Audio Clustering Report

1. Data

The dataset consists of a set of .mp3 files with different durations, ranging from ~19 to ~30 seconds. All files share the same technical parameters:

- Sample rate
- Number of channels
- Bit depth
- Encoding format

2. Preprocessing

For both versions of the solution (classical features and deep learning), the same preprocessing steps are applied:

- Each audio file is split into 5-second chunks.
- If the last chunk is shorter than 5 seconds, it is padded with zeros (if allowed by the padding threshold).
- Features or embeddings are extracted from each chunk.

Version 1: Classical Features

Extracted features:

- **MFCC (mean & var)**: spectral features describing the timbre of the audio.
- **Spectral centroid**: indicates where the “center of mass” of the spectrum is, associated with brightness.
- **Spectral bandwidth**: width of the frequency spectrum.
- **Spectral rolloff**: the frequency below which 85% of the spectral energy is contained — represents sharpness.

- **Zero crossing rate**: how often the signal crosses zero, gives an idea of noisiness or tonality.
- **Tempo**: estimated beats per minute (BPM).
- **Energy**: mean squared signal value, representing loudness.
- **Chroma (mean)**: pitch class profile, represents harmonic content.

The final feature vector for each chunk is approximately **33 dimensions**.

Model & Pipeline

Pipeline:

Audio files → 5s chunks → Feature extraction (librosa) → StandardScaler → PCA → KMeans clustering

PCA is used to reduce dimensionality due to the small dataset size and relatively high feature count.

KMeans is chosen for its simplicity and effectiveness when the number of clusters is known.

Special Handling:

Because the audio files are of different lengths, the number of chunks per file also varies. To mitigate this:

- The number of chunks per track is controlled by using a lower quantile (e.g., 25%) of audio durations.
- The final chunk can be padded if it's close enough to the required length. A parameter padding_threshold allows you to control what percentage of the chunk duration must be present to allow padding.

Version 2: Deep Learning-based Features (CLAP)

CLAP is all you need!

At first, I considered using middle-layer embeddings from **Wav2Vec2**, but this approach has some issues:

- Most speech models like Wav2Vec2 are pre-trained on human voice data, which limits generalization for music.

- CLAP (Contrastive Language-Audio Pretraining), on the other hand, is trained on diverse audio including music, using large datasets like **FSD50K**.

Moreover, CLAP uses supervised learning with natural language descriptions, which helps the model learn semantic audio concepts. This makes CLAP a much better fit for music clustering compared to speech-oriented models.

Model & Pipeline

Pipeline:

Audio → 5s chunks → ClapProcessor → ClapModel → Embedding → Mean-pooling → (Optional PCA) → KMeans clustering

Uses the chunk_average strategy — each chunk is embedded independently, then averaged per song.

KMeans is again used for clustering.

Can be run on both GPU and CPU; automatically uses cuda if available.

Special Features:

- The class supports different chunking and aggregation strategies:
    - chunk_average: mean of all chunk embeddings
    - chunk_level: each chunk is clustered independently, then final label is assigned by majority vote
- The padding logic is preserved but less critical here since CLAP is more robust to variable-length input.
- Embeddings are extracted using ClapModel.get_audio_features().