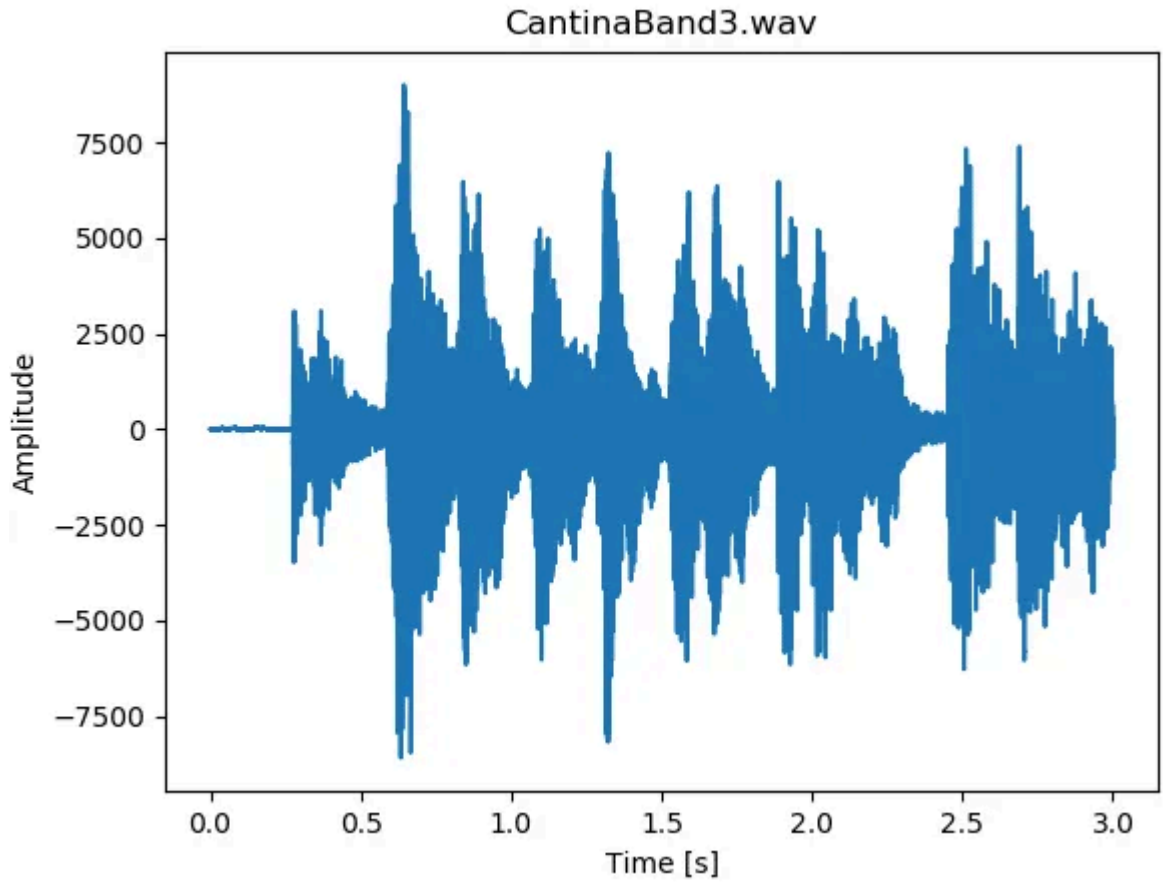
Member-only story

Extracting Features from audio files for Sound classification

Nausheen
Published in Towards Dev
4 min read · Mar 10, 2023



An audio file plotted in the time domain

With the increasing availability of audio data and advances in machine learning algorithms, sound classification is becoming an increasingly important tool in a wide range of fields.

Sound classification is assigning one or more categories or labels to a sound or audio clip. There are many applications of sound classification, including:

- **Speech recognition:** used to recognise different speech patterns and convert spoken words into text.
- **Music genre classification:** Sound classification can be used to identify the genre of a particular song based on its audio features.
- **Animal sound classification,** such as bird songs or whale calls, to aid in conservation efforts.
- **Environmental sound classification:** to detect and classify different environmental sounds, such as traffic noise, sirens, or even natural sounds like rain or wind.
- **Medical diagnosis:** it can also be used in medical applications, such as diagnosing respiratory disorders based on the sound of a patient's breathing.
- **Security and surveillance:** to identify specific sounds, such as gunshots or breaking glass, in security and surveillance applications.
- **Automotive and transportation**: can be used to detect and classify different vehicle sounds, such as engine noise, tire squealing, or honking horns, to improve vehicle safety and performance also car crashes and gravel road condition assessment.
- **Human activity recognition:** Sound classification can be used to recognize and classify different human activities, such as walking, running, or jumping, for health and fitness applications or to monitor human behavior in public spaces.

Sound feature extraction involves transforming a raw audio signal into a set of numerical features that can be used for further analysis, such as classification or clustering.

Mel-frequency cepstral coefficients (MFCCs) are a commonly used feature extraction technique in audio signal processing, particularly for speech recognition and music information retrieval.

The MFCC algorithm is based on the human perception of sound, which is known to be more sensitive to differences in frequency at lower frequencies and less sensitive

at higher frequencies. To capture this behavior, the algorithm first transforms the audio signal from the time domain to the frequency domain using the short-time Fourier transform (STFT). It then applies a filterbank that mimics the non-linear frequency resolution of the human ear, known as the Mel-scale. The Mel-scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another, and it is based on the ratio of the fundamental frequency of the sound to the critical bandwidth of the auditory system at that frequency.

After applying the Mel filterbank, the algorithm computes the logarithm of the filterbank energies, which are then transformed using the discrete cosine transform (DCT) to yield a set of MFCCs. The MFCCs represent the spectral envelope of the audio signal, capturing information about the distribution of energy across frequency bands.

MFCCs are commonly used as features for speech and music recognition tasks because they are robust to variations in speaker or instrument, and can capture information about the phonetic content or musical structure of the audio signal.

We here use the Librosa library in Python for extracting some common sound features. To extract features from multiple sound files in a folder and save them to an Excel file, you can use the following Python code:

```
import librosa
import pandas as pd
import os

# Set path to folder containing audio files
folder_path = "path/to/folder"

# Initialize empty list to store features
features = []

# Loop through all files in folder
for file_name in os.listdir(folder_path):
    if file_name.endswith(".wav"):
        # Load audio file
        file_path = os.path.join(folder_path, file_name)
        signal, sr = librosa.load(file_path, sr=None)

        # Extract features
        mfcc = librosa.feature.mfcc(signal, sr=sr, n_mfcc=13)
        spectral_contrast = librosa.feature.spectral_contrast(signal, sr=sr)
        spectral_centroid = librosa.feature.spectral_centroid(signal, sr=sr)
```

```
            zero_crossing_rate = librosa.feature.zero_crossing_rate(signal)

            # Append features to list
            features.append({
                "File Name": file_name,
                "MFCC": mfcc.flatten(),
                "Spectral Contrast": spectral_contrast.flatten(),
                "Spectral Centroid": spectral_centroid.flatten(),
                "Zero-crossing Rate": zero_crossing_rate.flatten()
            })

    # Convert list of features to a DataFrame
    df = pd.DataFrame(features)

    # Save DataFrame to a CSV file
    output_file = "output_file.csv"
    df.to_csv(output_file, index=False)
```

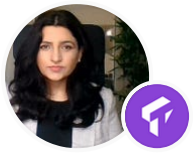
In this code, we first set the path to the folder containing the audio files. We then initialize an empty list `features` to store the extracted features for each file.

We loop through all files in the folder using `os.listdir()` and extract the same set of features as in the previous example for each file that ends with `.wav`. We append the extracted features to the `features` list as a dictionary with the file name and flattened feature arrays.

After looping through all files, we convert the `features` list to a DataFrame using `pd.DataFrame()` and save it to an CSV file using `df.to_csv()`. The output csv file will contain one row for each audio file, with the file name and the flattened feature arrays as columns.

In the above code, we first load an audio file using the `librosa.load()` function. We then extract four common sound features using different functions from the Librosa library: MFCC, spectral contrast, spectral centroid, and zero-crossing rate. Finally, we print the shape of each feature to verify that they all have the same number of time frames. These features can then be used for further analysis, such as classification or clustering.

Sound Classification | Librosa | Python Programming | Data Science



Edit profile

Written by Nausheen

117 Followers · Writer for Towards Dev

PhD Researcher Data science, Sweden | Data Enthusiast

More from Nausheen and Towards Dev

Nausheen in Towards Dev

Building a Credit Card Approval Prediction Model with Logistic Regression

In this blog, we'll build a predictive model that determines credit scores using logistic regression. The model aims to predict which...

May 10