Shazam

# Introduction

Write catchy bullshit

We will be implementing and evaluating the performance of an audio fingerprinting and recognition system in python. Our main objective is to implement a system that can successfully fingerprint music files and recognize tunes that have already been fingerprinted with at least 80% accuracy. We also want find out through experiments how robust our system is against white noise. Another interesting question we want to answer is how many seconds of audio recording we need for successful recognition. We hope to learn a lot about audio preprocessing, fingerprinting, and matching techniques.

# Methods and Design Specification

## Data Collection and Preprocessing

All of the songs we used for were collected from freemusicarchive.org. All songs from this website are free to use and download. For our project, we collected 10 random songs for 10 different genres in order to have a wide array of different styles of music, while still having multiple samples from the same genre to test against similarity.

Our preprocessing pipeline is separated into two stages. First, we get the songs into a format tenable for use with the libraries and technologies we will be running. Specifically, we converted the song files which were in stereo mp3 format into mono wav files using AudioSegment from pydub. We then took the average of the two channels and wrote the result into a wav file. For the second part, we apply a bandpass filter to remove all frequencies below 10hZ and above 10kHz. From there, we downsample the data by a factor of 4, decreasing our sampling frequency from 44100Hz to 11025Hz.

ADD PREPROCESSING PIPELINE

## Fingerprinting

* Spectrogram
* Find peaks
* Generate Hashes

## Database

* Include database pseucode
* Explain what we will be storing in the database and general idea of how it will be used

## Matching

* Explain absolute and relative offset, and explain why we use the difference for matching purposes

# Experiment Setup

## 1. Recognition Accuracy as a Function of Recording Duration

## 2. Noise Resistance

Choose recording duration based off results from experiment 1. Then measure recognition accuracy when more and more noise is added to the original samples.

Signal to noise ratio

# Results and Analysis

## 1. Recognition Accuracy as a Function of Recording Duration

Measure accuracy of recognition when the recording is 1s,2s,…,15s.

## 2. Noise Resistance

Choose recording duration based off results from experiment 1. Then measure recognition accuracy when more and more noise is added to the original samples.

Signal to noise ratio

# Discussion

Lots of parameters we chose by inspection.

# Future Work

Use a real database, like mySQL faster query time, and don’t have to fingerprint everytime you want to recognize could just have a session running

Machine learning model to tune parametes. Bandpass high,low parameter tunig, etc. fingerprinting parameters, window\_size, fan\_value, min amplitude, etc which all play a role in determining the number of fingerprints generated per song, to maximize accuracy o;ver a large eough dataset with songs from many genres. We just hard coded some values, but this could easily be tuned to maximize performance.

More time analyzing the trade-off of having more vs less fingerprints, and which hyperparameters can help improve certain weaknesses in our algorithm.

## 3. Impact of Number of Fingerprints on Performance

For this experiment we choose a recording duration length where the recognition accuracy is not very high. This way, we can measure the change in accuracy when all we change is a set of parameters which increase the number of fingerprints. And we measure how much longer this takes to fingerprint/recognize.