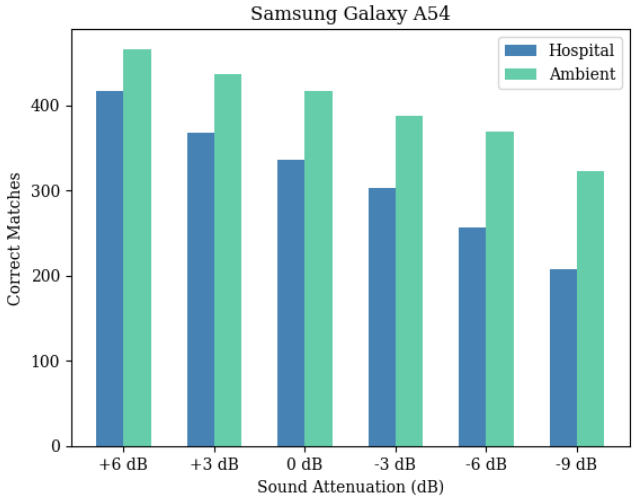

Figure 5: Tests on Samsung Galaxy A54 also with noise attenuation of 3 and 6 db

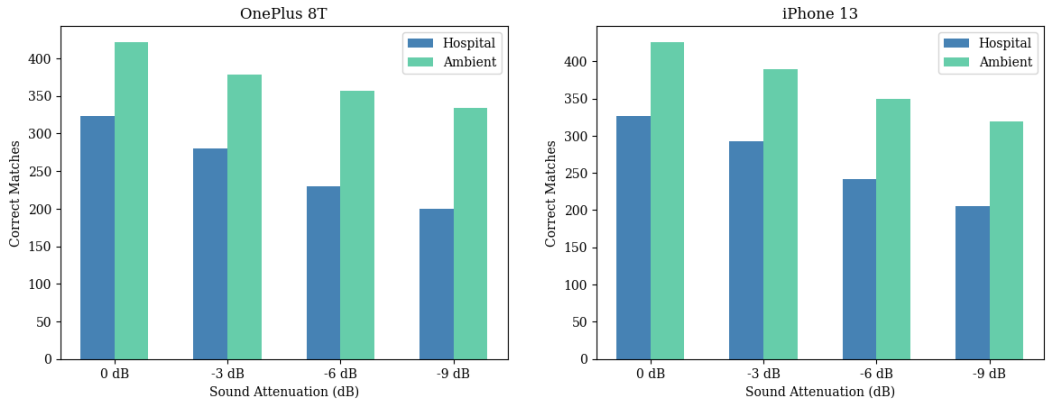

Figure 6: Tests on One Plus 8T and iPhone 13

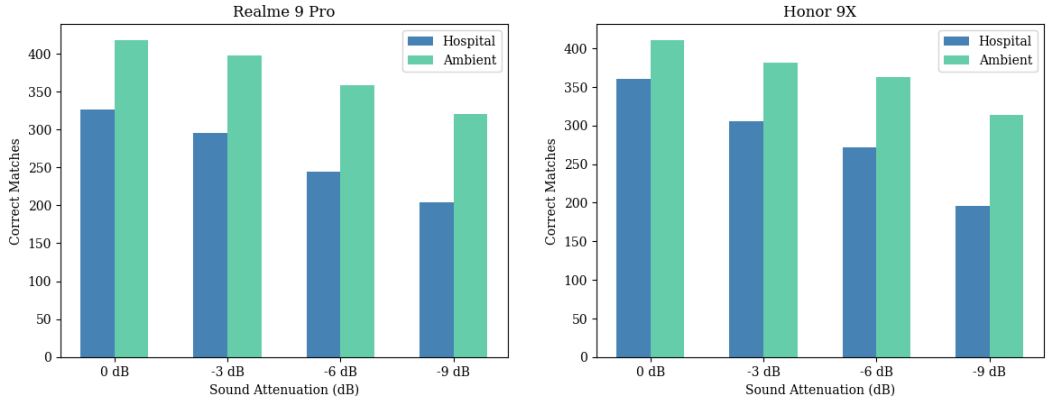

Figure 7: Tests on Realme 9 Pro and Honor 9X

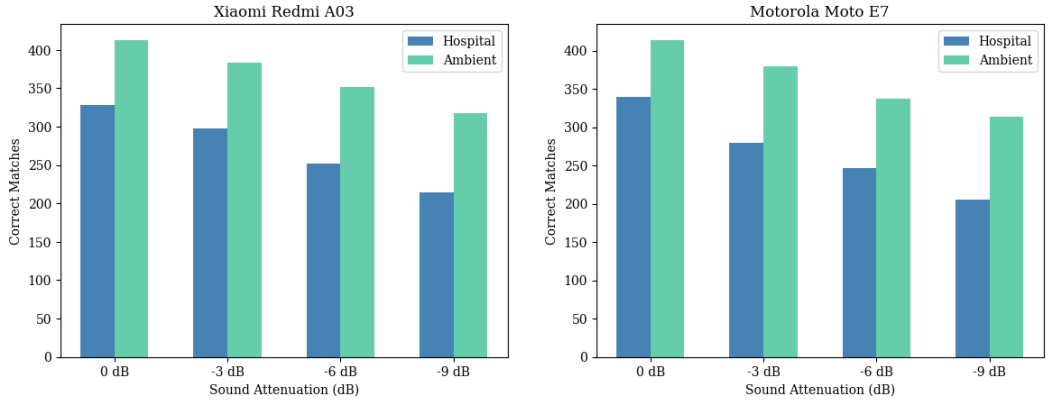

Figure 8: Tests on Xiaomi redmi A03 and Motorola Moto E7

5 Conclusion

This project explored the feasibility of using an audio fingerprinting system for the recognition of TIS (Transmission of Information Sound) codes, focusing on the challenges posed by varying ambient noises and the frequency response characteristics of different smartphone microphones. We began by generating a diverse dataset of TIScodes using the MusicGen model, ensuring a broad range of audio features to test the robustness of our system. The experimental setup included precise measurements in an anechoic chamber to account for the frequency response of various commercially available smartphones.

Our results demonstrated that while the basic audio fingerprinting system is capable of recognizing TIScodes with a reasonable success rate, particularly in less noisy environments, its performance degrades as noise levels increase. Notably, the system achieved up to 67% accuracy without any noise correction when the noise and TIScode were at equal intensity. However, performance improved significantly with reduced noise levels, achieving over 90% accuracy with a 6 dB noise attenuation.

Therefore, we can state that our fingerprinting system outperforms the state-of-the-art TIScode matching system, which relies on harmonic and pitch recognition and achieved only 11% correct matches in real-world testing. The study also highlighted the limitations of applying a frequency response derived from one signal scale to another, showing that such an approach could lead to poor filtering results. This underlines the importance of tailoring the frequency response to specific conditions, particularly when dealing with different spectral characteristics or scales.

Future work could focus on integrating more advanced noise reduction techniques and testing the system on a wider range of smartphones, including those with higher-end microphones, to enhance the overall recognition accuracy. Additionally, exploring more sophisticated audio fingerprinting methods and real-time processing capabilities could further improve the robustness and applicability of the TIScode system in diverse environments.

References

[1] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," 2024.

[2] S. Shuvo, "https://www.kaggle.com/datasets/nafin59/hospital-ambient-noise."

[3] J. H. Solòrzano, "https://www.kaggle.com/datasets/solorzano/ambient-noise/data."

[4] A. Wang, "An industrial strength audio search algorithm.," 01 2003.

[5] M. Strauss, "https://michaelstrauss.dev/shazam-in-python."

[6] S. Ciumac, "https://emysound.com/blog/open-source/2020/06/12/how-audio-fingerprinting-works.html."

[7] N. Kamuni, S. Chintala, N. Kunchakuri, J. S. A. Narasimharaju, and V. Kumar, "Advancing audio fingerprinting accuracy with ai and ml: Addressing background noise and distortion challenges," in *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*, pp. 341–345, 2024.

[8] K. Akesbi, D. Desblancs, and B. Martin, "Music augmentation and denoising for peak-based audio fingerprinting," 2023.

[9] B. A. y Arcas, B. Gfeller, R. Guo, K. Kilgour, S. Kumar, J. Lyon, J. Odell, M. Ritter, D. Roblek, M. Sharifi, and M. Velimirović, "Now playing: Continuous low-power music recognition," 2017.