Music Recognition System

Moein Sorkhei

Abstract

As the galloping technology of producing and playing music thrives, more people tend to listen to the music they like. Furthermore, it is quite common that we hear part of a song somewhere on a show or in a video on YouTube and feel we are interested to listen to the whole music. Hence, providing systems that can recognize what music is being played – namely *music recognition systems* – is of paramount importance. In this project, we implemented a music recognition system from scratch and evaluated it using a small dataset

1. Introduction

Recognizing a song being played has always been of interest to music lovers. The techniques for doing so, however, have recently been proved to be so successful given the advances in the field of digital signal analysis. There are popular music recognizers, namely *Shazam* and *TrackID*, which gained much popularity in light of their satisfactory performance in the task of recognizing a song being played. Hence, working in this interesting field and extending the current techniques is potentially an important task. To get familiar with the basics of this field and learn the fundamental techniques used in this field, we implemented a music recognition system with a limited dataset and evaluated its performance in noisy conditions.

In Section 2 we give some background in this area. Next, we explain our method and what approach we took to for implementation in Section 3, and then bring the results of the evaluation of the system in both noisy and noiseless environments in Section 4. In Section 5, we discuss the findings and possible reasons behind some weaknesses of the system. Finally, in Section 6 we present our conclusions.

2. Background

Analyzing audio, in any form, is an important task, as there is a lot of valuable hidden information in audio. In essence, the task of processing and analyzing audio and speech paves the way for us to extract hidden information, and use it for different purposes. One of the commonly used applications of speech recognition, for instance, is to authenticate the person who is talking. Similarly, analyzing a song being played could help us understand the main features of that song. These features can then be used to retrieve similar songs, which is a significant task for popular applications such as Spotify. Hence, learning the principle techniques in this area is potentially important. In this work, however, we use these features for the task of music recognition.

3. Method

When it comes to visualizing audio, the first image that strikes to our mind is a signal fluctuating over an axis. This common visualization is mainly in the *time domain*, that is, the value of the signal in each moment represented by a number. This could easily be obtained using almost any

tool for recording audio, like a microphone. The following shows the time domain representation of two popular songs for 30 seconds.

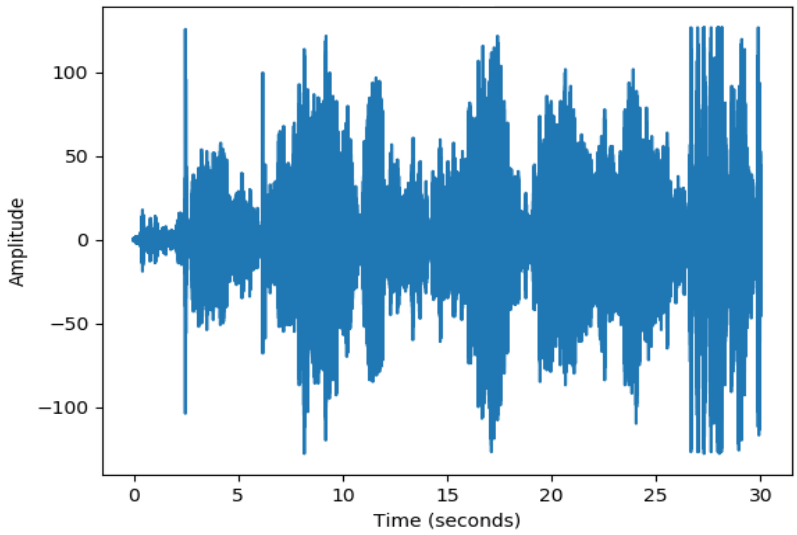

Figure 1: The time domain representaion of The Second Waltz

The time domain representation of another popular song, namely, *The Blue Danube*, is as follows:

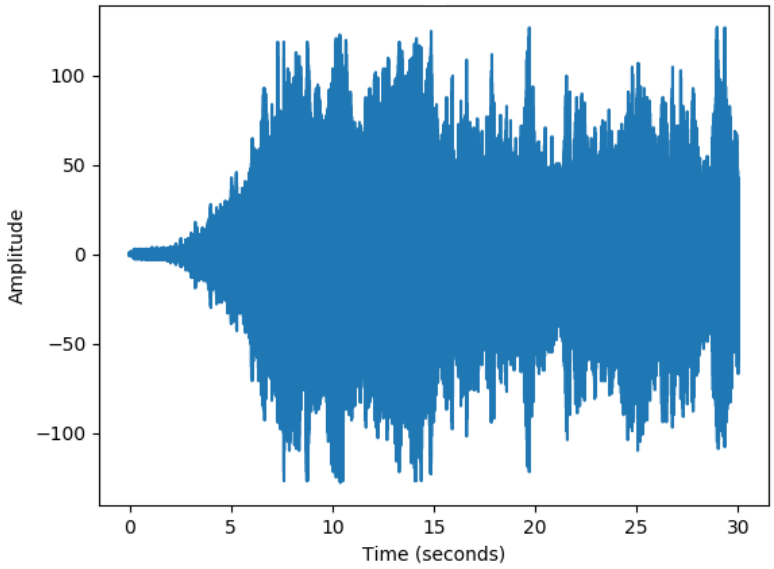

Figure 2: The time domain representation of The Blue Danube

While the time domain representation of the song, and in general audio, has some information in that, it is quite common to convert it into another representation called the *frequency domain*. There are a lot of ways to do so, and we use *fast Fourier transform (FFT)* to do so. Basically, this conversion allows us to extract the frequencies present in the audio, and also enables us to exploit these frequencies and use them as distinguishing factors of each song. Using [1], the frequency representation of these two songs (for their total length) are as follows.

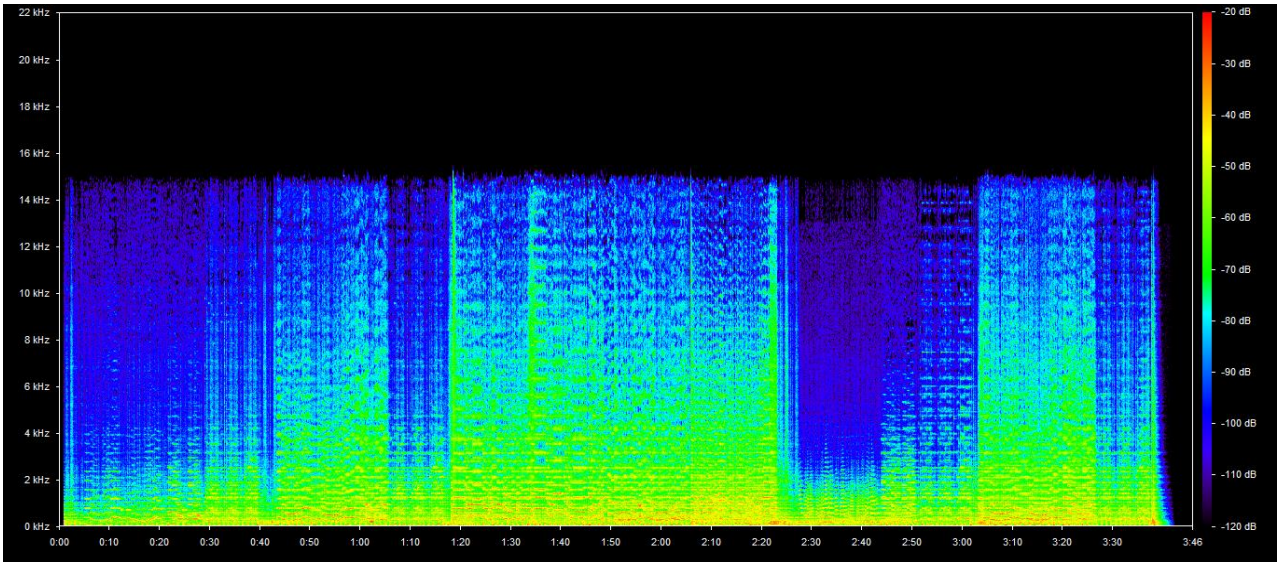

Figure 3: The frequency domain representation of The Second Waltz

The frequency domain representation of The Blue Danube is as follows:

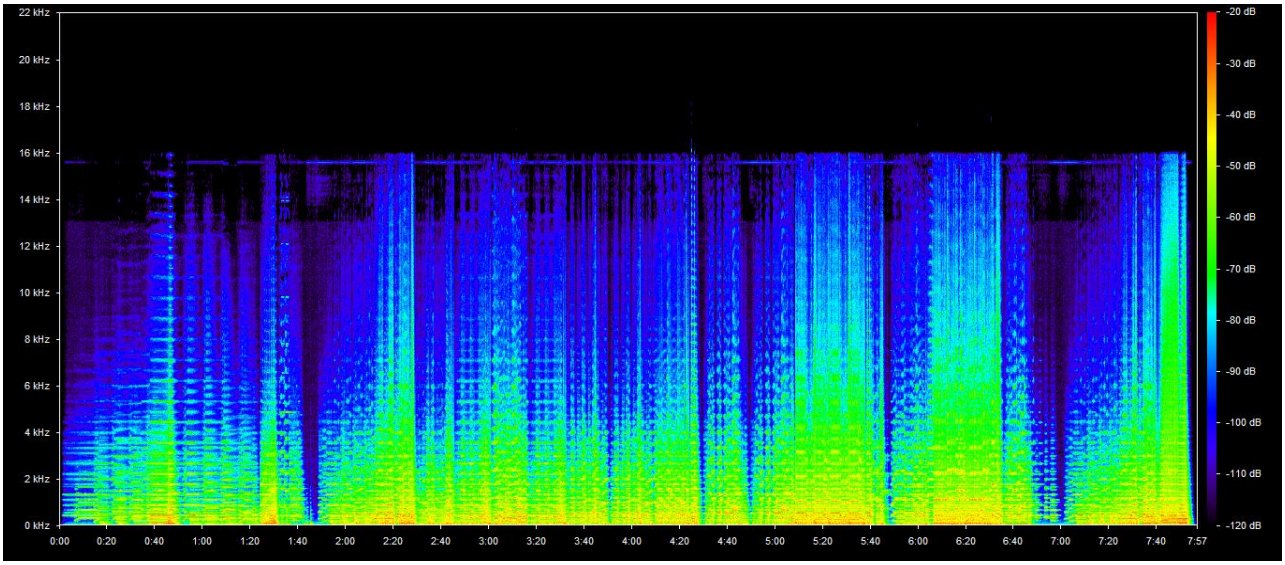

Figure 4: The frequency domain representation of The Blue Danube

It can be seen that in this domain, we have a lot of valuable information. In fact, other than knowing which frequencies exist in each interval, we know the amplitude of each frequency, which helps us distinguish the songs. A notable feature, which is common in a typical song, is that most of the red dots – the frequencies with the highest amplitudes – lie in the range below 1000Hz. The reason for that is the nature and acoustic structure of the typical musical instruments used in songs. This enables us to discard rare low-amplitude frequencies which leads to much more efficient computation. Of course we lose some amount of information by restricting our attention to such a frequency range, but it's in fact a trade-off between accuracy and computational cost.

The effective implementation of a music recognition system relies on splitting a song into different chucks, and for each chuck, extract the frequencies that have the highest amplitude values (red dots). We call these frequencies *major frequencies*.

Given the wide potential range of frequencies each song might have, we need to choose certain ranges of frequencies in which we believe most of the major frequencies of a typical song lie (as already described). Considering the most common musical instruments, we chose to only consider frequencies that lie in the range of 40Hz to 300Hz. Essentially, for each chunk of data, after applying the frequency domain conversion, we divided this span into five ranges, and used the highest frequencies in each range (representatives of such a range) as a fingerprint of that specific chunk. Hence, each song would consist of a number of chunk fingerprints which are unique to that song.

This technique could help increase the robustness of the system, as in the noisy environments, the frequencies with the lowest amplitudes are easily sacrificed, and those with higher amplitudes are hopefully the ones that survive and can be used to distinguish the song being played.

It is naturally possible to have a lot of common chunks of data for similar songs. However, even by having a limited number of chunks in similar between two songs, by taking into account a sufficient number of chunks in succession in the song being played, we are able to uniquely identify the song, as the “sequence” of chunks is inevitably unique to each song. This is the main technique that the system utilizes to differentiate between songs with potentially similar intervals. Having the chunk fingerprints, we are able to store each song in the database by a series of fingerprints unique to that song. The following is the top-down structure of the system.

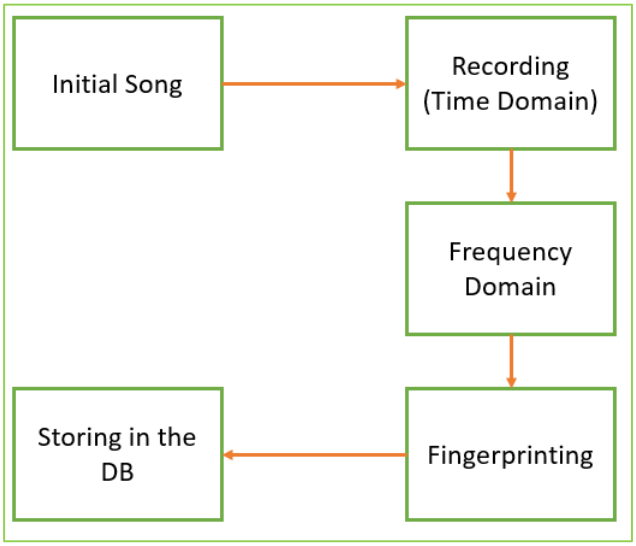

Figure 5: The top-down structure of the system

4. Experimental Results

In this section, we bring the experimental results of running the music recognition program in different environments. Particularly, we have chosen to evaluate the system in an ideal environment, that is, without any noise to see how the system performs. Other than the ideal environment, we have evaluated the system in noisy environments with two common noise types,

namely *babble noise* and *white noise*. Specifically, four different noise levels are chosen, and hence we evaluated the system using four different values of *signal to noise ratio (SNR).* We increased the level of noise to compare the performance of the system in different noise levels. Based on [2], a common way to compute SNR is using the following formula:

$$\mathrm{SNR} = \left(\frac{A_{signal}}{A_{noise}}\right)^2.$$

Here, *A* represents the amplitude of the signal or noise, so the formula is basically comparing the amplitude of the signal and noise. According to [3], in cases that the signal is swinging above and below a specific reference, but is not necessary sinusoidal, which is the case in our situation, the peak amplitude is often used, which is the maximum absolute value of the signal when the reference is zero. Finding the peak amplitude in a signal which corresponds to the song being played or the noise of the environment is straightforward, because it is already implemented in our recording sub-system which finds the values of the signal in different times. Hence, to compute the SNR for the evaluation process, we took the average over the peak amplitude of all the songs (which were approximately near each other, so the average is a reliable representative) and divided it by the peak amplitude of the noise (which we changed in each scenario to obtain different SNRs).

In the evaluation process, we chose to see the performance of the system in an ideal, noiseless environment as well as four levels of noise with corresponding SNRs respectively: 36, 9, 2.25, and 1.44. Clearly, the higher the SNR is, the more powerful the main signal is in comparison to the noise. As a result, we gradually increased the noise which led to lower SNRs, as can be seen. Next, we tested our system with all the 50 songs we had in our database, and computed the accuracy of recognition. As for the diversity of the songs in the database, we chose about half of the songs to be songs with smoothing piano being played (which possibly have a lot of silence in between and a slow-speed rhythm) and the other half to be energetic songs which have an active singer and a relatively high-speed rhythm). We used [4, 5] for the generation of such kinds of noises.

We tested the system on all the songs by playing the first 20 seconds of each of them. The following table shows the accuracy of recognition in different configurations: the ideal environment as well as four different noise levels of two different noise types.

SNR/Noise Type	Accuracy in White Noise	Accuracy in Babble Noise
No noise	62%	62%
36	48%	42%
9	34%	40%
2.25	16%	26%
1.44	12%	22%

Table 1: Accuracy of the recognizer in different noise levels

We also bring an illustrative figure which explains the behavior of the system in the noisy environments and compares the effect of the specific types of noise on the performance of the system.

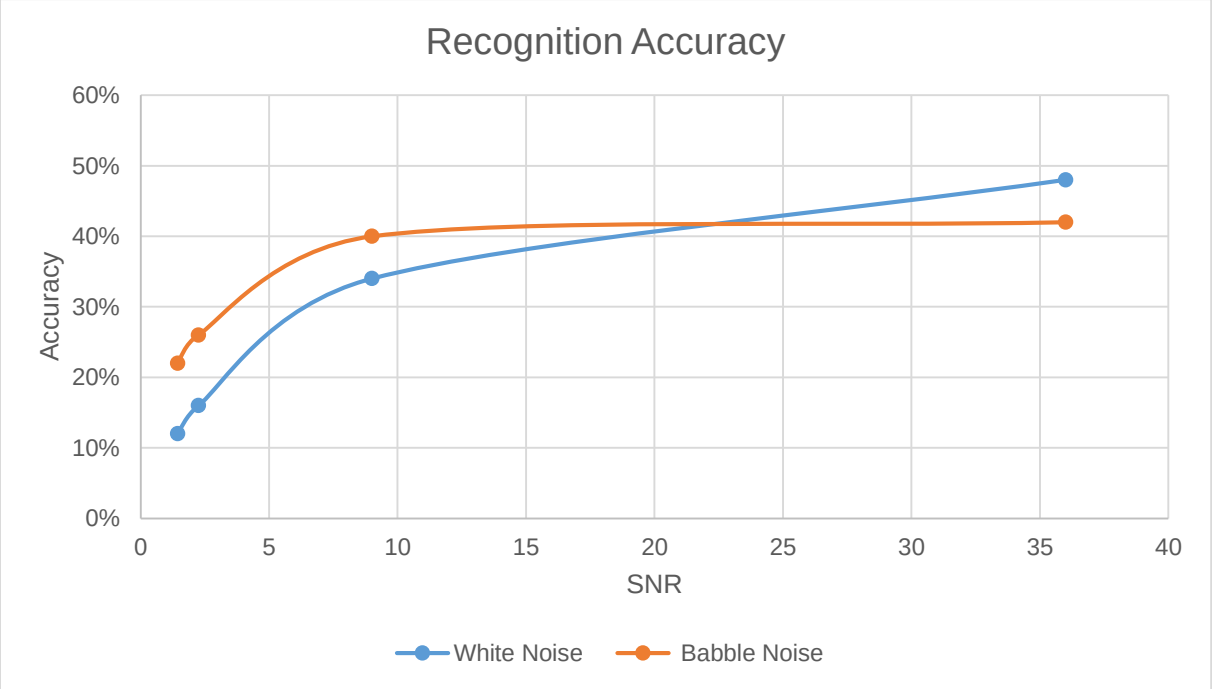

Figure 6: Accuracy of the recognizer with white and babble noise

Regarding the exact percentage of the accuracy of the recognizer, it is noteworthy to say that given the limited database we used, we should not heavily rely on the results to decide whether the current implementation is robust enough or not. As mentioned earlier, we have intentionally decided that half of the songs be “hard” to recognize and the other half be relatively easier for figuring out the weaknesses of the current state of the system. One could potentially use 50 songs with totally different nature and distinguishing features which make it quite easy for the recognizer to find the best match and achieve considerably high accuracy; however, we believe that the result in this case would be positively biased and hence we avoided to have such a database. Rather, having the current database paves the way to have a comparative analysis of the system with respect to each song, which we elaborate upon in the following section.

In order for better comparison, we show detailed results as for the performance of the system in such environments with respect to each song. Particularly, one may not be interested in the performance of the system for each individual song and may pay more attention to the overall performance of the system. However, given the limited database by which we evaluated the system, we are somewhat aware of the nature of each song we have in the database, and taking into account the performance of the system on each individual song gives rise to better understanding of where the system might be weak at, which is undoubtedly of paramount importance for further improvement of the system.

Song Name/SNR	No noise	36	9	2.25	1.44
Never Let You Go	Correct	Wrong	Wrong	Wrong	Wrong
2 Become 1 - 3rd Movement	Wrong	Wrong	Wrong	Wrong	Wrong
Revelations	Wrong	Wrong	Wrong	Wrong	Wrong
Stay	Wrong	Wrong	Wrong	Wrong	Wrong
Blooming Romance	Correct	Wrong	Wrong	Wrong	Wrong
Memories	Correct	Wrong	Wrong	Wrong	Wrong
Hope	Correct	Correct	Correct	Wrong	Wrong
Inspiring Piano	Wrong	Wrong	Wrong	Wrong	Wrong
Elegiac	Wrong	Wrong	Wrong	Wrong	Wrong
Going Home	Correct	Correct	Correct	Correct	Correct
Our World	Wrong	Wrong	Wrong	Wrong	Wrong
Crossroads	Wrong	Wrong	Wrong	Wrong	Wrong
All the Good Times	Correct	Correct	Correct	Correct	Wrong
Emotions	Correct	Correct	Wrong	Wrong	Wrong
See You Again	Correct	Correct	Wrong	Wrong	Wrong
I Waited for You	Wrong	Wrong	Wrong	Wrong	Wrong
Desolation	Correct	Correct	Wrong	Wrong	Wrong
Aurora	Wrong	Wrong	Wrong	Wrong	Wrong
No Name	Correct	Correct	Wrong	Wrong	Wrong
I Waited for You	Wrong	Wrong	Wrong	Wrong	Wrong
Heartsong	Correct	Correct	Wrong	Wrong	Wrong
Lovely	Wrong	Wrong	Wrong	Wrong	Wrong
A Light Within	Correct	Correct	Wrong	Wrong	Wrong
Hereafter	Wrong	Wrong	Wrong	Wrong	Wrong
Made of Stardust	Correct	Correct	Wrong	Wrong	Wrong
Square Groove	Correct	Correct	Correct	Wrong	Wrong
I'm Happy	Correct	Correct	Correct	Correct	Correct
El Paradiso	Wrong	Wrong	Wrong	Wrong	Wrong
Schwerelos	Correct	Correct	Correct	Wrong	Wrong
Tagebuch	Wrong	Wrong	Wrong	Wrong	Wrong
Tanzen	Correct	Correct	Correct	Wrong	Wrong
Ich will nur tanzen	Correct	Correct	Correct	Wrong	Wrong
Falling in Love with You	Correct	Wrong	Wrong	Wrong	Wrong
Ozean	Wrong	Wrong	Wrong	Wrong	Wrong
Nummer Eins	Wrong	Wrong	Wrong	Wrong	Wrong
Konfetti im Kopf	Correct	Correct	Correct	Wrong	Wrong
Nimm mich in dem Arm	Wrong	Wrong	Wrong	Wrong	Wrong
Queen	Correct	Correct	Correct	Wrong	Wrong
Knots (feat. klei)(feat. klei)	Correct	Wrong	Wrong	Wrong	Wrong
I Want It That Way	Correct	Wrong	Wrong	Wrong	Wrong
Mmm Mmm Mmm	Correct	Correct	Correct	Correct	Correct
Crush	Correct	Correct	Correct	Wrong	Wrong
Save The World (feat. Axelle)	Correct	Wrong	Wrong	Wrong	Wrong
Terra Titanic	Correct	Correct	Correct	Correct	Correct
Liebficken	Wrong	Wrong	Wrong	Wrong	Wrong
Be The One (feat. Perttu & Mogli)	Correct	Correct	Correct	Correct	Correct
Labe laut	Correct	Correct	Correct	Correct	Correct
Warriors	Correct	Correct	Correct	Wrong	Wrong
Midnight Sun	Wrong	Wrong	Wrong	Wrong	Wrong
Runaway Train	Correct	Correct	Correct	Correct	Wrong

Table 2: Detailed results of the performance of the system in white noise

Song Name/SNR	No noise	36	9	2.25	1.44
Never Let You Go	Correct	Correct	Correct	Wrong	Wrong
2 Become 1 - 3rd Movement	Wrong	Wrong	Wrong	Wrong	Wrong
Revelations	Wrong	Wrong	Wrong	Wrong	Wrong
Stay	Wrong	Wrong	Wrong	Wrong	Wrong
Blooming Romance	Correct	Wrong	Wrong	Wrong	Wrong
Memories	Correct	Correct	Correct	Correct	Correct
Hope	Correct	Correct	Correct	Correct	Correct
Inspiring Piano	Wrong	Wrong	Wrong	Wrong	Wrong
Elegiac	Wrong	Wrong	Wrong	Wrong	Wrong
Going Home	Correct	Correct	Correct	Correct	Correct
Our World	Wrong	Wrong	Wrong	Wrong	Wrong
Crossroads	Wrong	Wrong	Wrong	Wrong	Wrong
All the Good Times	Correct	Correct	Correct	Correct	Correct
Emotions	Correct	Correct	Correct	Correct	Wrong
See You Again	Correct	Wrong	Wrong	Wrong	Wrong
I Waited for You	Wrong	Wrong	Wrong	Wrong	Wrong
Desolation	Correct	Wrong	Wrong	Wrong	Wrong
Aurora	Wrong	Wrong	Wrong	Wrong	Wrong
No Name	Correct	Correct	Correct	Correct	Correct
I Waited for You	Wrong	Wrong	Wrong	Wrong	Wrong
Heartsong	Correct	Correct	Correct	Correct	Correct
Lovely	Wrong	Wrong	Wrong	Wrong	Wrong
A Light Within	Correct	Correct	Correct	Wrong	Wrong
Hereafter	Wrong	Wrong	Wrong	Wrong	Wrong
Made of Stardust	Correct	Wrong	Wrong	Wrong	Wrong
Square Groove	Correct	Wrong	Wrong	Wrong	Wrong
I'm Happy	Correct	Correct	Correct	Correct	Correct
El Paradiso	Wrong	Wrong	Wrong	Wrong	Wrong
Schwerelos	Correct	Correct	Correct	Wrong	Wrong
Tagebuch	Wrong	Wrong	Wrong	Wrong	Wrong
Tanzen	Correct	Wrong	Wrong	Wrong	Wrong
Ich will nur tanzen	Correct	Correct	Correct	Correct	Correct
Falling in Love with You	Correct	Wrong	Wrong	Wrong	Wrong
Ozean	Wrong	Wrong	Wrong	Wrong	Wrong
Nummer Eins	Wrong	Wrong	Wrong	Wrong	Wrong
Konfetti im Kopf	Correct	Correct	Correct	Wrong	Wrong
Nimm mich in dem Arm	Wrong	Wrong	Wrong	Wrong	Wrong
Queen	Correct	Correct	Correct	Wrong	Wrong
Knots (feat. klei)(feat. klei)	Correct	Correct	Correct	Correct	Wrong
I Want It That Way	Correct	Correct	Wrong	Wrong	Wrong
Mmm Mmm Mmm	Correct	Correct	Correct	Correct	Correct
Crush	Correct	Wrong	Wrong	Wrong	Wrong
Save The World (feat. Axelle)	Correct	Correct	Correct	Wrong	Wrong
Terra Titanic	Correct	Correct	Correct	Correct	Correct
Liebficken	Wrong	Wrong	Wrong	Wrong	Wrong
Be The One (feat. Perttu & Mogli)	Correct	Wrong	Wrong	Wrong	Wrong
Labe laut	Correct	Correct	Correct	Correct	Correct
Warriors	Correct	Correct	Correct	Wrong	Wrong
Midnight Sun	Wrong	Wrong	Wrong	Wrong	Wrong
Runaway Train	Correct	Wrong	Wrong	Wrong	Wrong

Table 3: Detailed results of the performance of the system in babble noise

5. Discussion

Having the results of the performance of the system, there are notable findings that we discuss in this section.

In the first place, the comparison between babble and white noise shows that each of them could potentially affect the performance of the system. Figures shows that increasing the level of white noise (or in other words reducing the SNR) has reduced the performance of the system, especially in the least two SNRs. While in the environment with the highest SNR we see that white noise has a less distorting effect than babble noise, by increasing the level of noise (decreasing SNR), white noise contributes considerably more to the distortion of the frequencies used by the system to recognize a given song, which results in a considerable decrease in the accuracy. In fact, the accuracy of recognition is much higher in the cases of low SNR with babble noise in comparison with white noise. Also, we see that there is no considerable difference in the performance of the system in the two highest SNR values of the babble noise case. Rather, the destructive effect of babble noise could clearly be seen in the least SNR cases where the accuracy has considerably dropped.

Furthermore, taking into account the nature of the songs in our database, we see in Table 1 and Table 2 that the number of wrongly recognized cases is much higher in the first half of the songs (for example, see the first SNR column). As a matter of fact, these songs are the peaceful ones with slow-speed melodies and a lot of silence intervals in between, which makes it difficult for the recognizer to find out the best match unless it listens to the song being played for a relatively long time. On the other hand, the second half of the songs, as mentioned earlier, basically have an active singer in them with intense sounds in them and high peak values which are not distorted as easily as the other ones. Hence, the recognizer generally performs better when tested using these songs.

The fact that we used the first 20 seconds of each song in the database for the evaluation of the recognizer may even better explain why it cannot perform well in the first half of the songs. These peaceful songs, like most of the soothing songs, start with relatively silent intervals and with very low-intensity signals which could easily be distorted even with a medium level of noise. Therefore, it is quite hard for the recognizer to find them in the database in a noisy environment simply by listening to the first 20 seconds. Conversely, the other type of songs start with high peaks from scratch, and hence the recognizer finds them more easily. Please note that the specific 20 seconds of each song for evaluation was simply a choice we have made, and extensive further evaluations can be done as discussed below.

Again, it is noteworthy that the evaluation process we went through enables us to have a comparative analysis, like what we discussed so far, and we believe it is not a decent measurement to draw absolute conclusions on the performance of the system. To do so, the first important step is to add a great number of songs to the database, since the similarities in the songs (like the case in our peaceful songs) would result in lower accuracy. Also, to get unbiased results of the performance of the system to rely on, one should split each song into some parts and test to see how the system performs on each of those parts and average the accuracy, as some long songs like may start and end with literally silent intervals.

6. Conclusions

In this project, we implemented a music recognition system and evaluated it under different noise conditions to see how it performs in those conditions. We reported the results, which potentially show how each of the noise types we used, babble and white noise, contributed to the distortion of the frequencies the system uses for recognizing songs. Further improvements and evaluations, as discussed in the report, could be done on the system. We could use re-define the range of frequencies we believe we can find the unique features in, and possibly consider much higher frequencies or consider smaller ranges, but this leads to a higher cost of computation for the algorithm and more time needed by the recognizer to identify songs. Furthermore, to more extensively test the performance of the system, one should add a great number of songs and tests the system with different intervals of diverse songs.

References

[1] http://spek.cc/

[2] https://en.wikipedia.org/wiki/Signal-to-noise_ratio

[3] https://en.wikipedia.org/wiki/Amplitude

[4] https://en.wikipedia.org/wiki/White_noise

[5] https://mynoise.net/NoiseMachines/babbleNoiseGenerator.php