194.039 Intelligent Audio and Music Analysis

Author:

Max Tiessler

2025-01-08

Contents

1 Introduction

1.1 Basics

1.1.1 What are the differences between sound, audio, and music

1.1.2 What is the frequency range for audio?

1.1.3 Give examples for objective properties in music!

1.1.4 What is an audio waveform and how can it be created from sound?

1.1.5 What is ADC and what steps does it consist of?

1.1.6 What influence do sampling frequency and bit depth for ADC have?

1.1.7 What are the requirements for the sampling frequency for ADC, when sampling a signal?

1.1.8 What does PCM stand for?

1.1.9 Explain how PCM encodes an audio signal!

1.2 Audio Features and Signal Framing

1.2.1 What are (audio-)features in the context of signal processing?

1.2.2 What are examples for basic time-domain audio features?

1.2.3 What are problems of global audio features, especially in the context of music?

1.2.4 Explain in your own words what signal framing is, including the relevant parameters!

1.2.5 How can signal framing be used to calculate time-local audio features?

1.2.6 What is the advantage of time-local audio features?

1.2.7 Give examples of time-local audio features and explain what information they reveal!

1.2.8 Explain the process of calculating a time-local feature like ZCR!

1.3 Frequency Domain Features

How does audio signal frequency analysis relate to the human hearing process?

1.3.1 Explain, in your own words, what the Fourier transform is!

1.3.2 Sketch a sine-frequency with 10Hz and the magnitude spectrum of this signal!

1.3.3 Sketch a signal that is a mixture of two sines and it's magnitude spectrum!

1.3.4 Explain amplitude, frequency, and phase for sinusoid!

1.3.5 What is a "frequency-domain representation" of an audio signal?

1.3.6 What is a spectrum, what is displayed on the x and y axis?

1.3.7 What is the role of the spectrum's phase values of a signal?

1.3.8 When calculating a DFT: what parameters influence the frequency resolution of the spectrum?

1.3.9 Give examples for frequency-domain features of an audio signal!

1.4 Time-Frequency Domain Features

What are the problems when calculating the Fourier transform on a longer piece of music?

1.4.1 How can we maintain time information when performing frequency analysis of a signal?

1.4.2 Explain in your own words what the STFT is and how it is calculated!

1.4.3 What is the advantage of using the STFT instead of simple Fourier transform?

1.4.4 What is the role of the window function in the context of the STFT?

1.4.5 What is a spectrogram and what values are shown on the x, y, and z axis?

1.4.6 Sketch a spectrogram of a signal that contains multiple short sine waves with different frequencies!

1.4.7 Give examples for time-frequency domain features!

1.5 Psychoacoustics

1.5.1 Explain why psychoacoustics are relevant for audio signal processing!

1.5.2 How are psychoacoustic scales created?

1.5.3 Give examples for psychoacoustic pitch scales! What are their characteristics?

1.5.4 How do psychoacoustics influence loudness perception?

1.5.5 How can we apply psychoacoustic loudness scaling to spectrograms?

1.5.6 How can we apply psychoacoustic frequency scales to spectrograms?

2 Onset Detection

2.1 Notes and Onsets

2.1.1 What are the (small-scale) structures that we can find in music?

2.1.2 How can we call the individual musical events in music?

2.1.3 What is the ADSR model?

2.1.4 Explain the phases of the ADSR model and how they relate to playing a note on an instrument!

2.1.5 List basic properties of musical notes that we discussed in the lecture!

2.1.6 What is the relation of onsets and transients of musical notes?

2.1.7 Explain the difference between percussive, harmonic, and percussive-harmonic sounds!

2.1.8 Discuss identifying onsets for percussive, percussive-harmonic, and purely harmonic sounds and the challenges for each sound type!

2.2 Onset Detection - I. Signal Pre-Processing

2.2.1 What is the goal of onset detection?

2.2.2 What are challenges of onset detection in real world signals?

2.2.3 Why is onset detection an important task / what can onsets be used for?

2.2.4 What are the steps in a typical processing pipeline for onset detection?

2.2.5 What is the goal of the signal preprocessing step for onset detection?

2.2.6 Give examples for signal preprocessing for onset detection!

2.3 Onset Detection - II. Detection Functions

2.3.1 What is an onset detection function?

2.3.2 What are the requirements for an onset detection function?

2.3.3 Give examples for onset detection functions!

2.3.4 Explain the fundamental idea of the phase-deviation onset detection function in your own words!

2.3.5 How is the spectral flux onset detection function calculated? No equation, only steps of algorithm.

2.3.6 Name three different types of audio signal contents and explain the challenges for onset detection for them!

2.4 Onset Detection - III. Peak Picking

2.4.1 What is the goal of peak picking in the context of onset detection?

2.4.2 What are the challenges of peak picking in the context of onset detection?

2.4.3 What are the processing steps for the peak picking method we discussed in the lecture?

2.5 Performance Evaluation

3 Beat Tracking

3.1 Introduction

3.1.1 What are beats, downbeats, and measures, in the context of music?

3.1.2 What is tempo (in the context of music) and how can we quantify it?

3.1.3 Explain the difference between local and global tempo of a music track?

3.1.4 What is the goal of beat tracking?

3.2 Tempo Estimation

3.2.1 What is the goal of tempo estimation?

3.2.2 What are inter-onset-interval histograms, and how can they be used for tempo estimation?

3.2.3 What is autocorrelation, and how can it be used for tempo estimation?

3.2.4 What are octave errors in the context of tempo estimation?

3.2.5 What is a tempogram?

3.3 Beat Tracking

3.3.1 What is beat phase detection, in the context of beat tracking?

3.3.2 Describe a method to detect the beat phase for beat picking!

3.3.3 Explain the greedy beat tracking algorithm discussed in the lecture!

3.3.4 Explain the multi-agent tracking approach discussed in the lecture!

3.4 ML Approaches

3.4.1 Explain the machine learning setup for beat tracking!

3.4.2 Why are RNNs a suitable architectural choice for beat tracking?

3.4.3 What is the task of the RNN in the RNN beat tracker discussed in the lecture?

3.4.4 What is the task of the DBN in the RNN beat tracker discussed in the lecture?

3.5 TCN Beat Tracker

3.6 Evaluation

3.6.1 Which metrics can we use for beat tracking evaluation?

3.6.2 In what range are tolerances for beat positions typically?

3.6.3 Why are tolerances for beat tracking higher than for onset detection?

4 F0 and Notes

4.1 Pitch and F0

4.1.1 Properties of Musical Notes Mentioned in the Lecture

- **Onset Time**: The time when a note begins.
- **Offset Time**: The time when a note ends.
- **Fundamental Frequency** (f_0): The pitch or base frequency of a note.
- **Relative Loudness** (Velocity): The volume or intensity of the note.
- **Instrument** (Timbre): The unique quality or color of the sound.
- **Playing Techniques**: Variations such as glissando, vibrato, tremolo, harmonic/percussive components, etc.

4.1.2 Pitch and Its Relationship to the Fundamental Frequency of a Note

- **Pitch**: A perceptual quantity that allows humans to order sounds on a frequency-related scale from low to high.
- **Relationship**: Pitch perception is closely tied to the fundamental frequency (f_0) of a sound, which is often derived logarithmically.

4.1.3 Harmonics or Harmonic Partials of a Note

- **Harmonics**: Integer multiples of the fundamental frequency (f_0).
- **Harmonic Partials**: Components of a sound that resonate at frequencies higher than f_0, contributing to the timbre of the note.

4.1.4 Fundamental Frequency of a Note

- The **fundamental frequency** (f_0) is the lowest frequency of a sound and is responsible for the perceived pitch of the note.

4.1.5 Difference Between Percussive, Harmonic, and Harmonic-Percussive Notes

- **Percussive Notes**: Do not have a distinct fundamental frequency; their sound arises from transient noise (e.g., drums).
- **Harmonic Notes**: Have a clear fundamental frequency and harmonics (e.g., violin or flute).
- **Harmonic-Percussive Notes**: Combine properties of both percussive and harmonic sounds (e.g., piano).

4.1.6 Visualization Method for Monophonic F0 Estimation/Annotations

- The f_0 **Contour**: A common visualization method for tracking f_0 over time for monophonic melodies, often calculated from spectrograms.

4.1.7 Visualization Methods for Polyphonic F0 Estimation/Annotations

- The **Piano Roll Representation**: Suitable for polyphonic music, often used in digital audio workstations. It is akin to a spectrogram but without harmonics and transients.

4.1.8 Difference Between Single- and Multi-F0 Estimation

- **Single-F0 Estimation**:
  - Detects the f_0 of monophonic signals (e.g., singing voice).
  - Accurate and robust algorithms exist; considered a solved problem.
- **Multi-F0 Estimation**:
  - Detects the f_0 of each tone in polyphonic signals (e.g., piano or mixed instruments).
  - Still an open research problem.

4.1.9 Three Levels for F0 Estimation Discussed in the Lecture

- **Frame-Level Detection**: Focused on detecting pitch contours. We estimate *f0* for each frame
- **Note-Level Detection**: Used for monophonic transcription. We need to do a post-processing for framewise detection. There is a simultaneous onset/pitch/offset detection.
- **Stream-Level Detection**: Applied in automatic music transcription. There is a post-processing for note detection and instrument note detection

4.2 Evaluation

4.2.1 Evaluation for F0 Estimation on Frame-Level

For F_0 estimation on the frame level, annotations are discretized to a frame rate. This results in a continuous F_0 contour that is used for evaluation. Each frame is evaluated individually, and each sounding pitch is assessed separately.

4.2.2 Evaluation for F0 Estimation on Note-Level

Note-level evaluation involves a piano-roll representation, where annotations are represented as **discrete quantized** notes. This method focuses on the musical properties such as the onset and offset times of the notes.

4.2.3 Evaluation for F0 Estimation on Stream-Level

Stream-level evaluation treats each instrument separately at the note level. For multi-instrument setups, each instrument is evaluated independently on its note-level representation.

4.2.4 Issues with Frame-Level F0 Estimation

Frame-level evaluation is convenient but problematic:

- Short missing sounding frames have marginal impact on overall performance but are very irritating to humans.
- Similarly, falsely detected short spurious notes are problematic in frame-level evaluation.

4.2.5 Metrics for Frame-Level F0 Estimation

The metrics commonly used for frame-level F_0 estimation include:

- Precision
- Recall
- F-measure
- Accuracy: acc $= \frac{TP+TN}{TP+FP+TN+FN}$, often assuming $TN = 0$.

4.2.6 Issues with Note-Level F0 Estimation

Note-level F_0 estimation has its own challenges:

- Determining accurate onset and offset times can be subjective and unclear from signal information alone.
- Errors in note segmentation or classification significantly affect the results.

4.2.7 Properties Evaluated for Note-Level F0 Estimation

The properties typically evaluated for note-level F_0 estimation include:

- Note onset time
- Note offset time
- Pitch accuracy
- Note duration

Various aspects of notes can be evaluated

- Onset (e.g. $\pm$50ms) + pitch (e.g. $\pm$50cents)
- Onset + pitch + offset (e.g. max $\pm$50ms, 20% of note length)

Usually measured independently (no “overall” metrics)

4.3 Fundamental Frequency Estimation

4.3.1 Which simple/naive methods could be used for F0 estimation?

- **Zero-Crossing Rate (ZCR)**: Measures how often the waveform crosses zero per unit of time. This was one of the first approaches to F_0-estimation. ZCR is directly related to the number of times the waveform repeats per unit time.
- **Autocorrelation**: Analyzes periodicity by finding the lag at which the signal best correlates with itself. Often used for pitch detection in periodic signals.

4.3.2 Explain the typical problems of simple F0 estimation methods!

- **Problems with ZCR**:
  - ZCR works only if the signal does not contain extra zero-crossings.
  - Partial-rich tones produce too many zero-crossings, leading to incorrect frequency estimation.
  - Estimated frequency can be off by a factor of 2 if there are 4 zero-crossings instead of 2 per cycle.
  - Requires preprocessing, such as low-pass filtering, to counteract the effects of additional crossings.
- **Problems with Autocorrelation**:
  - Partial-rich tones may produce a dominant peak at one of the harmonic partials instead of the fundamental frequency.
  - When the signal is pseudo-periodic with a low-power fundamental frequency, even humans may mistake a harmonic partial for the fundamental.
- **General Issues with Simple Methods**:
  - Most common errors are **octave errors**, where the first partial is detected instead of the fundamental ($f = 2 \cdot f_0$).
  - Naive methods struggle to cope with harmonic-rich signals or noise.

4.3.3 Explain the processing steps for the YIN F_0 estimation method!

1. **Difference Function Calculation:**
   - YIN operates in the time-domain, similar to the Autocorrelation Function (ACF) method.
   - Replaces ACF with the Average Magnitude Difference Function (AMDF):

$$d_t[\tau] = \sum_{t=1}^{M} (x[t] - x[t+\tau])^2$$

   - Utilizes the principle of cancellation: $d_t[T] \approx 0$ for periodic signals with period T.
2. **Normalization:**
   - Values of $d_t[\tau]$ depend on the range of $x[t]$, so a simple minimum search is insufficient.
   - Normalization produces the Cumulative Mean Normalized Difference Function $nd_t[\tau]$:

$$nd_t[\tau] = \begin{cases} 1 & \text{if } \tau = 0, \\ \frac{d_t[\tau]}{\frac{1}{\tau}\sum_{i=1}^{\tau} d_t[i]} & \text{otherwise.} \end{cases}$$

   - This reduces valleys caused by strong first partials and ensures the range of $nd_t[\tau]$ lies within $[0, 2]$, where 0 indicates perfect periodicity.

3. **Thresholding:**
   - After normalization, an absolute threshold is applied to identify candidates.
   - The smallest lag τ below the threshold is chosen as the fundamental frequency.
   - If no values fall below the threshold, the global minimum is used as a fallback.
4. **Interpolation:**
   - To address quantization errors, YIN employs parabolic interpolation.
   - Uses $nd_t[\tau - 1]$, $nd_t[\tau]$, and $nd_t[\tau + 1]$ to fit a parabola.
   - The minimum of the fitted parabola is selected as the refined lag τ.

4.3.4 What is the difference between YIN and pYIN?

- **YIN:**
  - A deterministic F_0 estimation algorithm based on the time-domain.
  - Outputs a single F_0 value or absence for each frame.
- **pYIN (Probabilistic YIN):**
  - Extends YIN by incorporating a probabilistic framework.
  - Produces a probability distribution over F_0 values for each frame instead of a single output.
  - Improves robustness against noisy signals and enhances pitch tracking by considering multiple candidates.

4.4 F0 Estimation in the Frequency Domain

4.4.1 What is the fundamental idea behind F_0 estimation in the frequency domain?

The fundamental idea is to analyze the spectrogram and locate the peaks corresponding to harmonic frequencies. The F_0 is estimated by identifying the fundamental frequency or by analyzing the spacing between harmonics. This method historically represents earlier approaches to pitch estimation and has the appeal of being generalizable to multiple F_0 estimation.

4.4.2 List some of the problems of F_0 estimation in the frequency domain!

- Simply picking the bin with the most energy is insufficient:
  - Harmonics may have a larger magnitude than the fundamental F_0.
  - In some cases, the fundamental F_0 may have no energy (missing fundamental).
- Common challenges include:
  - **Harmonic errors:** $f_n \neq F_0 \cdot k$, where $k \in \mathbb{N}$.
  - Missing harmonics that disrupt the detection of F_0.
  - Octave errors, where higher harmonics are mistaken for the fundamental.

- For multiple F_0:
  - Overlapping harmonics and transients make detection challenging.
  - Determining how many peaks (or notes) to select can be ambiguous.

4.4.3 How does the SHAPE F_0 estimation method work, in principle?

The SHAPE method works by using spectral templates:

- Define a **pitch template function** representing the expected harmonic structure of a given pitch.
- Compute the correlation of this template with the actual spectrum over a set of candidate pitches.
- Select the pitch with the **highest correlation** as the estimated F_0.

To address specific problems:

- Frequencies closer to F_0 are weighted more heavily.
- Penalize incorrect harmonics by considering the entire spectrum, not just harmonic peaks.
- Handle imperfect harmonics (non-integer multiples) by adapting the template to the specific signal.

This approach provides a more robust solution to issues like missing fundamentals and harmonic interference.

4.5 Multiple F0 Estimation

4.5.1 Explain the difference between single and multiple-F_0 estimation!

- **Single-F_0 Estimation:**
  - The task is to estimate a single fundamental frequency (F_0) present in a monophonic audio signal.
  - Assumes that there is only one source active at any given time.
  - Common in applications like voice pitch detection or simple melody tracking.
- **Multiple-F_0 Estimation:**
  - The task is to estimate multiple F_0 values simultaneously from a polyphonic audio signal.
  - Accounts for overlapping harmonics, transients, and the presence of multiple instruments or voices.
  - Common in applications like polyphonic music transcription or source separation.

4.5.2 What are the challenges of multiple-F_0 estimation?

- **Overlapping Harmonics:**

  - Harmonics from different sources can overlap, making it hard to distinguish individual F_0 values.
- **Diverse Instrument Sounds:**
  - Different instruments produce distinct harmonic structures and timbres, complicating harmonic identification.
- **Unknown Number of Notes:**
  - It is often unknown how many simultaneous F_0 values exist in the signal.
- **Noisy Spectrogram:**
  - Overlapping harmonics and fundamentals create a noisy spectrogram that is difficult to interpret.

4.5.3 Explain the three categories for multiple-F_0 estimation we discussed in the lecture!

1. **Spectrogram Analysis:**
   - Detect peaks corresponding to fundamental frequencies and harmonics.
   - Challenges include overlapping harmonics and missing fundamentals.
2. **Spectral Decomposition:**
   - Decompose the spectrogram into components using techniques like:
     - Non-Negative Matrix Factorization (NMF).
     - Probabilistic Latent Component Analysis (PLCA).
     - Sparse Coding.
   - These methods model the spectrogram as a combination of templates and activations.
3. **Machine Learning Approaches:**
   - Treat F_0 estimation as a multi-label classification problem.
   - Use algorithms like:
     - Support Vector Machines (SVMs).
     - Hidden Markov Models (HMMs).
     - Neural Networks (NNs).
   - Requires a large amount of training data to learn from examples.

4.6 Neural Network Based Methods

4.6.1 Explain how a CNN can be used to perform multi-f0 estimation!

A Convolutional Neural Network (CNN) can be used for multi-f0 estimation by operating on spectrogram representations of polyphonic audio. The spectrogram provides a time-frequency representation of the audio, capturing harmonic structures that are crucial for pitch detection. The CNN extracts patterns corresponding to harmonic frequencies using convolutional layers. Each frame of the spectrogram is processed independently, and the network predicts the active pitches for each frame. The output consists of a binary or continuous-valued vector representing the presence of each pitch (e.g., 88 piano keys) at a given time frame. This frame-wise transcription acts as an "acoustic model," focusing on the spectral features associated with pitch activation.

4.6.2 What issues can arise when doing frame-wise multiple-f0 estimation?

- **Temporal Continuity:** Frame-wise processing lacks temporal context, leading to inconsistencies in pitch activation across frames.
- **Overlapping Harmonics:** Harmonics from different pitches may overlap, making it challenging to distinguish individual pitches.
- **Transients:** The attack and decay phases of notes can be misinterpreted as new notes or missed entirely.
- **Note Sustain:** Sustained notes may be inconsistently detected, especially if harmonics decay unevenly.

4.6.3 What is the task of the HMM post-processing for the CNN multiple-f0 estimation approach, we discussed in the lecture?

The Hidden Markov Model (HMM) post-processing aims to enforce temporal consistency in the CNN's frame-wise predictions. Its tasks include:

- **Temporal Smoothing:** Reducing abrupt transitions in pitch activations to provide a smoother representation.
- **State Constraints:** Guiding transitions between pitch states (e.g., onset, sustain, release) based on predefined rules.
- **Noise Reduction:** Suppressing spurious activations caused by noise or overlapping harmonics by modeling the note activation process.

4.6.4 Why does explicit modeling of note phases (ADSR) improve performance for multiple-f0 estimation?

Explicit modeling of note phases, such as Attack, Decay, Sustain, and Release (ADSR), improves performance by:

- **Temporal Representation:** Capturing the temporal evolution of notes ensures accurate identification of note onsets, sustains, and releases.
- **Context Awareness:** Phase-specific outputs help the network distinguish between the start, middle, and end of a note, reducing false positives and negatives.
- **Granular Training Labels:** Providing phase-specific training labels allows the network to learn the dynamics of note transitions, improving transcription accuracy.

5 Music Classification

Recap: Audio Analysis So Far

- Extraction of features (low-level)
- Detection of events:
  - Onsets
  - Beats
- Description of tonal properties:
  - Pitch estimation

Yet to Come

- Transcription of instruments:
  - Drums
  - Multiple instruments

5.1 Semantic Description

Semantic description refers to the use of words that make sense to humans when describing music. These descriptions focus on musical properties that hold semantic meaning for the average listener. For example, terms like "genre," "mood," and "instruments" provide intuitive ways to characterize music. Semantic representation can be categorized into three levels: high-level, mid-level, and low-level. High-level representations involve musical concepts as humans perceive them, such as emotions or style. Mid-level representations serve as a bridge, combining low-level features into meaningful patterns. Low-level representations, on the other hand, consist of statistical descriptions of audio signals that are machine-readable.

5.2 Musical Genre

A musical genre is defined as a set of musical events, real or possible, governed by a set of socially accepted rules. These rules can be categorized into five types, as proposed by Fabbri in 1981. First, formal and technical rules pertain to content-based practices. Second, semiotic rules involve abstract concepts, such as emotions or political messages conveyed through music. Third, behavioral rules define how composers, performers, and audiences interact with the genre. Fourth, social and ideological rules describe the connection between genres and demographic factors, like age or culture. Finally, economic and juridical rules focus on the systems that support genres, such as contracts and performance venues.

5.3 Why Genre Classification?

Genre classification is essential for grouping songs based on common criteria and labeling musical trends. This helps in organizing and facilitating the production, circulation, and reception of music. Archetypal genres, such as rock or jazz, are often immediately recognizable, sometimes within just 300–400 milliseconds. In Music Information Retrieval (MIR), genre classification is a common evaluation method. It serves as a proxy for music similarity tasks, relying on fixed ground truths like the "Cranfield paradigm." Additionally, it simplifies experimental setups compared to pairwise similarity testing.

5.4 Genre Classification

Genre classification is modeled as a machine learning task. It involves extracting features from musical data and using them to predict the genre of a piece of music. Features can range from low-level signal attributes, like spectral flatness, to more abstract descriptors, such as rhythm patterns. A classifier is trained on labeled examples and then used to categorize unseen data.

5.5 Case Study: k-NN for Genre Classification

A notable case study explored the use of a k-Nearest Neighbors (k-NN) classifier for genre classification. In this study, $k = 3$ was chosen, and features like MFCC and spectral flatness were used. The GTZAN dataset served as the basis for evaluation, with a 10-fold cross-validation setup. The classifier achieved an accuracy of 80.8%, though it misclassified 192 examples. This demonstrates the effectiveness of k-NN while also highlighting the challenges in achieving perfect classification.

5.6 Investigating Classification Problems

To investigate the reasons for misclassification, a user experiment was conducted involving 20 experts from the Icelandic music industry, including musicians, producers, and DJs. Participants were tasked with assigning genre labels to 30-second snippets of the 192 misclassified examples. These snippets were presented without metadata in a controlled environment, using a quiet room and a Hi-Fi stereo system. The objective of the study was to determine the most suitable genre label for each snippet and to analyze the discrepancies between human perception and machine classification.

5.7 Why NOT Genre Classification?

Genre classification can be problematic for several reasons. Cultural differences often lead to varying interpretations of the same music, making universal agreement difficult. Additionally, expert disagreements arise, with inter-listener agreement reported at only 80%. Another issue is that intensional definitions of genres may not exist, as genres are fluid and dynamic. A notable example is the "Bohemian Rhapsody effect," where a single label is insufficient to capture the complexity of the music. For instance, a track may simultaneously fall under categories like ballad, pop, rock, musical, and metal.

5.8 Tags

Tags are short descriptions of specific aspects of music, such as genre, style, instrumentation, or mood. These can describe the track itself, the performer, or even personal categories like "favorite" or "seen live." Tags often rely on music-targeted vocabularies that may be explicit (assigned by experts) or implicit (derived from user interactions). Weighted tags are common, reflecting the number of users who agree on a specific tag. There are several approaches to obtaining tags, including expert annotations, collaborative tagging, and games designed for this purpose.

5.9 Games With A Purpose (GWAP)

An innovative approach to manual tagging involves using games to engage users while collecting meaningful data. These are known as Games With A Purpose (GWAP). Some tasks, such as semantic image labeling or audio tagging, are challenging for AI to perform effectively. GWAP leverages human computational power by integrating the task into a game, encouraging users to put effort into completing the task while enjoying the gameplay. A classic example is the ESP Game, introduced by von Ahn and Dabbish (2004).

5.10 ESP Game

The ESP Game uses a collaborative approach to ensure availability, diversity, and challenge in tagging tasks. The system simulates players by replaying sessions and includes features like ”taboo words,” which restrict commonly used terms to ensure more diverse input. It also removes overused or skipped images, ensuring a wide variety of content is tagged. This method increases the quality and breadth of data collected.

5.11 TagATune

TagATune is another tagging game, focusing on the concept of ”input agreement” rather than ”output agreement.” In this game, participants describe a tune to determine whether they are listening to the same or a different one as their partner. This approach avoids semantic gaps commonly found in output-based systems. However, the game revealed that output agreement did not work as well for music, highlighting the challenges in bridging the semantic gap.

5.12 MagnaTagATune Dataset

The MagnaTagATune dataset is a product of the TagATune game. It includes over 25,000 snippets from more than 5,000 songs, with detailed annotations. These annotations cover tag-based descriptions, similarity matrices, and expert-provided genre labels. This dataset has been widely used in research for tasks like music recommendation and classification, providing valuable resources for advancing the field.

5.13 Semantic Description from Audio Analysis

Semantic description combines low-level and mid-level features to extract meaningful information from audio. Common features include spectral descriptors, MFCCs, chroma features, and rhythm patterns. Machine learning methods fit these features to specific tasks or taxonomies. Alternative approaches include extracting stylistic information from symbolic music and employing deep learning for end-to-end learning.

5.14 Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs, rooted in speech recognition, represent the spectral envelope of audio, capturing timbral aspects critical for sound similarity. They are computed using the following steps: 1. Framing the signal. 2. Applying the Discrete Fourier Transform (DFT). 3. Mapping the spectrum to the Mel scale for perceptual relevance. 4. Taking the logarithm of Mel-scaled amplitudes to reflect perceived loudness. 5. Performing the Discrete Cosine Transform (DCT) to decorrelate features, resulting in coefficients representing the spectral shape.

5.15 The Mel Scale

The Mel scale is a perceptual pitch scale, mapping frequencies in Hertz to their corresponding Mel values. The formula:

$$m = 2595 \log_{10}(1 + \frac{f}{700})$$

ensures equal perceptual distances between pitches. Typically, 40 Mel filter bins are used in MFCC computation.

5.16 Bag-of-Frames Modeling

In Bag-of-Frames modeling, an audio track is represented as a set of MFCC vectors. These vectors are aggregated using statistical measures like mean, variance, or covariance. Comparisons are made using distances between feature distributions, e.g., Earth Mover's Distance or KL Divergence. While effective, this approach loses temporal information.

KL Divergence for MFCCs

KL Divergence measures the difference between probability distributions. For Gaussian models of MFCCs, a closed-form solution exists, enabling efficient comparison. Applications include using MFCC "Bag-of-Frames" with Support Vector Machines (SVMs) for genre classification.

5.17 Deep Learning Approaches

Deep learning enables end-to-end learning by directly extracting features from raw signals, using spectrograms or melspectrograms as input. Convolutional Neural Networks (CNNs), including Deep CNNs and CRNNs, are common architectures. These models predict single or multi-class labels (e.g., genres or tags). However, careful design is necessary to match the task's requirements, such as handling percussive versus harmonic properties or ensuring pitch invariances.

5.18 Deep Learning for Music Tagging and Classification

Deep learning has revolutionized the field of music tagging by introducing end-to-end learning frameworks that automatically extract features from raw or preprocessed audio signals. These models eliminate the need for handcrafted features by using architectures like convolutional neural networks (CNNs), which process spectrograms and mel-spectrograms to learn relevant patterns for tagging and classification tasks.

5.18.1 End-to-End Learning for Tags

End-to-end learning utilizes CNNs to automatically extract audio features, ensuring translation, distortion, and locality invariance. This is especially relevant for musical features, as events can occur at any frequency or time range. The architecture processes log-amplitude mel-spectrograms, leveraging multiple convolutional layers with activation functions like ReLU, batch normalization, and dropout for regularization. The final output predicts multiple tags associated with each audio clip.

5.18.2 CNN Architectures and Design

The architecture for tagging involves:

- Input: 29.1-second audio clips downsampled to 12 kHz, resulting in 1,366 frames per clip.
- Features: Log-amplitude mel-spectrograms with 96 mel bands.
- Processing: Multiple convolutional layers with varying filter sizes (e.g., 3x3) and max pooling layers for downsampling.
- Optimization: Batch normalization, dropout, and ADAM optimizer.
- Output: 50 tags per clip using a sigmoid activation function.

5.18.3 Comparison of Input Representations and CNN Models

CNNs provide competitive results for music tagging, being lightweight and easy to train. They often involve many layers or large filters to achieve a wide receptive field. Dilated convolutions enhance the receptive field by skipping pixels, making tagging reliant on global properties. Experiments reveal the influence of input representation on tagging performance:

- Preprocessing methods include raw waveform (PCM), STFT, mel spectrogram, CQT, and MFCCs.
- Evaluations performed on datasets like MagnaTagATune and MTG-Jamendo.
- Dilated CNNs perform better on two-dimensional input representations like mel-spectrograms but struggle with raw waveforms.
- Statistical tests, such as two-way ANOVA, confirm significant influences of model architecture and input representation.

5.18.4 Results from Choi et al. (2016)

- The proposed system (FCN-4) achieved an AUC of 0.894 on the MagnaTagATune dataset.
- Comparison with previous methods showed that end-to-end CNN models outperform hand-crafted feature-based systems.
- Dilated CNNs excel in training efficiency and produce robust results on diverse input formats.

5.18.5 Experimental Setup and Analysis

Five preprocessing methods were evaluated using corresponding CNN models, including VGG-16, ResNet, SENet, MusiCNN, and Dilated CNN. The experiments used:

- Datasets: MagnaTagATune (25,000 samples) and MTG-Jamendo (55,000 samples).
- Metrics: ROC-AUC and PR-AUC.

- Statistical Analysis: Multiple runs per configuration with two-way ANOVA and Tukey-HSD tests.

Results showed:

- STFT preprocessing achieved the best performance, while raw waveforms performed the worst.
- Lightweight MFCC-based approaches yielded consistent results.

6 Multimodality

Music is represented in various formats across multiple domains, impacting how it is perceived. It is not merely treated as a signal but as a cultural phenomenon. This perspective allows for deeper analytical and cultural explorations. Semantic tagging and description approaches are heavily influenced by the availability and quality of data, enabling richer interpretations and applications.

6.1 Diverse Music Data Sources

Music data comes from a variety of sources: - **Content-based**: Includes audio, symbolic data, and lyrics. Features can be extracted directly from audio files, enabling recommendation systems without reliance on community data or cultural biases. - **Meta-data**: Editorial and curatorial data, as well as multi-modal sources like album covers, enrich metadata-driven recommendations. - **User-generated data**: Tags, reviews, blogs, biographies, and personal stories allow for personalized insights. - **Interaction data**: Logs, feedback mechanisms (e.g., thumbs), and purchase history help model user preferences. - **Curated collections**: Playlists, radio channels, and album compilations provide structured user preferences.

6.2 Collaborative Filtering (CF)

Collaborative Filtering leverages user interaction data to recommend music based on preferences. Key points include: - CF relies on implicit data (e.g., plays) and explicit data (e.g., likes/dislikes). - The task involves completing a sparse user-item matrix by predicting unknown interactions. - It assumes users with similar tastes in the past will exhibit similar preferences in the future. - Two main methods: 1. *User-based CF*: Relies on the similarity of users. 2. *Item-based CF*: Relies on the similarity of items.

6.3 Factors Hidden in the Data

Matrix factorization-based recommender systems assume that observed data (e.g., ratings) represent interactions between users and items. Latent factors extracted through matrix decomposition represent underlying patterns, aiding in user-item interaction predictions.

6.4 Audio Content Analysis

Audio-based recommendation systems provide true content-based recommendations by extracting features directly from audio files, removing the need for community or cultural data. Key aspects: - High-level semantic descriptors are learned from low-level features using machine learning. - Representation learning and temporal modeling are state-of-the-art approaches. - Advantages include no cultural biases, no popularity biases, and independence from subjective ratings.

6.5 Recap: Audio Content Analysis (2017 State-of-the-Art)

Audio content analysis involves the following: - **Timbre**: Features like MFCCs are used for genre classification and “more-of-this” recommendations. - **Beat/Downbeat**: Tempo extraction (e.g., using madmom for 85 bpm). - **Tonal Features**: Pitch-class profiles for melody extraction or cover version identification (e.g., Essentia). - **Semantic Categories**: Machine learning enables predictions like *not_danceable*, *mood_not_happy*, etc.

6.6 Text Analysis Methods (Basic IR)

Text analysis leverages user-generated content and lyrics to capture aspects beyond the audio signal. Methods transform content similarity tasks into text similarity tasks using tools such as: - Bag-of-Words, Vector Space Models, TFIDF. - Advanced methods like topic modeling, word2vec, BERT, and representation learning. Applications include tag-based similarity, sentiment analysis, and mood detection in lyrics.

6.7 Textual Sources of Music Similarity

Text processing of user-generated data (reviews, blogs, tags) is critical for music similarity. However, the process involves noisy and potentially erroneous data. Similarity measures (e.g., cosine distance) are applied, but personalization remains limited as it depends on user-generated content.

6.8 Multimodal Approaches

Multimodal approaches integrate multiple sources of information to enhance recommendation systems, particularly to address cold-start problems. Examples include: - Combining text embeddings (e.g., artist biographies) with CNN-trained audio embeddings [**Oramas2017**]. - Fusing deep features from audio, image (e.g., album covers), and text.

6.9 Feedback-Transformed Content

Content features are aligned with collaborative filtering (CF) models to address cold-start problems. Approaches include: - Predicting CF data from audio features. - Personalizing mixtures of content features. - Learning item-based CF similarity functions using metric learning [**McFee2012**]. - Weighted matrix factorization with CNN inputs (e.g., mel-spectrograms) [**vanOord2013**]. - Tag-trained neural networks integrated with matrix factorization to emphasize content [**Liang2015**].

6.10 Beyond Items

Describing items through non-personalized models helps create generic recommendations. These models use aggregated user data to focus on aspects like cultural or contextual metadata, independent of individual users.

6.11 Listener Background

Psychological and sociological factors are vital for building predictive user models and addressing cold-start issues. Factors include: - User personality, music preferences, demographics, and cultural context (e.g., age, mood, individualist vs. collectivist cultures). - Findings: Preferences vary by age, mood, and cultural background. For example, introverts may prefer sad music, while open individuals may seek uplifting music when sad.

6.12 Listener Context

The user's context significantly affects music recommendations. Types of context include: - **Environment-related**: Factors like time, location, weather, and traffic conditions. - **User-related**: Includes activity, emotion, and cultural context. Context data is obtained through: 1. Explicit methods (e.g., user input via questionnaires). 2. Implicit methods (e.g., sensors in smart devices like accelerometers or noise detectors). 3. Inference using rules or machine learning techniques (e.g., inferring mood from skipping behavior or activity from movement data).

7 Drum Transcription

7.1 Introduction

7.1.1 Explain the automatic drum transcription (ADT) task!

7.1.2 What are the differences between drum transcription and harmonic instrument transcription?

7.1.3 What kind of data is needed for automatic drum transcription (training and evaluation)?

7.1.4 Which metrics are used for ADT evaluation?

7.2 Methods Overview

7.2.1 What are the three categories of ADT methods we discussed in the lecture?

7.2.2 Explain the segment and classify approach for ADT!

7.2.3 Explain the separate and detect approach for ADT!

7.2.4 Explain the activation function based approach for ADT!

7.3 Non Negative Matrix Factorization

7.3.1 Explain in your own words the algorithm for non-negative matrix factorization!

7.3.2 What are the challenges that come with NMF for automatic drum transcription?

7.3.3 Name variants for NMF in the context of ADT, what are the differences?

7.4 NN Based Methods

7.4.1 Explain the setup for neural network based ADT!

7.4.2 Explain the difference between the multi-task and single-task NN approach for ADT!

7.4.3 What are questions to answer when trying to choose a model for a given task?

7.4.4 Explain the differences between dense, recurrent, convolutional, and dilated-conv neural networks! What are the advantages and disadvantages of each architecture category?

7.4.5 Explain how a CNN/RNN/TCN can be used to perform automatic drum transcription! How is the spectrogram processed in the context for each model?