

2023 September 13

Deep Learning for Music Analysis and Generation

Representations

for musical scores and audio



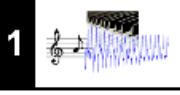
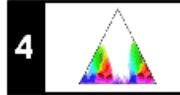
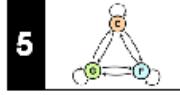
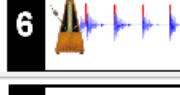
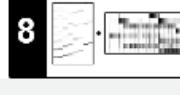
Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Outline

- Sheet music & symbolic representations for music
- Audio representation for music
- Math in STFT: frequency and temporal resolution

FMP Notebook

<https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1.html>

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
B	 Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
0	 Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
1	 Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
2	 Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
3	 Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]
4	 Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5	 Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
6	 Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
7	 Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
8	 Musically Informed Audio Decomposition	Harmonic/percussive separation, signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Outline

- **Sheet music & symbolic representations for music**
 1. Sheet music
 2. Piano roll
 3. MIDI
 4. MusicXML
- Audio representation for music
- Math in STFT: frequency and temporal resolution

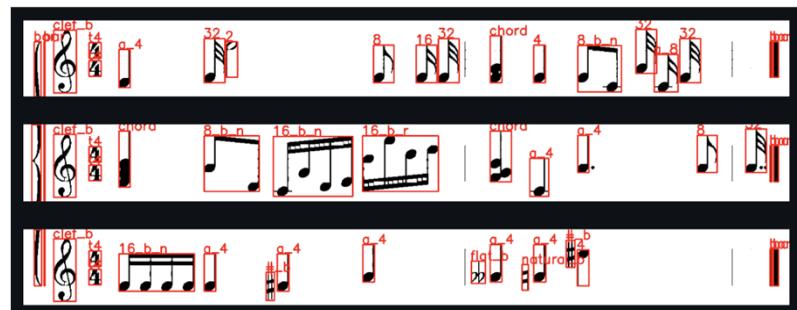
Sheet Music

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html

- A *visual* representation (*.PDF, *.PNG, etc)
 - A **guide** for performing a piece of music leaving room for different interpretations
 - Musicians may vary the **tempo**, **dynamics**, and **articulation**, thus resulting in a personal interpretation of the given musical score
- **Rarely** directly used as input to neural network models
 - Exception: *optical music recognition* (OMR)



Figure 1.1 from [Müller, FMP, Springer 2015]



Sheet Music: Key Signature & Note Duration

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html



Figure 1.5 from [Müller, FMP, Springer 2015]

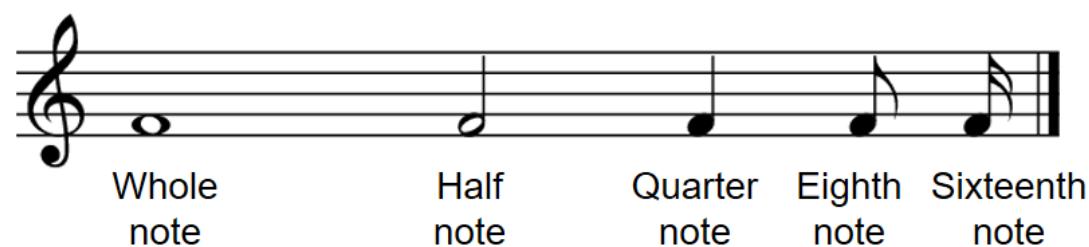


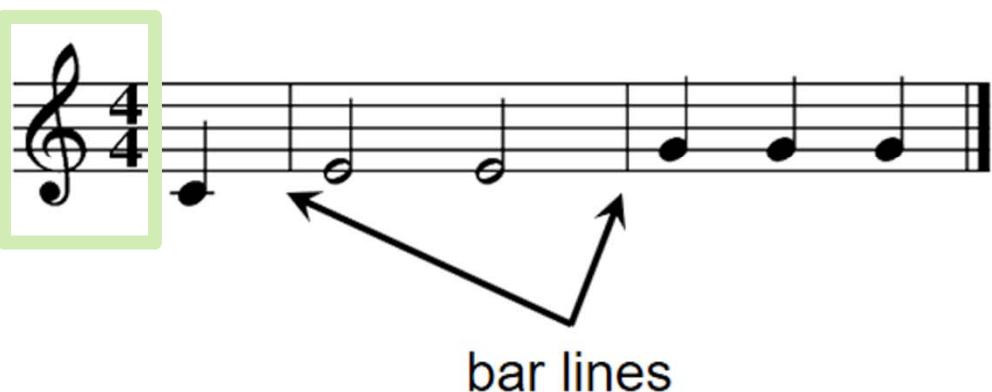
Figure 1.7 from [Müller, FMP, Springer 2015]

Sheet Music: Meter, Bar, Beat, Rhythm

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html

- Dividing music into measures (bars) not only reflects its rhythmic nature, but also provides regular reference points within it

Figure 1.6 from [Müller, FMP, Springer 2015]



Sheet Music: Articulation Marks

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html

- How certain notes are to be played

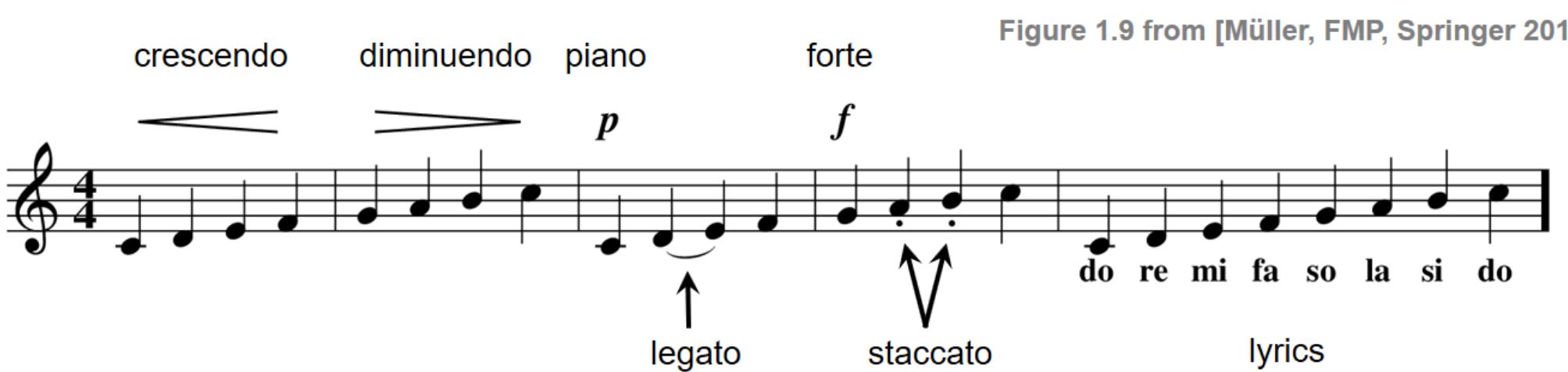
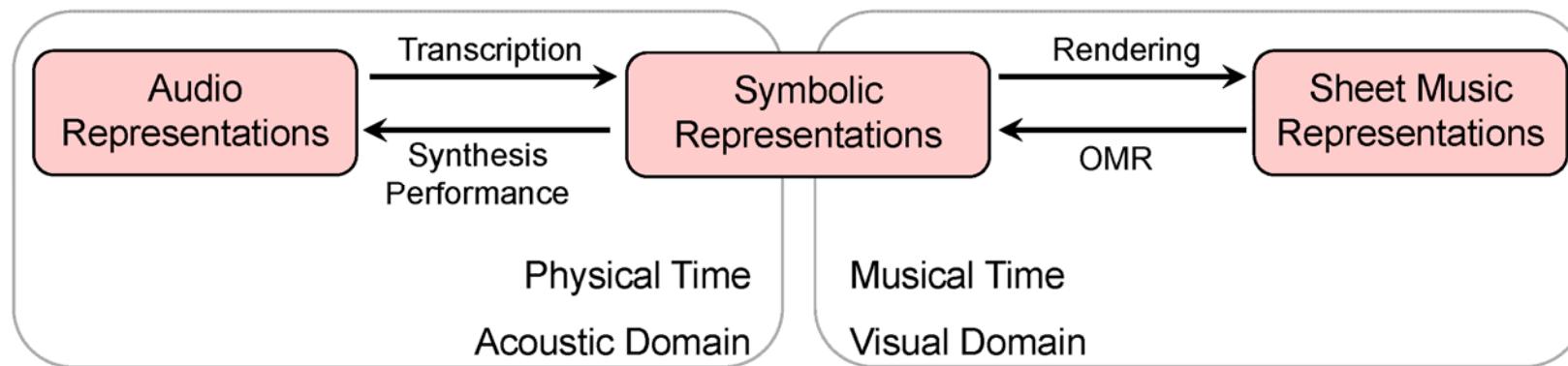


Figure 1.9 from [Müller, FMP, Springer 2015]

Sheet Music and Symbolic-domain Representations

Ref: <https://www.mdpi.com/2624-6120/2/2/18>



- **Musical time:** 1/4, 1/8, 1/16 etc, good for representation **scores**
- **Sheet music:** visual representations of a musical score either given in printed form or encoded digitally in some image format
- **Physical time:** in milliseconds or alike, good for representation **performances** (*tempo, dynamics, and articulation*)
- **Symbolic representations**

Piano Roll Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_PianoRoll.html

- A *visual, image-like* representation of *.MIDI (also how DAWs visualize MIDI files)

- Horizontal axis: **Time**
 - Vertical axis: **Pitch**
 - Rectangle: **Note**
 - Leftmost point: **Onset**
 - Lowermost point: **Pitch**
 - Width: **Duration**
 - (Can also indicate **Velocity**)

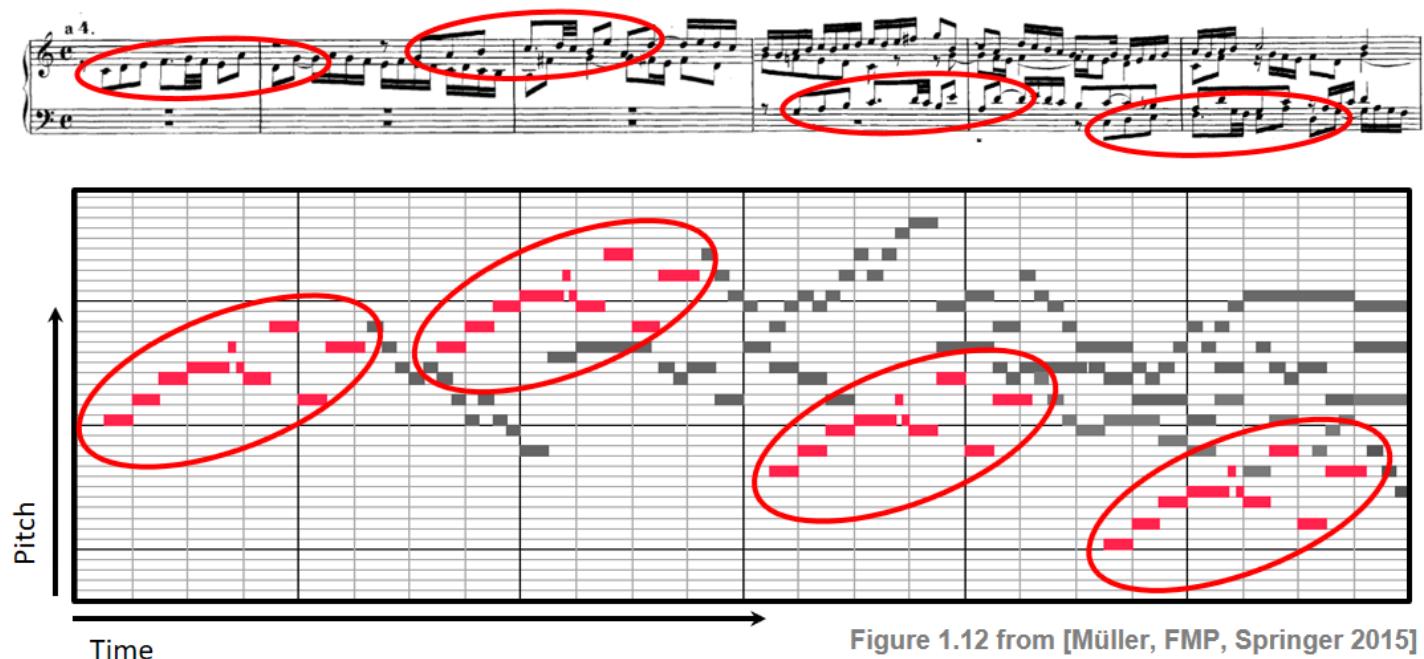
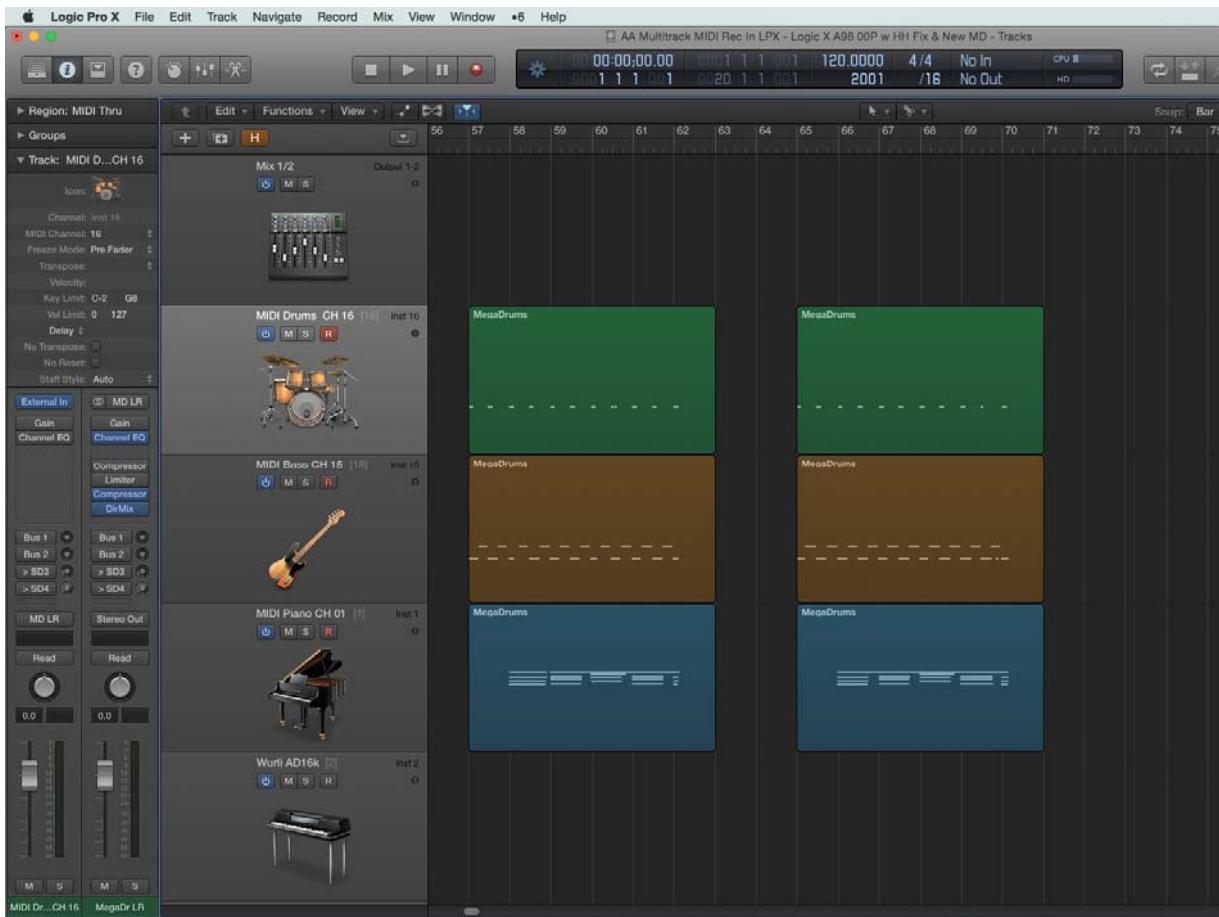


Figure 1.12 from [Müller, FMP, Springer 2015]

ps. The four occurrences of the theme of the music are highlighted

Piano Roll Representation



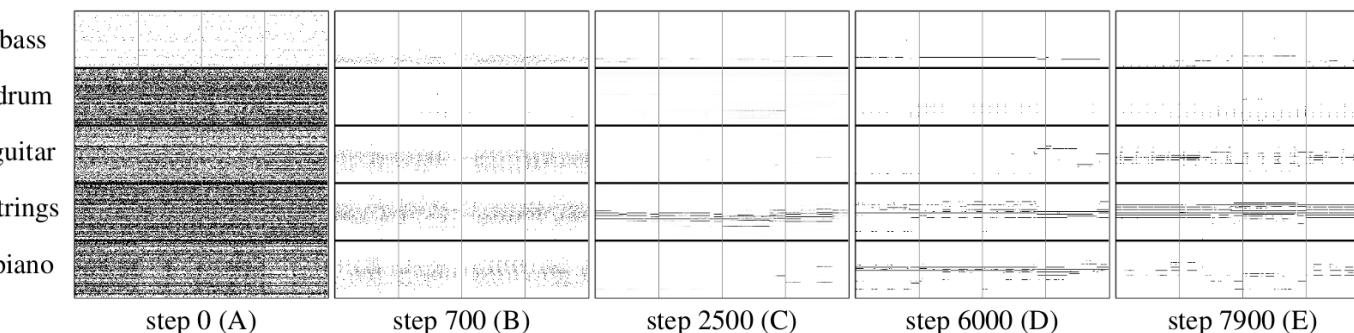
Source: <https://www.macprovideo.com/article/audio-software/logic-pro-x-a-guide-to-multitrack-midi-recording>

Piano Roll Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_PianoRoll.html

- A *visual, image-like* representation of *.MIDI
- **Widely** used by deep generative neural networks
 - *MidiNet* (2017) [GAN-based]
 - *MuseGAN* (2018) [GAN-based]: <https://salu133445.github.io/musegan/results>
 - *DiffRoll* (2023) [diffusion-based]: <https://polyffusion.github.io/>

MuseGAN

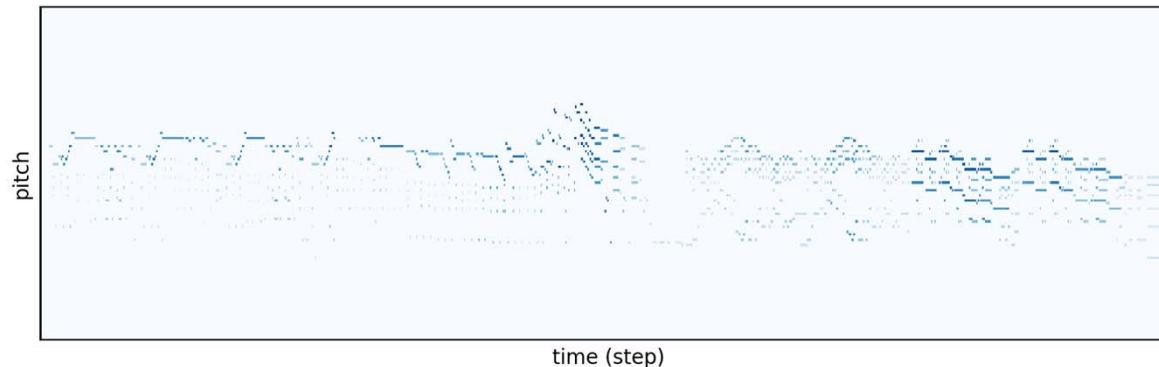


DiffRoll



Library: PyPianoroll

<https://salu133445.github.io/pypianoroll/>



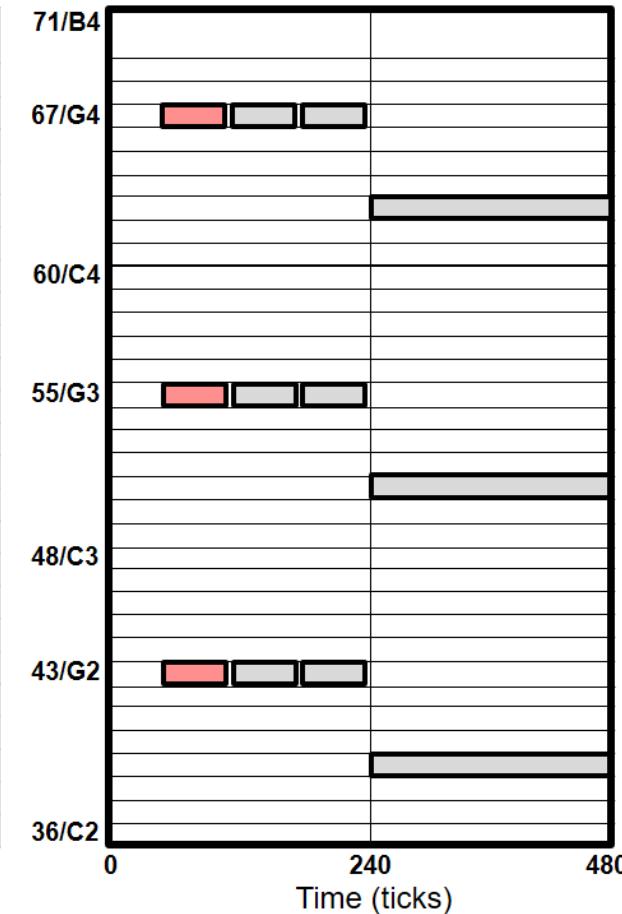
- Features
 - Manipulate multitrack piano rolls intuitively
 - Visualize multitrack piano rolls beautifully
 - Save and load multitrack piano rolls in a space-efficient format
 - Parse **MIDI** files into multitrack piano rolls
 - Write multitrack piano rolls into **MIDI** files

MIDI Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MIDI.html

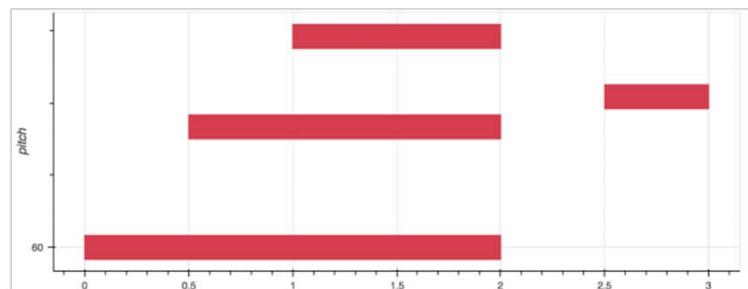
- A ***text-like*** representation of
*.MIDI, using **MIDI messages**
and **timestamps**
 - *MIDI note number* (0-127)
 - *Key velocity* (0-127): intensity
of the sound
 - *MIDI channel* (different
instruments)
 - *Time stamp*: how many *clock
pulses or ticks to wait* before
the command is executed

Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0



MIDI Representation

- A *text-like* representation of *.MIDI
- **Widely** used by deep generative neural networks, especially Transformers
 - By viewing the MIDI messages as “tokens”
 - *Music Transformer* (2019) : <https://magenta.tensorflow.org/music-transformer>
 - *MuseNet* (2019): <https://openai.com/research/musenet>
 - *Pop Music Transformer* (2020): <https://github.com/YatingMusic/remi>
 - *MuseCoco* (2023): <https://ai-muzic.github.io/musecoco/>



```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

Timing in Piano Roll & MIDI

- The time axis can either be in *absolute timing* or in *symbolic timing*
 - For **absolute** timing, the actual timing of note occurrence is used, for example by indicating the tempo for each bar or each beat
 - For **symbolic** timing, the **tempo** information is removed and thereby each beat has the same length
- MIDI subdivides a **quarter note** into basic time units referred to as *clock pulses* or *ticks*
 - Pulses per quarter note (PPQN): commonly 120
 - PPQN determines the resolution of the time stamps associated to note events

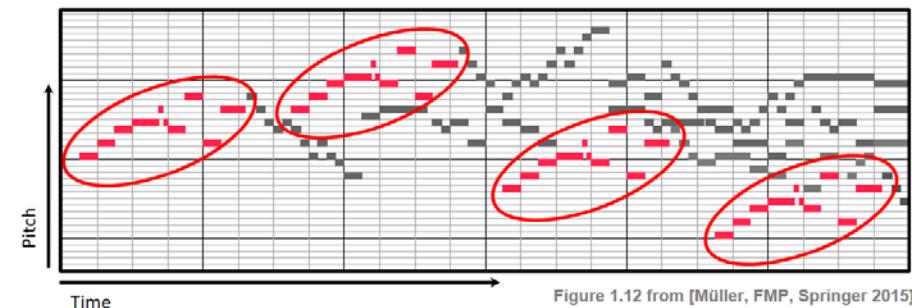
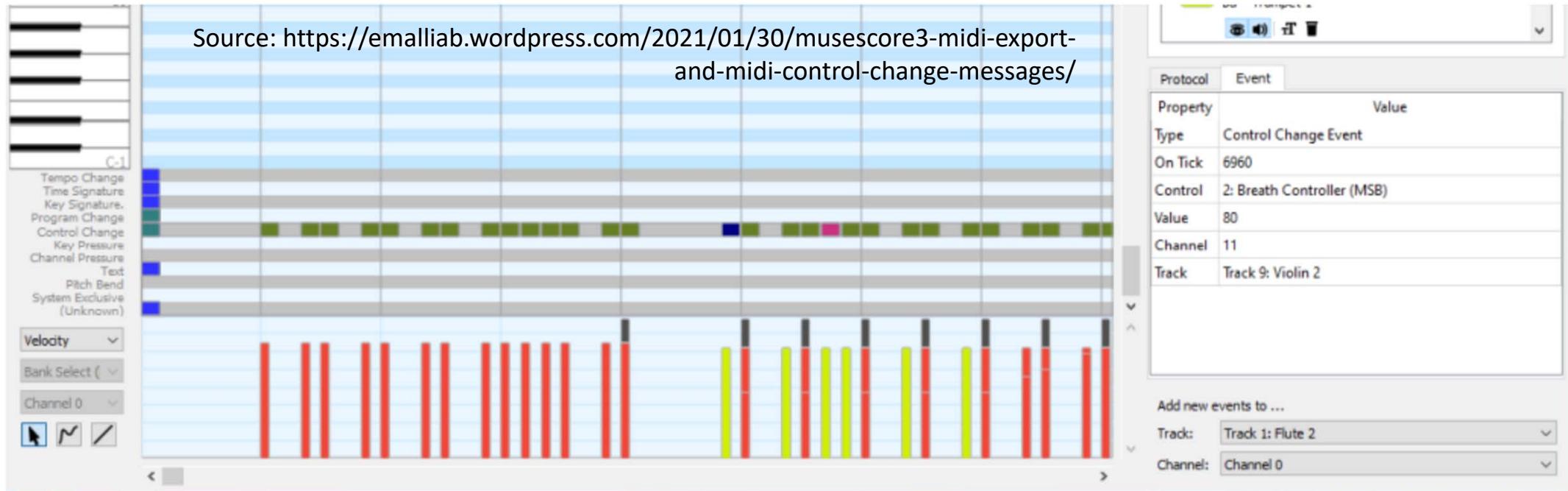


Figure 1.12 from [Müller, FMP, Springer 2015]

Ref1: <https://salu133445.github.io/lakh-pianoroll-dataset/representation.html>

Ref2: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MIDI.html

Program Change or Control Change (CC) Messages in MIDI

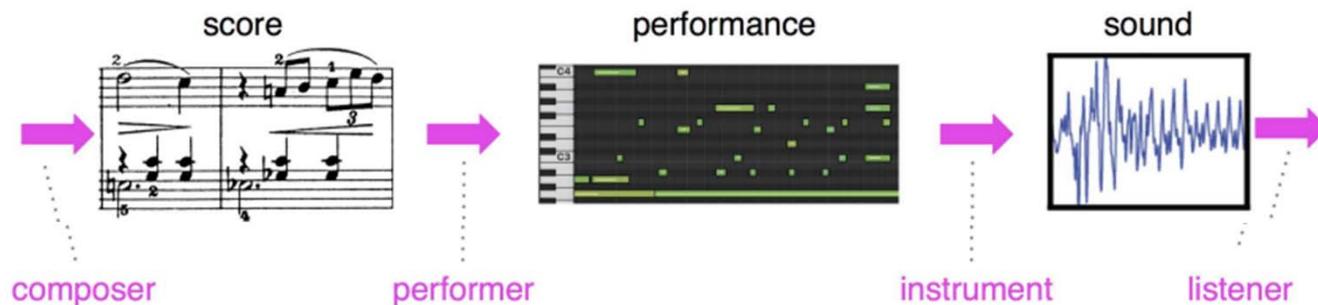


- Tempo change
- Program change
- Control change

Protocol	Event	Protocol	Event	Protocol	Event
Property	Value	Property	Value	Property	Value
Type	Program Change Event	Type	Program Change Event	Type	Program Change Event
On Tick	768	On Tick	1535	On Tick	3069
Program	1: Bright Acoustic Piano	Program	2: Electric Grand Piano	Program	3: Honky-tonk Piano
Channel	0	Channel	0	Channel	0
Track	Track 0: Asturias(Leyenda)	Track	Track 0: Asturias(Leyenda)	Track	Track 0: Asturias(Leyenda)

Source: <https://gigperformer.com/how-to-automate-switching-rackspace-variations-and-song-parts-using-the-midi-file-player/>

MIDI Score vs MIDI Performance



- **MIDI score**
 - A score that has been rendered **directly** into a MIDI file
 - Rendered with **NO** dynamics and **NO** expressive timing (i.e., exactly according to the written metrical grid)
- **MIDI performance**
 - A score has been performed, and that performance has been encoded into a MIDI stream, through either **MIDI keyboards** or by **automatic music transcription**
 - With expressive timing, tempo changes, dynamics, and articulation

Library: Miditoolkit

<https://github.com/YatingMusic/miditoolkit>

- **Miditoolkit** is designed for handling MIDI in **symbolic timing** (ticks), which is the native format of MIDI timing
- **PrettyMIDI** (<https://github.com/craffel/pretty-midi>) can parse MIDI files and generate pianorolls in absolute timing
- **PyPianoroll** can parse MIDI files into pianorolls in symbolic timing
- **mido** (<https://github.com/mido/mido>) processes MIDI files in the lower level such as messages and ports
- **music21** (<https://web.mit.edu/music21/>) provides analysis modules

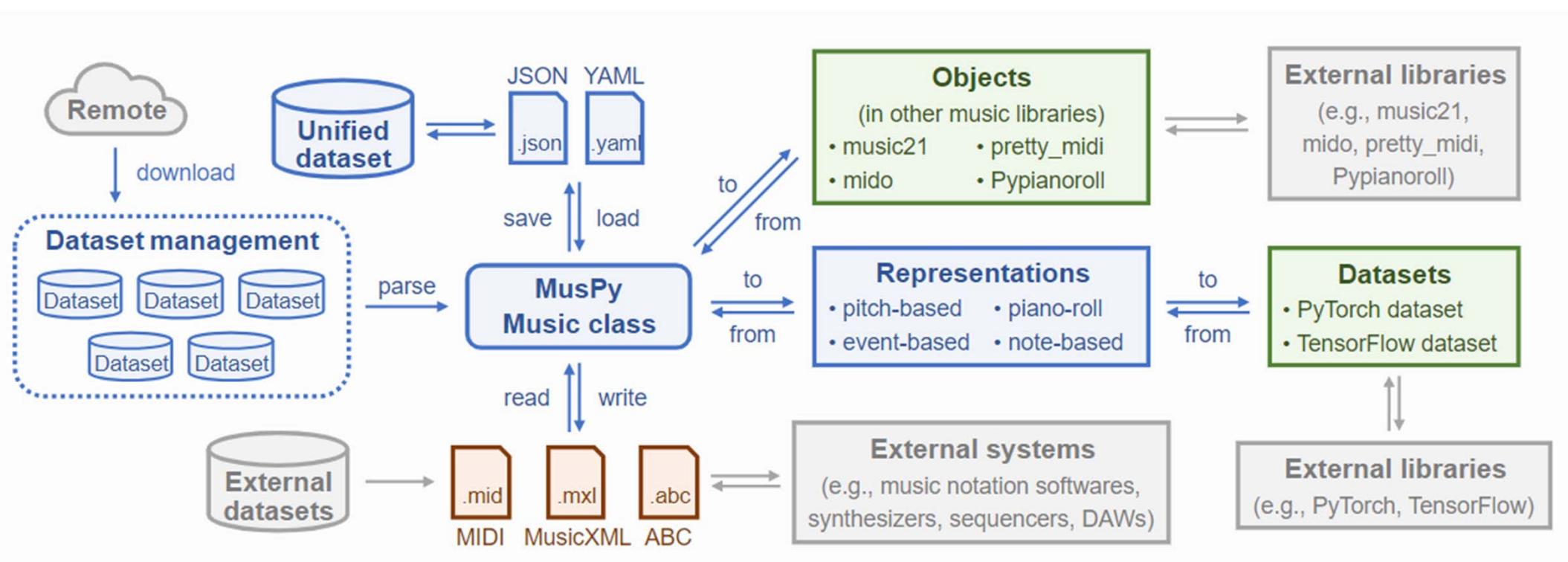
Library: MidiTok

<https://github.com/Natooz/MidiTok>

- “MidiTok can take care of converting (tokenizing) your MIDI files into **tokens**, ready to be fed to models such as **Transformer**, for any generation, transcription or MIR task”
 - “MidiTok features most known MIDI tokenizations (e.g. REMI, Compound Word...), and is built around the idea that they all share common parameters and methods”
 - “It supports Byte Pair Encoding (BPE) and data augmentation.”

Library: MusPy

<https://salu133445.github.io/muspy/index.html>

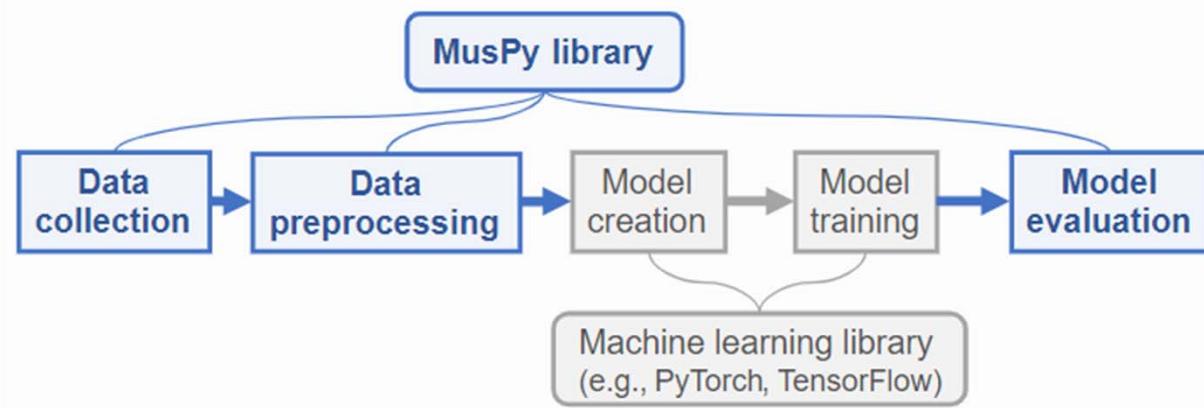


Library: MusPy

<https://salu133445.github.io/muspy/index.html>

- **Features**

- Dataset management system for commonly used **datasets** with interfaces to PyTorch and TensorFlow
- Data **I/O** for common symbolic music formats and interfaces to other symbolic music libraries
- Implementations of common **representations** for music generation, including the pitch-based, the event-based, the piano-roll and the note-based representations
- Model **evaluation tools** for music generation systems, including audio rendering, score and piano-roll visualizations and objective metrics



MusicXML

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MusicXML.html

- A ***text-like*** representation of **sheet music**
 - To represent a <note>
 - <pitch>: <step>, <alter> <octave>
 - <duration>
 - <type>
- Can be **tokenized**
- Musical (not physical) onset times are specified
- Avoid information loss of the piano roll representation
 - Example 1, the notes **D♯4** and **E♭4** are distinguishable in MusicXML but not in MIDI

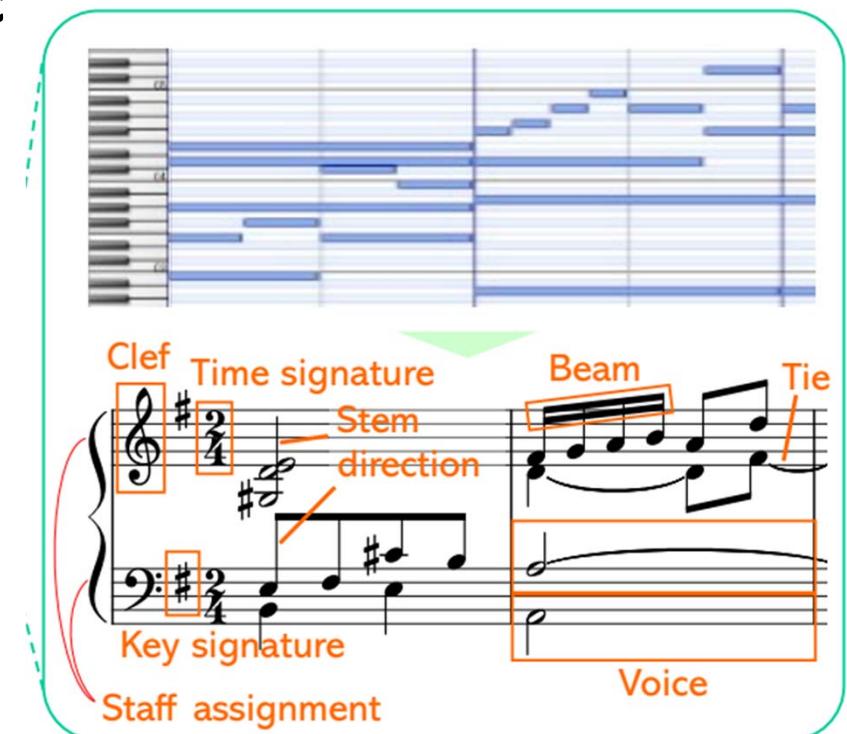
```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <type>half</type>
</note>
```



MusicXML

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MusicXML.html

- A ***text-like*** representation of **sheet music**
- Can be tokenized
- Musical (not physical) onset times
- Avoid information loss of the piano roll representation
 - Example 1, D♯4 and E♭4
 - Example 2: can describe **voices**



Ref: Suzuki, “Score Transformer: Generating musical score from note-level representation,” MMAAsia 2021

MusicXML

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MusicXML.html

- A *text-like* representation of **sheet music**
- *Relatively less* used by deep generative neural networks
 - *Score Transformer* (2022)



(a) Score

R bar clef_treble key_sharp_1 time_2/4 note_E4
note_D4 note_G#3 len_2 stem_up bar ... L bar
clef_bass key_sharp_1 time 2/4 <voice> note_E3
len_1/2 stem_up beam_start note_F#3 len_1/2
stem_up beam_continue note_C#4 stem_up
beam_continue note_B3 len_1/2 stem_up beam_stop
</voice> <voice> note_B2 len_1 stem_down note_E3
len_1 stem_down </voice> bar ...

(b) Notation-level Representation

Ref: Suzuki, "Score Transformer: Generating musical score from note-level representation," MMAAsia 2021

Software: MuseScore

The screenshot shows the MuseScore application interface. A modal window titled "Download this score" is open, listing five download options: Musescore, PDF, MusicXML, MIDI, and Audio. The score itself is for "Amazing Grace" by matt1738, with a key signature of F# and a tempo of 72 BPM. The score includes two staves of musical notation.

Search for Sheet music

Browse Learn Start Free Trial

Musescore

Open in Musescore

PDF

View and print

MusicXML

Open in various software

MIDI

Open in editors and sequencers

Audio

Listen to this score

Amazing Grace

matt1738

35K 2.3K 39 ★

Download Print

Favorite Share

Please rate this score

Try Shutter FLEX for f

Software: KernScores

Kern Scores

A library of virtual musical scores in the Humdrum **kern data format.
Total holdings: 7,866,496 notes in 108,703 files.

search: [browse](#) | [shortcuts](#) [Text](#) anchored

A guided tour of the KernScores website
Recent additions to the KernScores library
Data Collection Highlights

Online Humdrum Editor
CCARH Humdrum Portal
Contribute kern scores

Composers				
Adam	Chopin	Giovannelli	Lassus	Schubert
Alkan	Clementi	Grieg	Liszt	Schumann
J.S. Bach	Corelli	Haydn	MacDowell	Scriabin
Banchieri	Dufay	Himmel	Mendelssohn	Sinding
Beethoven	Dunstable	Hummel	Monteverdi	Sousa
Billings	Field	Isaac	Mozart	Turpin
Bossi	Flecha	Ives	Pachelbel	Scarlatti
Brahms	Foster	Joplin	Prokofiev	Vecchi
Buxtehude	Frescobaldi	Josquin	Ravel	Victoria
Byrd	Gershwin	Landini	Scarlatti	Vivaldi
				Weber

Genres				
Ballate	Etudes	Motets	Scherzos	Symphonies
Ballads	Fugues	Preludes	Sonatas	Virelais
Chorales	Madrigals	Ragtime	Sonatina	Waltzes
Contrafacta	Mazurkas	Quartets		

VerovioHumdrumViewer Scarlatti, Sonata in C minor, L.10, K.84

File View Edit Analysis Scores Help A z

```

1 !!!COM: Scarlatti, Domenico
2 !!!CDT: 1685-1757
3 !!!OTL: Sonata in C minor, L.10, K.84
4 !!!SCT: K. 84
5 !!!SCT: L. 10
6 !!!OMD: Allegro ([quarter]=152)
7 **kern **kern **dynam
8 *staff2 *staff1 *staff1/2
9 *Icemb a *Icemb a *
10 *I" L.10 (K.84) *I" L.10 (K.84)
11 *>[A,A1,A,A2,B,B1,B,B2] *>[A,A1,A,A2,B,
12 *>norep[A,A2,B,B2] *>norep[A,A2,B,
13 *>A *>A *>A
14 *clefF4 *clefG2 *
15 *k[b-e-a-] *k[b-e-a-] *
16 *M3/4 *M3/4 *
17 *MM152 *MM152 *
18 =1- =1- =1-
19 4CC 4C 8r f
20 . 8c'L .
21 4r 8e' .
22 . 8g' .
23 4r 8cc' .
24 . 8ee'-J .
25 =2 =2 =2
26 2.r (8gg^L .
27 . 8ccc) .
28 . 8gg' .
29 . 8ee'- .
30 . 8cc' .
31 . 8g'J .
32 =3 =3 =3
33 8r 4c' .
34 8c'l

```

Summary

- **Sheet Music / MusicXML**
 - Support **score** only, not performance
 - Use **music timing** only, not absolute timing
 - Richer information about the score itself
- **Piano roll / MIDI**
 - Support either score or **performance**
 - Use either music timing (after quantization) or **absolute timing**
 - Rich performance information
 - Datasets more easily accessible and thereby **more widely used**

Outline

- Sheet music & symbolic representations for music
- **Audio representation for music**
- Math in STFT: frequency and temporal resolution

Audio Waveforms

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Waveform.html

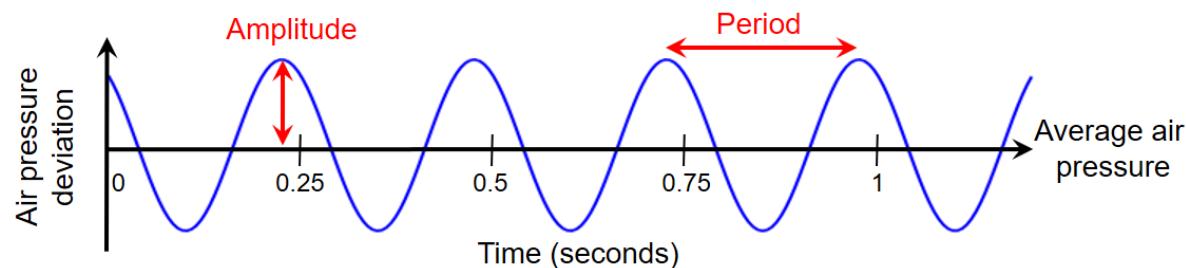
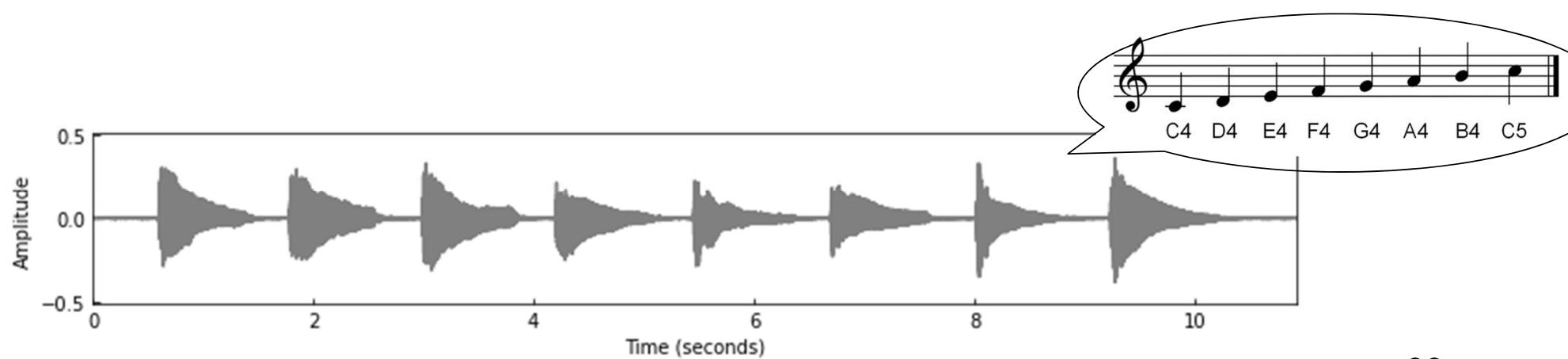
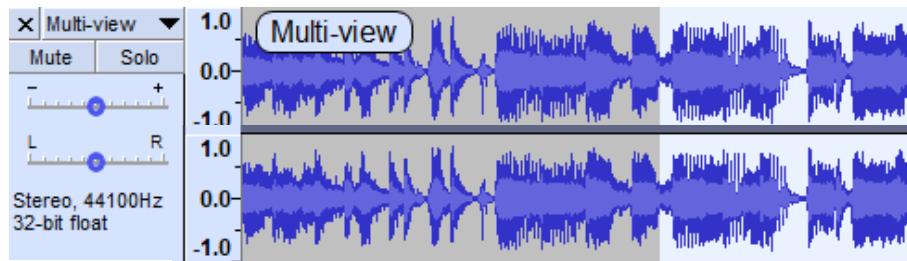


Figure 1.19 from [Müller, FMP, Springer 2015]



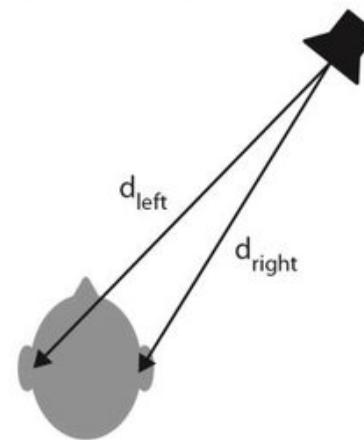
Mono vs Stereo

- Monaural
- Binaural

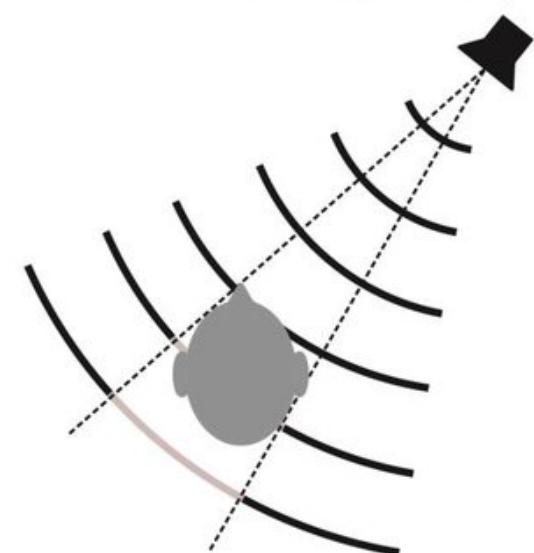


- Stereo-to-mono conversion
 - By taking the average
- Mono-to-stereo conversion
 - Need research
https://www.youtube.com/watch?v=aWxmQKm_s8Q

a) Binaural localization cue:
interaural time difference (ITD)



b) Binaural localization cue:
interaural intensity difference (IID)



Source: https://www.researchgate.net/figure/Binaural-and-monaural-cues-used-for-sound-localization-a-For-sound-sources-off-the_fig1_299281975

Notes and Pitches

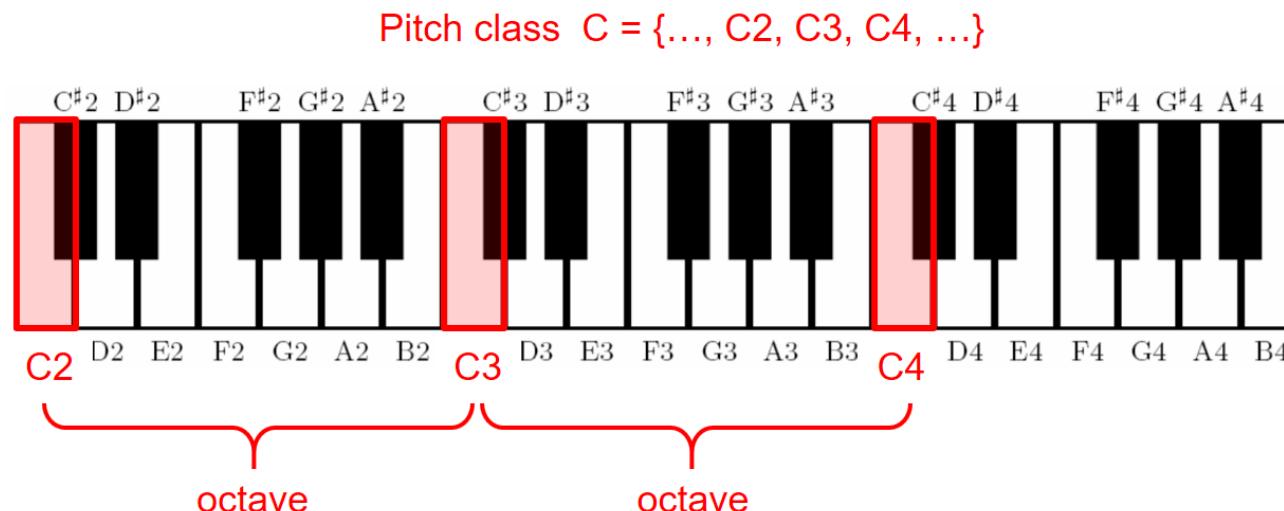
<https://newt.phys.unsw.edu.au/jw/notes.html>

Notes and Pitches

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_MusicalNotesPitches.html

- **Pitch Class**

- Two notes with *fundamental frequencies* in a ratio equal to *any power of two* (e.g., half, twice, or four times) are perceived as very **similar**
- All notes with this kind of relation can be grouped under the same *pitch class*



MIDI Note Numbers

https://en.wikipedia.org/wiki/MIDI_tuning_standard

$$f = 2^{(d-69)/12} \cdot 440 \text{ Hz} \quad d = 69 + 12 \log_2 \left(\frac{f}{440 \text{ Hz}} \right)$$

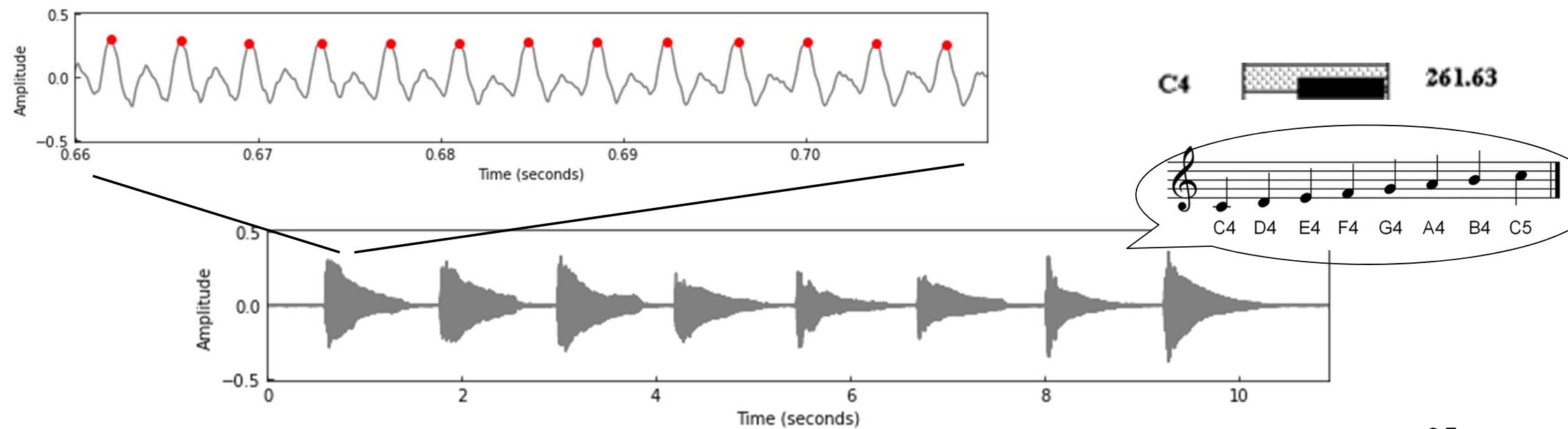
MIDI number	Note name	Keyboard	Frequency Hz						
21	A0		27.500						
23	B0		30.868	29.135					
24	C1		32.703						
26	D1		36.708	34.648					
28	E1		41.203	38.891					
29	F1		43.654						
31	G1		48.999	46.249					
33	A1		55.000	51.913					
35	B1		61.735	58.270					
36	C2		65.406						
38	D2		73.416	69.296					
40	E2		82.407	77.782					
41	F2		87.307						
43	G2		97.999	92.499					
45	A2		110.00	103.83					
			123.47	116.54					

Audio Waveforms (Cont')

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Waveform.html

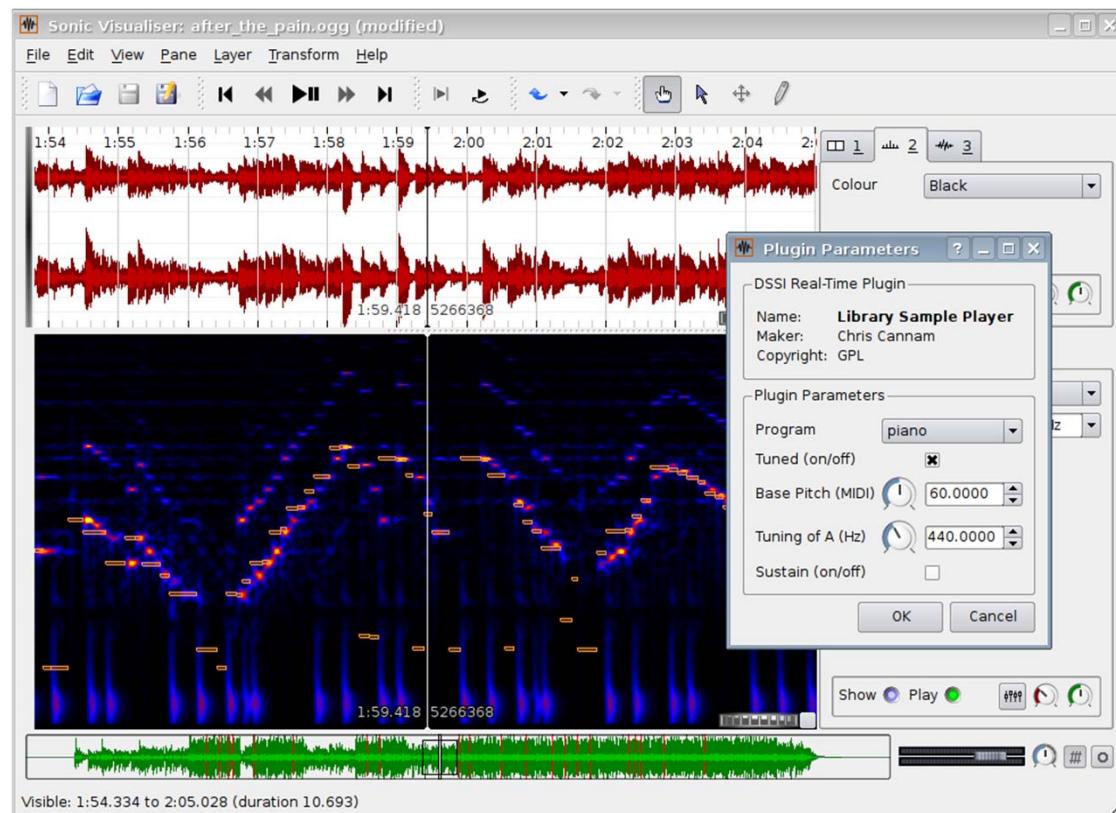
Zoom-in of the section between 0.66 and 0.71 seconds

- Highly repetitive
- 13 high-pressure (red) points $\rightarrow 20 * 13 = 260$ Hz



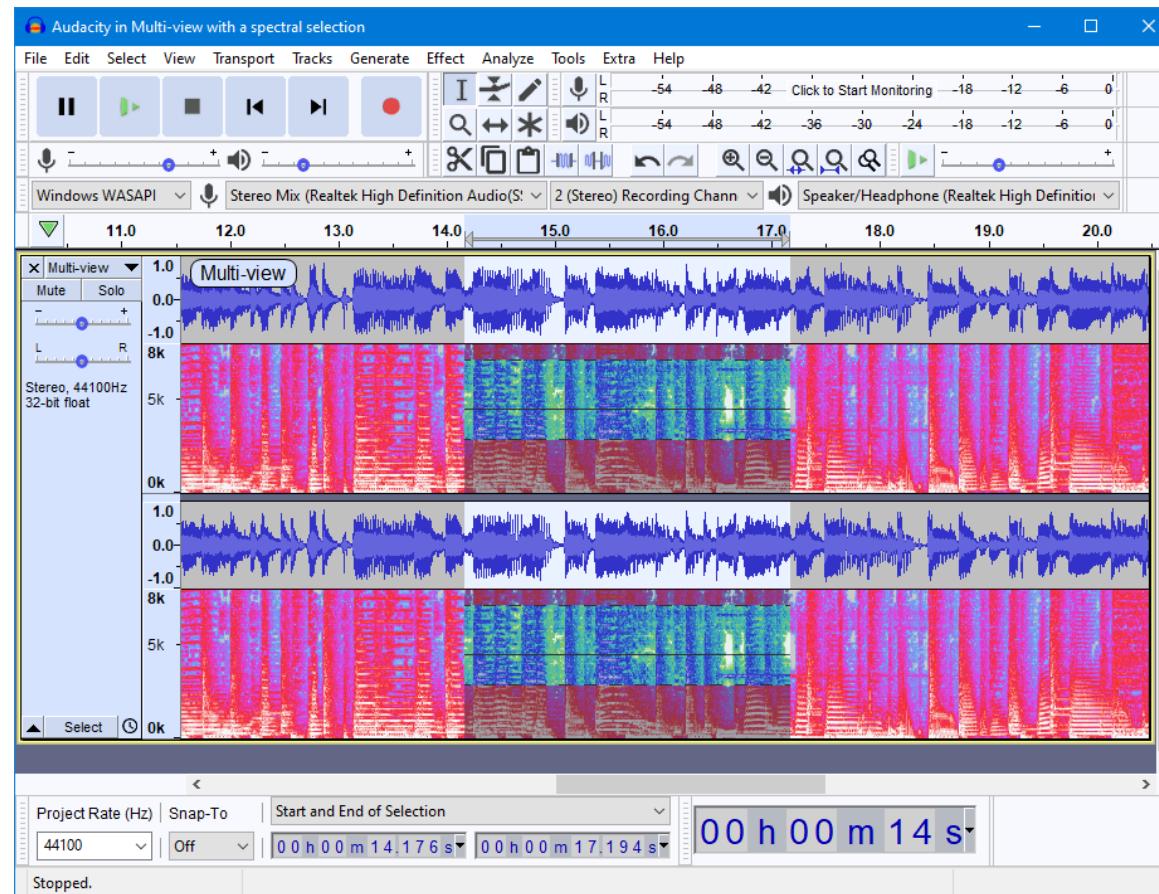
Software: Sonic Visualiser

<http://www.sonicvisualiser.org/>



Software: Audacity

<https://www.audacityteam.org/>

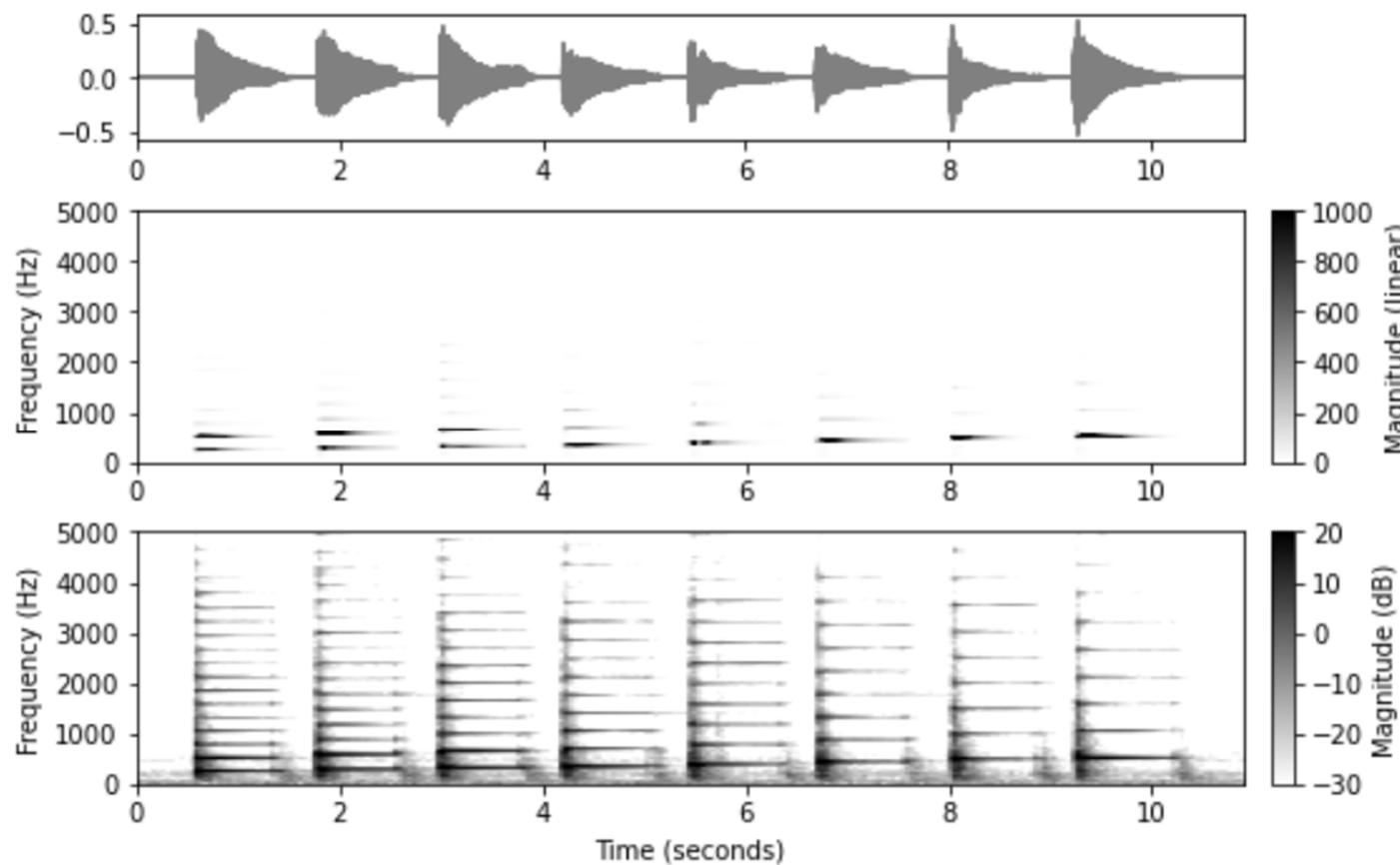


Let's Find Some Audio Recordings

- <https://www.freesound.org/>
 - <https://www.freesound.org/people/acclivity/sounds/22347/>
 - https://www.freesound.org/people/Rudmer_Rotteveel/sounds/316915/
 - <https://www.freesound.org/people/Jaylew1987/sounds/321112/>
 - <https://www.freesound.org/people/mickel11/sounds/90803/>

Discrete Short-Time Fourier Transform

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2_STFT-Basic.html

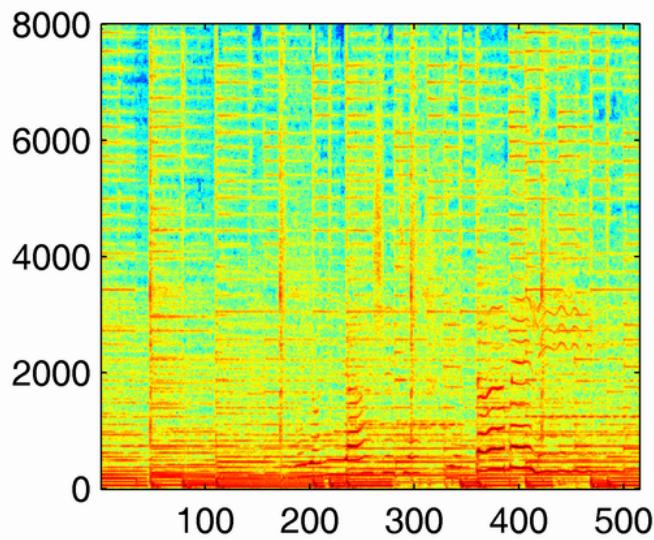


- **Time-frequency representation of audio**
 - Linear frequency, linear magnitude
 - Linear frequency, logarithmic magnitude (dB)

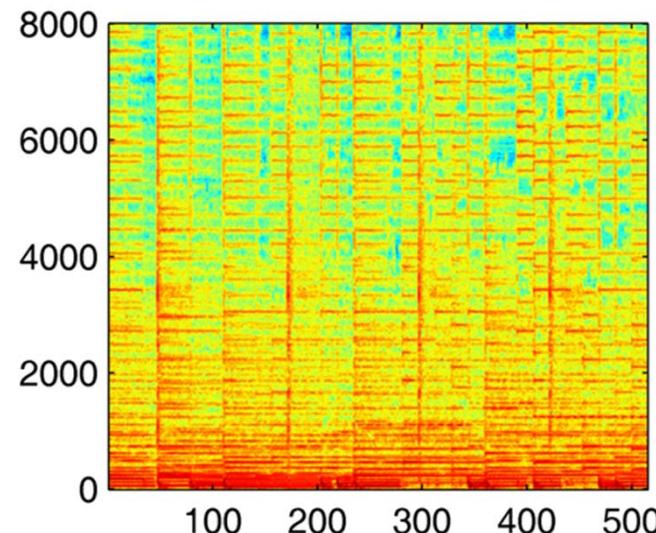
Discrete Short-Time Fourier Transform

- Time-frequency representation of audio

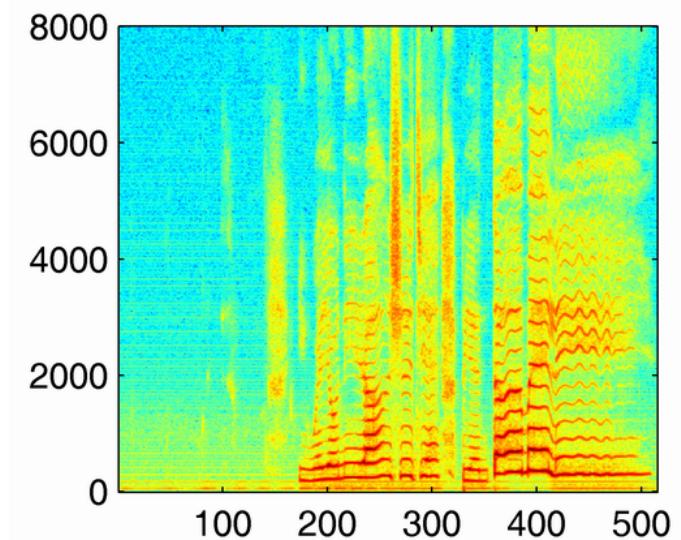
mixture



instrumental (clean)



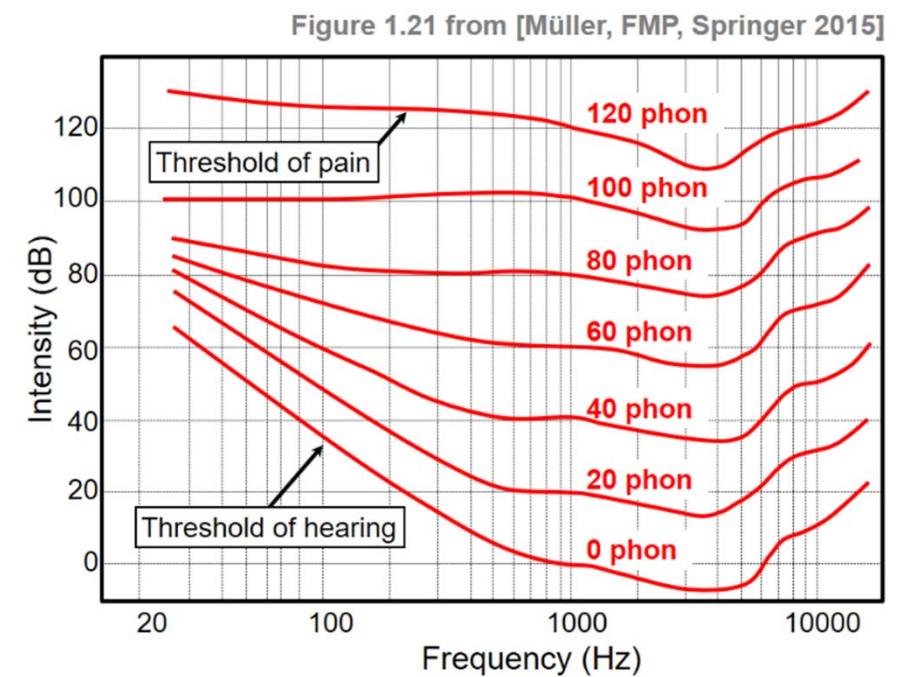
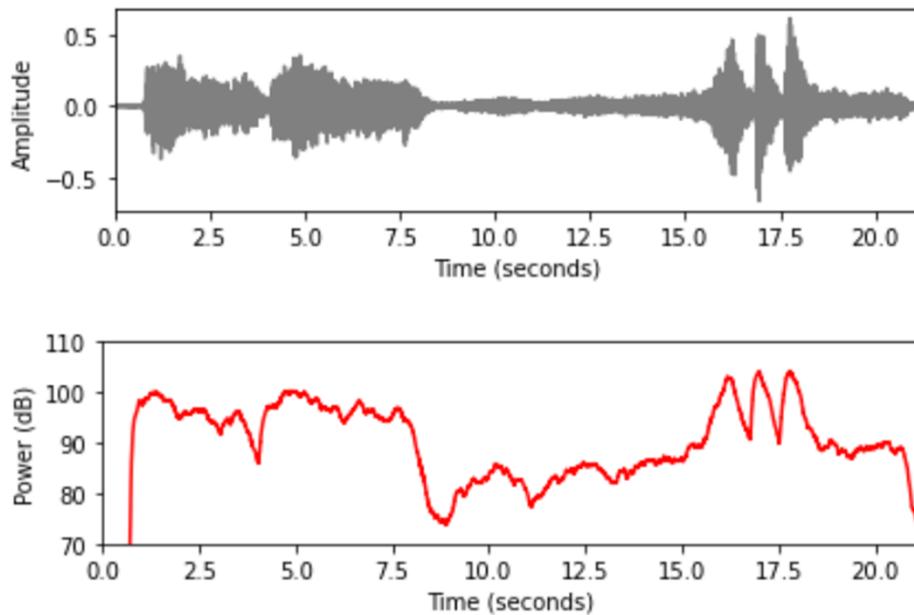
vocal (clean)



Dynamics, Intensity, and Loudness

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Dynamics.html

- **Intensity:** a physical property
 - Other names: power (dB); energy
- **Loudness:** a perceptual property
 - Less used in deep learning research



Library: LibROSA

<https://librosa.org/doc/latest/index.html>

https://colab.research.google.com/github/stevetjoa/musicinformationretrieval.com/blob/gh-pages/ipython_audio.ipynb

```
[ ] import librosa  
x, sr = librosa.load('audio/simple_loop.wav')  
  
[ ] X = librosa.stft(x)  
Xdb = librosa.amplitude_to_db(abs(X))  
plt.figure(figsize=(14, 5))  
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
```

Library: torchaudio

https://pytorch.org/audio/0.11.0/tutorials/audio_feature_extractions_tutorial.html

```
waveform, sample_rate = get_speech_sample()

n_fft = 1024
win_length = None
hop_length = 512

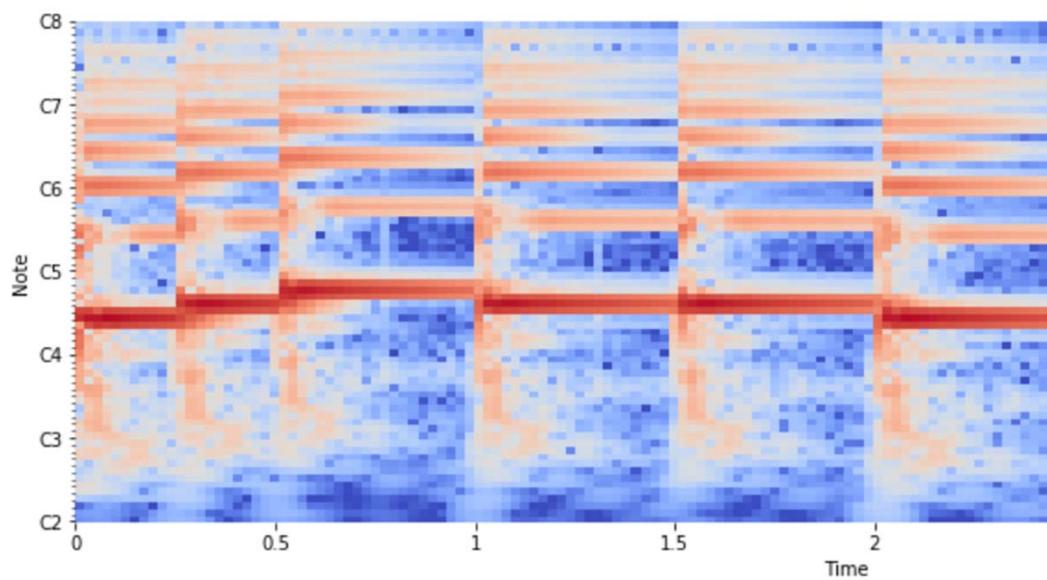
# define transformation
spectrogram = T.Spectrogram(
    n_fft=n_fft,
    win_length=win_length,
    hop_length=hop_length,
    center=True,
    pad_mode="reflect",
    power=2.0,
)
# Perform transformation
spec = spectrogram(waveform)

print_stats(spec)
plot_spectrogram(spec[0], title="torchaudio")
```

Constant-Q Transform (CQT)

<https://musicinformationretrieval.com/chroma.html>

- STFT (*linearly*-spaced frequencies)
- CQT (*logarithmically*-spaced, closer to human auditory perception)

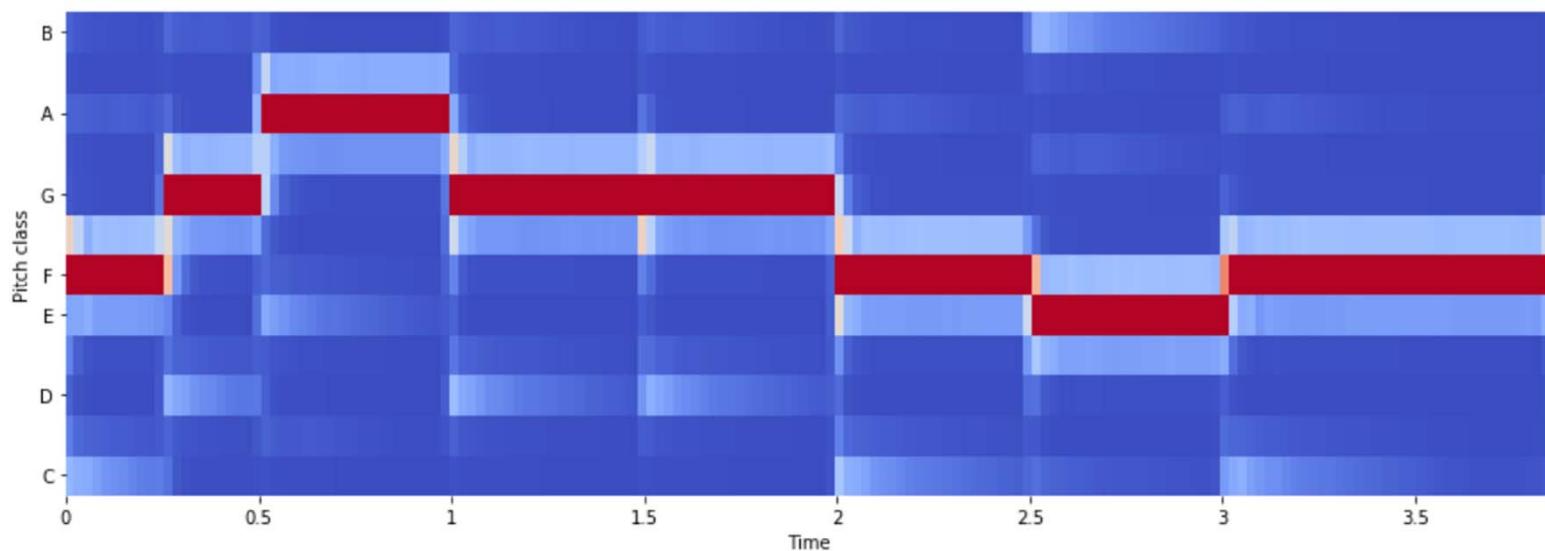


- Logarithmic frequency, logarithmic magnitude

Pitch Class Profile / Chromagram

<https://musicinformationretrieval.com/chroma.html>

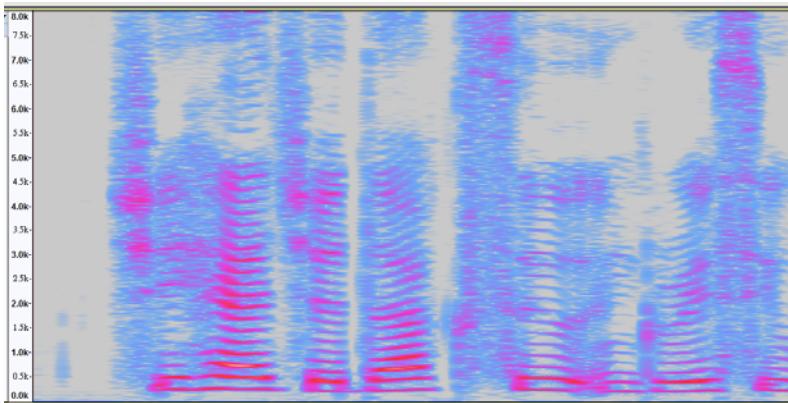
- “A **chroma vector** ([Wikipedia](#)) is typically a 12-element feature vector indicating how much energy of each pitch class, {C, C#, D, D#, E, ..., B}, is present in the signal”
 - i.e., ignore octaves



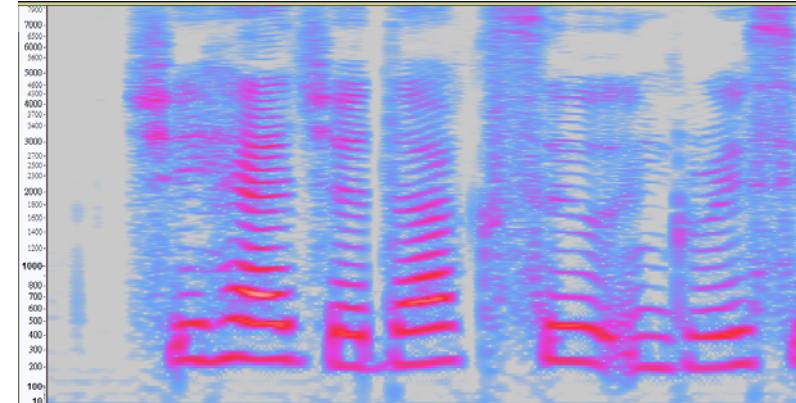
Mel-Spectrogram

- The mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another
- **Finer resolution in the low-frequency range** (NOT exactly logarithmic scale)
- Dimension reduction

linear scale



mel scale



Feature Extraction

- **Timbre representation:** Spectrogram → mel-spectrogram → MFCC
- **Harmonic representation:** Spectrogram → CQT → chroma feature

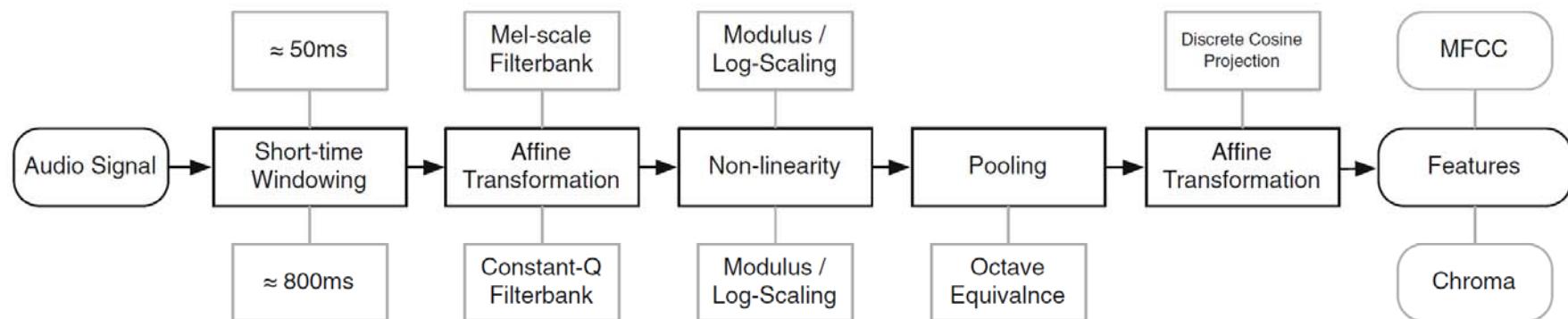


Fig. 3 *State of the art:* standard approaches to feature extraction proceed as the cascaded combination of a few simpler operations; on closer inspection, the main difference between chroma and MFCCs is the parameters used

Feature Learning by Convolutional Layers

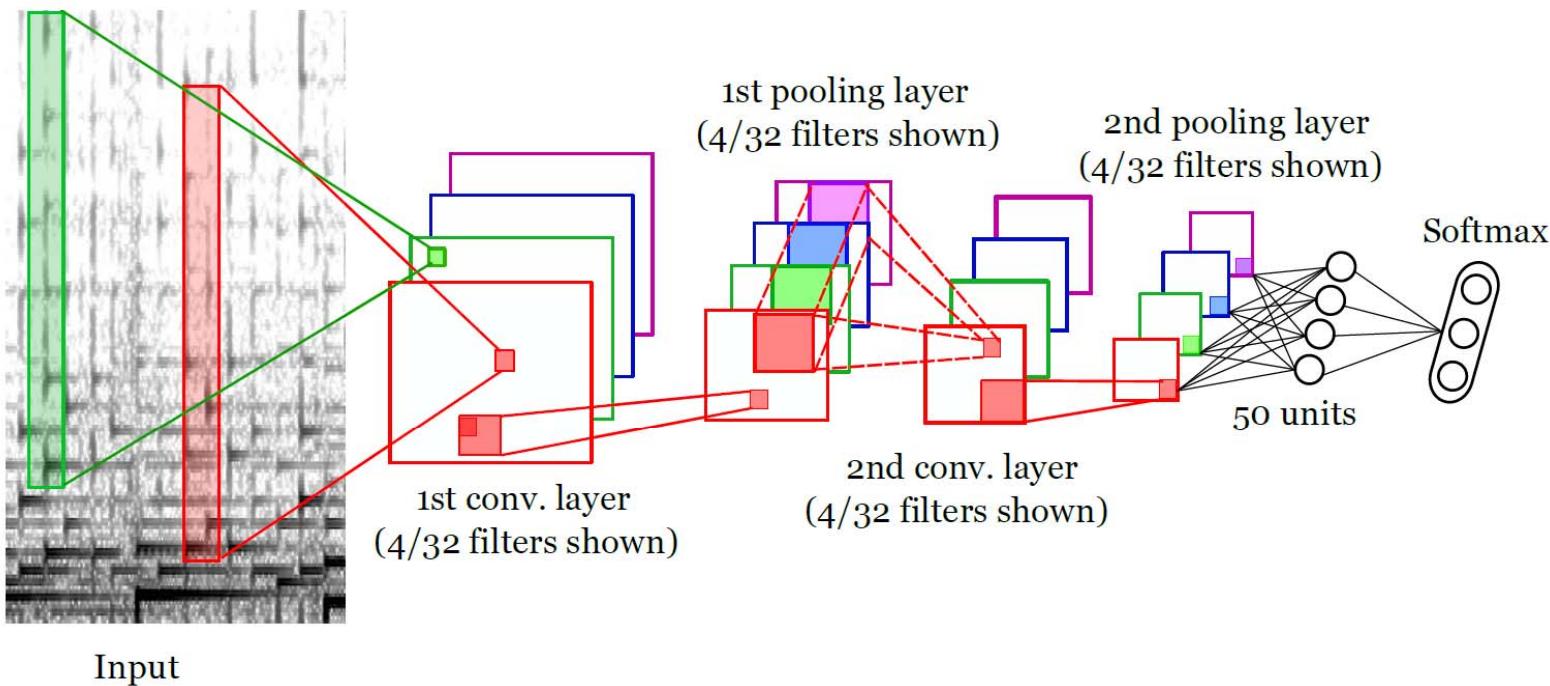
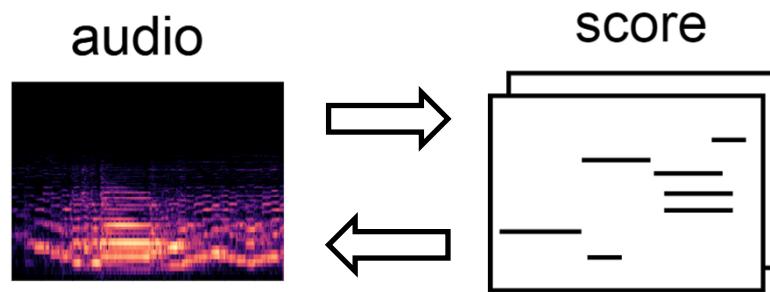


Fig. 1. Illustration of the CDNN architecture we use for our experiments. The CDNN first applies narrow vertical filters to the input sonogram (left) to capture harmonic structure. Then, it applies 32 different filters in the first convolutional layer (we show only 4). This is followed by the first max-pooling layer, and then a 2nd pair of convolutional and max-pooling layers. Finally, the output of the final max-pooling layer is fully connected to a final hidden layer of 50 units, followed by a softmax output unit. The input spectrogram contains 100 time slices, which means that the final layer of the CDNN summarises information over a total duration of 2.35 seconds.

Summary

- Audio representation
 - Waveforms
 - Spectrograms

Music AI Research



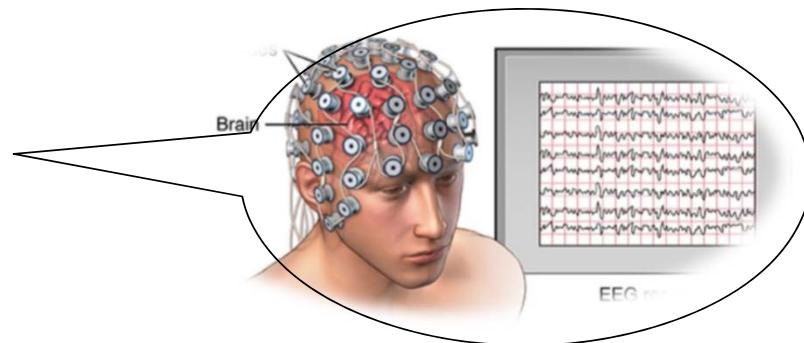
- Topics
 - **audio → audio:** signal processing
 - **audio → score:** transcription (similar to automatic speech recognition)
 - **score → score:** composition
 - **score → audio:** synthesis (similar to text-to-speech)
 - **audio → knowledge:** audio analysis
 - **score → knowledge:** symbolic-domain analysis

Outline

- Sheet music & symbolic representations for music
- Audio representation for music
- **Math in STFT: frequency and temporal resolution**

Sampling Rate

- Definition: number of samples per second
- Why: analog to digital
- Examples
 - EEG signal: **128 Hz**
 - Telephone audio: **8k Hz**
 - Music audio: **44k Hz**

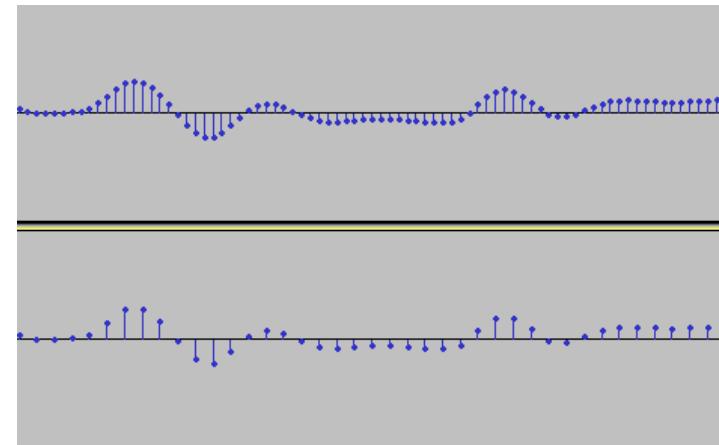


<https://www.brightbraincentre.co.uk/electroencephalogram-eeg-brainwaves/>

- MATLAB code
 - `[a,sr] = wavread('...')` % sr = sampling rate
 - `length(a)` % length of the signal in 'number of samples'
 - `length(a)/sr` % length of the signal in 'seconds'

Sampling Rate (Cont')

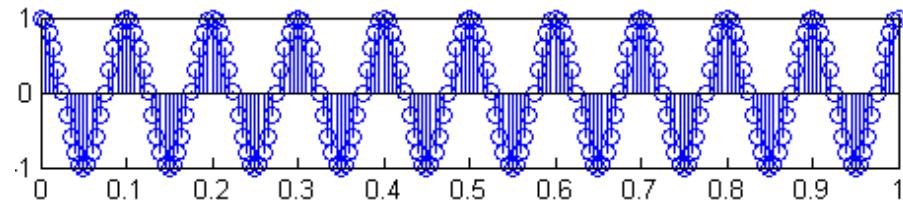
- MATLAB code
 - `a2 = downsample(a,2);`
 - `sr2 = sr/2;`
 - `length(a2)` % length of the signal in ‘number of samples’
 - `length(a2)/sr2` % length of the signal in ‘seconds’
 - `wavwrite(a2,sr2,’test.wav’)`



Sinusoids

- MATLAB code

- $sr = 200;$
 - $t = 0:1/sr:1;$
 - $f_0 = 10;$ % frequency
 - $a = 1;$ % amplitude
 - $y = a * \sin(2 * \pi * f_0 * t + \pi/2);$
 - $\text{stem}(t, y)$

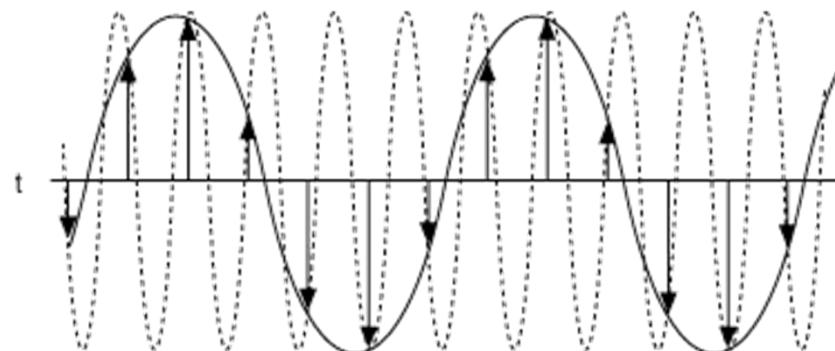


- Why?

- $\sin(2 * \pi * f_0 * t + \pi/2) = 1$, when $t = 1/f_0, 2/f_0, 3/f_0, 4/f_0, \dots$
 - frequency = inverse of the period

Nyquist–Shannon Sampling Theorem

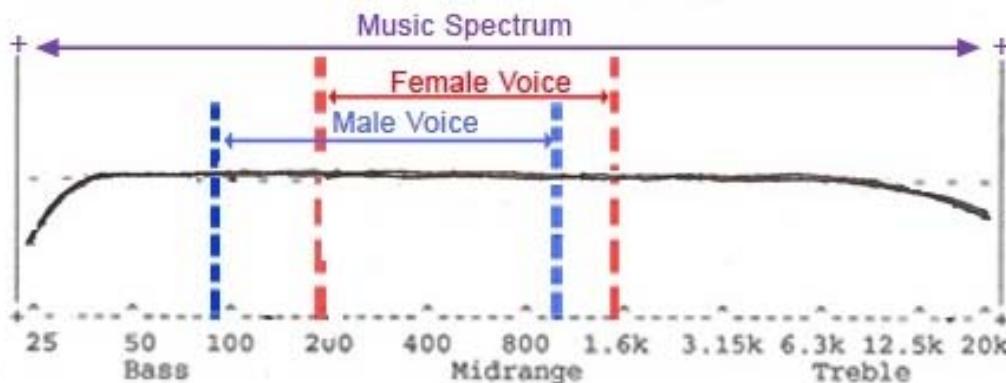
- A signal must be sampled at least **twice** as fast as the bandwidth of the signal to accurately reconstruct the waveform; otherwise, the high-frequency content will alias at a frequency inside the spectrum of interest
- **Sampling freq > 2* the highest freq in the signal**



http://zone.ni.com/reference/en-XX/help/370524T-01/siggenhelp/fund_nyquist_and_shannon_theorems/

Nyquist–Shannon Sampling Theorem

- Telephone audio: 8k Hz
 - Via phone, we cannot hear frequency higher than 4k Hz



<https://www.quora.com/How-do-HRT-sex-reassignment-and-other-such-procedures-affect-vocal-production-particularly-the-singing-voice>

- **Question:** With $\text{sr}=128 \text{ Hz}$, we assume that we don't need to care freq higher than __ Hz in brain waves

Nyquist–Shannon Sampling Theorem

Brainwaves, Frequencies and Functions

Unconscious		Conscious		
Delta	Theta	Alpha	Beta	Gamma
0,5 – 4 Hz	4 – 8 Hz	8 – 13 Hz	13 – 30 Hz	30-42 Hz
Instinct	Emotion	Consciousness	Thought	Will
Survival Deep sleep Coma	Drives Feelings Trance Dreams	Awareness of the body Integration of feelings	Perception Concentration Mental activity	Extreme focus Energy Ecstasy

<http://altered-states.net/barry/update236/>

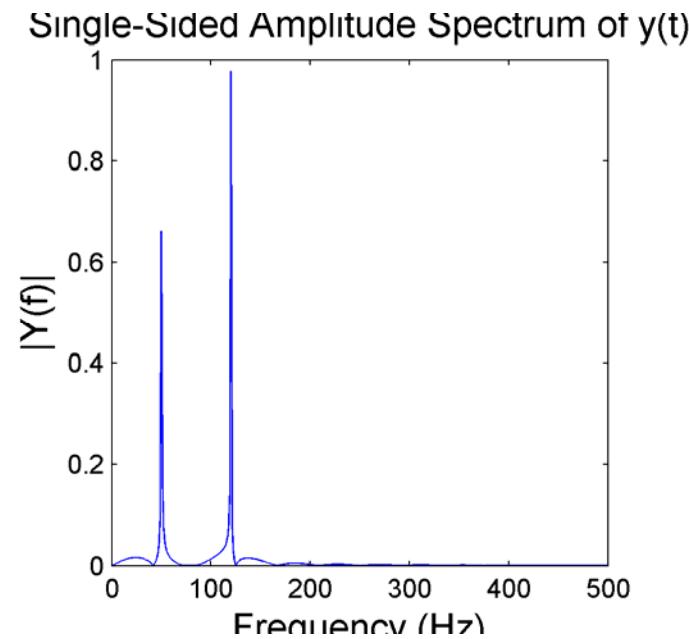
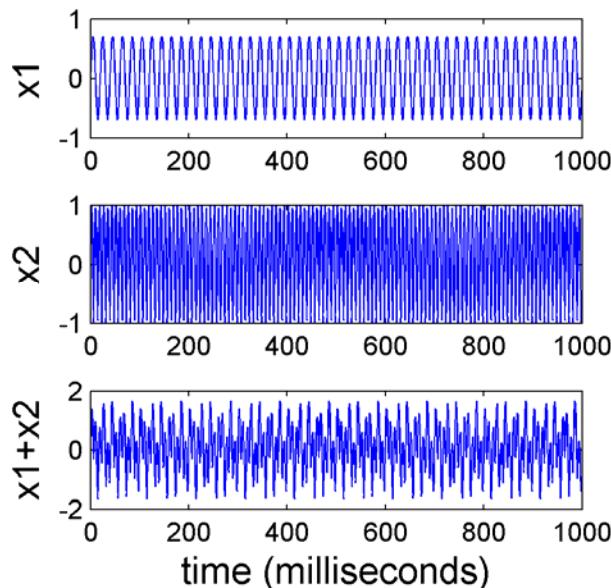
- **Question:** With $sr=128$ Hz, we assume that we don't need to care freq higher than 64 Hz in brain waves

Fourier Transform

- To get the spectrum of a signal
- MATLAB code
 - <https://www.mathworks.com/help/matlab/ref/fft.html>
 - doc fft
 - $Y = \text{abs}(\text{fft}(y));$

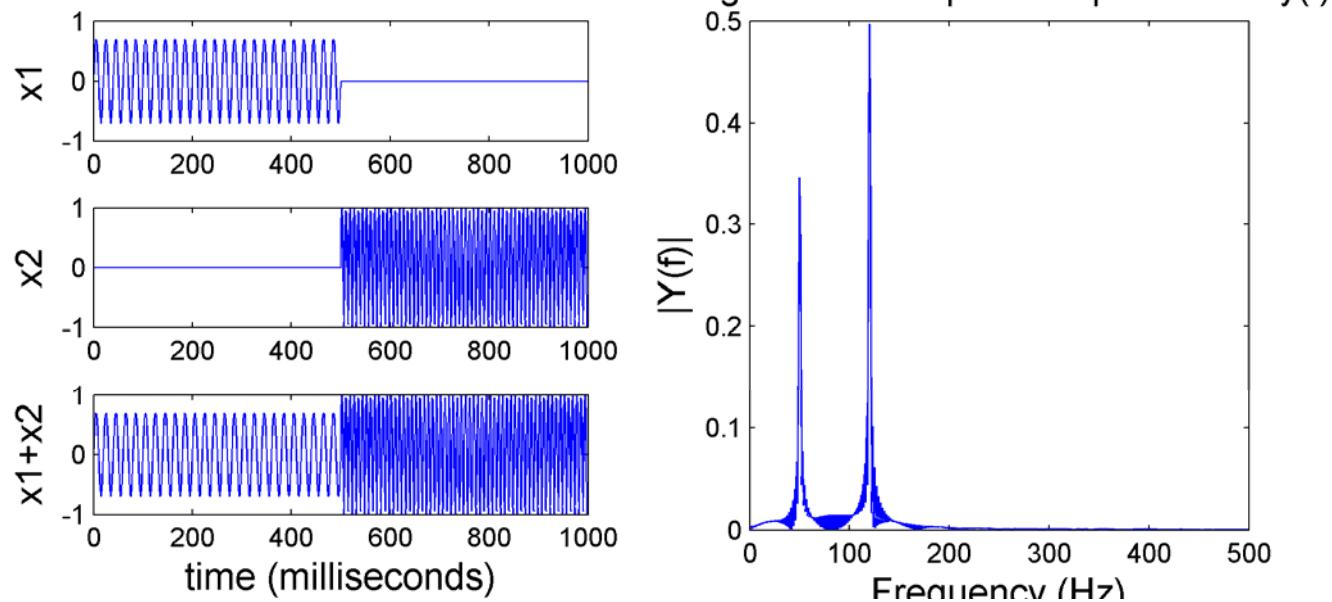
Fourier Transform

- MATLAB code
 - $x1 = 0.7 * \sin(2 * \pi * 50 * t);$
 - $x2 = \sin(2 * \pi * 120 * t);$



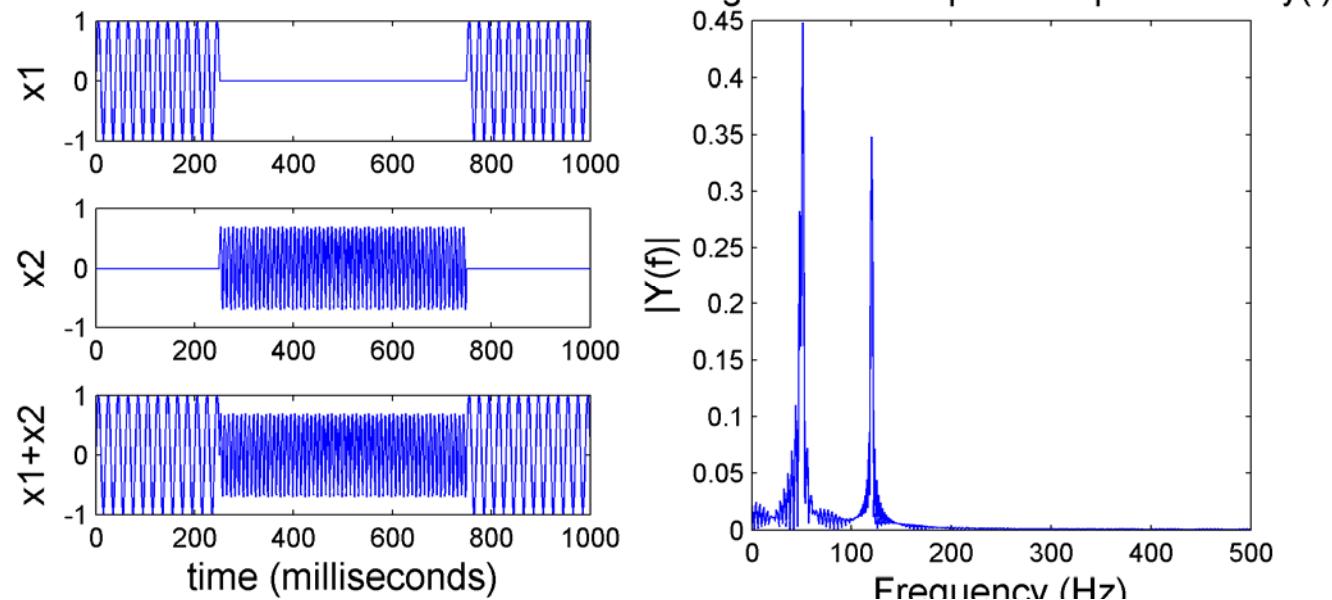
Fourier Transform

- Problem: cannot “localize” signal of interest



Fourier Transform

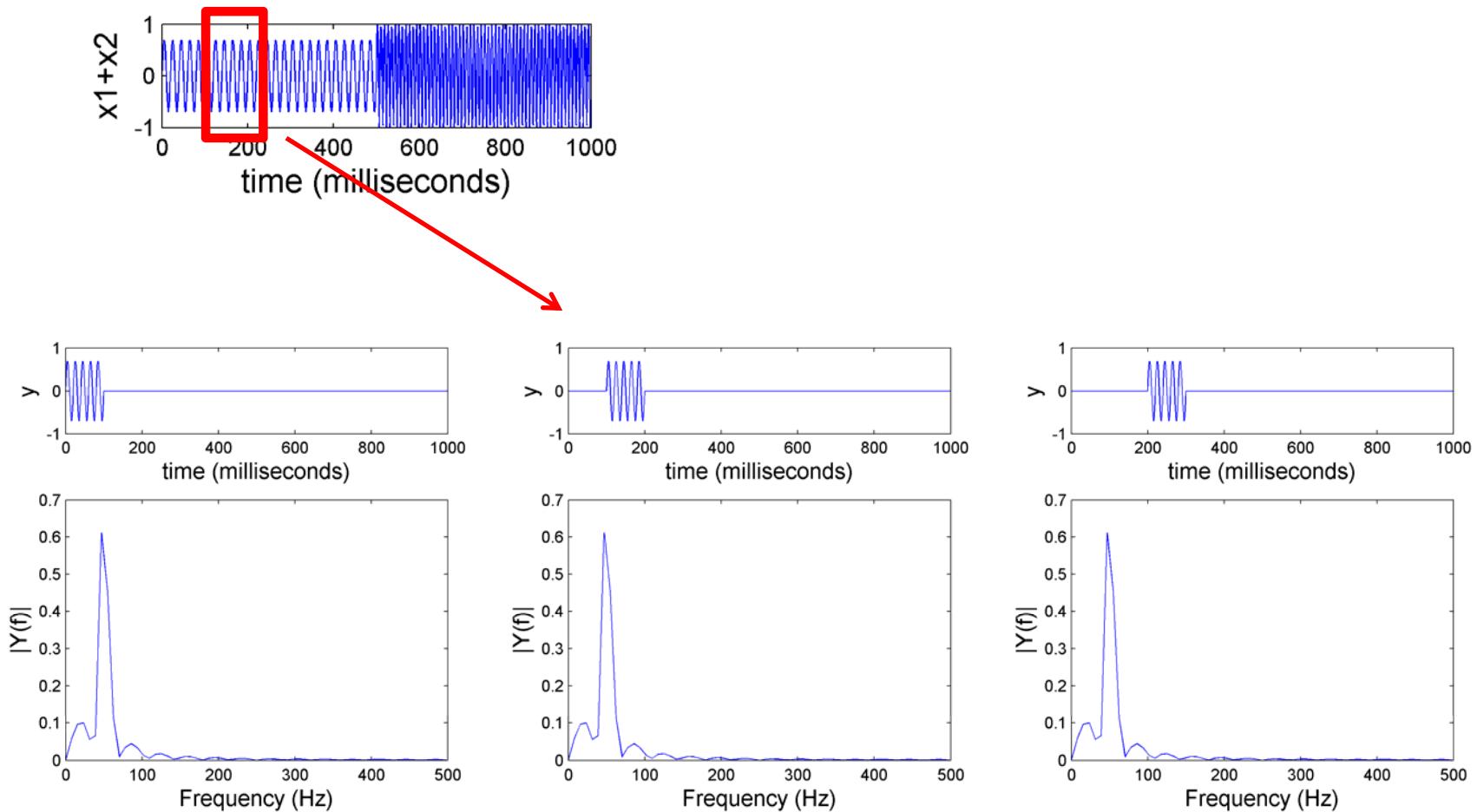
- Problem: cannot “localize” signal of interest



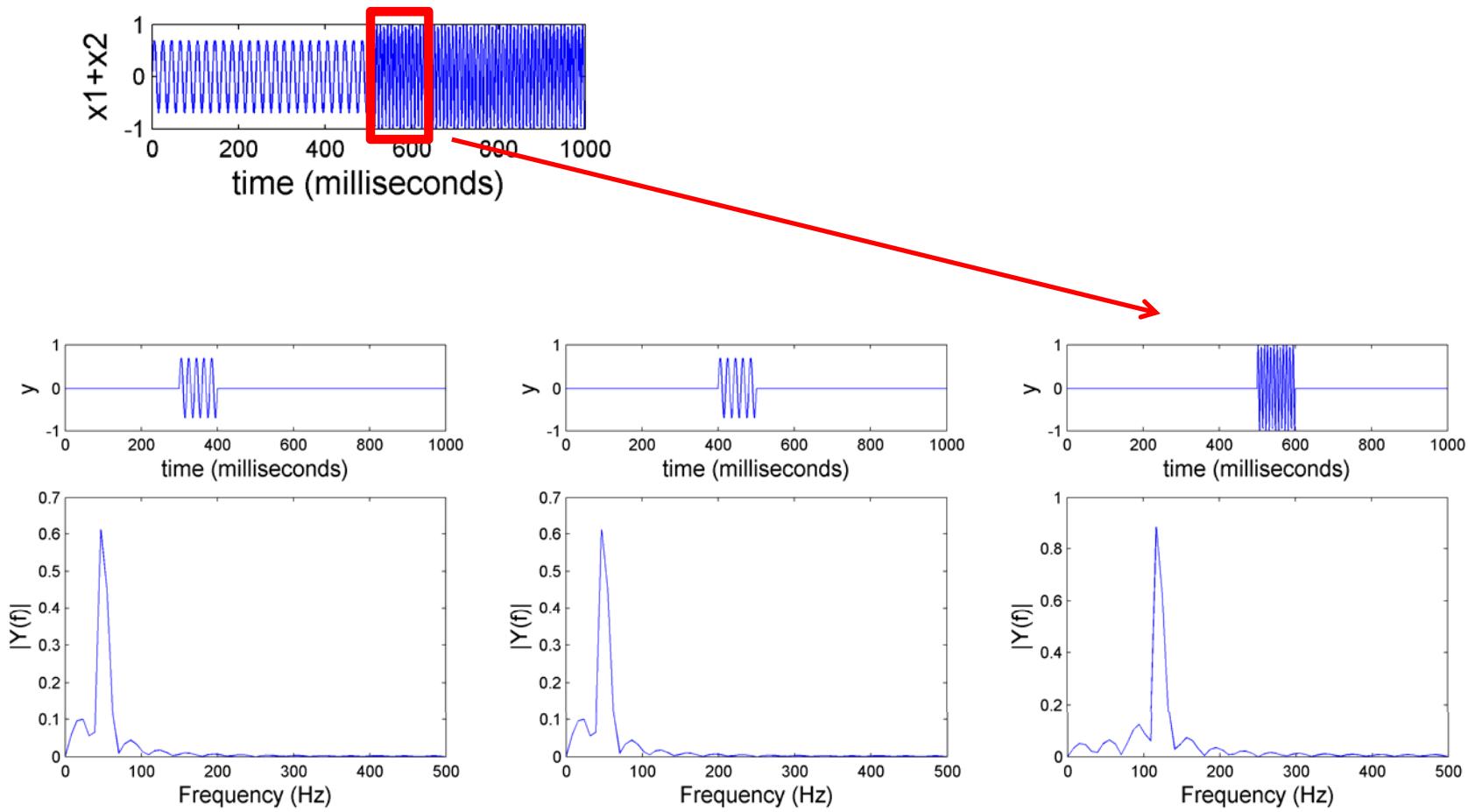
Short Time Fourier Transform (STFT)

- “**Windowed**” version of the Fourier Transform
- Output: a **time-frequency representation**
- MATLAB code
 - <https://www.mathworks.com/help/signal/ref/spectrogram.html>
 - doc spectrogram
 - spectrogram(y,window,noverlap,nfft)
 - spectrogram(y,100,50,100,sr,'yaxis')

Short Time Fourier Transform (STFT)

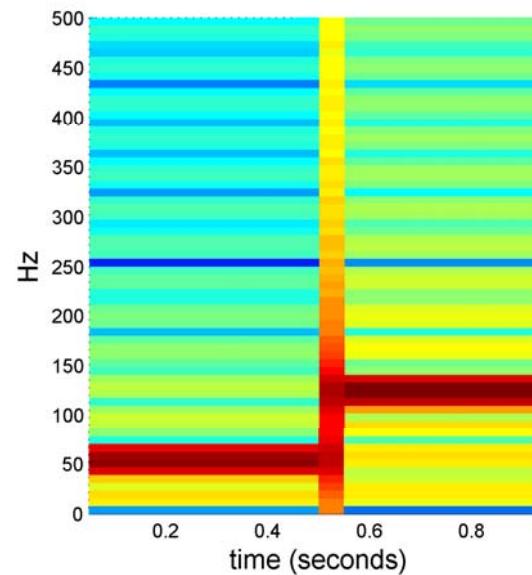
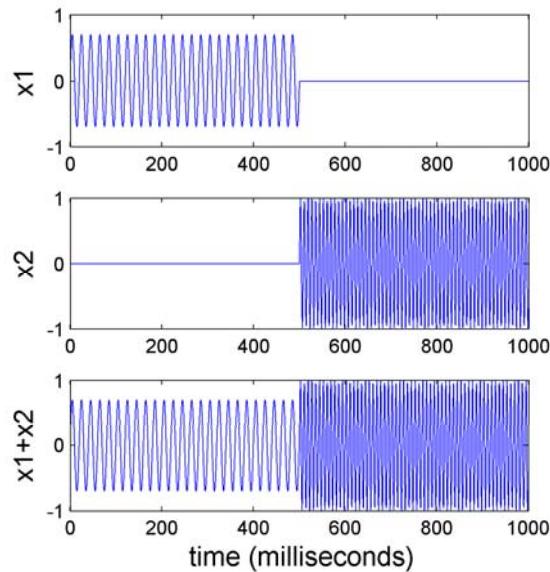


Short Time Fourier Transform (STFT)



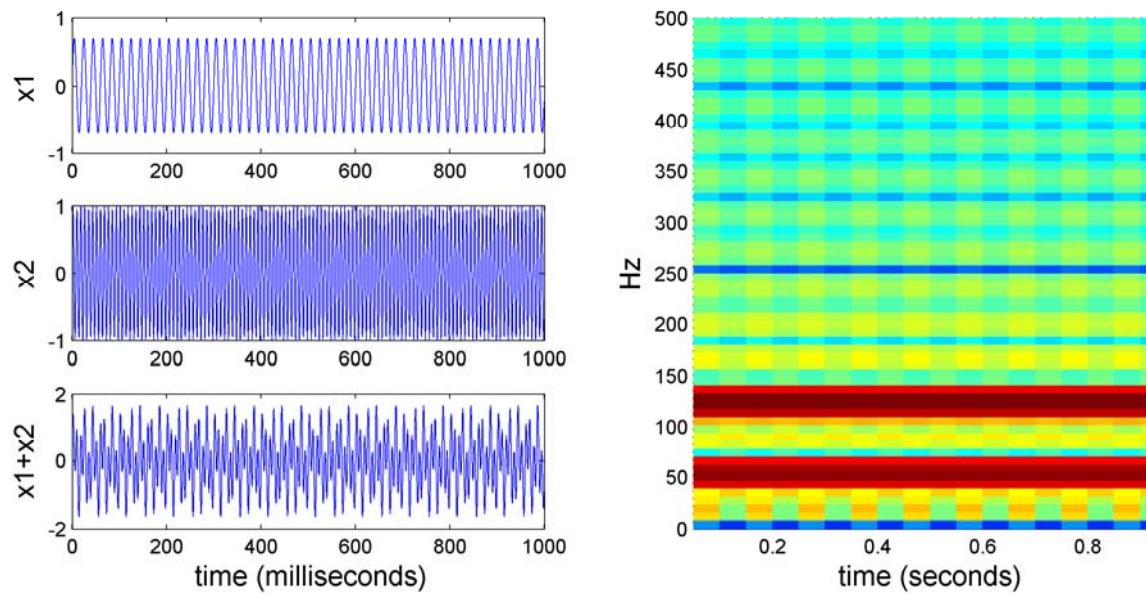
Short Time Fourier Transform (STFT)

- window size = 100



Short Time Fourier Transform (STFT)

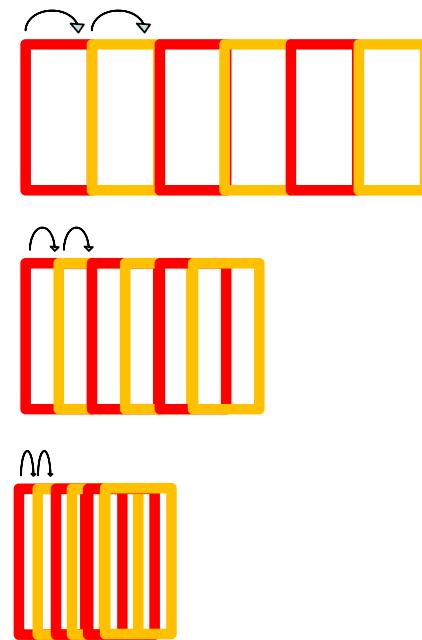
- window size = 100



Short Time Fourier Transform (STFT)

- Hop size
 - Can be proportional to the window size

- $\text{hop_size} = \text{win_size}$
- $\text{hop_size} = 0.5 * \text{win_size}$
- $\text{hop_size} = 0.1 * \text{win_size}$

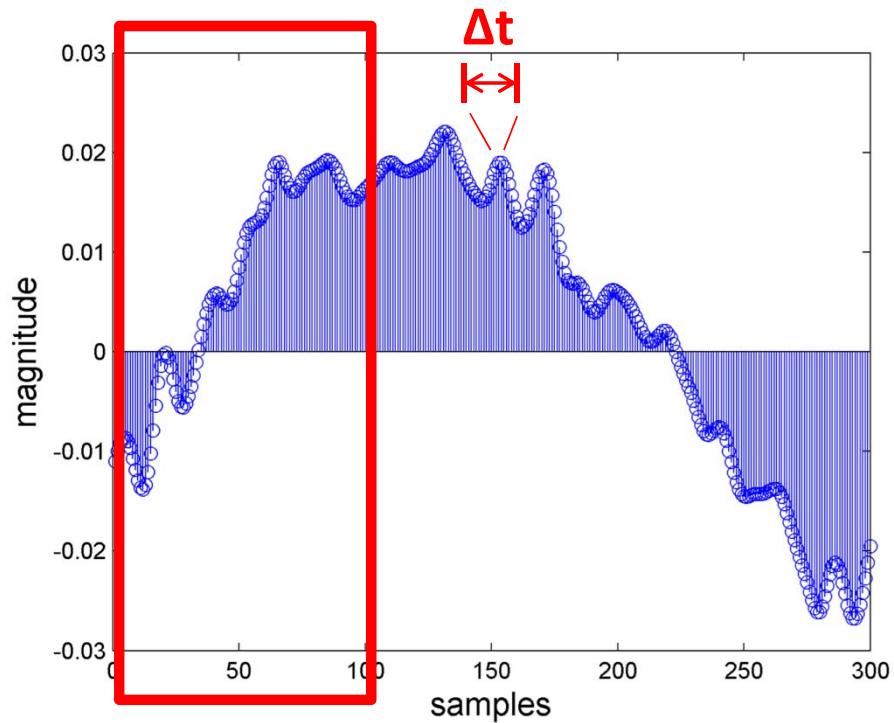


- Can also be not proportional to the window size

Quiz Time

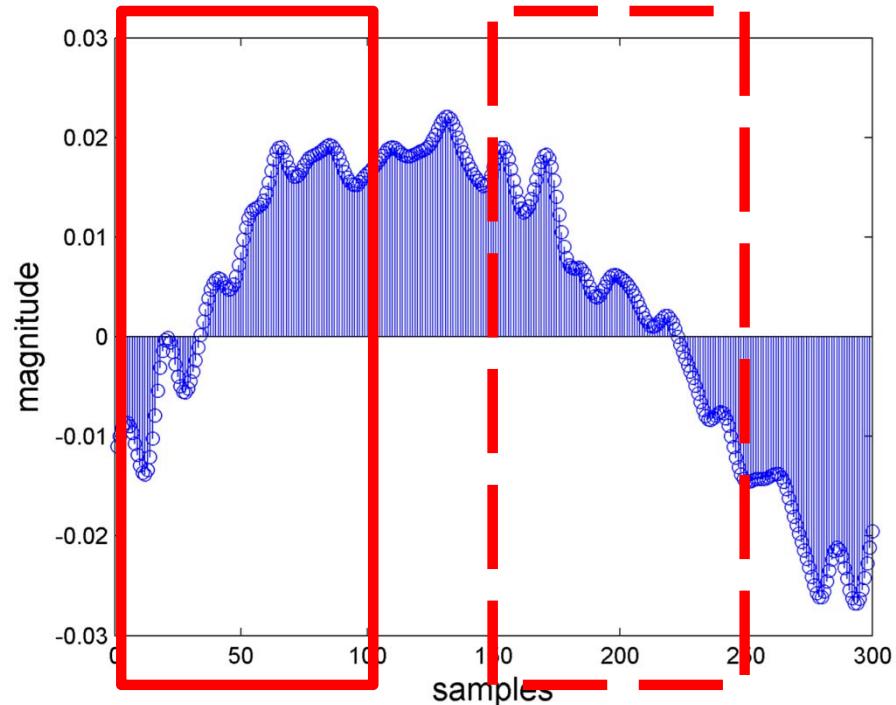
1. When the sampling rate (sr) is 1k Hz, what would be the time interval (in seconds) between two neighboring samples?

2. When the $sr=1k$ Hz, if we use a window size of 100 samples for the STFT, what is the actual duration of the window (in seconds)?



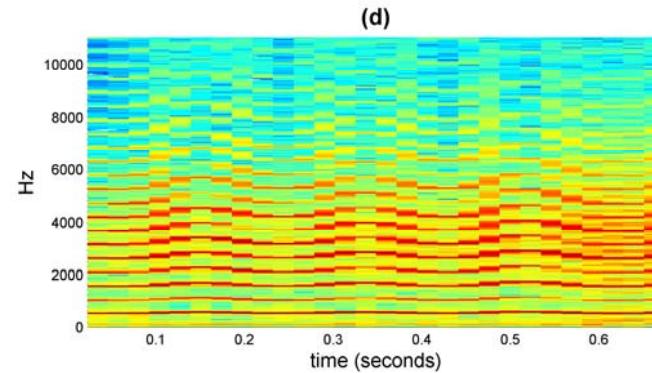
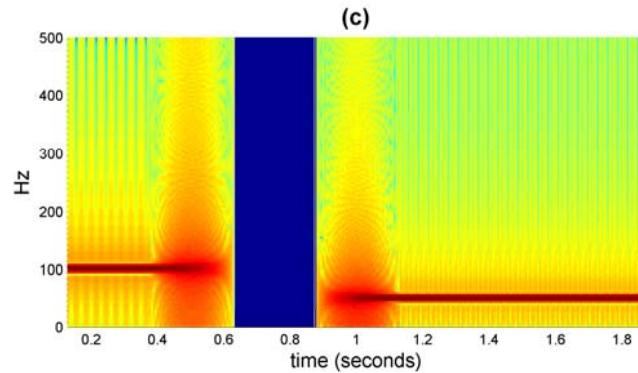
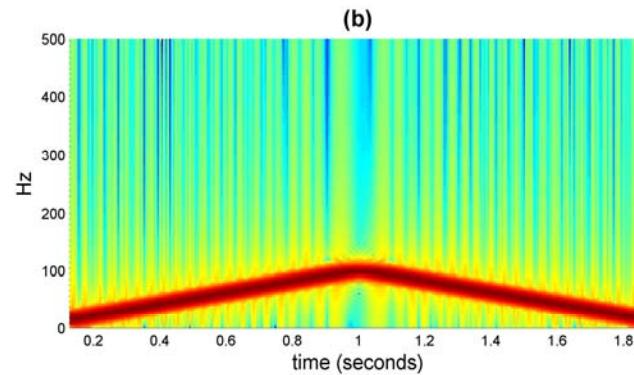
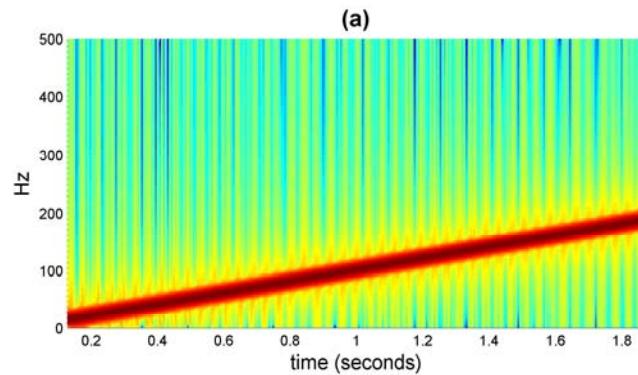
Quiz Time

3. When the $sr=1k$ Hz and we use a STFT window size of 100 samples with no overlaps between consecutive windows, how many times do we need to move the window to cover a signal with 300 samples?
4. And, if there is 50% overlaps between windows, how many times do we need to move the window?



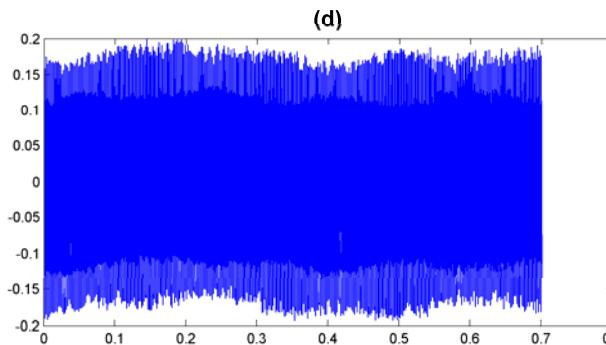
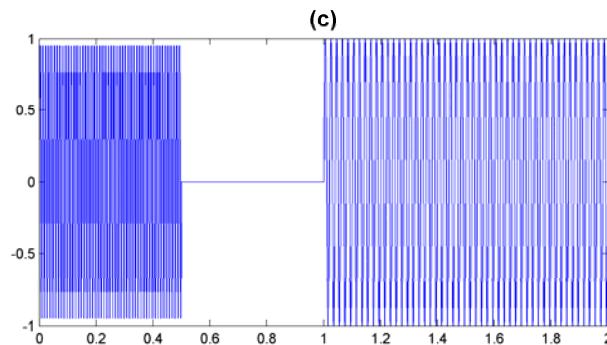
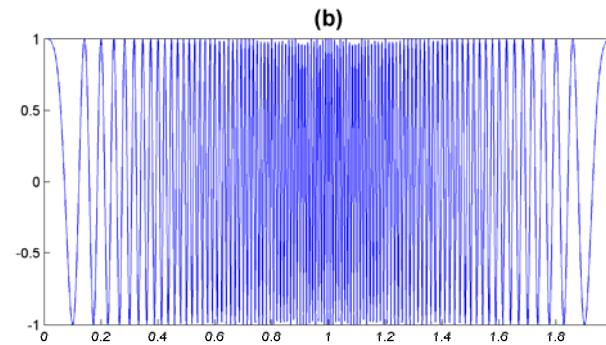
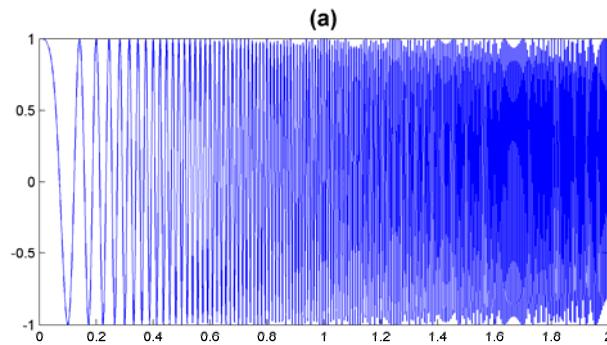
Quiz Time

5. Given the following spectrograms, try to draw the corresponding waveforms



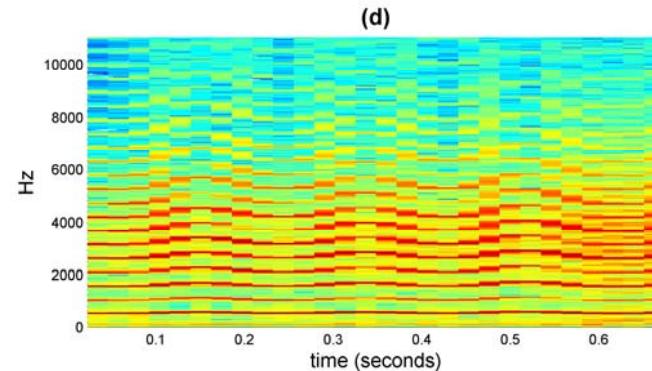
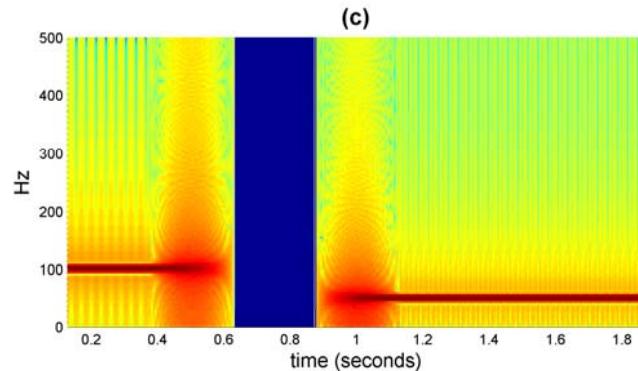
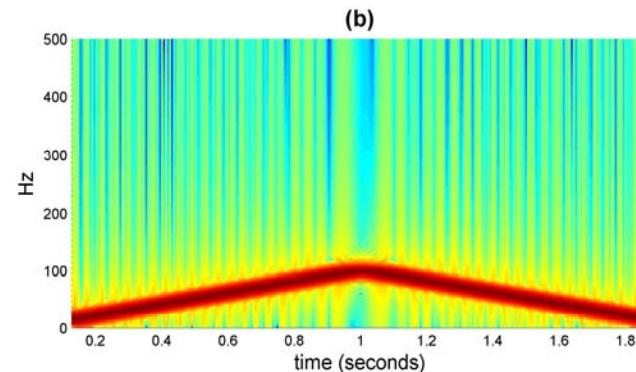
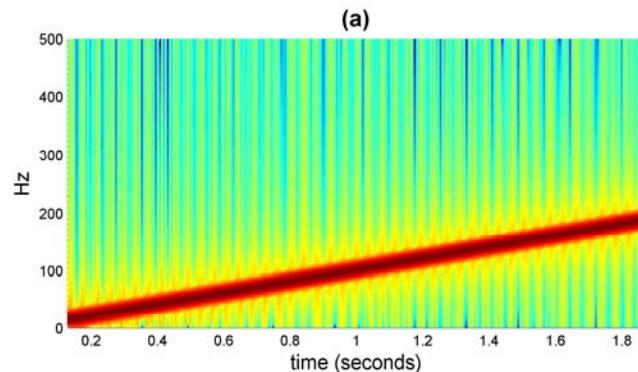
Quiz Time

5. Given the following spectrograms, try to draw the corresponding waveforms (SOLUTION)



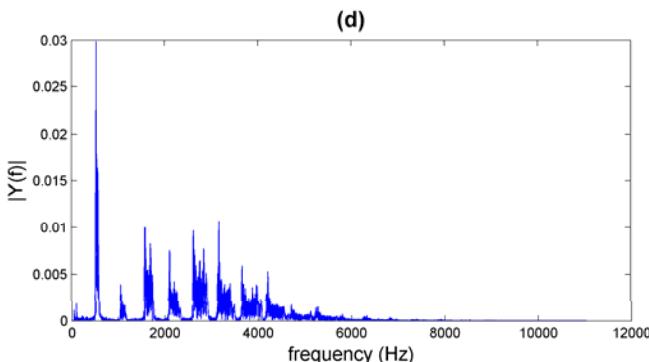
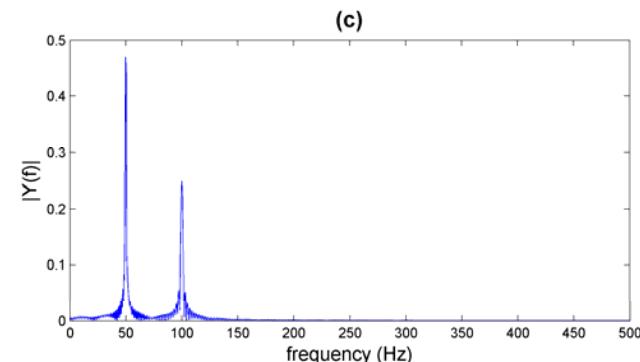
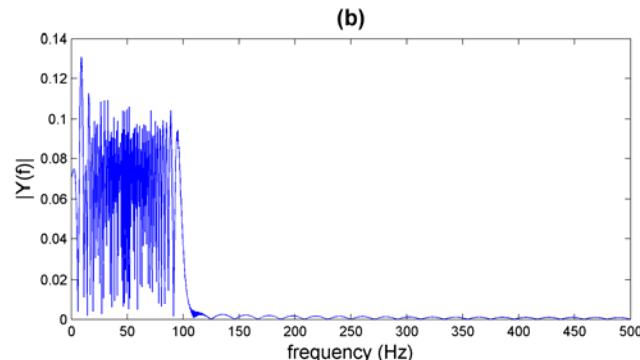
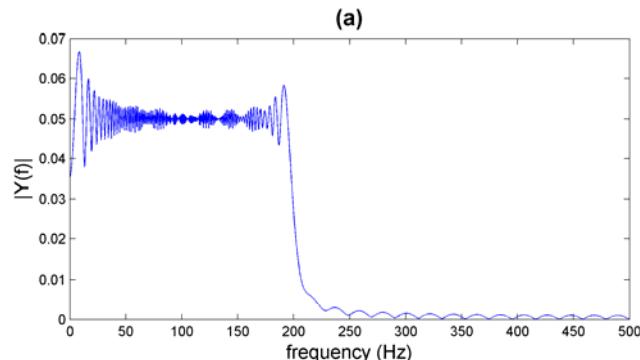
Quiz Time

6. Given the following spectrograms, try to draw the corresponding the spectra computed by Fourier Transform



Quiz Time

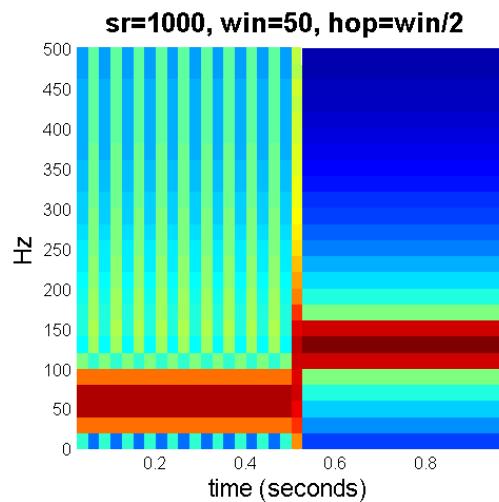
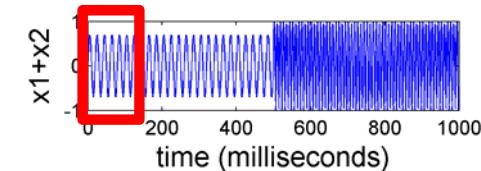
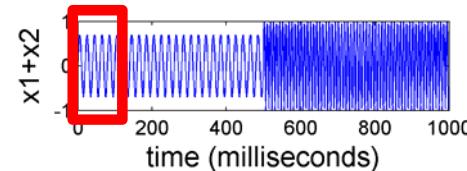
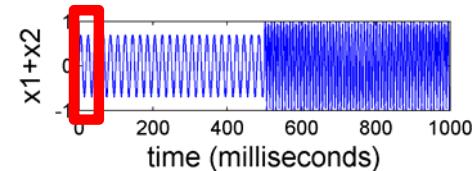
6. Given the following spectrograms, try to draw the corresponding the spectra computed by Fourier Transform (SOLUTION)



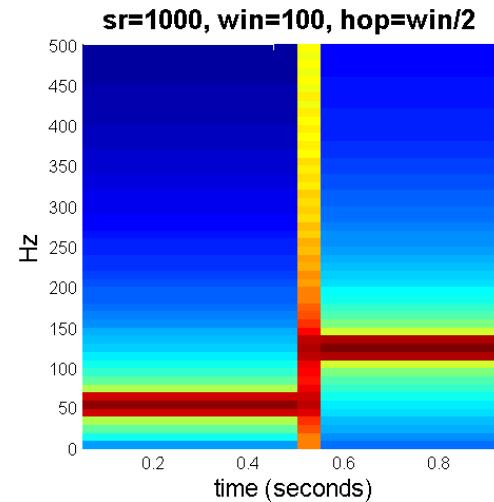
Understanding STFT

(assuming the hop size is proportional to the window size)

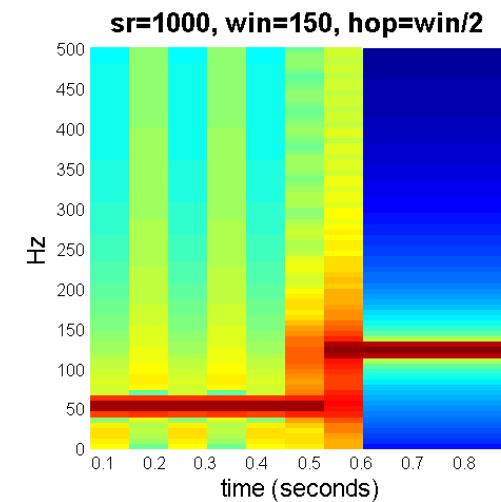
- Different window size (win_size = 50, 100, 150)



size: 26 x 39



size: 51 x 19

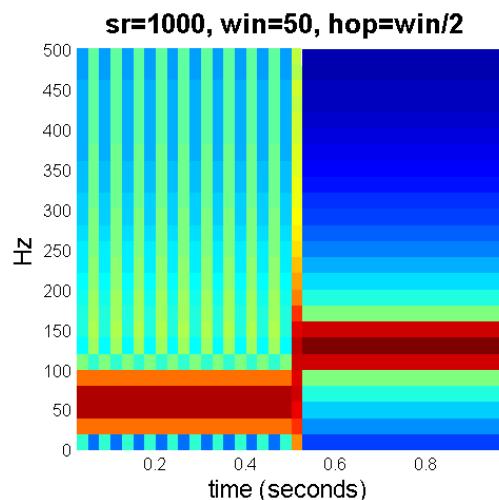


size: 76 x 12

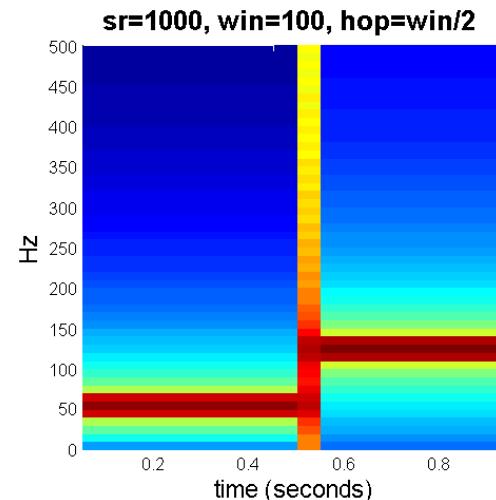
Understanding STFT

(assuming the hop size is proportional to the window size)

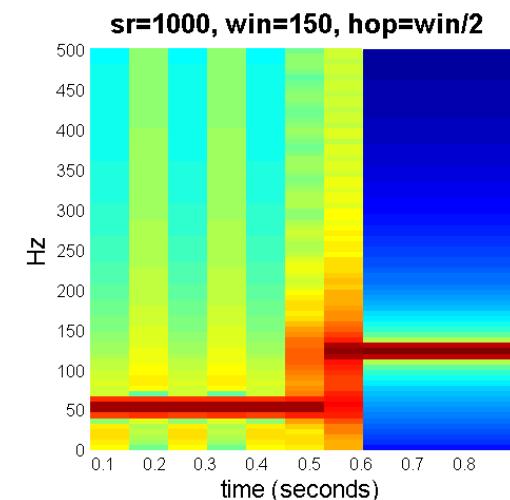
- Shorter window → worse **frequency resolution**
- Longer window → worse **temporal resolution**



size: 26 x 39



size: 51 x 19

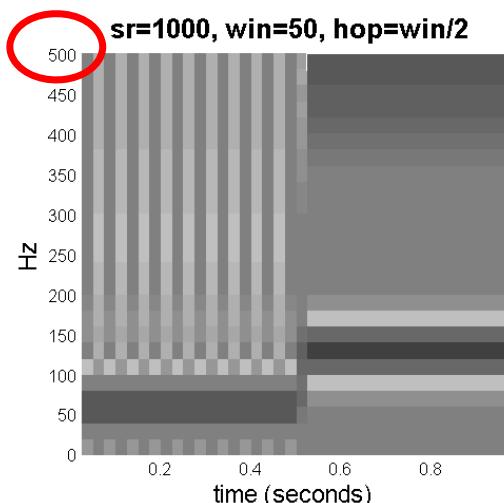


size: 76 x 12

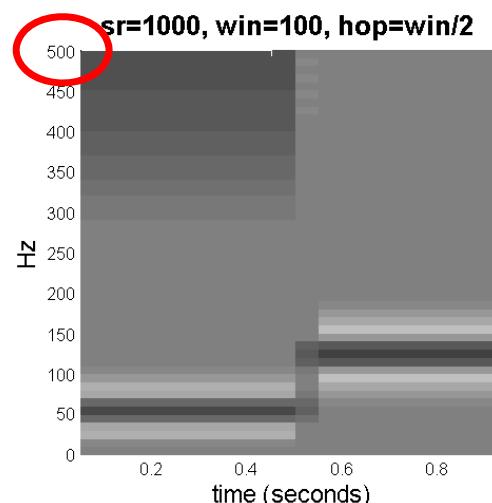
Understanding STFT

- $f_{\text{max}} = sr/2$

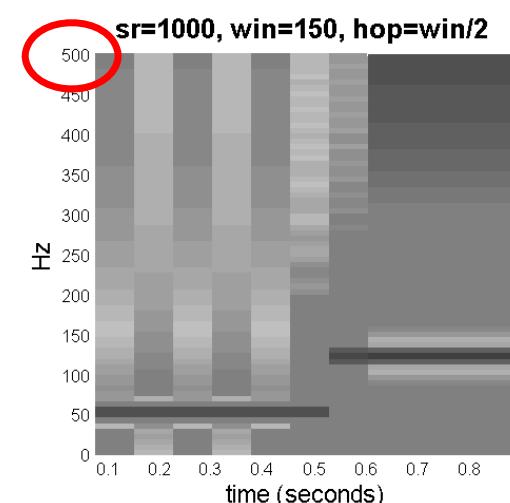
- sampling freq > 2* the highest freq in the signal
(Nyquist–Shannon sampling theorem)



size: 26 x 39



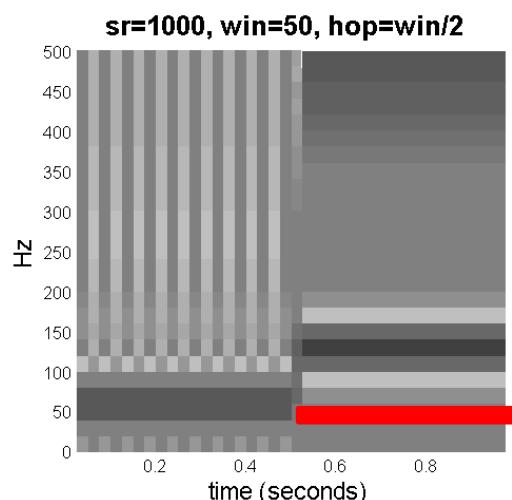
size: 51 x 19



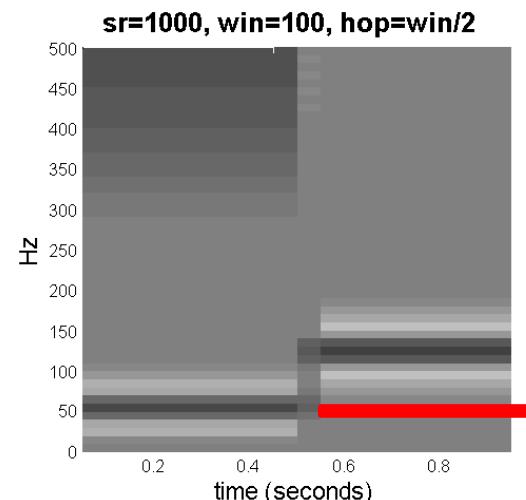
size: 76 x 12

Understanding STFT

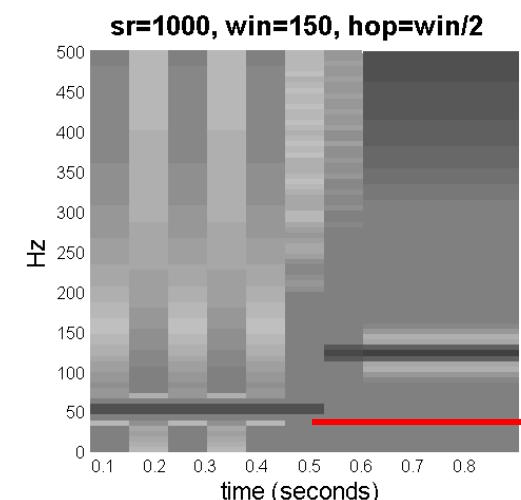
- **freq_resolution = sr/win_size**
 - longer window → better frequency resolution
 - freq_resolution = 20, 10, 6.6667 (Hz), respectively



size: 26 x 39



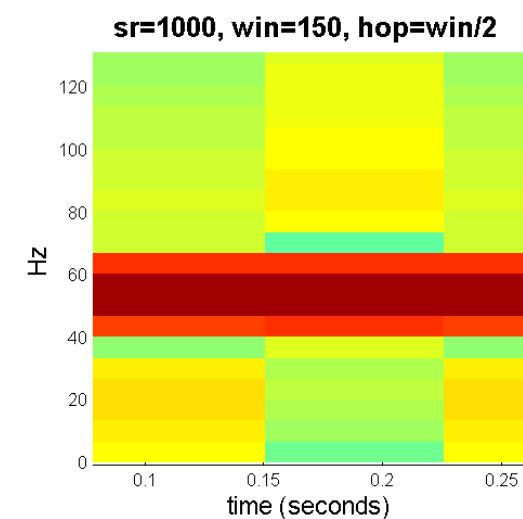
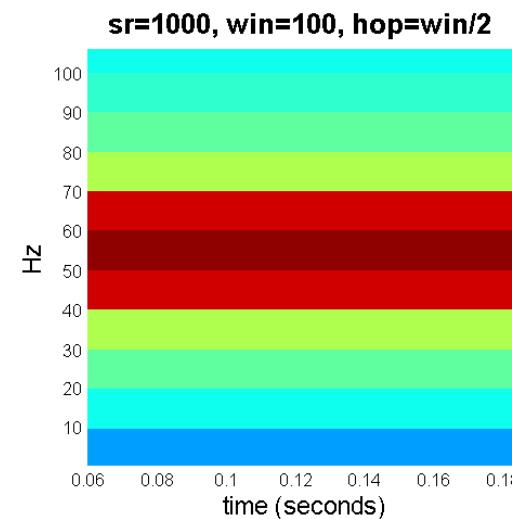
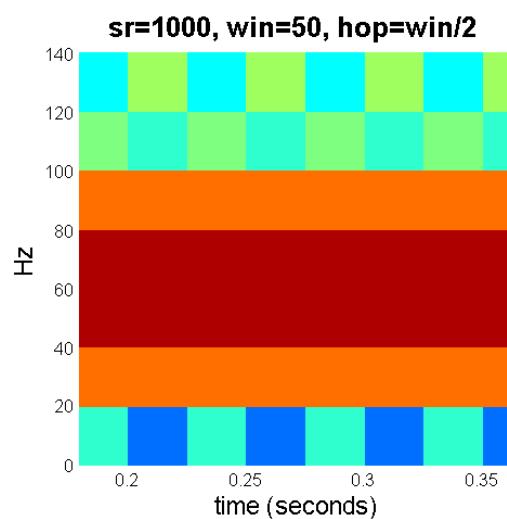
size: 51 x 19



size: 76 x 12

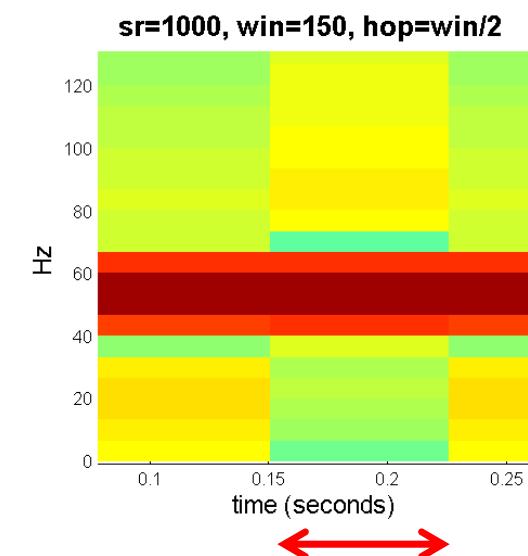
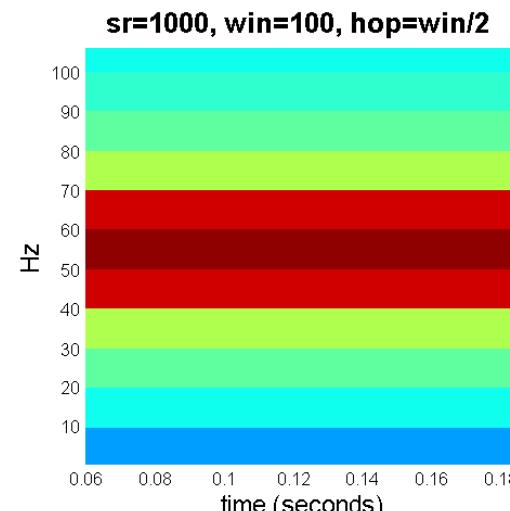
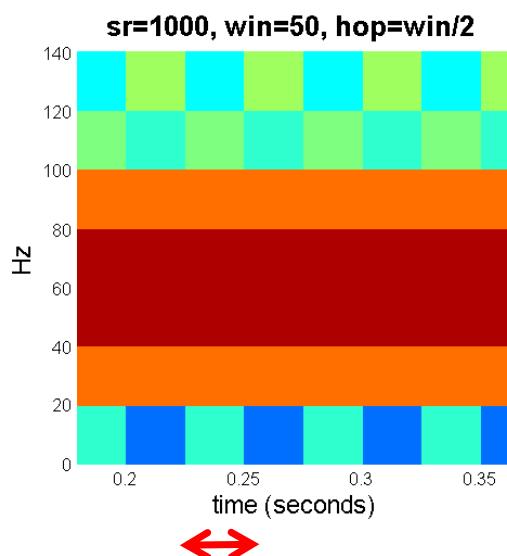
Understanding STFT

- **freq_resolution = sr/win_size**
 - longer window → better frequency resolution
 - freq_resolution = 20, 10, 6.6667 (Hz), respectively



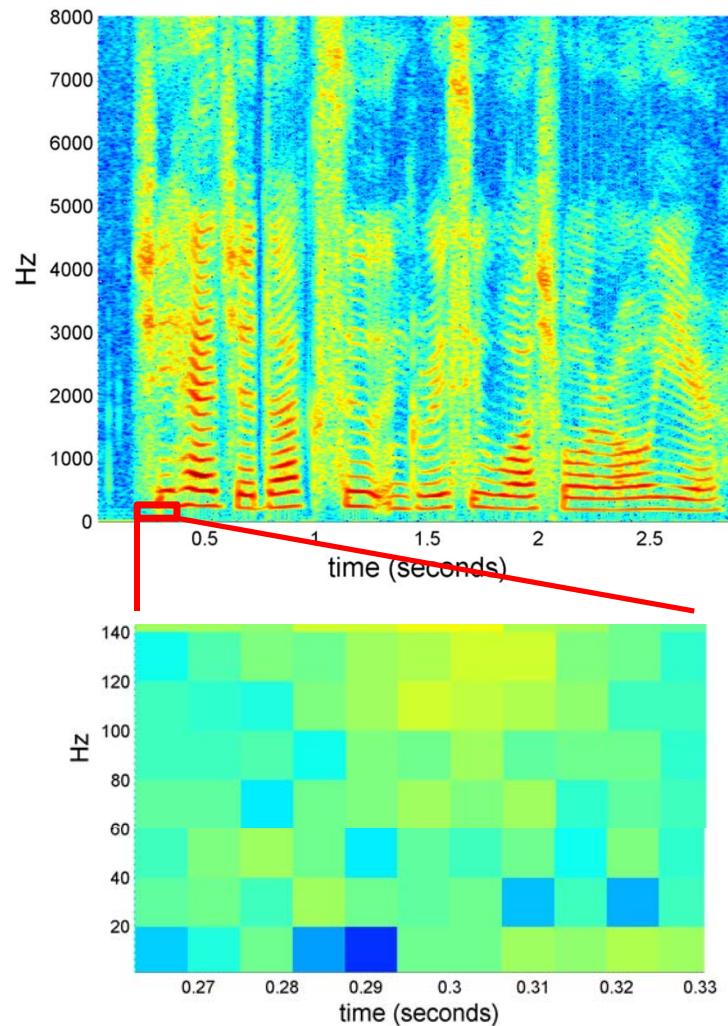
Understanding STFT

- **temporal_resolution: hop_size**
 - longer window → worse temporal resolution
 - $\text{temp_resolution} = 25, 50, 75 \text{ (ms)}$, respectively



Quiz Time

1. The figure on the top-right is the spectrogram of a signal. What is the underlying sampling rate?
2. The figure on the bottom-right is a zoom-in of the above figure. We can see that the frequency resolution is 20 Hz. What is the window size (in samples)?
3. The temporal resolution is close to 6.6 ms. What's the hop size (in samples), approximately?



Quiz Time

4. Given an EEG headset that samples signals at 128 Hz, if we want to be able to discriminate frequency components that differ by 0.5 Hz in frequency, what is the minimal window size (in samples) we need to use? What is the length of such a window in seconds then?
5. Following the previous question, if we further want to discriminate events that differ in time by 0.5 second, what is the maximal hop size (in samples) we need to use?

Brainwaves, Frequencies and Functions

Unconscious		Conscious		
Delta	Theta	Alpha	Beta	Gamma
0,5 – 4 Hz	4 – 8 Hz	8 – 13 Hz	13 – 30 Hz	30-42 Hz
Instinct	Emotion	Consciousness	Thought	Will

Quiz Time

6. Given a music signal with $sr = 44,100$ Hz, when we use a window size of 1,024 samples, what would be the frequency resolution?
7. According to the figure on the right, we know that the fundamental frequency (f_0) of A1 is 55 Hz, that of A#1 is 58.27 Hz, etc. Following the previous question, which notes does the first frequency bin of the STFT would cover?

Note name	Keyboard	Frequency Hz
A0		27.500
B0		30.868
C1		32.703
D1		36.708
E1		41.203
F1		43.654
G1		48.999
A1		55.000
B1		61.735
C2		65.406
D2		73.416
E2		82.407
F2		87.307
G2		97.999
A2		110.00
B2		123.47
C3		130.81
D3		146.83
E3		164.81
F3		174.61
G3		196.00
A3		220.00
B3		246.94
C4		261.63
D4		293.67
		311.13

Quiz Time

6. Given a music signal with $sr = 44,100$ Hz, when we use a window size of 1,024 samples, what would be the frequency resolution?

Sol: 43.1 Hz

7. According to the figure on the right, we know that the fundamental frequency (f_0) of A1 is 55 Hz, that of A#1 is 58.27 Hz, etc. Following the previous question, which notes does the first frequency bin of the STFT would cover?

Note name	Keyboard	Frequency Hz
[0, 43.1)		27.500
B0		30.868
C1		32.703
D1		36.708
E1		38.891
[43.1, 86.2)		43.654
G1		48.999
A1		55.000
B1		61.735
C2		65.406
D2		73.416
E2		82.407
[86.2, 129.3)		87.307
G2		97.999
A2		110.00
B2		123.47
[129.3, 172.4)		130.81
C3		146.83
E3		164.81
[172.4, 215.5)		174.61
G3		196.00
A3		220.00
B3		246.94
C4		261.63
D4		293.67
		211.12

Quiz Time

8. Following the '6' and '7';
how if we use a window size of 4,096 samples?

[0, 10.8)
[10.8, 21.5)
[21.5, 32.3)
[32.3, 43.1)
[43.1, 53.8)
...

(Note: Musical notes after F#3 would be covered by only one frequency bin now)

Note name	Keyboard	Frequency Hz
[21.5, 32.3)		27.500 30.868
[32.3, 43.1)		32.703 36.708
D1		34.648
E1		41.203
[43.1, 53.8)		43.654 48.999
G1		46.249 51.913
A1		55.000
B1		61.735
C2		65.406
D2		73.416
E2		82.407
F2		87.307
G2		97.999
A2		110.00
B2		123.47
C3		130.81
D3		146.83
E3		164.81
F3		174.61
G3		196.00
A3		220.00
B3		246.94
C4	261.63
D4		293.67
		311.13

Real Example 1: Piano Transcription

- Curtis Hawthorne et al., “**Onsets and Frames: Dual-objective piano transcription**,” ISMIR 2018

<https://archives.ismir.net/ismir2018/paper/000019.pdf>

Q: What is the frequency resolution?
And the temporal resolution?

Our onset and frame detectors are built upon the convolution layer acoustic model architecture presented in [13], with some modifications. We use *librosa* [15] to compute the same input data representation of mel-scaled spectrograms with log amplitude of the input raw audio with 229 logarithmically-spaced frequency bins, a hop length of 512, an FFT window of 2048, and a sample rate of 16kHz. We present the network with the entire input sequence, which allows us to feed the output of the convolutional frontend into a recurrent neural network (described below).

Real Example 2: Beat Tracking

- Sebastian Böck, Florian Krebs, and Gerhard Widmer, “**Joint beat and downbeat tracking with recurrent neural networks**,” ISMIR 2016

http://www.cp.jku.at/research/papers/Boeck_etal_ISMIR_2016.pdf

120 BPM = 120 beats per minute
= 2 beats per second

(16th note in 4/4 meter: 125 ms)

2.1 Signal Pre-Processing

The audio signal is split into overlapping frames and weighted with a Hann window of same length before being transferred to a time-frequency representation with the Short-time Fourier Transform (STFT). Two adjacent frames are located 10 ms apart, which corresponds to a rate of 100 fps (frames per second). We omit the phase portion of the complex spectrogram and use only the magnitudes for further processing. To enable the network to capture features which are precise both in time and frequency, we use three different magnitude spectrograms with STFT lengths of 1024, 2048, and 4096 samples (at a signal sample rate of 44.1 kHz). To reduce the dimensionality of the features, we limit the frequencies range to [30, 17000] Hz and process the spectrograms with logarithmically spaced

Summary

- **Sampling rate, window size, hop size** all matter
 - Frequency resolution
 - Temporal resolution
 - Number of samples, number of frames, and physical time (in milliseconds)
- Make sure the sampling rate is “right”
 - Do “resample” if the sampling rate of your audio files is different from what is assumed by an open-source model
 - Speech: 16k Hz
 - Music: 22k, 44k, or 48k Hz
 - Similarly for window size, hop size etc

Library: SoXR

<https://github.com/dofuuz/python-soxr>

```
import soxr

y = soxr.resample(
    x,          # 1D(mono) or 2D(frames, channels) array input
    48000,      # input samplerate
    16000       # target samplerate
)
```

Library	Time on CPU (ms)
soxr (HQ)	7.2
scipy.signal.resample	13.4
soxr (VHQ)	15.8
torchaudio	19.2

Library: pyloudnorm

<https://github.com/csteinmetz1/pyloudnorm>

Loudness normalize and peak normalize audio files

Methods are included to normalize audio files to desired peak values or desired loudness.

```
import soundfile as sf
import pyloudnorm as pyln

data, rate = sf.read("test.wav") # load audio

# peak normalize audio to -1 dB
peak_normalized_audio = pyln.normalize.peak(data, -1.0)

# measure the loudness first
meter = pyln.Meter(rate) # create BS.1770 meter
loudness = meter.integrated_loudness(data)

# loudness normalize audio to -12 dB LUFS
loudness_normalized_audio = pyln.normalize.loudness(data, loudness, -12.0)
```

