194.039 Intelligent Audio and Music Analysis

Author:

Max Tiessler

2025-01-15

Contents

1 Fundamentals

1.1 Basics

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

- **Sound**: Generated by a **vibrating object**, causing **displacements** and **oscillations** of air molecules. Measured in **Hertz (Hz)**, which represents cycles per second.
- **Audio**: The transmission, reception, or reproduction of **sound** within the human hearing range (20 Hz to 20 kHz). Includes dynamic and tonal information essential for accurate sound reproduction.
- **Music**: Structured arrangement of **audio events** in terms of **time** and **pitch**, often with artistic or emotional intent.

1.1.2 What is the frequency range for audio?

- The frequency range for human hearing is typically 20 Hz to 20 kHz.

1.1.3 Give examples for objective properties in music!

- **Pitch**: Frequency of the sound wave.
- **Duration**: Time the sound is held.
- **Loudness**: Perceived intensity or amplitude of the sound wave.
- **Timbre**: Unique quality or color of the sound.

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

- An **audio waveform** is a representation of sound, with **time** on the x-axis and **amplitude** on the y-axis.
- It is created by converting sound pressure variations into an electrical signal, typically using a microphone.

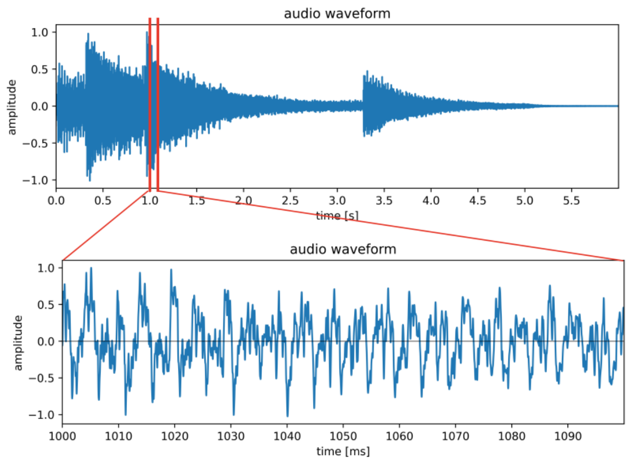

Figure 1: Audio Waveform

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

- **Analog-to-Digital Conversion (ADC)** converts analog signals into digital data.
- Steps include:
  1. **Sampling**: Capturing the signal at discrete time intervals.
  2. **Quantization**: Mapping amplitude values to discrete levels.

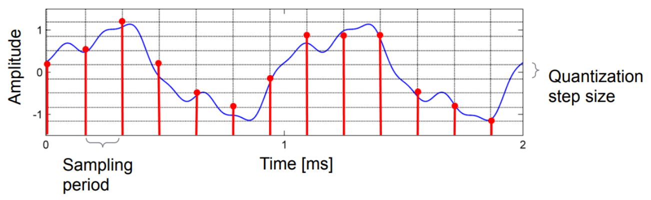

Figure 2: ADC example

  3. **Encoding**: Representing the quantized values as digital data.

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

- **Sampling Frequency**: Determines the temporal resolution and the highest frequency that can be accurately captured (**Nyquist frequency**).
- **Bit Depth**: Affects the amplitude resolution and dynamic range. Higher bit depth reduces **quantization error**.

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

- The sampling frequency must be at least twice the highest frequency in the signal, as per the **Nyquist–Shannon Sampling Theorem**. Frequencies above half of sampling frequency (Nyquist frequency) cannot be captured.
- Common audio sampling rates: 44.1 kHz (CD quality), 48 kHz, 96 kHz.

1.1.8 What does PCM stand for?

- **PCM** stands for **Pulse-Code Modulation**.

1.1.9 Explain how PCM encodes an audio signal!

- PCM encodes an audio signal by:
  1. Sampling the analog signal at equidistant intervals.
  2. Quantizing the amplitude of each sample to a finite set of values.
  3. Encoding the quantized values as binary data.
- PCM is the foundation of most digital audio formats.

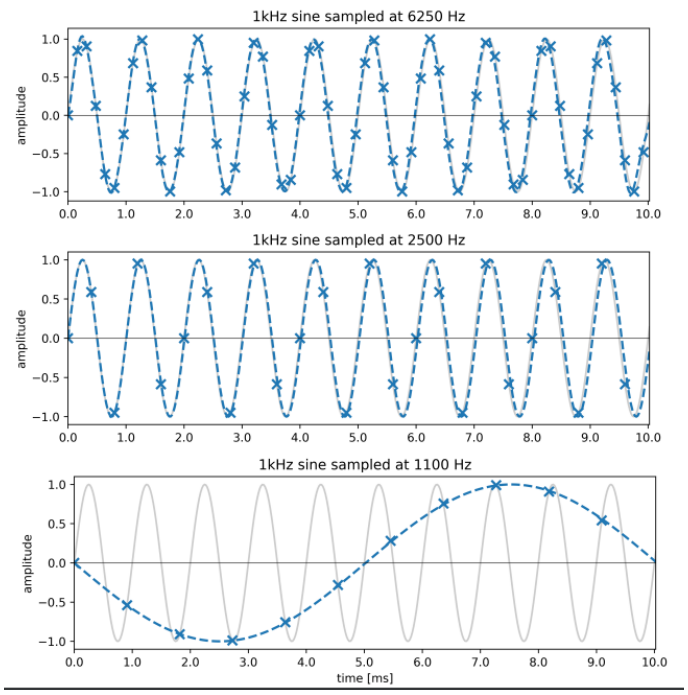

Figure 3: Nyquist–Shannon Sampling Theorem

1.2 Audio Features and Signal Framing

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

- **Audio features** are numerical representations or summaries of the properties of an audio signal.
- They are used to extract meaningful information from audio signals for analysis or classification.
- Features can be categorized as:
  - **Time-domain features**: Derived directly from the audio waveform.
  - **Frequency-domain features**: Derived from the signal's spectral representation (e.g., Fourier Transform).
  - **Time-frequency domain features**: Combine time and frequency information (e.g., spectrogram).

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

- **Root Mean Square (RMS) Energy**: Measures the average energy of the signal; correlates with perceived loudness.
- **Zero Crossing Rate (ZCR)**: Counts the number of times the signal crosses the zero-amplitude line; indicates signal noisiness or pitch.
- **Amplitude Envelope**: Tracks the maximum amplitude within short time frames; useful for analyzing dynamics.

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

- **Limited Temporal Information**: Global features represent the entire signal with a single value, ignoring temporal changes.
- **Dynamic Nature of Music**: Music often changes in pitch, rhythm, and timbre over time; global features cannot capture these variations.
- **Dependency on Signal Length**: Some global features may vary with the length of the signal, reducing their reliability for comparison.

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

- **Signal framing** is the process of dividing an audio signal into short, overlapping segments called **frames**.
- Key parameters:
  - **Frame length**: Number of samples in each frame (e.g., 512–8192 samples, corresponding to 11–186 ms at 44.1 kHz).
  - **Hop size**: Number of samples between the start of consecutive frames; determines the overlap.
  - **Overlap**: Difference between frame length and hop size; ensures smooth transitions between frames.

  The main idea is: `overlap = frame_len - hop_size`
- Purpose: To capture short-time properties of the signal while preserving time-local information.

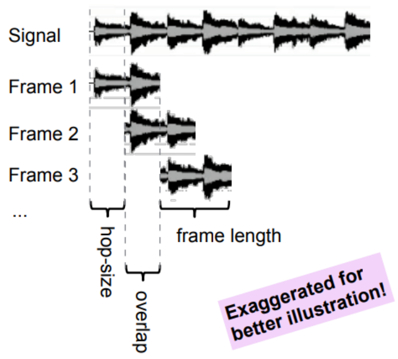

Figure 4: Signal Framing

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

- The signal is divided into frames, and features (e.g., ZCR, RMS, spectral features) are calculated for each frame individually.
- By analyzing each frame, variations in features over time can be captured, revealing the dynamic nature of the signal.

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

- Time-local features preserve information about changes in the signal over time, crucial for dynamic audio content like music.
- They enable tasks such as beat tracking, onset detection, and speech analysis, where temporal variations are critical.

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

- **Zero Crossing Rate (ZCR)**:
  - Measures the rate at which the signal crosses the zero line.
  - Reveals information about pitch or noisiness of the signal.
- **Short-Time Energy (RMS)**:
  - Measures the signal's power within a frame.
  - Indicates loudness and energy dynamics.
- **Spectral Centroid**:
  - Represents the center of gravity of the spectrum.
  - Indicates whether the sound is dominated by high or low frequencies.

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

- **Zero Crossing Rate (ZCR)**:
  1. Divide the signal into overlapping frames.
  2. For each frame, calculate the number of times the signal changes sign (crosses zero).
  3. Normalize the count by the frame length to obtain the ZCR for that frame:

$$ZCR_t = \frac{1}{2} \sum_{k=tK}^{(t+1)K-1} |\mathrm{sgn}(s(k)) - \mathrm{sgn}(s(k+1))|$$

     where $s(k)$ is the amplitude of the k-th sample, and K is the frame length.
- **Interpretation**: A high ZCR indicates a noisy or high-frequency signal, while a low ZCR suggests smoother or lower-frequency content.

1.3 Frequency Domain Features

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

The human auditory system analyzes audio signals in terms of frequency, much like how frequency-domain analysis works in signal processing. Within the cochlea, specific regions respond to distinct frequency ranges, decomposing complex sounds into simpler frequency components. Similarly, frequency analysis methods such as the Fourier Transform mimic this process by breaking down audio signals into their frequency components, helping to understand characteristics like pitch, timbre, and harmonics.

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

The Fourier Transform is a mathematical method that converts a signal from its **time-domain representation** (amplitude over time) into its **frequency-domain representation** (amplitude and phase over frequency). It essentially breaks a signal down into a sum of sinusoidal waves of varying frequencies, amplitudes, and phases. This makes it possible to identify the underlying components of complex signals, which is invaluable for tasks like pitch detection and spectral analysis.

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

1.3.4 Sketch a signal that is a mixture of two sines and its magnitude spectrum!

1.3.5 Explain amplitude, frequency, and phase for a sinusoid!

A sinusoid is fully described by its amplitude, frequency, and phase.

- The amplitude represents the maximum strength or height of the wave, correlating with how loud the sound appears.
- The frequency defines how often the sinusoid oscillates per second, measured in Hertz (Hz), and directly relates to the pitch of the sound.
- The phase describes the relative starting point of the wave, measured in radians or degrees, and influences how the sinusoid aligns with others when combined.

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

A frequency-domain representation describes an audio signal in terms of its frequency components rather than its time-dependent waveform. It reveals how much energy is present at each frequency, making it easier to analyze harmonic content, detect dominant pitches, and study spectral characteristics of the signal.

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

A spectrum is a graphical representation of a signal's frequency content. The x-axis typically shows the frequencies in Hertz, while the y-axis indicates the magnitude, representing the energy or amplitude of the signal at each frequency. It is a powerful tool for visualizing and analyzing the frequency components of an audio signal.

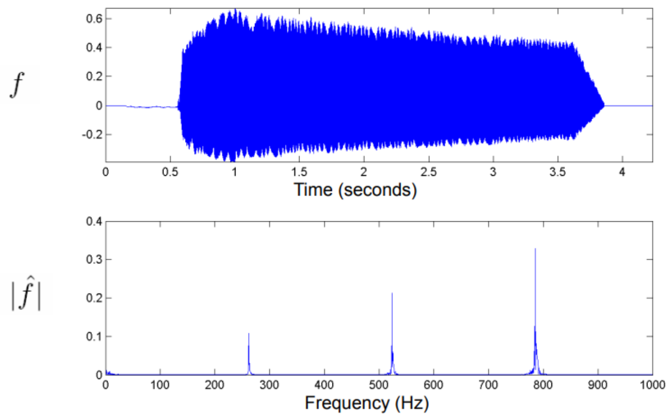

Figure 5: Time-domain to Frequency-domain example

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

The phase values in a spectrum indicate the relative timing of frequency components. While the magnitude spectrum shows the strength of each frequency, the phase spectrum is essential for reconstructing the original signal accurately. Phase affects how the frequency components combine in time, influencing the overall shape and alignment of the waveform.

1.3.9 When calculating a DFT, what parameters influence the frequency resolution of the spectrum?

The frequency resolution of a Discrete Fourier Transform (DFT) depends on two key parameters:

1. The sampling frequency
2. The number of samples

A higher number of samples results in smaller frequency intervals, improving resolution. Similarly, the sampling frequency determines the range of frequencies that can be analyzed. Together, these parameters control the ability to distinguish between closely spaced frequencies in the spectrum.

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

Frequency-domain features include:

1. **Spectral centroid**, which represents the center of gravity of the spectrum and indicates whether a sound is dominated by high or low frequencies
2. **Bandwidth**, which describes the range of significant frequencies in the signal.

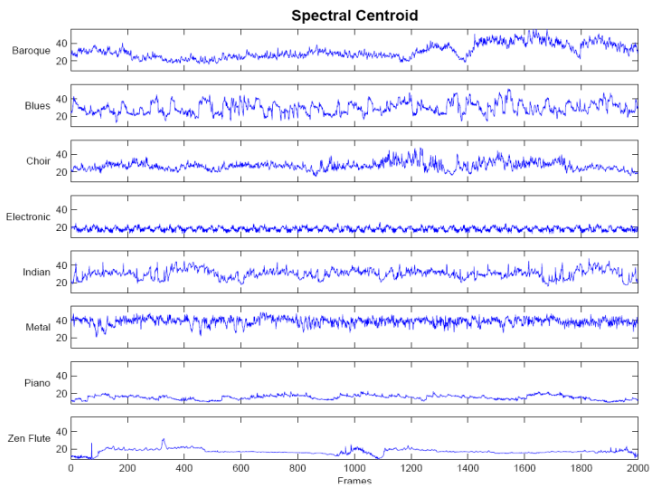

Figure 6: Spectral Centroid example

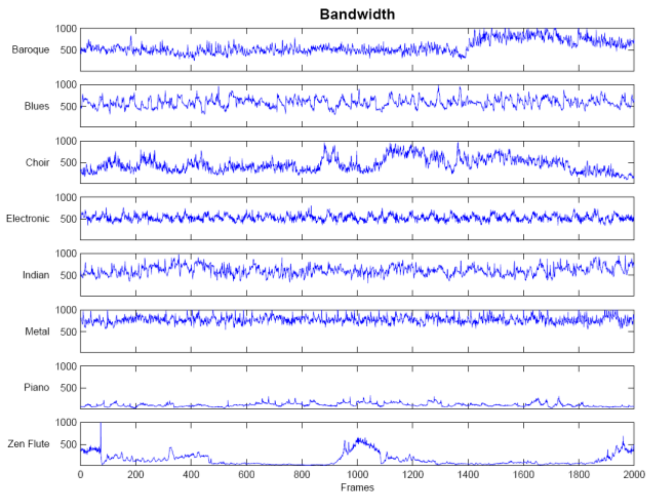

Figure 7: Bandwidth Example

1.4 Time-Frequency Domain Features

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

The Fourier Transform provides information about the frequency components of a signal but averages this information over the entire duration of the signal, losing time-local information.
Music and other audio signals are dynamic, with changing frequencies, harmonics, and rhythms over time, which the Fourier Transform cannot capture.
For long signals, the assumption of **stationarity** (unchanging frequency components) is invalid, making the analysis less meaningful.

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

By dividing the signal into short, overlapping segments (frames) and performing frequency analysis on each segment individually. This approach, called **the Short-Time Fourier Transform (STFT)**, allows frequency components to be analyzed in small time intervals, maintaining temporal resolution.

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

The Short-Time Fourier Transform (STFT) is a technique that combines time and frequency analysis by dividing a signal into short overlapping frames and performing a Fourier Transform on each frame.

The steps are:

1. Divide the signal into overlapping frames of fixed length.
2. Multiply each frame by a window function to reduce edge effects.
3. Perform the Fourier Transform on each windowed frame to obtain its frequency content.

The result is a two-dimensional representation (time vs. frequency) of the signal.

1.4.4 What is the advantage of using the STFT instead of a simple Fourier Transform?

The STFT provides both time and frequency information, allowing analysis of non-stationary signals.

It enables tracking of changes in frequency content over time, making it particularly useful for analyzing music, speech, and other dynamic signals.

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

The window function determines how much of the signal is included in each frame and reduces artifacts caused by abrupt frame boundaries.

Common window functions, like the Hann or Hamming window, ensure smooth transitions between frames, reducing spectral leakage.

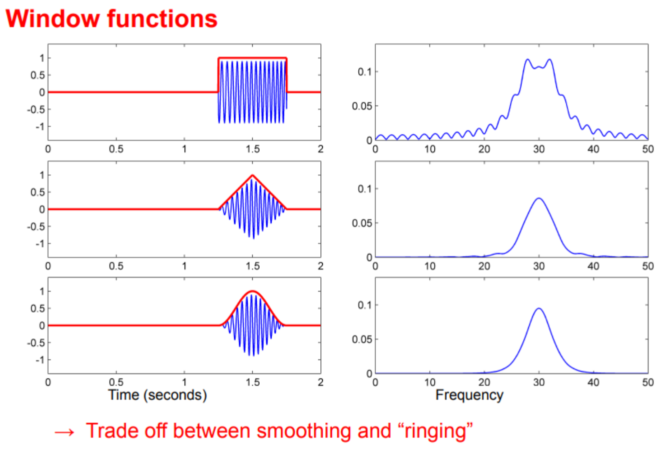

Figure 8: STFT Window Functions

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

A spectrogram is a visual representation of a signal's time-frequency content.

1. **x-axis**: Time (seconds).
2. **y-axis**: Frequency (Hertz).
3. **z-axis (color intensity)**: Magnitude or energy of the signal at each frequency and time.

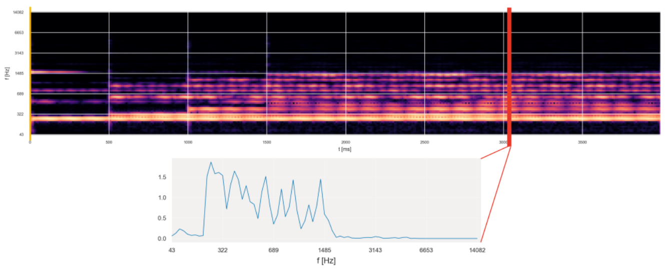

Figure 9: Spectogram Example

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

1.4.8 Give examples for time-frequency domain features!

- **Spectral Centroid**: Indicates the "center of gravity" of the spectrum at each time frame.
  Description: The center of gravity of the magnitude spectrum of the DFT, i.e., the frequency (band) region where most of the energy is concentrated.
  Remarks:
  - Used as a measure of sound sharpness (strength of high-frequency energy).
  - Sensitive to low-pass filtering (downsampling) as the high-frequency bands are given more weight.
  - Sensitive to white noise (for the same reason).
- **Spectral Bandwidth**: Represents the spread of the frequency spectrum around the centroid.
  Remarks:
  - The average bandwidth of a piece of music may serve as an indicator of aggressiveness.
  - Does not provide information about the perceived rhythmic structure.
  - Not suitable to distinguish different parts of a piece of music (e.g., vocal part in a metal piece may not be visible).
- **Spectral Flux**: Measures changes in the spectrum over time.
- **Mel-Frequency Cepstral Coefficients (MFCCs)**: Capture perceptually meaningful features derived from the spectrogram.

1.5 Psychoacoustics

1.5.1 Explain why psychoacoustics are relevant for audio signal processing!

Psychoacoustics studies how humans perceive sound, helping to design systems that align with human hearing capabilities.
By understanding perception, we can prioritize relevant audio information (e.g., important frequencies or loudness levels) and optimize compression, synthesis, and playback systems.

1.5.2 How are psychoacoustic scales created?

Psychoacoustic scales are derived from experiments that map physical sound properties (e.g., frequency, intensity) to perceived characteristics (e.g., pitch, loudness).

Examples include:

- Mapping frequency to perceptual pitch (e.g., Mel scale).
- Relating sound pressure to perceived loudness (e.g., Equal Loudness Contours).

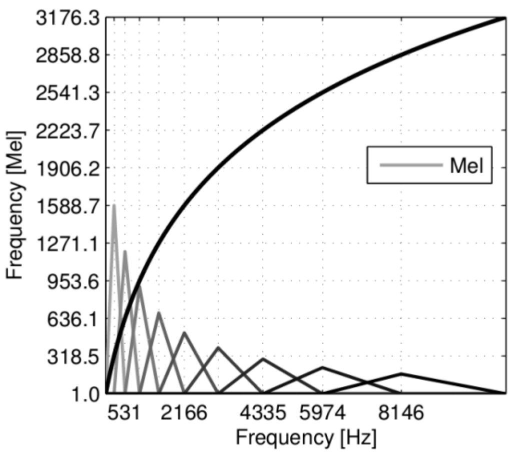

Figure 10: Mel Scale Example

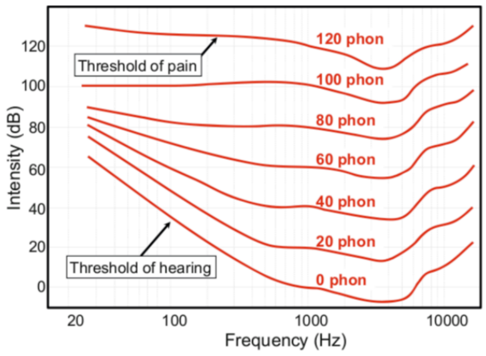

Figure 11: Equal Loudness Contour

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

Psychoacoustic pitch scales represent how humans perceive pitch, focusing on the nonlinear nature of auditory perception. Key scales include:

- **Mel Scale**: Reflects perceived pitch with equal spacing corresponding to equal perceptual differences. It is linear at low frequencies and logarithmic at high frequencies. Typically, 40 Mel-frequency bins are used, spaced equally on this scale.
- **Bark Scale**: Divides the frequency spectrum into critical bands, approximating human auditory filters. It is useful for studying auditory masking and sound interactions.
- **Cent Scale**: A logarithmic pitch scale where 1200 cents equal one octave. It provides precise representation of small pitch intervals.

- **Semitone Scale**: Maps frequencies to semitone intervals with a reference frequency (usually 440 Hz). Uses a filter bank with triangular filters centered on semitone frequencies, each spanning 100 cents.
- **dB Loudness Scaling**: A logarithmic scale for perceived loudness. Sound pressure levels in decibels are calculated as:

  $$L_p = 20 \cdot \log_{10}\left(\frac{p}{p_0}\right)$$

  For spectrograms, power values ($|S|^2$) are expressed as:

  $$S_{dB} = 10 \cdot \log_{10} |S|^2$$

  This allows relative loudness visualization when the reference value is omitted.

1.5.4 How do psychoacoustics influence loudness perception?

Loudness perception depends on frequency, with humans being more sensitive to mid-frequencies (1–5 kHz) and less sensitive to very low or very high frequencies.

The **Equal Loudness Contours** describe this non-linear perception, showing how different sound pressures are required for equal perceived loudness at various frequencies.

1.5.5 How can we apply psychoacoustic loudness scaling to spectrograms?

- Use logarithmic scaling (e.g., decibels) to reflect human perception of loudness.
- Normalize intensity values to emphasize perceptually important parts of the spectrum.

1.5.6 How can we apply psychoacoustic frequency scales to spectrograms?

- Transform the linear frequency axis into a psychoacoustic scale such as Mel or Bark.
- Apply a filter bank (e.g., Mel filter bank) to group and smooth frequency bins according to perceptual relevance.
- The resulting spectrogram reflects how humans perceive the frequency content of the signal.

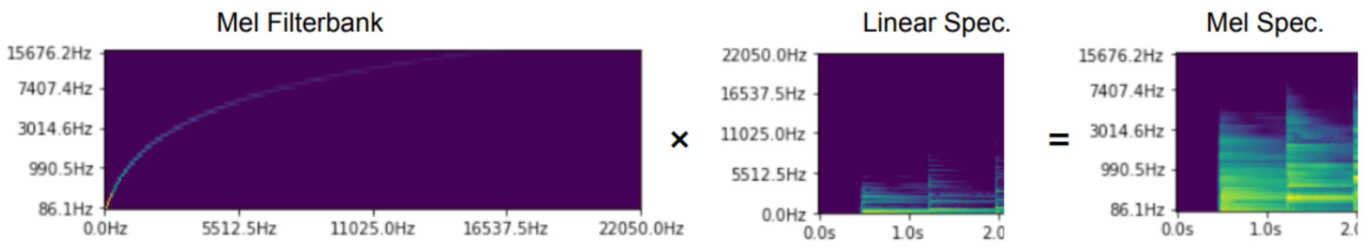

Figure 12: Filterbank example

2 Onset Detection

2.1 Notes and Onsets

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

Small-scale structures in music include:

- Events, individual notes.
- Chords
- Rhythmic patterns

These elements create the detailed framework of a piece, contributing to its texture and expressiveness.

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

Individual musical events are referred to as:

- *Notes* when describing pitch and duration
- *Onsets* when focusing on the timing of their initiation
- *Attacks* when emphasizing the initial phase of sound production.

2.1.3 Phases of the ADSR Model and Their Relation to Playing a Note on an Instrument

The ADSR model describes the envelope of a sound in four distinct phases, which correspond to how the amplitude of a sound changes over time. These phases are crucial for understanding how musical instruments produce and shape their sound.

- **Attack** (*transient*): The initial phase where the sound's amplitude increases rapidly. For example, on a piano, this corresponds to the moment the hammer strikes the string, creating a sudden onset of vibration and sound.

- **Decay**: Following the attack, this phase sees the amplitude decrease to a steady level. In a piano, this corresponds to the string's vibration stabilizing after the initial strike.

- **Sustain** (*resonance*): This is the phase where the sound is held at a relatively constant amplitude. The duration of this phase depends on the instrument and playing technique, such as bowing a violin string continuously or holding down a piano key.

- **Release**: The final phase where the sound fades to silence. On a piano, this happens when the key is released, and the damper stops the string's vibration.

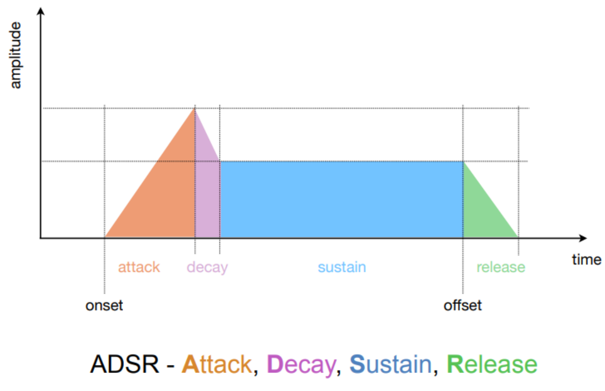

Figure 13: ADSR Model

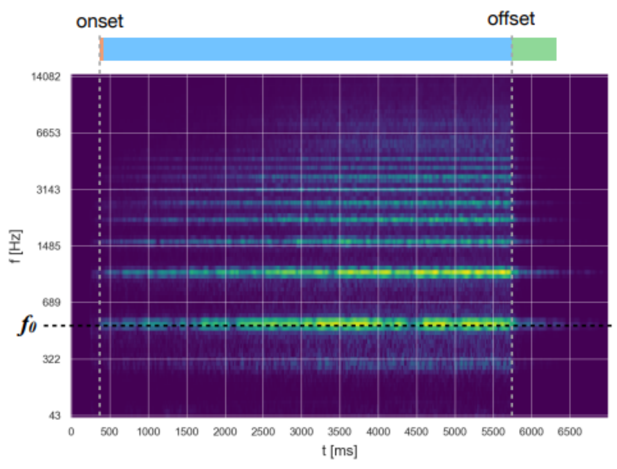

Figure 14: ADSR Spectrogram Example

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

Musical notes are characterized by:

- Pitch (frequency)
- Duration
- Intensity (loudness)
- Timbre
- Onset time

These properties determine how notes are perceived and how they interact in a musical context.

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

The *onset* marks the **beginning of a note**, corresponding to the point in time when sound energy significantly increases. The *transient* refers to the brief, non-stationary period at the start of the note, often encompassing the attack phase, where the spectral content changes rapidly.

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

- **Percussive sounds** are transient-rich and characterized by sharp, short attacks, such as drums or clapping.
- **Harmonic sounds** are sustained and exhibit stable periodic frequencies, like a bowed violin or a singing voice.
- **Percussive-harmonic sounds** combine both qualities, featuring distinct transients followed by sustained harmonic content, such as a piano or a plucked guitar string.

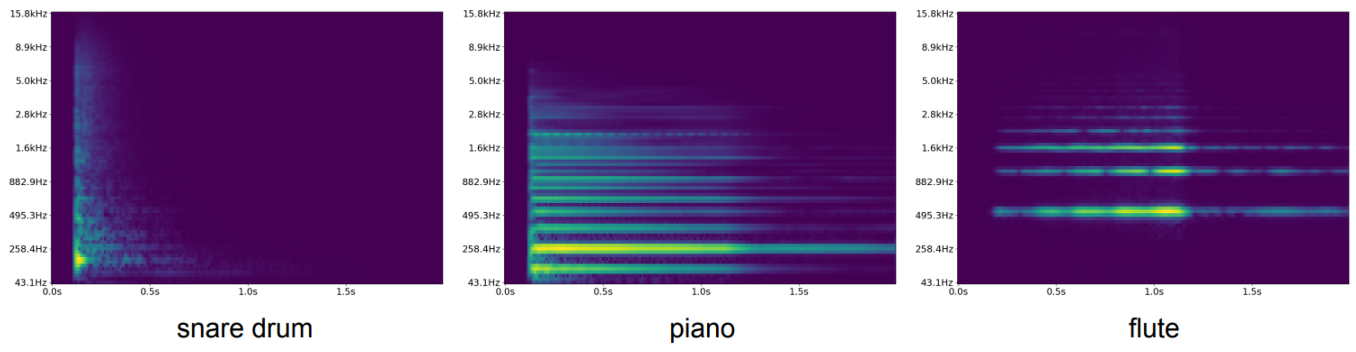

Figure 15: Different onsets example

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

Identifying onsets in **percussive sounds** is relatively straightforward due to their clear and sudden amplitude changes. For **harmonic sounds**, the challenge lies in detecting gradual energy increases or subtle spectral shifts, as the attack phase is less defined. **Percussive-harmonic sounds** present intermediate difficulty, where transient detection is feasible, but distinguishing overlapping harmonic content may require additional spectral analysis techniques.

2.2 Onset Detection - I. Signal Pre-Processing

2.2.1 What is the goal of onset detection?

The goal of onset detection is to identify the precise moments in time when new musical events or sounds begin. These moments, called onsets, are critical for analyzing rhythmic structures, extracting features, and enabling tasks such as music transcription or beat tracking.

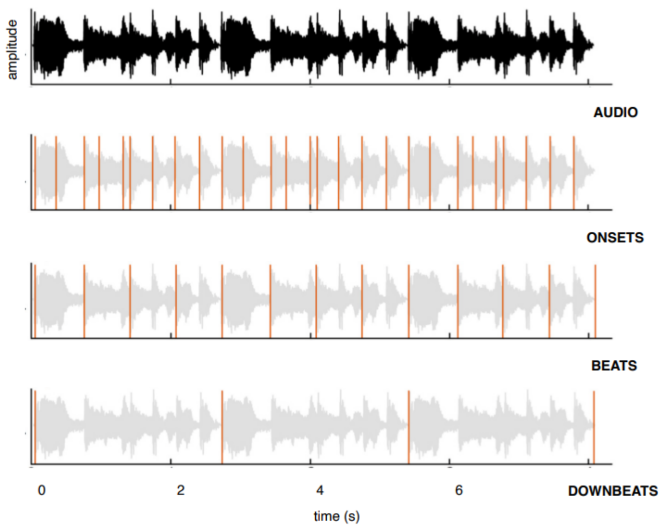

Figure 16: Onset, Beats and Downbeats detection

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

Challenges include:

1. **Variability in sound types**, such as percussive, harmonic, and mixed signals, each with unique characteristics.
2. **Overlapping sounds or polyphonic textures**, making it difficult to isolate individual onsets.
3. **Noise and distortions in real-world recordings**, which can obscure onset features.
4. **Gradual onsets**, particularly in harmonic or sustained sounds, where energy changes are subtle.

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

Onset detection is fundamental for many audio processing tasks, such as:

1. **Automatic music transcription**.
2. **Beat and tempo analysis**.
3. **Audio segmentation** for music structure analysis.
4. **Feature extraction** for machine learning applications.
5. **Synchronization of audio and video** or audio effects in production.

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

1. Signal preprocessing to enhance relevant features and suppress noise.
2. Computation of an onset detection function to highlight energy or spectral changes.
3. Application of peak picking algorithms to locate the exact onset positions.

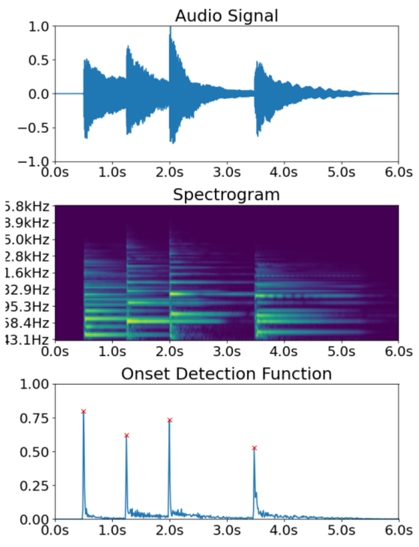

Figure 17: Pipeline

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

The goal of signal preprocessing is to prepare the audio signal for analysis by enhancing characteristics associated with onsets and reducing irrelevant or distracting components, such as noise or steady background signals.

2.2.6 Give examples for signal preprocessing for onset detection!

Examples of preprocessing techniques include:

- **High-pass filtering**: Removes low-frequency noise to focus on higher-frequency components relevant to onset detection.
- **Envelope extraction**: Captures the amplitude variations over time, which are often indicative of onsets in the signal.
- **Short-Time Fourier Transform (STFT)**: Analyzes the time-frequency content, providing a spectrogram that reveals changes in energy over time.
- **Spectral smoothing**: Reduces noise and emphasizes abrupt changes in spectral content.
- **Adaptive whitening**: A technique to normalize the magnitudes $|X(n,k)|$ of each frequency bin based on past peak values. This is achieved using an iterative algorithm for causal (realtime) processing:

  $$X_{\text{norm}}(n,k) = \frac{X(n,k)}{\max(r, m \cdot P(n-1,k))}$$

  Here:

  - $X(n,k)$: Value of the STFT at frame index n and frequency bin index k.
  - $P(n-1,k)$: Past peak values.
  - m: Memory coefficient controlling the influence of past values.
  - r: Floor parameter to prevent division by very small values.

  This approach dynamically adjusts to the signal's characteristics, enhancing its suitability for onset detection.

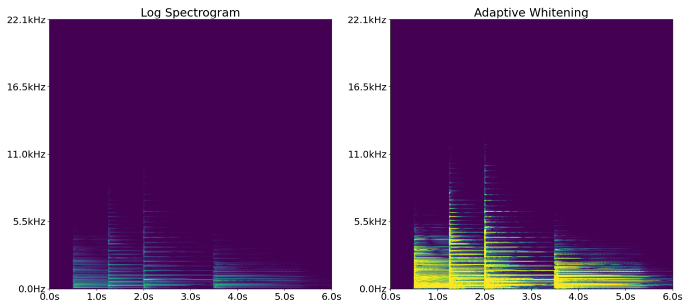

Figure 18: Adaptative Whitening example

2.3 Onset Detection - II. Detection Functions

2.3.1 What is an onset detection function?

An onset detection function is a time-domain signal derived from the audio input that emphasizes potential onsets by highlighting significant changes in energy, spectral content, or phase.

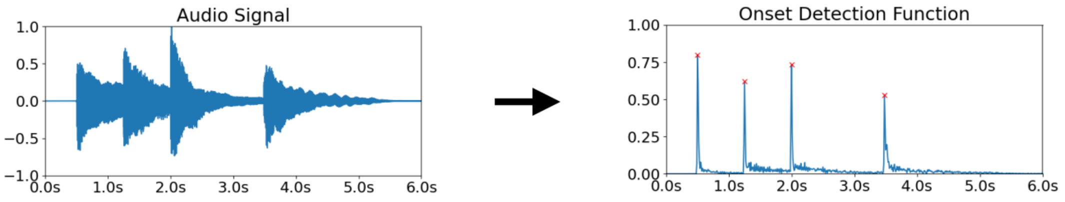

Figure 19: Example of onset detection function

2.3.2 What are the requirements for an onset detection function?

An effective onset detection function should:

1. Accurately capture changes in the audio signal that correspond to onsets.
2. Minimize false positives and negatives, even in noisy or complex signals.
3. Be computationally efficient for real-time applications.

2.3.3 Give examples for onset detection functions!

1. **Signal Envelope**: Tracks amplitude increases using a smoothed version of the signal. Improvements include using first-order differences or psychoacoustic "A-weighting." Simple but less reliable.
2. **High-Frequency Content**: Focuses on the broadband nature of transients by emphasizing high-frequency energy using weighted STFT magnitudes.
3. **Spectral Difference**: Measures changes between successive spectrogram frames using norms (e.g., L_1 or L_2). Spectral flux applies half-wave rectification for better results with logarithmic spectrograms.
4. **Phase Deviation**: Uses the second derivative of unwrapped phase from the STFT. Detects transients as deviations from constant phase increments, with "mean absolute phase deviation" being a common metric.

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

The phase-deviation onset detection function identifies onsets by analyzing sudden and significant changes in the phase relationships of frequency components. These changes often occur when a new sound event disrupts the steady phase progression of ongoing frequencies.

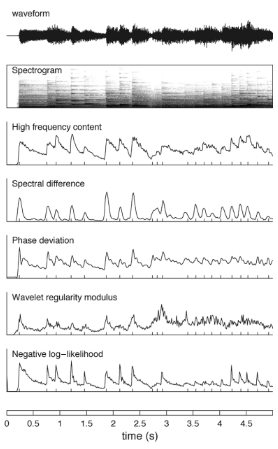

Figure 20: Example of onset detection functions

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

1. Compute the Short-Time Fourier Transform (STFT) of the audio signal to obtain its spectral representation.
2. Calculate the magnitude spectrum for each time frame.
3. Measure the frame-to-frame changes in the magnitude spectrum by comparing consecutive frames.
4. Retain only positive changes (increases in energy) to focus on potential onsets.
5. Aggregate the changes across frequencies for each time frame to generate the detection function.

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

1. **Percussive signals**: Characterized by sharp transients, making onsets easier to detect but susceptible to false positives from noise.
2. **Harmonic signals**: Gradual onsets with stable frequencies make detection challenging, requiring more sophisticated spectral analysis.
3. **Polyphonic signals**: Multiple simultaneous onsets or overlapping events increase complexity, necessitating advanced methods to distinguish individual onsets.

2.4 Onset Detection - III. Peak Picking

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

The goal of peak picking is to identify the exact positions of local maxima in the onset detection function that correspond to actual sound onsets.

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

Challenges include:

- Differentiating true peaks from noise-induced artifacts.
- Managing closely spaced onsets in fast sequences.
- Handling variable peak prominence due to dynamic changes in signal energy.

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

1. **Post-processing**: Smooth the onset detection function to reduce noise and fluctuations. Apply also Normalization and DC (offset) removal.
2. Detect Local Maxima, using a **moving maximum filter**
3. Apply a threshold to filter out low-magnitude peaks.
4. **Apply adaptative thresholding** for detecting peaks to refine onset timing and eliminate spurious detections.

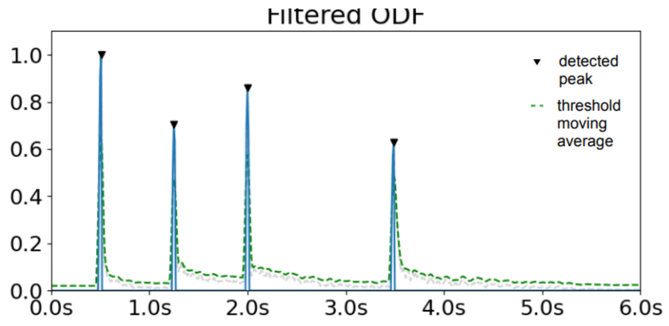

Figure 21: Post Processed ODF example

2.5 Performance Evaluation

2.5.1 What kind of annotations do we need to evaluate an onset detection method?

To evaluate an onset detection method, we need **ground-truth annotations** that specify the exact timing of each onset in the audio signal. These annotations are typically created manually by experts or derived from trusted sources.

2.5.2 What is a typical tolerance window for onset detection evaluation?

A typical tolerance window is **±50 ms** around the ground-truth onset. This accounts for slight variations in annotation precision and the temporal resolution of the detection method.

2.5.3 What evaluation metrics can be used for onset detection?

- **Precision**: The ratio of correctly detected onsets to all detected onsets.
- **Recall**: The ratio of correctly detected onsets to the total number of ground-truth onsets.
- **F-measure**: The harmonic mean of precision and recall, providing a single metric for overall performance.

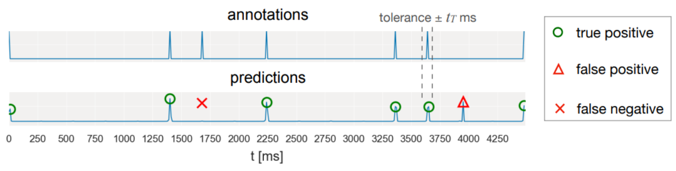

Figure 22: Example of evaluation

2.5.4 What are true positives, false positives, and false negatives in the context of onset detection evaluation?

- **True positives (TP)**: Onsets correctly detected within the tolerance window of ground-truth annotations.
- **False positives (FP)**: Detected onsets that do not correspond to any ground-truth onset within the tolerance window.
- **False negatives (FN)**: Ground-truth onsets that are not detected by the method.

2.5.5 What are true negatives, and why are they not used for onset detection evaluation?

True negatives (TN): Instances where no onset is present, and none is detected.
They are not used in onset detection evaluation because the number of true negatives is overwhelmingly large compared to other outcomes, making it irrelevant to precision, recall, and F-measure calculations.

2.5.6 Explain precision, recall, and f-measure. How are they calculated?

- **Precision**: Measures the proportion of detected onsets that are correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall**: Measures the proportion of ground-truth onsets that are correctly detected.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F-measure**: Combines precision and recall into a single score.

$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.6 State of the Art Examples

2.6.1 Give an example of a concrete implementation for an onset detector!

An example of a concrete implementation for an onset detector is **LibROSA's onset detection function**, which utilizes spectral flux as the detection function. LibROSA provides various options for preprocessing, including filtering and logarithmic scaling, as well as peak picking strategies.

A state-of-the-art approach based on spectral flux is the **LogFiltSpecFlux** method. It combines psycho-acoustically motivated preprocessing, spectral flux-based detection, and optimized peak picking to achieve high performance, particularly for certain signals.

- **Preprocessing**: Computes the STFT (window size 23ms, hop size 10ms), applies a filter bank to map the STFT to a semitone scale, and logarithmically scales the filtered magnitude spectrogram to compress the dynamic range:

  $$LFSf[t] = \log(1 + \lambda \cdot FSf[t])$$

  where $FSf[t]$ is the semitone-filtered spectrogram, and λ is a scaling parameter.

- **Detection Function**: Uses half-wave rectified differences to compute the spectral flux of the logarithmically scaled spectrogram:

  $$dLFSF[t] = \sum_{f=0}^{F-1} \max(LFSf[t] - LFSf[t-1], 0)$$

- **Peak Picking**: A point in the detection function is considered an onset if it satisfies:
  - It is a local maximum in a window (w_1, w_2).
  - It exceeds the mean in a surrounding window (w_3, w_4).
  - It maintains a minimum temporal distance (w_5) from other detected onsets.

The **LogFiltSpecFlux** method demonstrates state-of-the-art performance for onset detection in certain signal types, making it a robust and reliable implementation for audio analysis tasks.

2.6.2 Give an example for a vibrato compensation method for onset detection!

A vibrato compensation method addresses the challenge of vibrato (modulation in frequency) that can cause false positives in onset detection. One effective approach is the **SuperFlux** method, an extension of the "LogFiltSpecFlux." This method applies a **maximum filter** (± 1 quarter tones) along the frequency axis before computing the spectral difference. By compensating for frequency modulations, SuperFlux reduces false detections caused by vibrato.

Another approach is applying a **moving average filter** to smooth frequency fluctuations caused by vibrato, further minimizing the impact of these periodic variations on onset detection.

2.6.3 What is the difference between offline, online, and real-time processing?

- **Offline processing**: Analyzes the entire signal after it has been recorded, allowing access to future information for more accurate detection.
- **Online processing**: Processes the signal incrementally as it is received, without access to future data, but with some latency.
- **Real-time processing**: Processes the signal with minimal latency, suitable for live applications like live performances or interactive systems.

2.6.4 What difficulties bring online processing in the context of onset detection?

Online processing challenges include:

- Lack of future context, making it harder to confirm onsets with high confidence.
- The need for efficient algorithms to ensure low latency.
- Increased susceptibility to noise and false positives due to real-time constraints.

2.7 Onset Detection via ML

2.7.1 Why is machine learning a suitable technique for onset detection?

Machine learning is suitable for onset detection because it can learn complex patterns and features directly from data, including subtle variations in amplitude, frequency, and phase that traditional methods might overlook.

2.7.2 Describe a typical ML pipeline for onset detection!

- Preprocess audio data to create input representations (e.g., spectrograms or MFCCs).
- Annotate data with onset timings for supervised learning.
- Train a machine learning model, such as a convolutional neural network (CNN) or recurrent neural network (RNN), using the annotated data.
- Evaluate the model's performance using metrics like precision, recall, and F-measure.
- Deploy the trained model for onset detection on unseen audio signals.

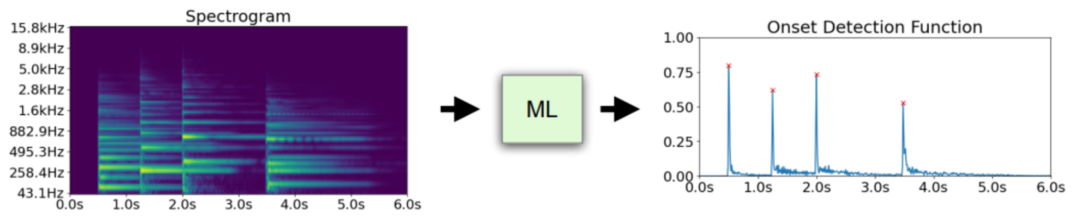

Figure 23: ML Pipeline diagram

2.7.3 How can we represent audio input for CNNs?

Audio input for CNNs can be represented as:

- **Spectrograms**: Two-dimensional time-frequency representations.
- **Mel-spectrograms**: Spectrograms transformed using the Mel scale.
- **MFCCs**: Mel-frequency cepstral coefficients for compact, perceptually relevant features.

2.7.4 What are the advantages of using ML for onset detection?

- Ability to learn complex, non-linear patterns from data.
- Adaptability to diverse audio content without manual feature engineering.
- Improved performance on challenging cases like polyphonic signals or gradual onsets.

2.7.5 What are the drawbacks of using ML for onset detection?

- Requires large, annotated datasets for effective training.
- High computational cost for training and inference.
- Potential lack of interpretability compared to traditional methods.

3 Beat Tracking

3.1 Introduction

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

- **Beats** represent the most prominent and regular pulses in music, often aligned with foot-tapping or clapping.
- **Downbeats** are the first beats in a measure, marking the start of a bar and often emphasizing rhythmic patterns and harmonic changes.
- **Measures**, also called bars, are recurring time intervals in music that group beats according to the meter.

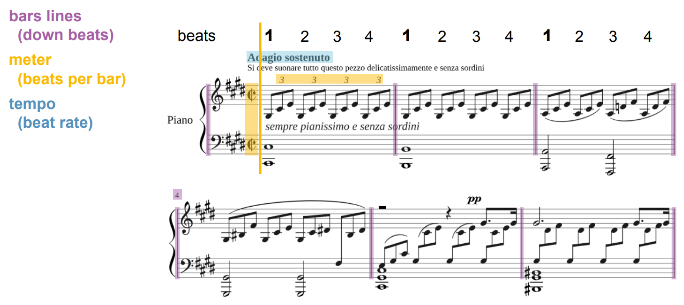

Figure 24: Musical Notation

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

Tempo is the speed at which beats occur in a musical piece. It is quantified in beats per minute (BPM), calculated as:

$$\text{BPM} = \frac{60}{\Delta t}$$

where Δt is the time interval between consecutive beats.

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

Local tempo refers to the tempo of a specific section or moment in the music, which may vary significantly. Global tempo represents the average tempo of the entire track, assuming a constant or dominant tempo throughout the piece.

3.1.4 What is the goal of beat tracking?

The goal of beat tracking is to **determine the precise timing of beats in a musical signal**, aligning them with the music's temporal structure and enabling tasks such as tempo estimation, rhythm analysis, and music synchronization.

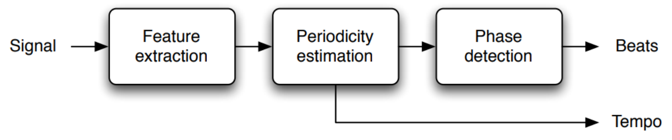

Figure 25: Pipeline

3.2 Tempo Estimation

3.2.1 What is the goal of tempo estimation?

The goal of tempo estimation is to determine the **periodicity of beats within a musical signal**, typically expressed in BPM, **to quantify the speed** of the music.

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

Inter-onset-interval (IOI) histograms are graphical representations of the time intervals between consecutive onsets in a musical signal. Peaks in the histogram correspond to dominant periodicities, which can be mapped to tempo values in BPM to estimate the tempo.

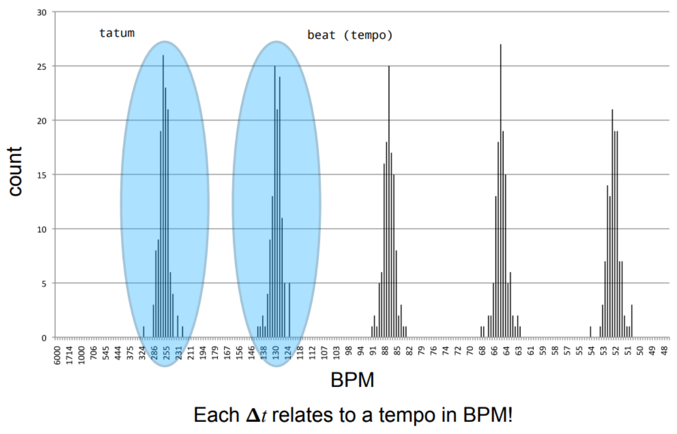

Figure 26: IOI example

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

Autocorrelation calculates the similarity of a signal with itself at varying time lags. It identifies repeating patterns in the signal, such as periodicities in an onset detection function, which can directly indicate the tempo by associating lag values with BPM.

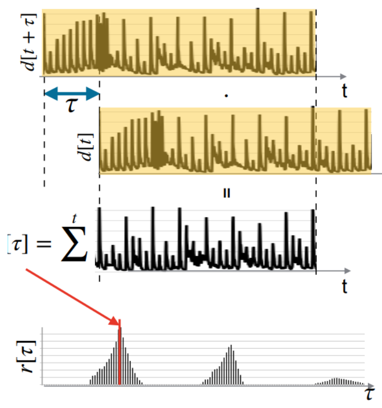

Figure 27: Autocorrelation example

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

Octave errors occur when the estimated tempo is an integer multiple or fraction (e.g., half or double) of the true tempo. These errors arise due to ambiguity in periodicity, where multiple tempos fit the rhythmic structure.

3.2.5 What is a tempogram?

A tempogram is a time-varying representation of tempo, calculated by analyzing periodicities in a sliding window. It shows changes in tempo over time, making it useful for tracks with significant tempo variations.

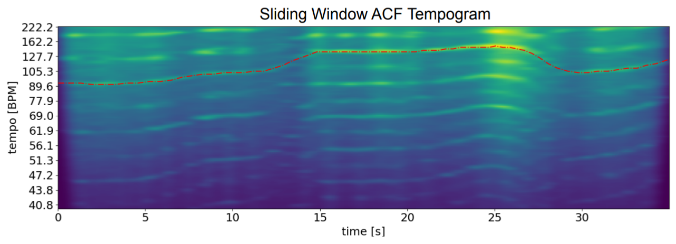

Figure 28: Tempogram example

3.3 Beat Tracking

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

Beat phase detection identifies the precise timing of individual beats within a periodic structure, aligning them with the underlying rhythm and tempo.

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

Beat phase detection can be performed using two methods:

1. **Cross-Correlation Method**: This approach uses the **onset detection function** (ODF) and a **pulse train generated from a tempo hypothesis**. The cross-correlation of the ODF with

the pulse train identifies the lag corresponding to the maximum correlation, which indicates the beat phase.

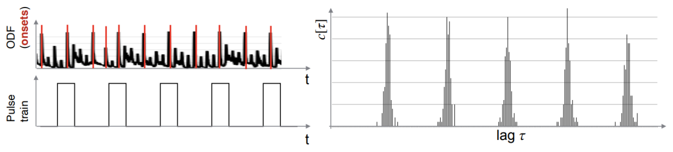

Figure 29: Pulse Train Cross-Correlation

2. **ACF Beat Tracker**: This method employs the **autocorrelation function** (ACF) of a median-filtered ODF. The filtering emphasizes salient events and reduces noise, enhancing periodicity estimation. Beats are aligned block-wise, which accommodates minor tempo variations but may struggle with large tempo changes or noisy signals. The performance of this method relies on multiple preprocessing steps, following a "weakest link" principle.

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

The greedy algorithm initializes the beat phase based on maximum cross-correlation. It iteratively advances by one beat period, aligning subsequent beats to the nearest onsets within a maximum allowed distance. If no onset is found, the algorithm approximates the beat position.

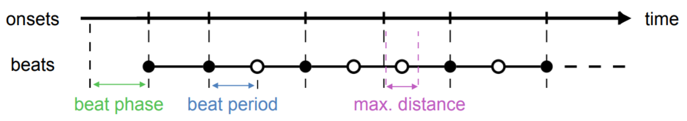

Figure 30: Greedy beat tracking

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

The multi-agent beat tracking approach generates agents corresponding to various tempo and phase hypotheses, leveraging IOI histograms to initialize tempo candidates. Each agent tracks beats independently, using a greedy algorithm to align beats with detected onsets. New agents spawn when events do not align with the current hypotheses, while similar agents can merge to consolidate tracking. A selective onset detector with high precision enhances performance. After tracking, surviving agents are scored based on beat fitness, and the beats from the highest-scoring agent are selected as the final sequence.

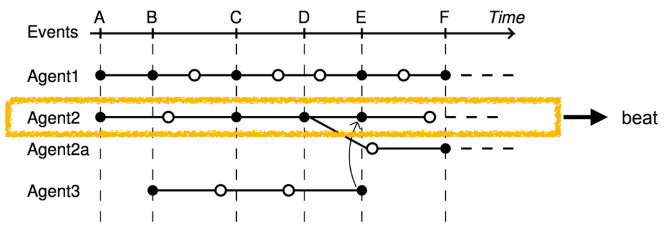

Figure 31: Multi-Agent Beat Tracker

3.3.5 Explain the machine learning setup for beat tracking.

In ML-based beat tracking, audio signals are converted into features (e.g., spectrograms) and labeled with beats and downbeats. A neural network (e.g., RNN) is trained to classify frames as beats, downbeats, or non-beats. The output probabilities are post-processed, often using a dynamic Bayesian network (DBN), to refine the predictions.

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

RNNs are well-suited for beat tracking due to their ability to model temporal dependencies in sequential data. Bidirectional RNNs, in particular, can consider both past and future context, making them effective for capturing rhythmic patterns.

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

The RNN predicts framewise probabilities for beats, downbeats, and non-beats using temporal dependencies in input features, such as spectrograms.

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

The DBN refines the RNN's output by jointly inferring tempo, meter, and phase, ensuring consistency across beats and downbeats while allowing tempo changes.

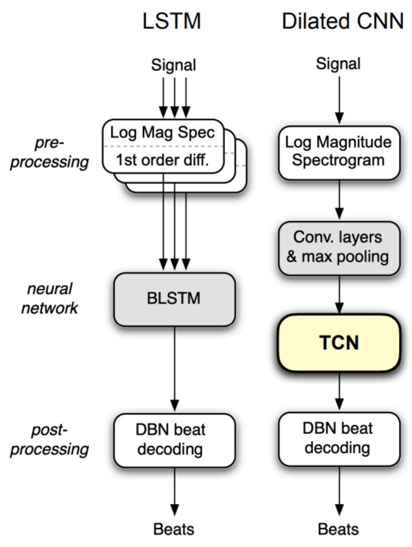

Figure 32: LSTM & Dilated CNN architectures

3.4 Evaluation

3.4.1 Which metrics can we use for beat tracking evaluation?

Metrics for evaluation include:

- **F-measure**: Measures precision and recall within a ±70 ms tolerance.

- **P-score**: Evaluates tracking accuracy based on deviations within 20% of the annotated beat interval.
- **CMLc and CMLt**: measure the longest continuously segment (CMLc) or all correctly tracked beats (CMLt) at the correct metrical level. A beat is considered correct if it is reported within a 17.5% tempo and phase tolerance, and the same applies for the previously detected beat.
- **AMLc and AMLt**: like CMLc & CMLt, but additionally allow offbeat and double/half as well as triple/third tempo variations of the annotated beats.

3.4.2 In what range are tolerances for beat positions typically?

Tolerances for beat tracking are typically in the range of **±70ms**, depending on the application.

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

Beat tracking tolerances are higher because beats may not align perfectly with onsets. Rhythmic structure and musical context allow for flexibility, making exact timing less critical 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.

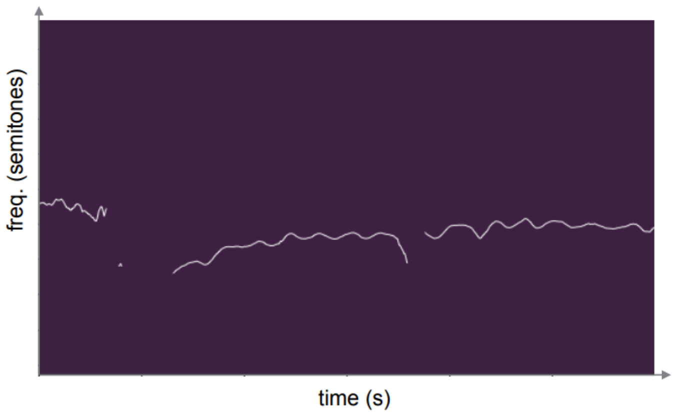

Figure 33: F0 contour

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.

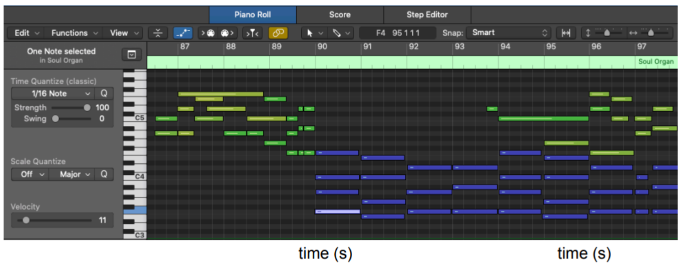

Figure 34: Piano Roll representation

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

1. **Frame-Level Detection**: Focused on detecting pitch contours. We estimate *f0* for each frame
2. **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.

3. **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.
- Each sounding pitch is evaluated separately.
- An absolute pitch difference of less than 50 cents (i.e., a quarter tone) is typically considered correct.
- Common metrics include Precision, Recall, F-measure, or Accuracy.

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 emphasizes musical properties such as the onset and offset times of the notes.

- Solves some of the problems associated with frame-wise evaluation.
- Suitable only for discrete, quantized notes (not continuous pitch contours).
- Various aspects of notes can be evaluated, such as:
  - Onset (e.g., within ±50ms) and pitch (e.g., within ±50 cents).
  - Onset, pitch, and offset (e.g., max ±50ms, 20% of note length).
- Metrics are usually measured independently, without an overall combined metric.

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: $\text{acc} = \frac{TP+TN}{TP+FP+TN+FN}$, 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. ±50ms) + pitch (e.g. ±50cents)
- Onset + pitch + offset (e.g. max ±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.

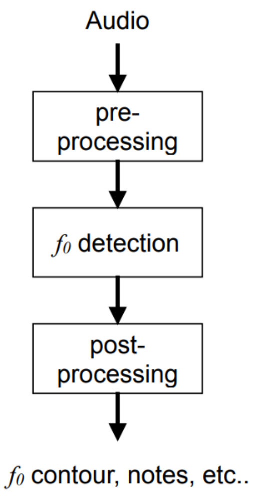

Figure 35: F0 estimation pipeline

  - 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.

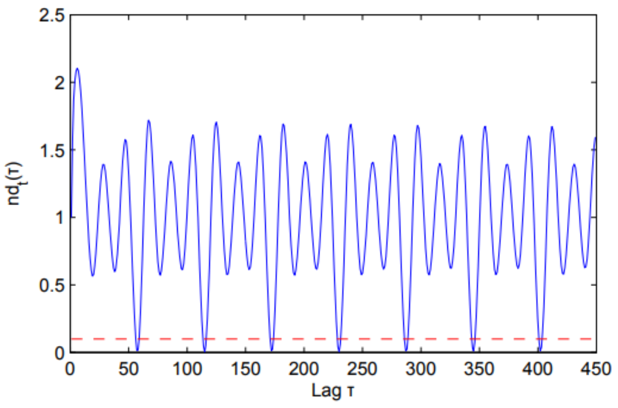

Figure 36: Thresholding in YIN

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.

Feature	**YIN**	**pYIN**
Core Methodology	Deterministic, time-domain analysis	Probabilistic extension of YIN
Output	Single F_0 value or absence per frame	Probability distribution over F_0 values
Noise Robustness	Moderate	High
Voicing Detection	Implicit	Explicit
Continuity in Output	May have jumps or discontinuities	Smoother pitch contours
Computational Cost	Lower	Higher
Use Case	Clean, monophonic signals	Noisy, real-world, or polyphonic signals

Table 1: Comparison of YIN and pYIN

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!

The automatic drum transcription (ADT) task aims to analyze audio signals to identify and transcribe drum events, including their timing and instrument classification (e.g., bass drum, snare drum, hi-hat). It enables applications such as generating sheet music, remixing, and higher-level tasks like genre classification and song segmentation.

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

Drum transcription is generally easier because: - Drum onsets are well-defined and often broadband. - There is no need for pitch (f0) estimation or offset detection. - The task focuses primarily on timing and instrument classification. Challenges include: - Higher temporal accuracy (10–20 ms) requirements. - Instrument classification is difficult due to spectral overlap. - Simultaneous onsets and masking effects complicate detection.

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

ADT requires: - Audio drum tracks (real recordings, synthesized, or sampled). - Annotations with high temporal accuracy (e.g., ± 10 ms), specifying drum instrument types and playing techniques. - Accompaniment tracks or isolated drum tracks for better generalization in machine learning models.

7.1.4 Which metrics are used for ADT evaluation?

Metrics include: - **Precision**: Proportion of correctly detected drum events. - **Recall**: Proportion of annotated drum events detected. - **F1-score**: Harmonic mean of precision and recall, calculated as:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

7.2 Methods Overview

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

1. **Segment and classify**: Detect onsets and classify them into drum instrument categories. 2. **Separate and detect**: Separate audio sources corresponding to different drum instruments and detect onsets within these sources. 3. **Activation function-based**: Use a single end-to-end system to extract onset detection functions for each instrument directly from the spectrogram.

7.2.2 Explain the segment and classify approach for ADT!

This approach involves two steps: 1. Detect onsets using basic methods (e.g., energy-based detection). 2. Classify each onset into a drum instrument category using rule-based methods (e.g., filters) or machine learning classifiers like support vector machines (SVMs).

7.2.3 Explain the separate and detect approach for ADT!

This approach separates audio sources corresponding to different drum instruments before onset detection. Techniques include: - Bandpass filtering for simple separation. - Advanced source separation methods like Independent Component Analysis (ICA) or Non-negative Matrix Factorization (NMF).

7.2.4 Explain the activation function based approach for ADT!

This state-of-the-art approach directly maps spectrograms to activation functions for each drum instrument using neural networks or NMF. Peaks in these activation functions indicate drum onsets, simplifying the processing pipeline.

7.3 Non-Negative Matrix Factorization

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

Non-negative Matrix Factorization (NMF) approximates a spectrogram X as the product of two matrices: - B: Basis matrix representing spectral templates for instruments. - G: Activation matrix indicating the presence of each template over time. The algorithm iteratively updates B and G using multiplicative update rules to minimize a loss function, typically the Kullback-Leibler divergence.

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

Challenges include: - Defining the number of templates (basis size). - Adapting to varying spectral energy distributions over time. - Handling harmonic and accompaniment content, which can overlap with drum components. - Assigning templates to specific instruments for meaningful transcription.

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

- **Fixed templates**: Predefined templates based on isolated drum hits. Works well for solo tracks but lacks adaptability. - **Semi-adaptive NMF**: Combines fixed templates with adaptive updates for better flexibility. - **Convolutional NMF**: Uses 2D templates to model spectral and temporal variations, offering improved accuracy at higher computational costs.

7.4 NN Based Methods

7.4.1 Explain the setup for neural network based ADT!

The setup involves: - Preprocessing audio to generate spectrograms. - Training a neural network (e.g., CNN, RNN, or TCN) with annotated drum events to predict activation functions for each instrument. - Post-processing activation functions to identify drum onsets and classify events.

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

- **Single-task NN**: A separate network is trained for each drum instrument. - **Multi-task NN**: A single network with multiple output layers predicts activation functions for all instruments simultaneously, improving efficiency and exploiting shared features across instruments.

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

1. Can the architecture model the problem efficiently? 2. Does it have a sufficiently large receptive field for context-aware predictions? 3. Is it computationally feasible with acceptable model size? 4. Can it be trained efficiently on the available data?

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?

- **Dense NN**: Fully connected layers; simple but unsuitable for time-series data. - **Recurrent NN (RNN)**: Captures temporal dependencies; effective for sequential data but hard to train due to vanishing gradients. - **Convolutional NN (CNN)**: Efficient for local feature extraction; limited temporal context unless stacked deeply. - **Dilated CNN (TCN)**: Provides a large receptive field with fewer layers, balancing efficiency and context modeling.

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?

- **CNN**: Processes spectrograms frame-by-frame, extracting spatial features representing drum events. - **RNN**: Processes sequential frames, capturing temporal dependencies to predict drum onsets over time. - **TCN**: Uses dilated convolutions to model long-range dependencies in spectrograms while maintaining computational efficiency.