LIBROSA et découverte audio

Sommaire

Environnement virtuel + installation

Créer un environnement virtuel, readme etc...

installer numpy librosa streamlit plotly soundfile

Interface Streamlit – chargement ou enregistrement

`st.file_uploader()` pour charger un `.wav`

`sounddevice` ou `pyaudio` pour l'enregistrement micro

Sauvegarde du fichier audio dans un dossier `./audio_inputs/`

Ajouter des boutons pour les questions suivantes

Visualiser le signal audio (temps)

librosa.load() pour obtenir le signal et Fe (Fréquence d'échantillonnage)

Affichage avec Plotly : `t = np.arange(len(y)) / Fe`
pour avoir l'axe **temps en secondes**

Afficher durée, échantillonnage, forme du signal

Créer une sinusoïde

Utiliser `np.sin(2 * np.pi * f * t)`

Exemple : f = 440 (440 Hz est le LA de base en musique) sur 2 secondes avec du coup **t** en secondes comme au dessus

Écoute avec soundfile.write() puis st.audio()
(attention les oreilles ne pas mettre le volume trop fort au cas ou)

Spectre d'un son

Le spectre d'un signal représente la répartition de son énergie ou de sa puissance en fonction des fréquences. Il est obtenu par la transformée de Fourier, qui décompose le signal en sinusoïdes. Le spectre permet d'identifier les composantes fréquentielles d'un son, comme les harmoniques d'une voix ou d'un instrument.

- Calcul de la transformée de Fourier (FFT) avec `np.fft.fft()`
- Affichage de l'amplitude et des fréquences (axe en Hz : `f = np.fft.fftfreq(N, 1/Fe)`)

```
import librosa
import numpy as np
import matplotlib.pyplot as plt

# Charger l'audio (mono)
y, Fe= librosa.load("mon_audio.wav")

# Appliquer une fenêtre de Hann (optionnel mais courant)
window = np.hanning(len(y))
y_windowed = y * window

# FFT
Y = np.fft.rfft(y_windowed)
frequencies = np.fft.rfftfreq(len(y), d=1/Fe)

# Spectre de puissance
power = np.abs(Y) ** 2

# Affichage
plt.figure(figsize=(10, 4))
plt.plot(frequencies, power)
plt.title("Spectre de puissance")
plt.xlabel("Fréquence (Hz)")
plt.ylabel("Puissance")
plt.grid()
plt.tight_layout()
plt.show()
```

Le décibel (dB) est une unité logarithmique utilisée pour exprimer un rapport de puissance ou d'amplitude, souvent en acoustique ou traitement du signal. Il permet de mieux représenter les grandes variations d'intensité perçues par l'oreille humaine. Pour convertir un spectre de puissance en décibel, on utilise la formule : 10 * log10(power) ou, avec Librosa, librosa.power_to_db(power).

STFT sur un son réel

La STFT (Short-Time Fourier Transform) permet d'analyser comment le contenu fréquentiel d'un signal évolue dans le temps. Elle découpe le signal en petites fenêtres temporelles,

puis applique une FFT sur chaque segment. Cela produit un spectrogramme, utile pour visualiser les sons non stationnaires comme la parole ou la musique.

- Utiliser `librosa.stft()`
- Puis `librosa.amplitude_to_db()` pour la visualisation
- Affichage avec `librosa.display.specshow()` ou `plotly.imshow()`

Analyse complète avec un son Kaggle

Charger un son depuis un dataset (ex : UrbanSound8K ou ESC-50)
afficher:

- STFT
- Spectrogramme Mel
- MFCC avec librosa.feature.mfcc()

de quelques fichiers de différentes classe (3 fichiers des classes 0 et 1 par exemple)

Classification de sons ambiants avec MFCC

Les MFCC (Mel-Frequency Cepstral Coefficients) sont des descripteurs audio qui modélisent la manière dont l'oreille humaine perçoit les sons. Ils s'obtiennent en appliquant une transformée de Fourier, un filtrage Mel, puis une transformée en cosinus. Très utilisés en reconnaissance vocale, ils capturent les caractéristiques du timbre plutôt que la hauteur.

- Créer une petite base (5-10 fichiers dans 2 classes)
- Extraire les MFCC moyens de chaque fichier
- Entraîner un **classifieur simple** : KNN ou LogisticRegression (`scikit-learn`)
- Prédiction sur un nouveau son