[COSC-454-S22]
A Pitch-Robust Fingerprint Extraction Scheme

Daniel Flores García

May 24 2022

link to code: https://github.com/dafg05/pitch-robust-fingerprinting

Abstract

As digital media becomes widely available, especially through online distribution, copyright infringement has become more of a concern for owners of intellectual property. In the world of copyrighted audio, a number of copyright detection techniques have been proposed; one of these is fingerprinting. In this paper, we consider a fingerprint extraction scheme devised by Haitsma, et. al. and we formulate an attack to a copyright detection system implemented using this scheme. Such an attack would consist of pitch-shifting a copyrighted audio signal to avoid detection. Furthermore, we present an augmentation of the extraction scheme that consists of extracting pitch-robust fingerprints in addition to the original fingerprint. Finally, we demonstrate that this augmented scheme is robust to pitch-shifting attacks, unlike the original scheme.

1 Introduction

Based on on the idea of using human fingerprints to identify people, an audio fingerprint consists of a collection of (typically handpicked) features of an audio signal that represent the entire signal. A fingerprint serves as index to the metadata of an audio signal. As part of a copyright detection system, we can extract the fingerprint (or potentially multiple fingerprints) of copyrighted audio and store it in a data base. An input signal to a copyright detection system is flagged as copyrighted if its own fingerprint matches an entry in the copyrighted fingerprints database.[1]

Copyright detection systems are subject to evasion-type attacks: in the case of a fingerprinting system, an attacker could degrade the audio signal in some way that would meaningfully change its fingerprint and not be picked up by the system, but to a human the audio would still be considered to be a case of copyright infringement. An example of this would be that of an attacker applying heavy mp3 compression to a popular song to deceive a fingerpringing system. While the audio signal has been changed, a human listening to the song would still recognize the original song, and the audio is legally still considered to be a case of copyright infringement in the US. As such, fingerprinting systems are designed to be robust to common signal degradations: some of these include mp3 compression, linear speed change, equalization, etc.

An example of a fingerprinting scheme is one proposed by Haitsma, et. al., empirically shown to be robust to a number of common signal degradations[2]. This paper is concerned with: 1. showing that this fingerprinting scheme is not robust to pitch shifting —that is, an evasion-type attack that pitch shifts the original signal by more than 50 cents will generally not be flagged as copyrighted— and 2. proposing that by computing some additional fingerprints per audio, we can defend against pitch-shifting attacks.

2 Fingerprint Extraction

2.1 Algorithm

The algorithm for the fingerprinting scheme proposed by Haitsma, et. al. works as follows: We first segment the audio signal into frames; from each of these frames, we extract a 32-bit subfingerprint. The length of a frame is 0.37 seconds, but all frames overlap with a factor of 31/32. Ultimately, a subfingerprint is extracted for every 11.6 milliseconds. For simplicity purposes, we will use the collection of all of the subfingerprints for all frames of a signal as a basic fingerprint. While in practice only storing one fingerprint per copyrighted audio results in a weak granularity for the fingerprinting scheme (we would need an input signal to be the same length as a copyrighted

signal to have a fingerprint match), it suffices to show describe the robustness of the fingerprint scheme.

To choose the features that will dictate the value of each 32-bit subfingerprint, we look at the frequency domain, as the most important perceptual audio features live in it. In our implementation this fingerprinting scheme, we compute an **energy mel-scaled-spectrogram** of the entire signal. The specifications of the spectrogram are as follows:

- Number of FFTs: $0.37 \cdot$ sample rate
- Hop Length = $\frac{0.37}{32} \cdot$ sample rate
- Number of bands: 33
- Min. Frequency: 300Hz
- Max. Frequency: 2000Hz

While Haitsma, et. al. used the Bark Scale to select the 33 bins between 300Hz and 2000Hz, the Mel Scale consists of logarithmically spaced bins, the choice of which approximates human perception in a similar way that the Bark Scale does. As such, in practice, using the Mel Scale instead of the Bark Scale should result in similar classification.

It has been experimentally verified that calculating a subfingerprint using the sign of energy differences (simultaneously along time and frequency axes) results in a fingerprint that is robust to common signal degradations. More precisely, let the energy of band m of frame n be equal to $E(n, m)$, and the mth bit of the subfingerprint of frame n be equal to $F(n, m)$. Each bit of the subfingerprint is defined as:

$$F(n,m) = \begin{cases} \mathbf{1} \text{ if } (E(n,m) - E(n,m+1)) - \\ \quad (E(n-1,m) - E(n-1,m+1)) > 0 \\ \mathbf{0} \text{ if } (E(n,m) - E(n,m+1)) - \\ \quad (E(n-1,m) - E(n-1,m+1)) \leq 0 \end{cases} \quad (1)$$

To compare two fingerprint, we calculate the (BER) Bit Error Rate between two fingerprints. Each subfingerprint is computed using the information found in the mel-scaled spectrogram.

Get BER Pseudocode

```
get_BER(fp1, fp2):
   total_bits = 32 * len(fp1)
   bit_errors = 0
   for in range(len(fp1))
      bit_errors +=
      hamming_distance(fp1[i], fp2[i])
   return bit_errors/total_bits
```

where `fp1` and `fp2` are 2 fingerprints: lists of 32-bit subfingerprints. Note that `len(fp1) == len(fp2) == number of frames` in the audio.

If the BER between two fingerprints is less than 0.35 —our BER threshold— , then the fingerprints are considered to be a match.

2.2 Robustness to common signal degradations

This fingerprinting extraction scheme is empirically shown to be robust to the following signal degradations:

- **MP3 Encoding/Decoding** at 128 Kbps and 32Kbps
- **Real Media Encoding/Decoding** at 20 Kbps
- **GSM Encoding at Full Rate**
- **All-pass Filtering**
- **Amplitude Compression**
- **Equalization**
- **Band-pass Filtering**
- **Time Scale Modification** of +4% and -4%.
- **Linear Speed Change** of +1% and -1%.
- **Noise Addition**
- **Resampling**
- **D/A and A/D Conversion**

Haitsma, et. al. showed that when these degradations where applied to a signal, a match was found between the fingerprints of the altered and original signals (i.e. their BER was below 0.35).

2.3 Pitch-shifting attack

Consider a pitch-shifting attack, which consists of pitch-shifting the original signal by a certain number of cents (note that 100 cents = 1 semitone). Experimentally, we show that the original fingerprinting scheme is not robust to pitch shifting attacks: the BER between the fingerprint of a signal pitch-shifted by more than 50 cents is generally higher than 0.35 To test this, we used excerpts shorter than 6 seconds from audio examples provided by the `librosa` python package. Table 1 shows BERs between the original fingerprint corresponding to each excerpt and a pitch

shifted fingerprint (the amount of pitch-shift is denoted in the first column).

We note that pitch-shifting by an audio signal by a "large number" of semitones subjectively compromises the integrity of the original audio signal. Thus, we limit the amount of pitch-shifting in our experiments to 200 cents in either the positive or negative directions.

3 Pitch-Robust Fingerprinting Scheme

We now propose a pitch-robust fingerprinting scheme. In this scheme, we not only compute the fingerprint of the original, unaltered audio signal, but we also compute a number of fingerprints of pitch-shifted versions of the original signal. We call this collection of fingerprints *pitch-robust fingerprints*. Instead of storing a single fingeprint per copyrighted signals, we store several pitch-robust fingerprints per signal in our fingerprint database.

To determine if an audio signal matches one of the copyrighted audio signals in our data base, we calculate the BERs between the fingerprint of an input audio signal and each of the pitch-robust fingerprints of the copyrighted audio signal. We say that an input audio signal matches another signal if the minimum of the BERs is less than the BER threshold.

Is Match Pseudocode

```
is_match(input_audio, cr_audio):
   BERs = []
   input_fp = get_fp(input_audio)
   cr_fps = get_robust_fps(cr_audio)
   for fp in cr_fps:
      BERs.append(get_BER(input_fp, fp))
   return min(BERs) < 0.35
```

The choice of how many pitch-robust fingerprints and how much to pitch shift each one is an important question. We want to minimize the number of pitch-robust fingerprints we store, as each fingerprint stores a collection of subfingerprints, which are stored at a rate of one subfingerprint per 11.6 milliseconds. As such, a 3 minute song would require $(180s/0.0116s) \approx 15517$ 32-bit subfingerprints per fingerprint. Too many pitch-robust fingerprints per signal would consume a significant amount of storage space.

Experimentally, we observed that if an audio signal is pitch shifted by 40 cents or less, then the BER of the original signal and the pitch shifted signal is going to be less than 0.35. As such, we choose the amounts of pitch shifing applied to the original signal for each pitch-robust-fingerprint to be (in cents)

```
robust_shifts = [-160, -80, 0, 80, 160].
```

These choices ensure that any attack that consists of pitch shifting a signal between -200 cents and 200 cents will fail, as a pitch-shift in that range will be 40 cents or less away from the pitch shift applied to one of the pitch-robust fingerprints. Therefore, the BER between the fingerprint of the input signal and at least one of these robust fingerprints should be less than our chosen BER threshold of 0.35.

Extract Pitch-Robust Fps: Pseudocode

```
get_robust_fps(audio):
   robust_fps = []
   for cents in robust_shifts:
   shifted = pitch_shift(audio, cents)
      robust_fps.append(get_fp(shifted))
   return robust_fps
```

See table 2 for results of pitch shfiting `brahms.wav` and using it as an input to the pitch-robust fingerprinting schemes.

4 Conclusions

We have shown that a copyright detection system that uses the Haitsma, et. al. fingerprinting scheme is vulnerable to pitch-shifting attacks, but this vunerability can be defended against by precomupting pitch-robust-fingerprints. The idea behind the pitch-robust fingerprint extraction scheme is applicable to other degradations; if someone were to discover that another signal degradation d evades the simple fingerprinting scheme, then precomputing d-robust fingerprints could be a simple yet effective defense.

For implementations of the simple fingerprinting scheme that compute multiple *fingerprint blocks* per audio signal instead of computing a single fingerprint, the memory requirements for our pitch-robustness augmentation could be too large. For this reason, further research could focus on devising a more rigoruous mathematical method of accurately flagging pitch-shifted copyrighted audio that does not require storing extra fingerprints per signal.

A worry one could have about our pitch-robust fingerprinting scheme is that it could increase the false postive of a copyright detection system. Although we guess that it doesn't, future work could construct experiments that show this.

Pitch-shifting (in cents)	Brahms	Choice	Trumpet	Fishin
28.6	0.23	0.23	0.19	0.23
-28.4	0.24	0.22	0.16	0.24
31.2	0.26	0.25	0.2	0.23
-117.6	**0.59**	**0.56**	**0.47**	**0.58**
123.3	**0.6**	**0.55**	**0.52**	**0.57**

Table 1: BERs between fingerprint of original signal and fingeprint of pitch-shifted signal; pitch shift in cents. BERs of successful attacks are bold.

Pitch shifting (in cents)	-160	-80	0	80	160
49.2	0.528	0.603	0.365	**0.278**	0.578
96.7	0.521	0.555	0.583	**0.194**	0.439
-88.2	0.479	**0.101**	0.543	0.562	0.537
196.1	0.522	0.513	0.526	0.605	**0.305**

Table 2: BERs between fingerprint of pitch-attacked input signal and each of the pitch-robust fingerprints. Pitch shift in cents. Audio signal: `brahms.wav`
Min BER shown in bold.

References

[1] Saadatpanah, P., Shafahi, A., and Goldstein, T. (2020, November). Adversarial attacks on copyright detection systems. In International Conference on Machine Learning (pp. 8307-8315). PMLR.

[2] Haitsma, J., and Kalker, T. (2002, October). A highly robust audio fingerprinting system. In Ismir (Vol. 2002, pp. 107-115).