2023 September 25

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]

Part	Title	Notions, Techniques & Algorithms	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

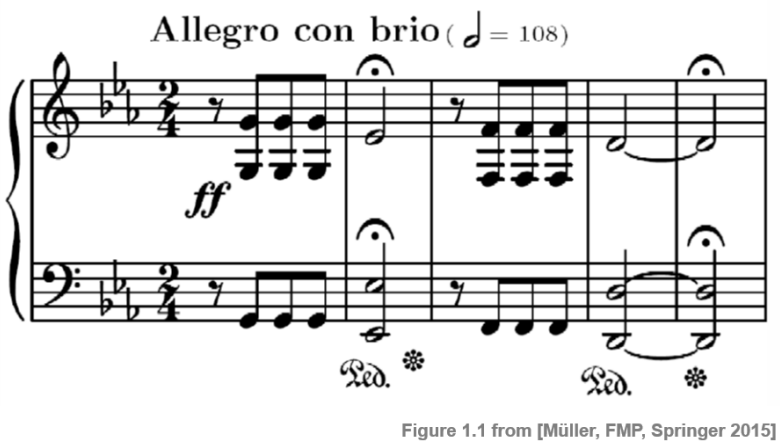
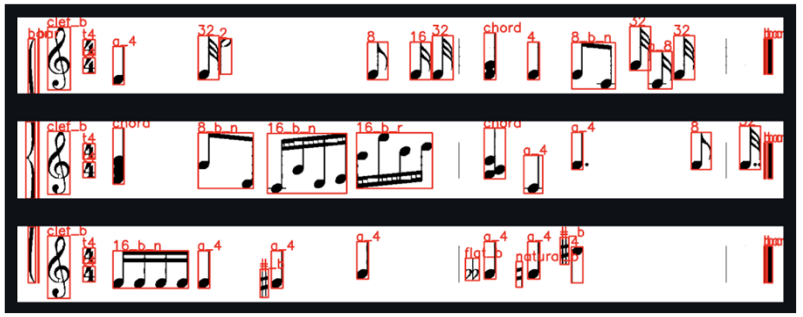


Figure 1.1 from [Müller, FMP, Springer 2015]

- ***Rarely*** directly used as input to neural network models
  - Exception: *optical music recognition* (OMR)



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]

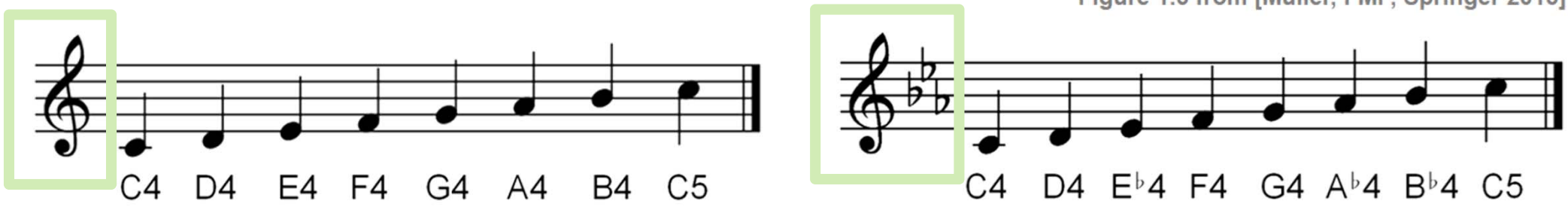
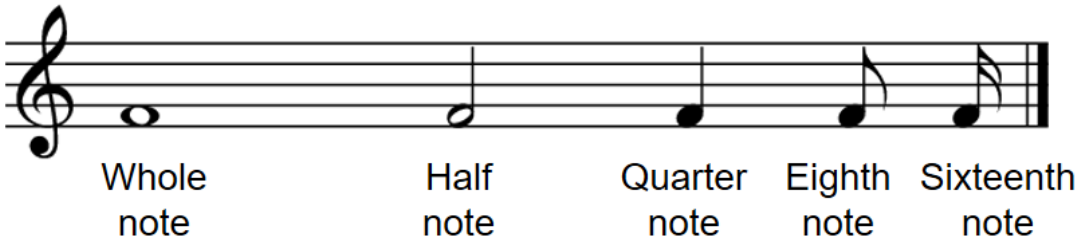


Figure 1.7 from [Müller, FMP, Springer 2015]

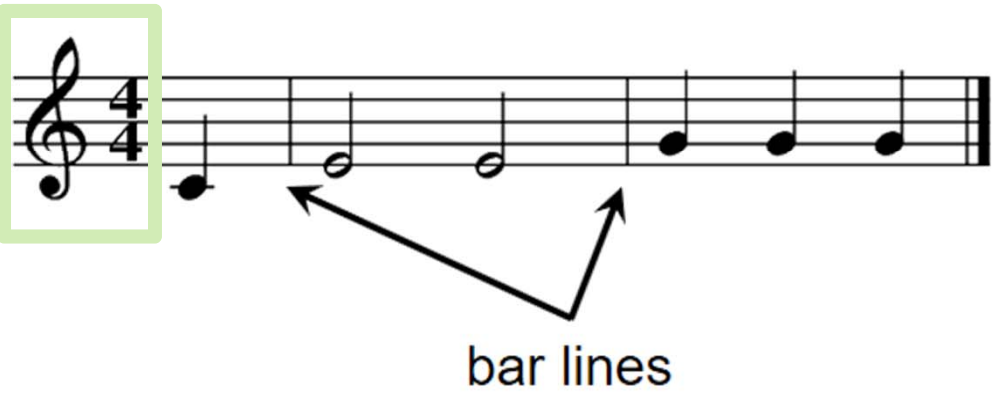


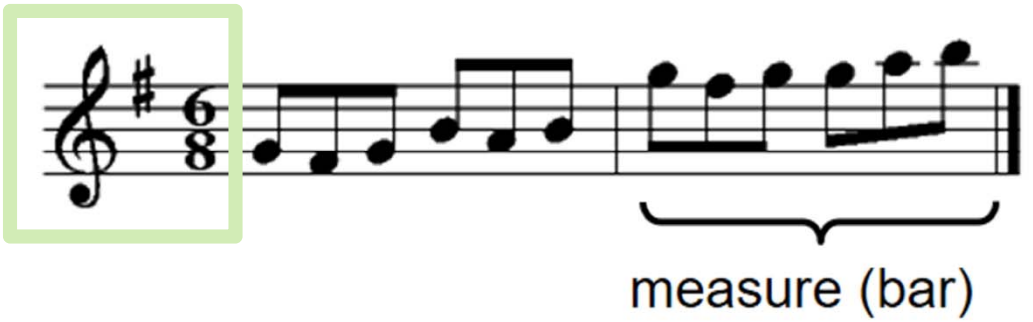
Sheet Music: Meter, Bar, Beat, Rhythm

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]



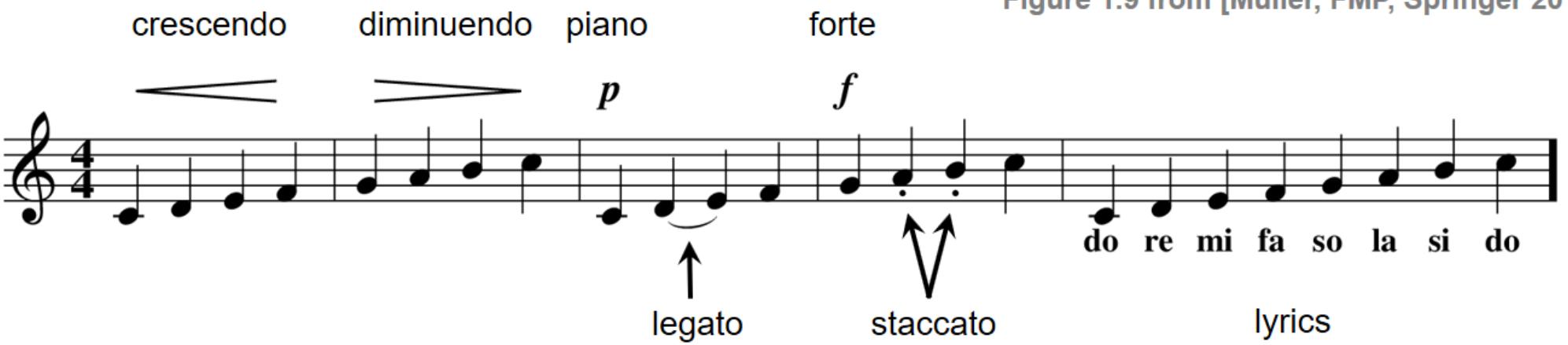


Sheet Music: Articulation Marks

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

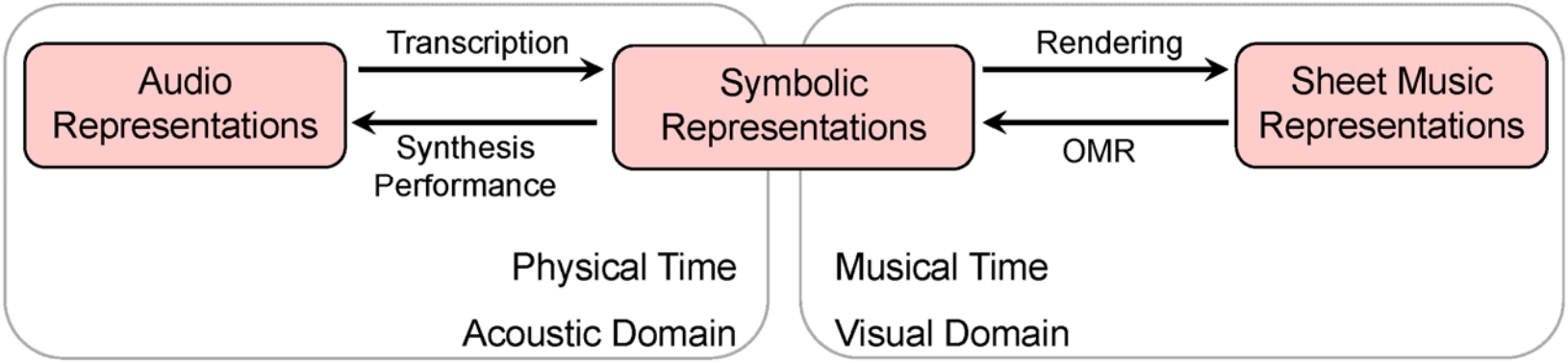
- How certain notes are to be played

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**)

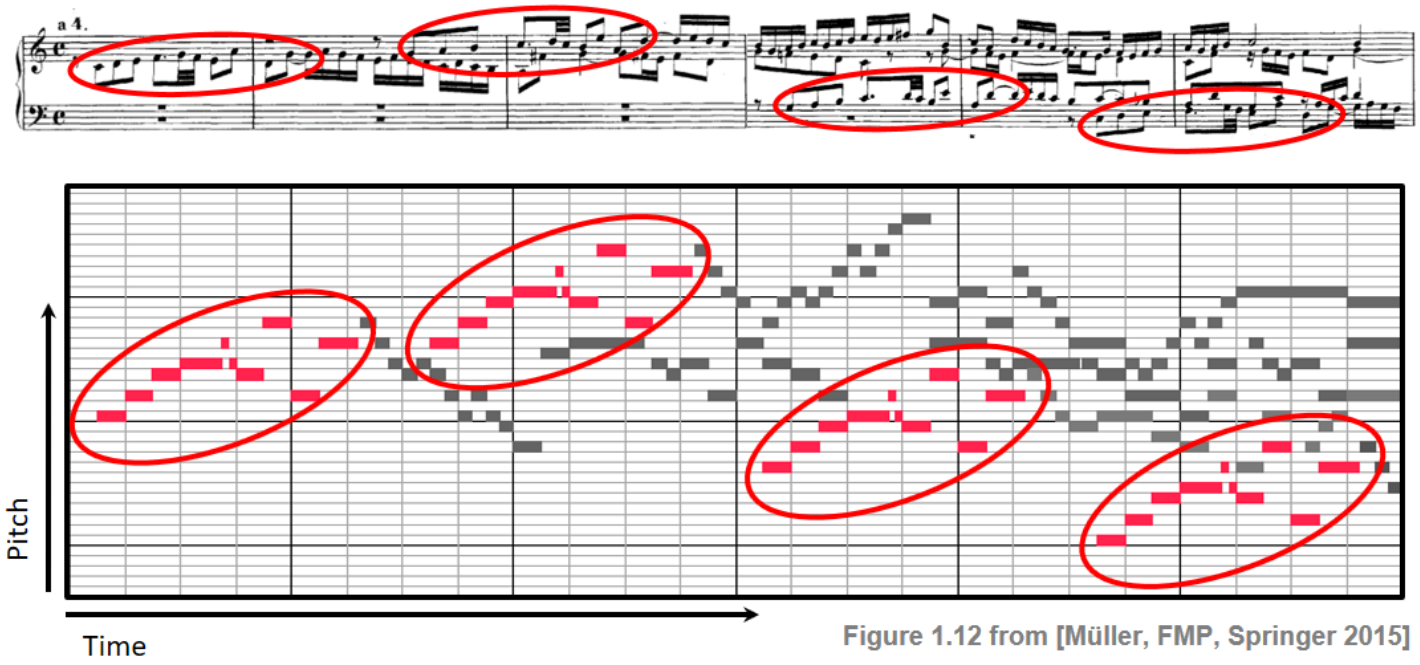
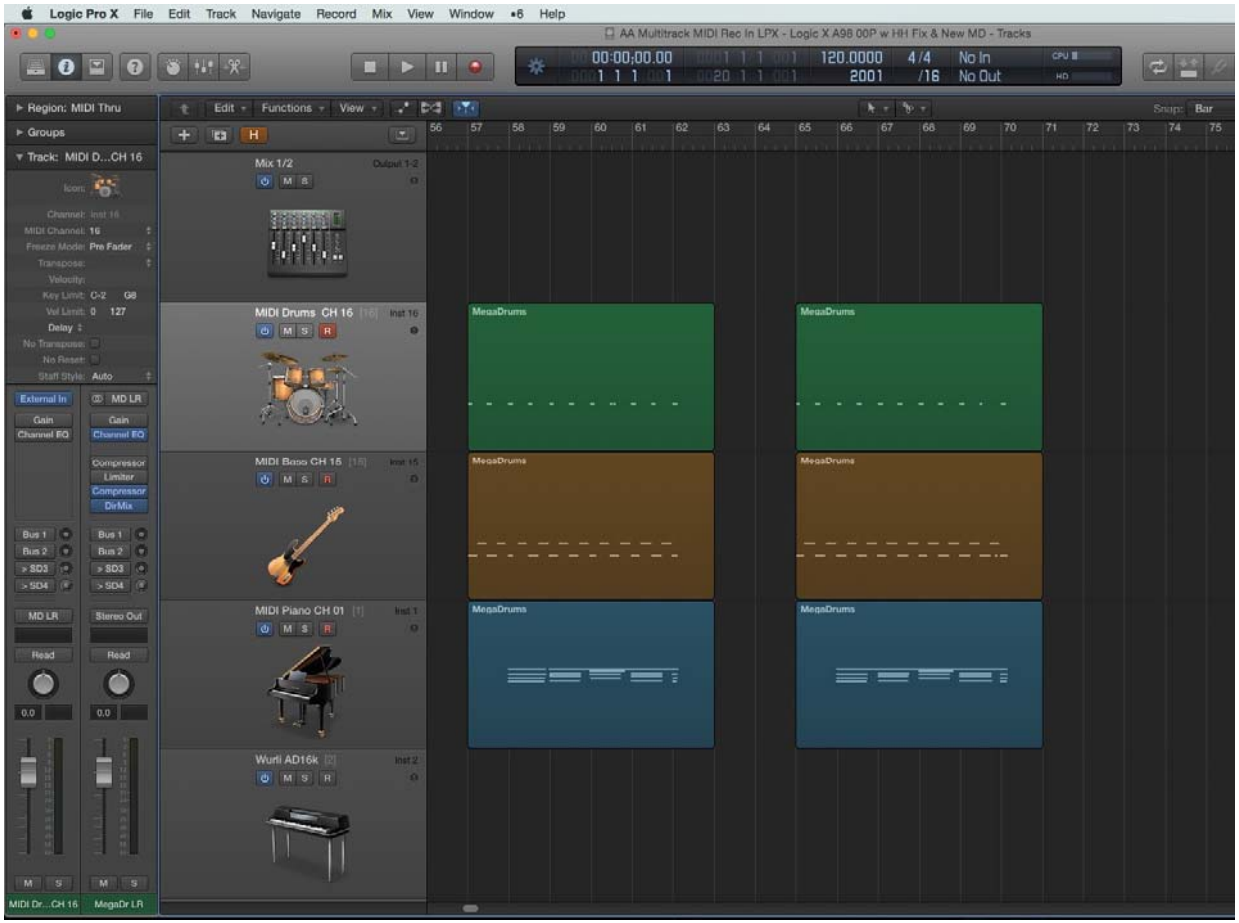


Figure 1.12 from [Müller, FMP, Springer 2015]

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

Piano Roll Representation



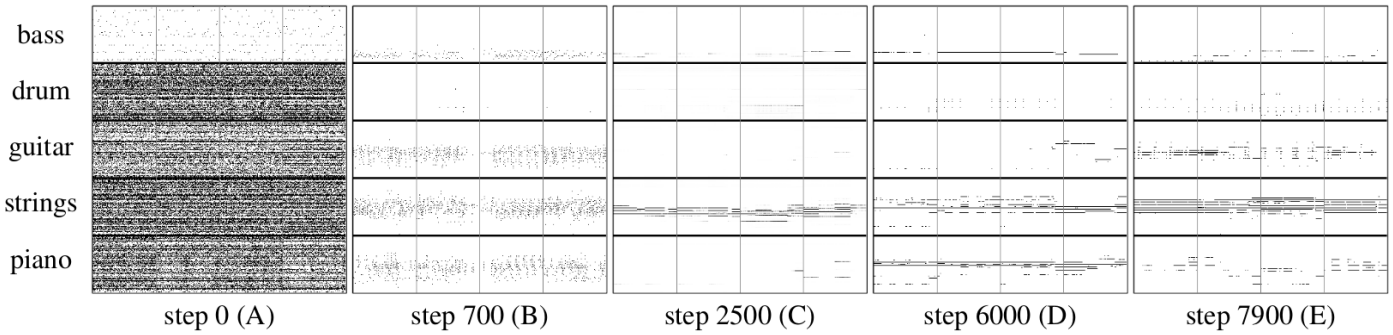
Source: https://www.macprovideo.com/article/audio-software/logic-pro-x-a-guide-to-multitrack-midi-recording

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/

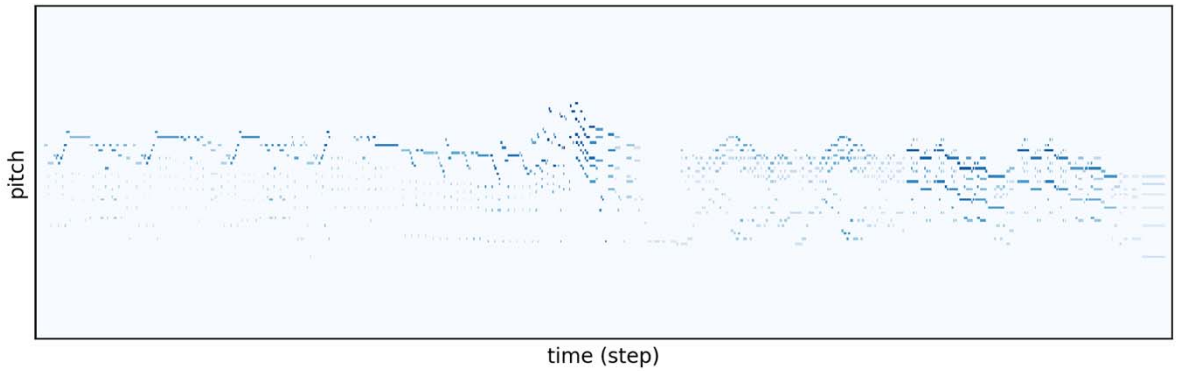
MuseGAN



DiffRoll

Library: PyPianoroll

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



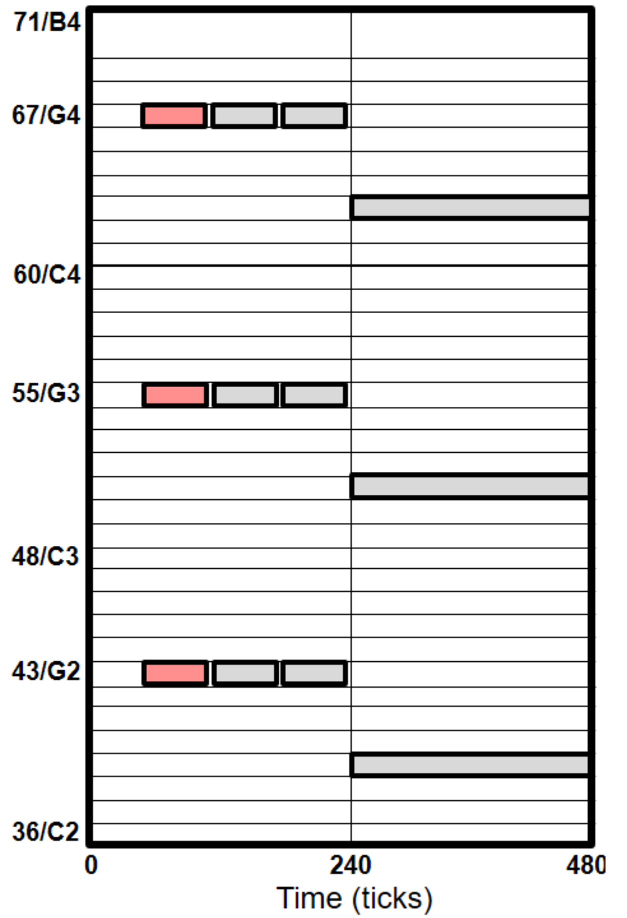
- Features
  - Manipulate multitrack piano rolls intuitively
  - Visualize multitrack piano rolls beautifully
  - Save and load multitrack piano rolls in a space-efficient format
  - Parse **MIDI** files into multitrack piano rolls
  - Write multitrack piano rolls into **MIDI** files

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

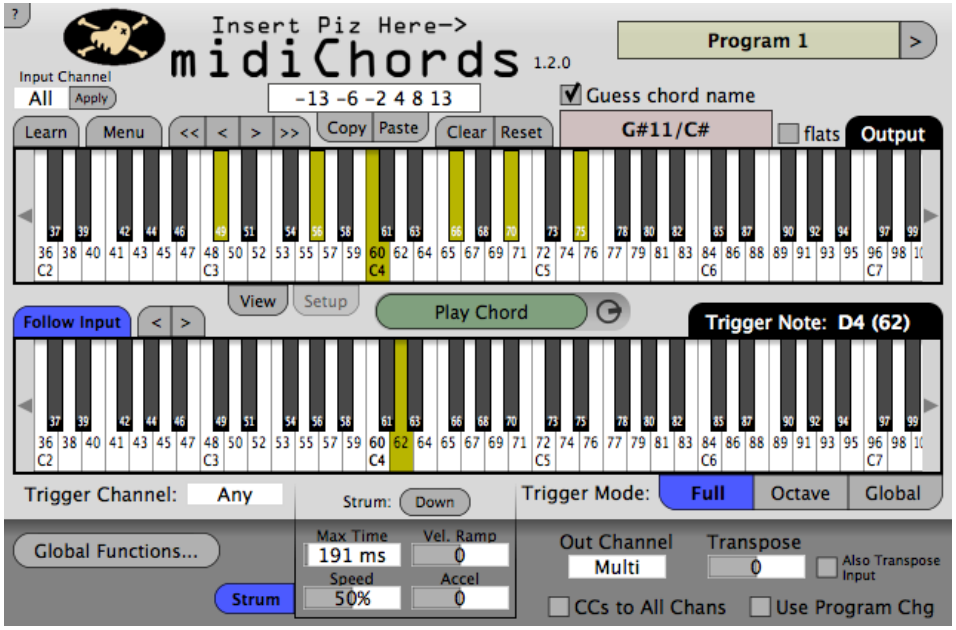
Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE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



Piano Roll vs MIDI Representation

- Both represent a *.MIDI file
  - Piano roll: image-like
  - MIDI Representation: text-like

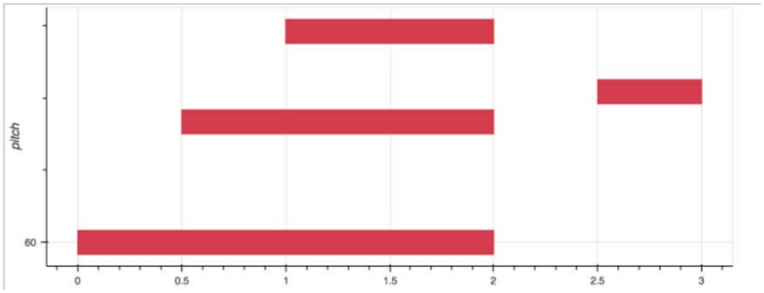
- Piano roll are like fixed-shape images
- MIDI representation is more flexible as it allows for *additional information* to be added
  - For example, **chord tokens, key tokens**, etc



Ref: Figure from https://www.kvraudio.com/product/midichords-by-insert-piz-here

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

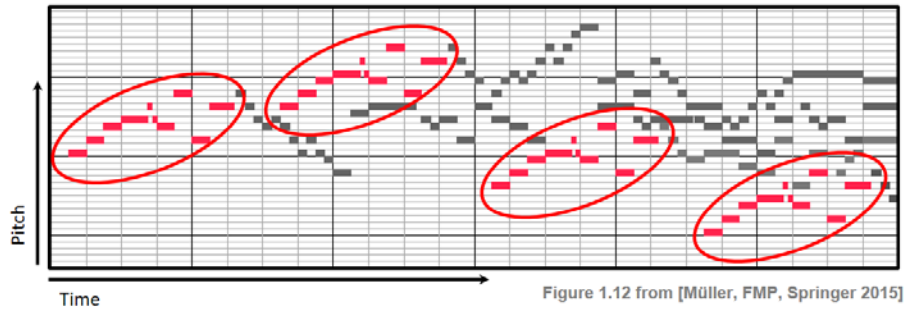
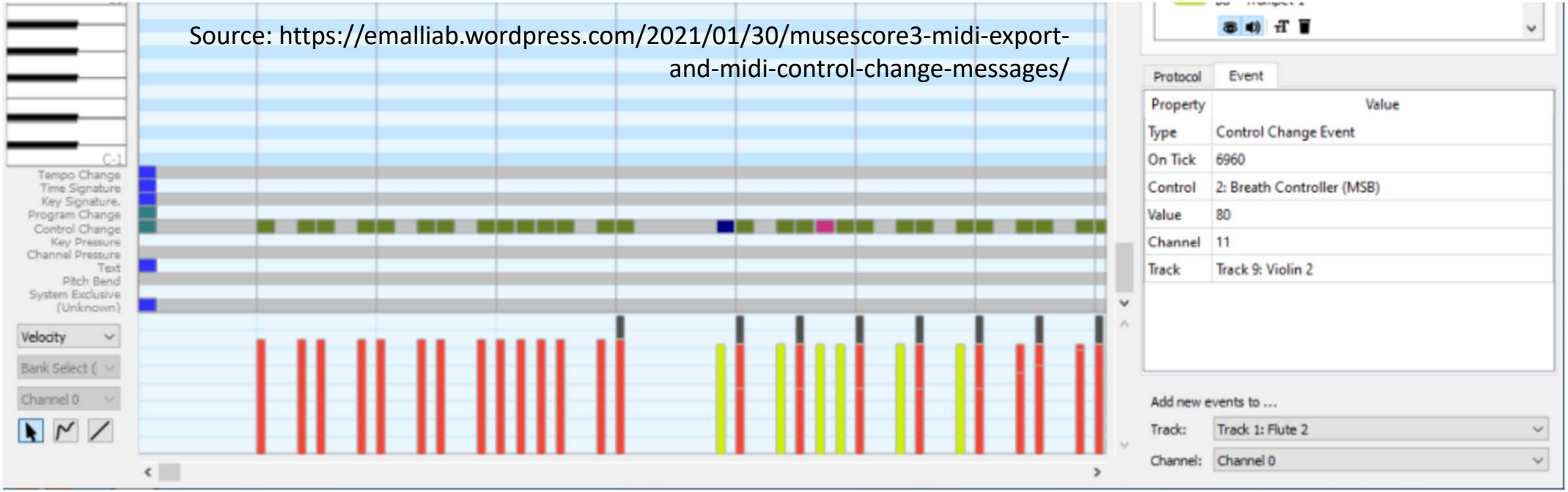


Figure 1.12 from [Müller, FMP, Springer 2015]

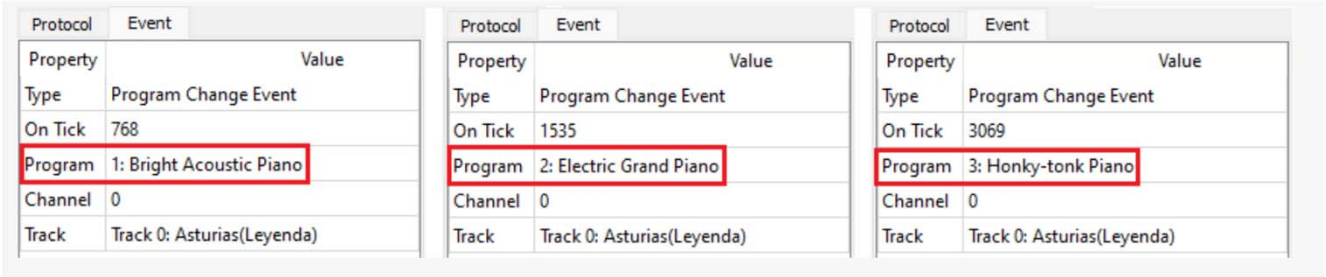
- MIDI subdivides a **quarter note** into basic time units referred to as *clock pulses* or *ticks*
  - Pulses per quarter note (PPQN): commonly 120
  - PPQN determines the resolution of the time stamps associated to note events

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

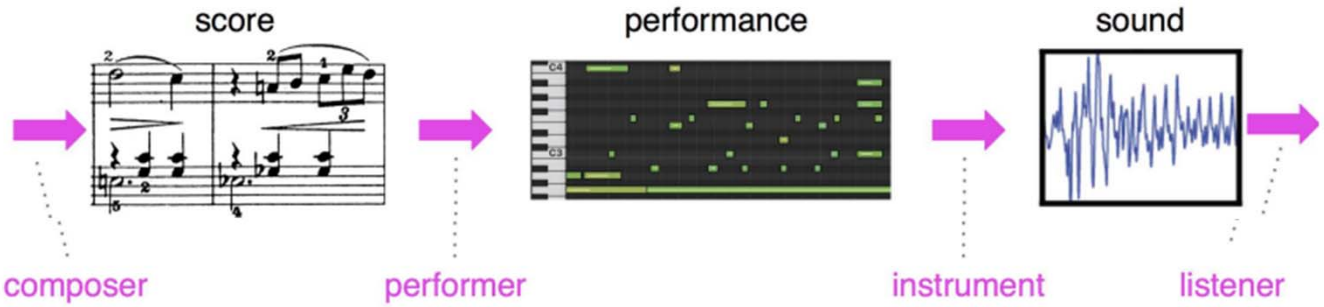


- Tempo change
- Program change
- Control change



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

Ref: Oore et al, "This time with feeling: Learning expressive musical performance," arXiv 2018

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

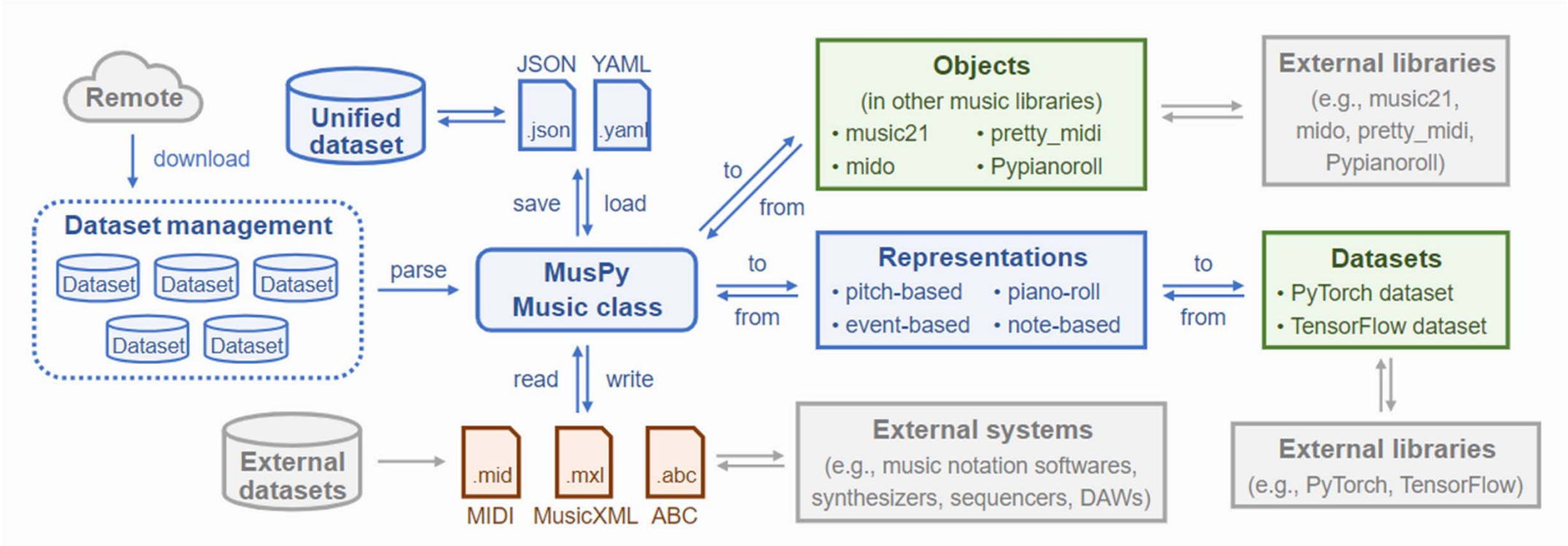
Library: MidiTok

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.”

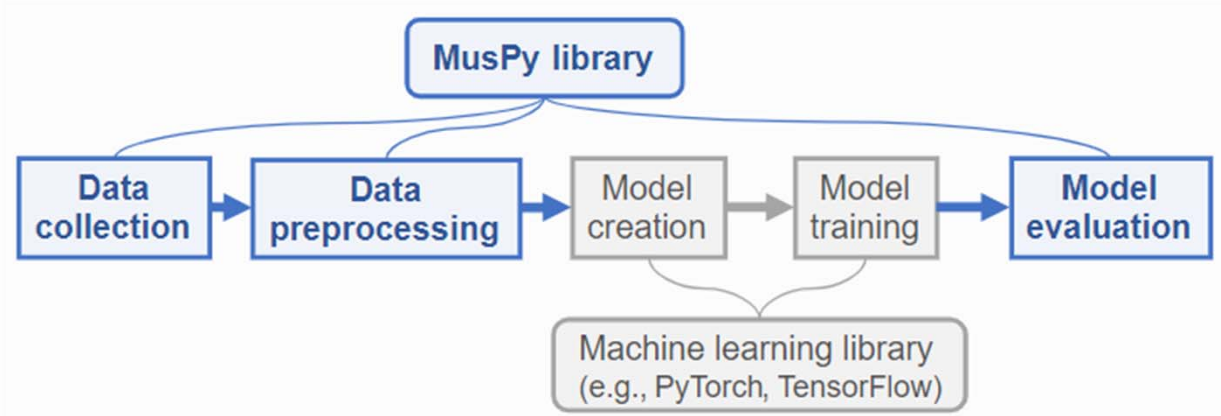
Library: MusPy

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

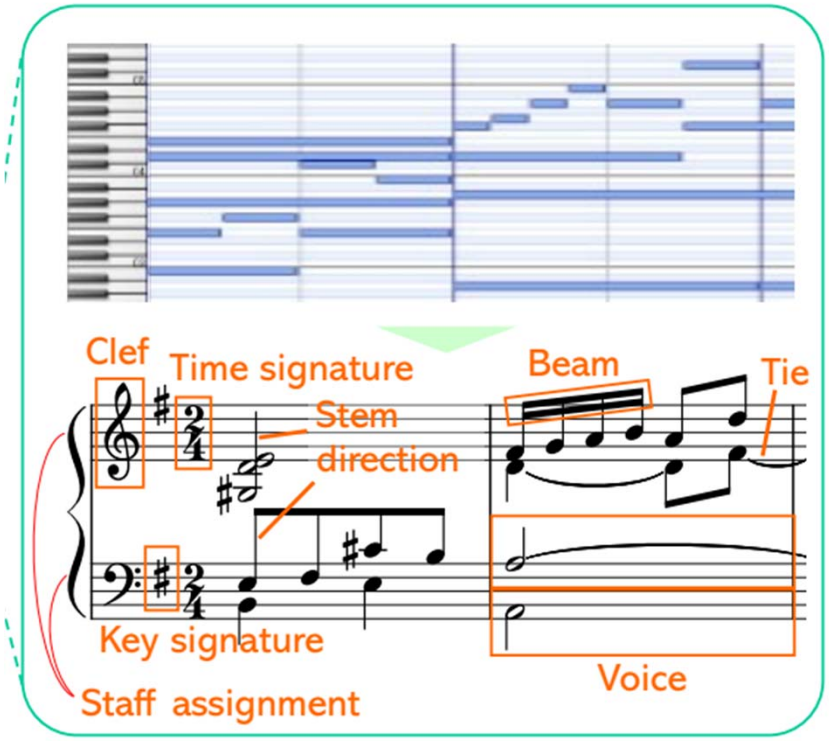
```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <type>half</type>
</note>
```

- 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

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**

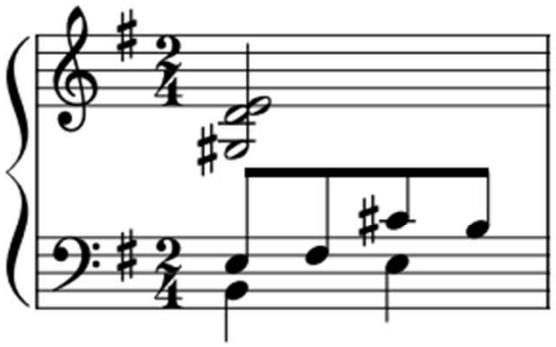


Ref: Suzuki, “Score Transformer: Generating musical score from note-level representation,” MMAsia 2021

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," MMAsia 2021

Score Transformer

Genre	# Pieces	# Systems	# Measures
Popular piano	7,161	105,132	364,129
Classical piano	354	10,124	40,496

Table 1: Symbols and their variations in the proposed score token representation.

Symbol	Example	Variations
Staff	*R*	*R/L*
Barline	*bar*	*bar*
Clef	*clef_treble*	*clef_{bass/treble}*
Key Signature	*key_flat_2*	*key_{sharp/flat/natural}_{1, 2, ..., 6}*
Time Signature	*time_4/4*	*time_{2/4, 3/4, 4/4, etc.}*
Voice	*<voice>*	*<voice>, </voice>*
Rest	*rest*	*rest*
Pitch	*note_C4*	*note_{A, B, ..., G}{##/#/b/bb/(none)}{0, 1, ..., 8}*
Duration	*len_1/2*	*len_{1/24, 1/16, ..., 4}*
Stem Direction	*stem_up*	*stem_{up/down}*
Beams	*beam_stop*	*beam_{start/stop/continue/partial-left/partial-right}_...*
Tie	*tie_start*	*tie_{start/continue/stop}*

- Expression symbols (e.g., articulations, dynamics, and ornaments) and repeat symbols (can be but) are not included

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

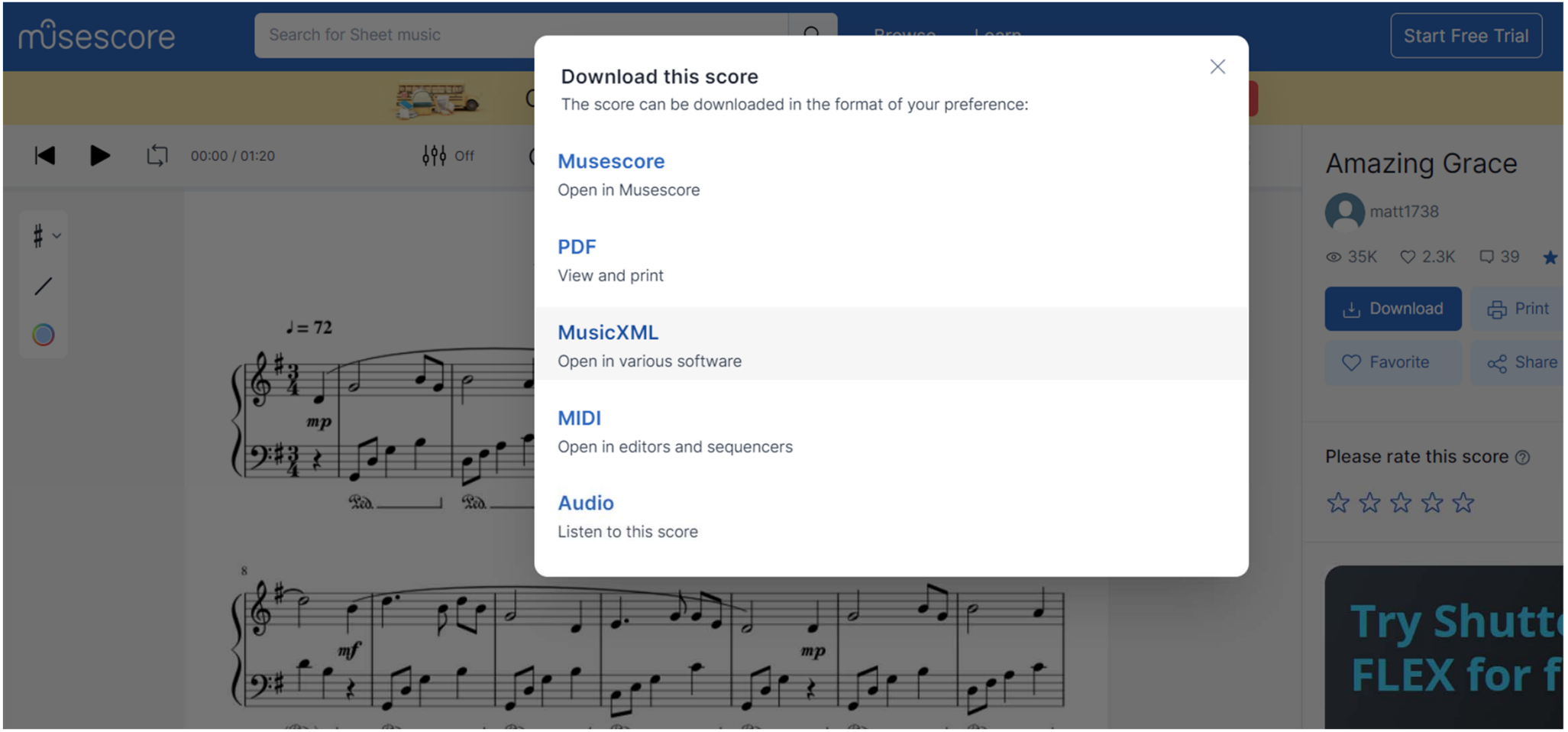
Score Transformer

Table 3: Overall error rates in % (measured based on the difference between the original and generated scores) for the popular piano dataset. Values in boldface show the lowest error rates. *"w/ABC"* uses ABC notation instead of our score token representation and the same for subsequent methods (see Section 3.2). For *Stem Direction*, models that do not specify directions are excluded from evaluation because the metric cannot measure them properly.

Method	Note Preservation		Note Segregation		Score Attributes			Note Attributes					Average
	Insertion	Deletion	Staff Assignment	Voice Separation	Clef	Key Signature	Time Signature	Duration	Spelling	Stem Direction	Beams	Tie	
CTD [4]	155.62	41.99	9.14	73.51	43.42	54.70	59.01	51.74	11.20	-	46.07	24.01	51.86
Finale 26	53.68	7.11	27.12	8.18	17.56	26.02	14.77	13.38	9.13	36.22	14.81	5.89	19.49
MuseScore 3	25.93	3.74	15.07	32.97	11.27	31.00	8.25	9.02	7.31	24.91	13.54	3.68	15.56
Score Transformer (ST)	**1.99**	**2.14**	0.91	**1.63**	1.26	**8.00**	0.88	0.49	**3.04**	2.57	**1.23**	2.07	**2.18**
w/ABC	14.05	18.35	0.90	6.25	1.64	9.07	1.12	19.41	4.74	-	13.06	2.91	8.32
w/Humdrum (row)	2.85	8.82	**0.63**	5.61	**0.98**	12.46	**0.65**	**0.48**	4.44	14.46	1.28	**1.78**	4.54
w/Humdrum (spine)	3.53	9.46	0.82	6.57	1.25	12.21	0.88	0.73	4.60	14.91	1.56	1.90	4.87
w/LilyPond	5.13	5.05	0.87	7.64	1.07	8.84	0.78	1.35	4.26	**2.50**	1.46	5.24	3.68

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

Software: MuseScore



Software: KernScores

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

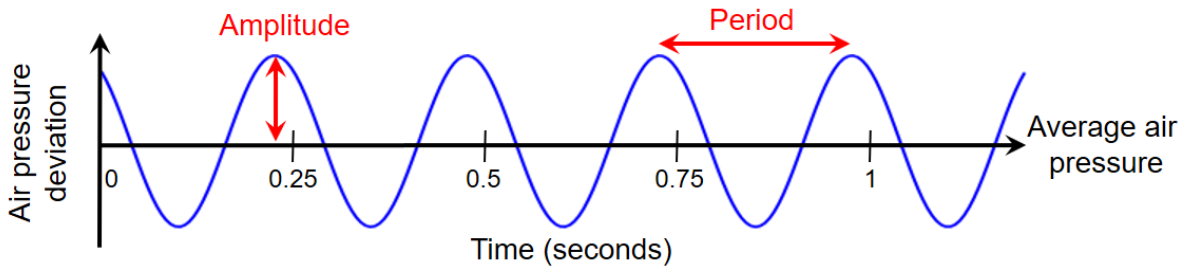
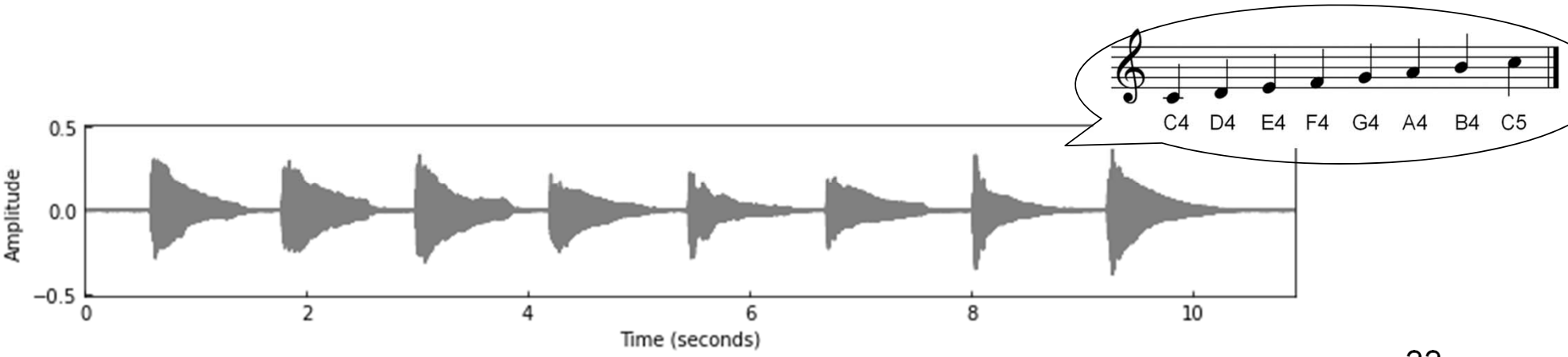
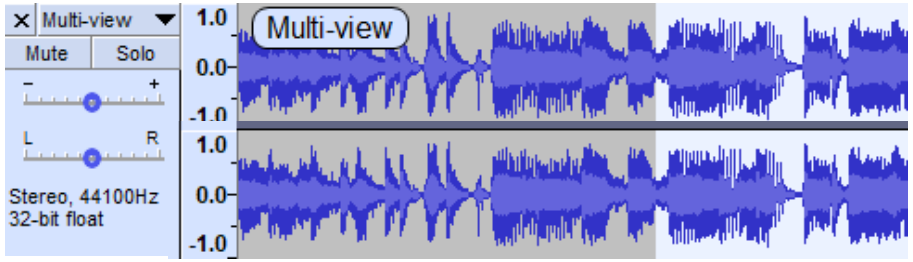


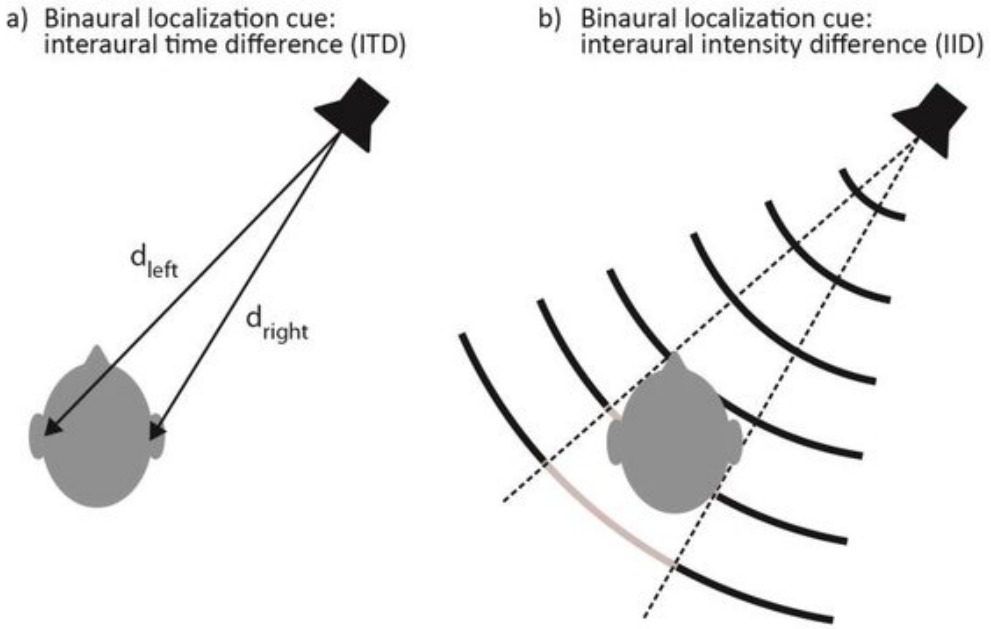
Figure 1.19 from [Müller, FMP, Springer 2015]



Mono vs Stereo

- Monaural
- Binaural





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

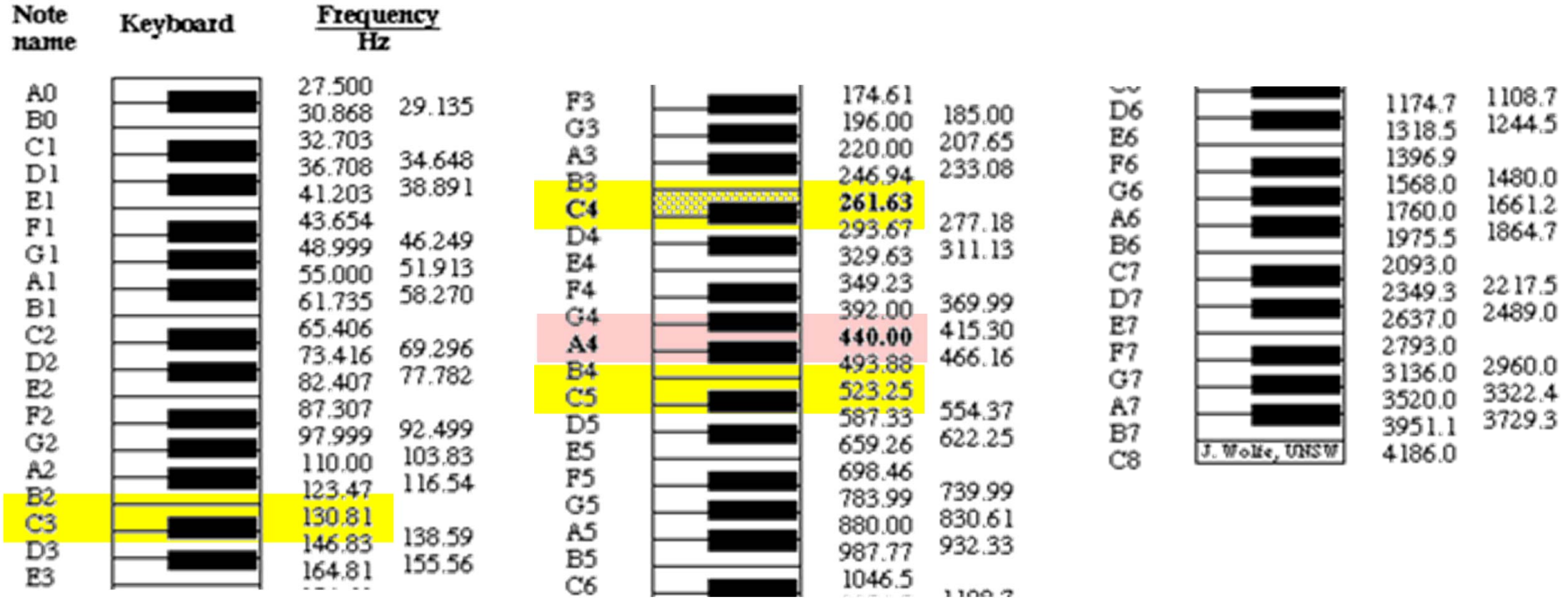
- Stereo-to-mono conversion
  - By taking the average
- Mono-to-stereo conversion
  - Need research
    https://www.youtube.com/watch?v=aWxmQKm_s8Q

Ref: Richard et al, "Neural synthesis of binaural speech from mono audio," ICLR 2021

Notes and Pitches

(Assuming A4=440 Hz; but there are other *tunings*)

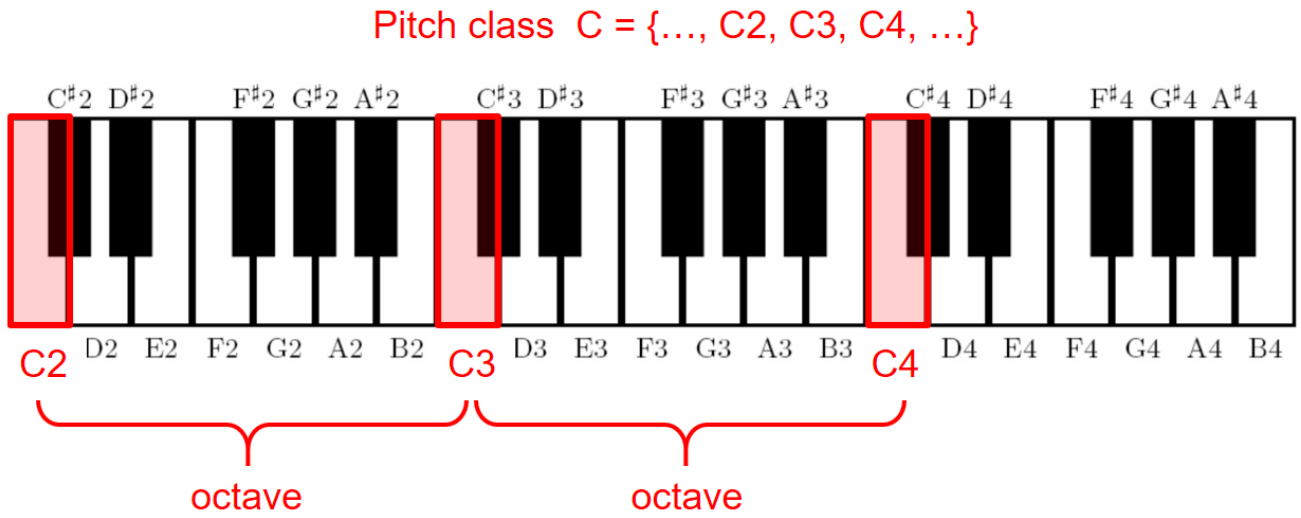
https://newt.phys.unsw.edu.au/jw/notes.html



Notes and Pitches

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

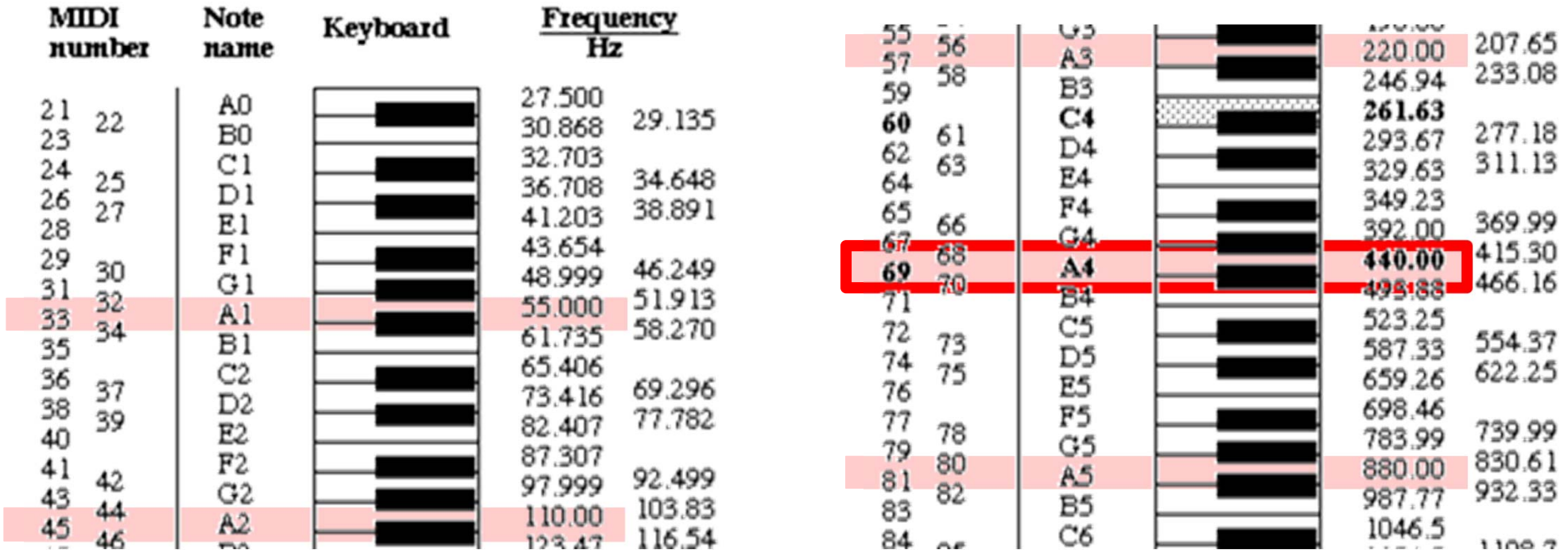
- **Pitch Class**
  - Two notes with *fundamental frequencies* in a ratio equal to *any power of* ***two*** (e.g., half, twice, or four times) are perceived as very **similar**
  - All notes with this kind of relation can be grouped under the same *pitch class*



MIDI Note Numbers

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

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

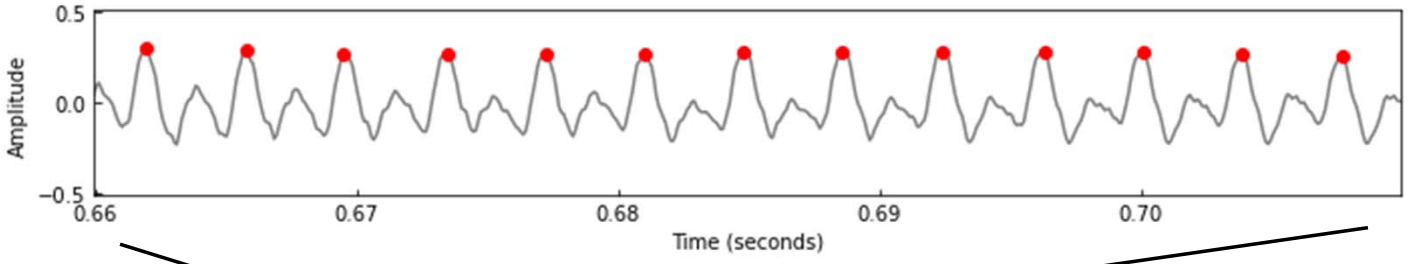


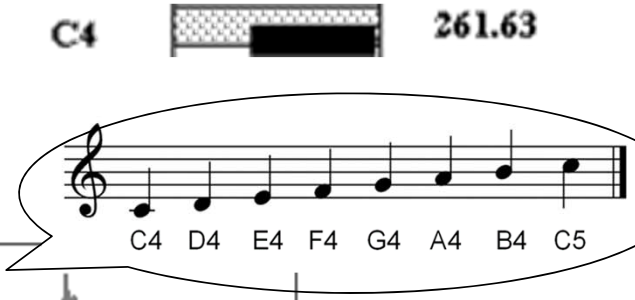
Audio Waveforms (Cont')

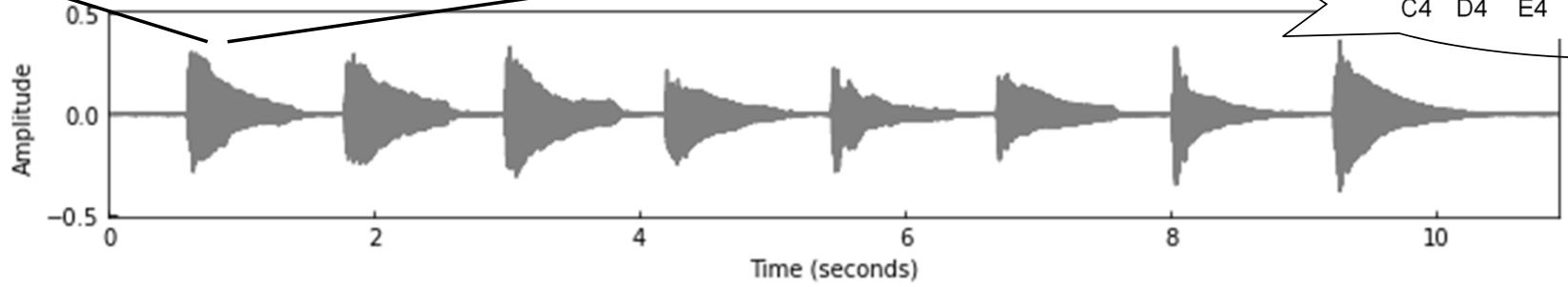
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Waveform.html

Zoom-in of the section between 0.66 and 0.71 seconds

- Highly repetitive
- 13 high-pressure (red) points → 20*13=260 Hz

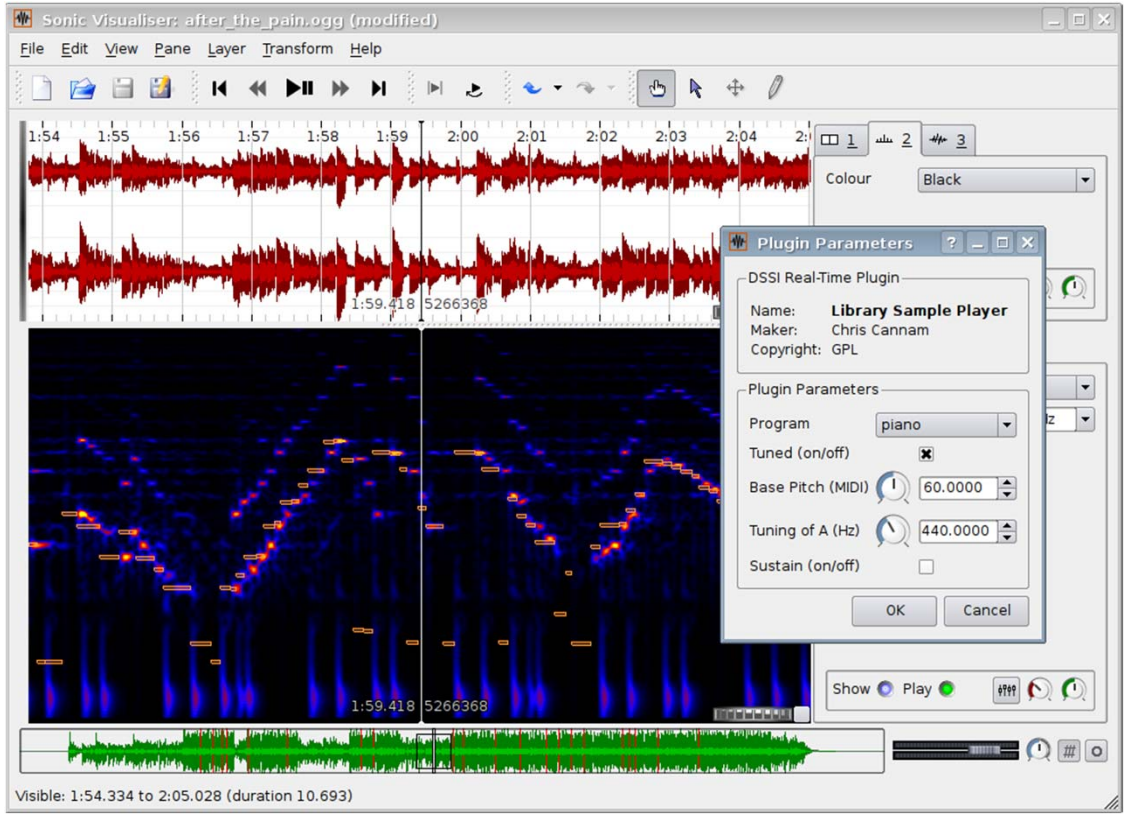






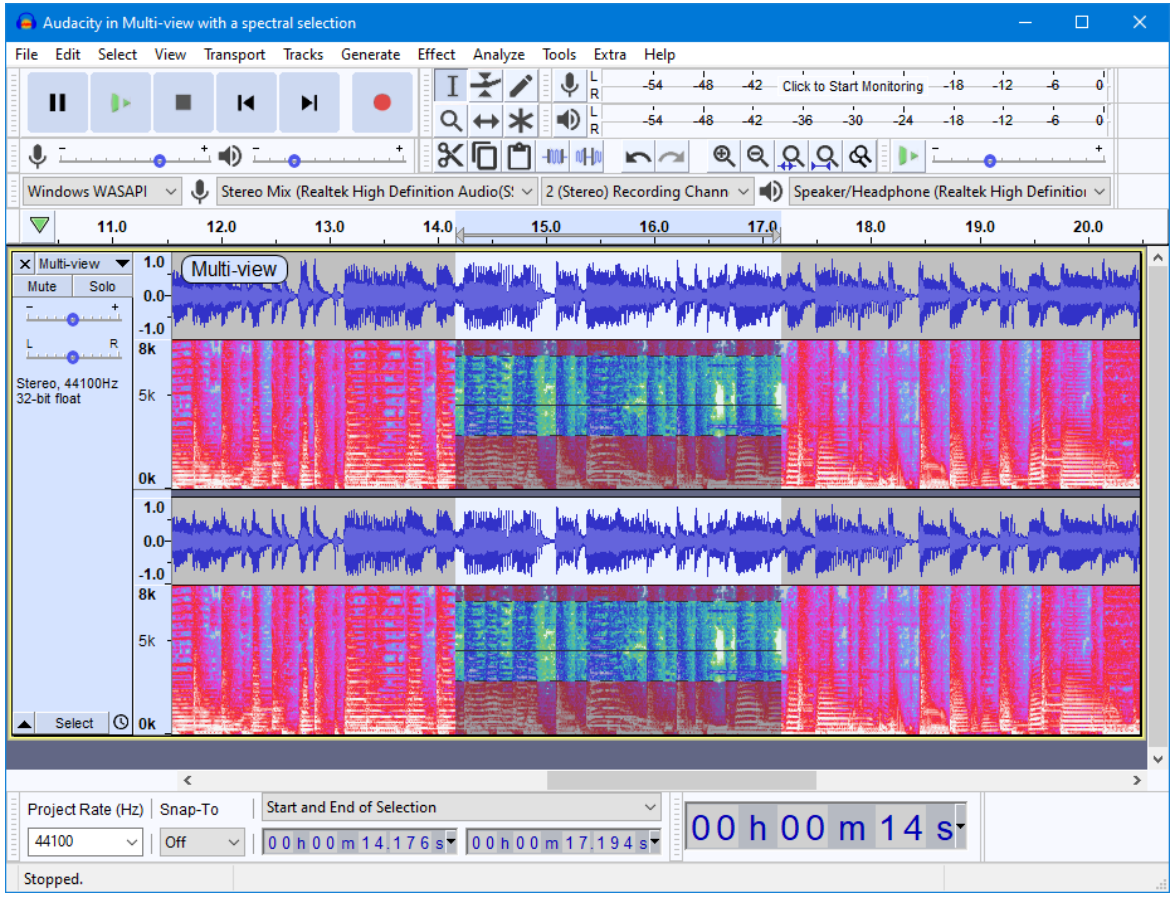
Software: Sonic Visualiser

http://www.sonicvisualiser.org/



Software: Audacity

https://www.audacityteam.org/

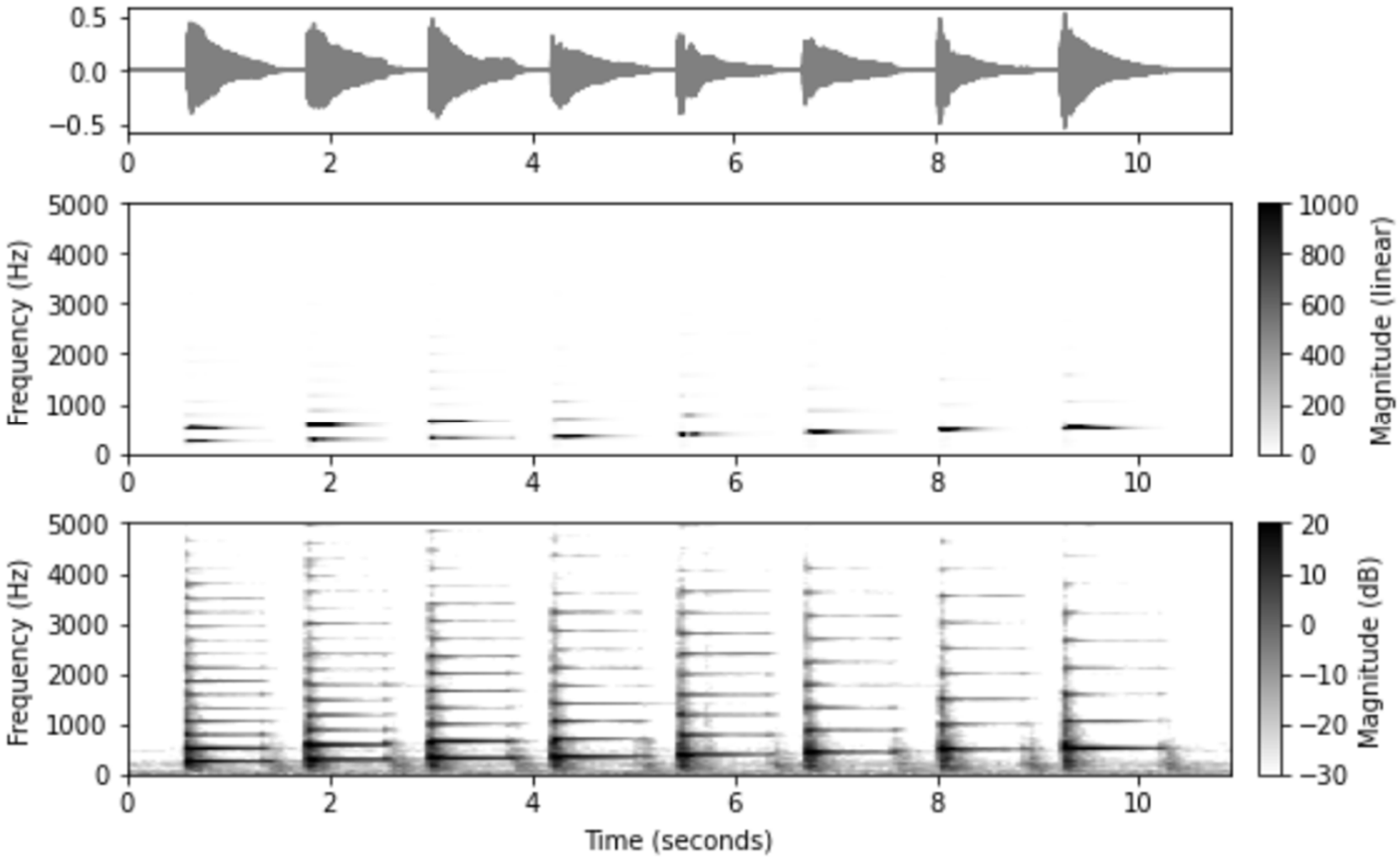


Let's Find Some Audio Recordings

- 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

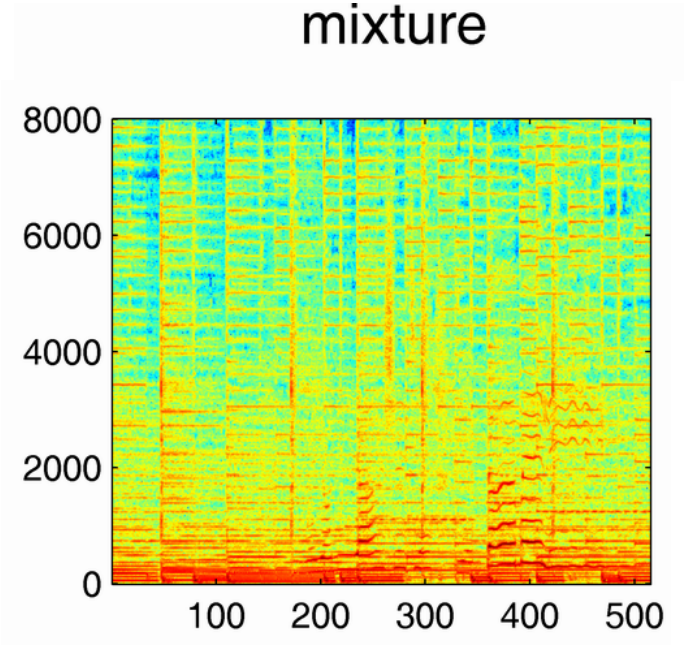
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2_STFT-Basic.html

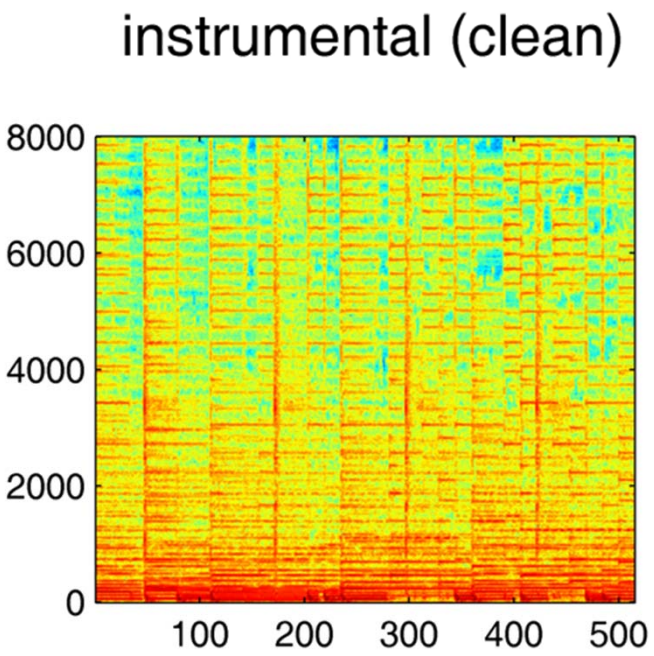


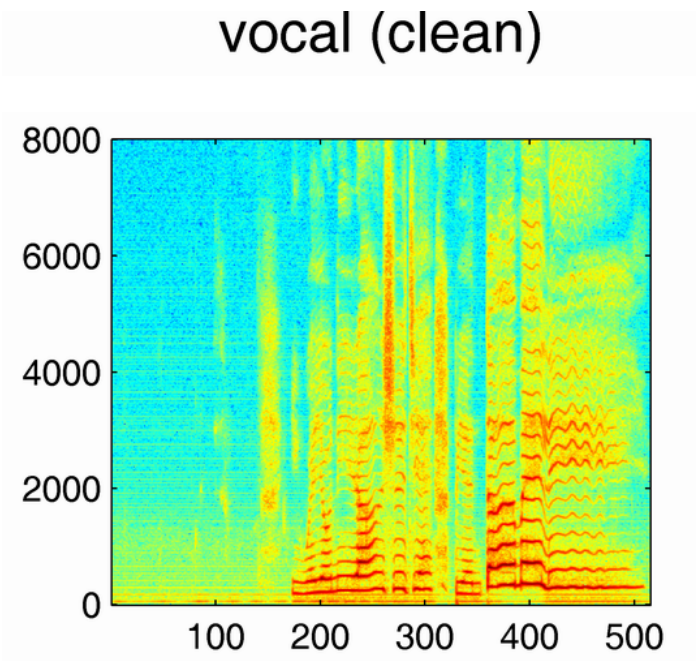
- **Time-frequency representation** of audio
  - Linear frequency, **linear** magnitude
  - Linear frequency, **logarithmic** magnitude (dB)

Discrete Short-Time Fourier Transform

- Time-frequency representation of audio



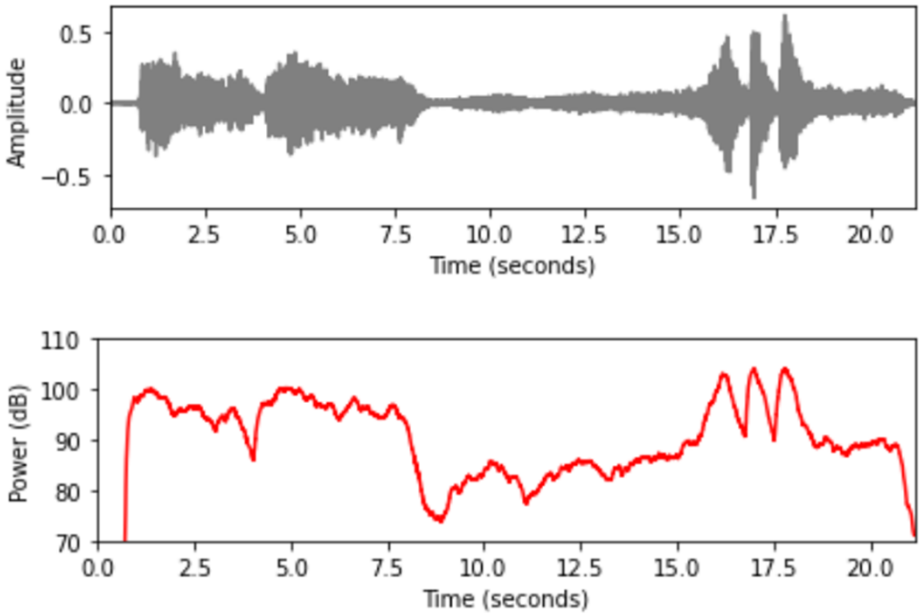


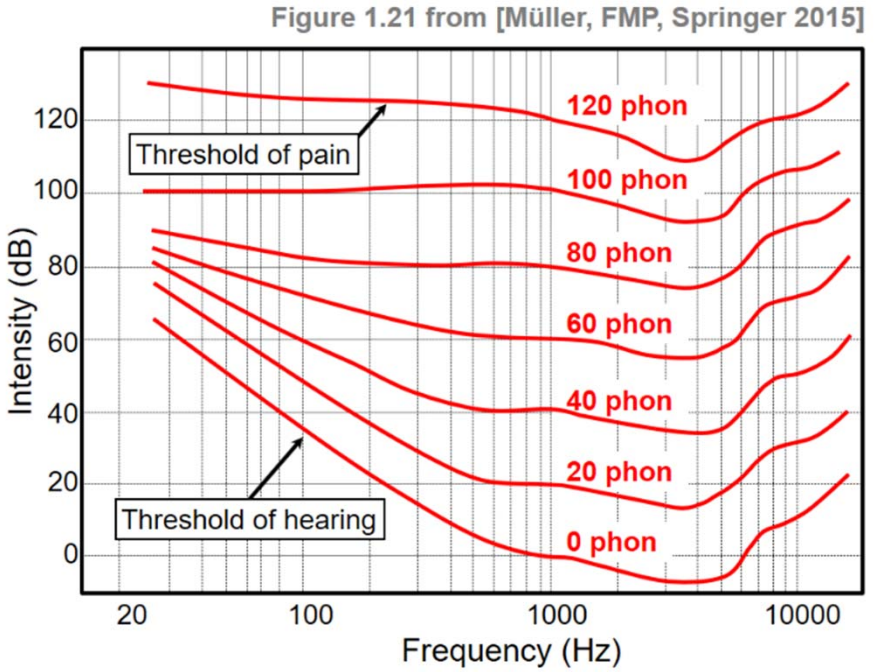


Dynamics, Intensity, and Loudness

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: 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")
```

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: LibROSA

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

Audio loading

- `load` (path, *[, sr, mono, offset, duration, ...])
- `stream` (path, *, block_length, frame_length, ...)
- `to_mono` (y)
- `resample` (y, *, orig_sr, target_sr[, ...])
- `get_duration` (*[, y, sr, S, n_fft, ...])
- `get_samplerate` (path)

Time-domain processing

- `autocorrelate` (y, *[, max_size, axis])
- `lpc` (y, *, order[, axis])
- `zero_crossings` (y, *[, threshold, ...])
- `mu_compress` (x, *[, mu, quantize])
- `mu_expand` (x, *[, mu, quantize])

Signal generation

- `clicks` (*[, times, frames, sr, hop_length, ...])
- `tone` (frequency, *[, sr, length, duration, phi])
- `chirp` (*, fmin, fmax[, sr, length, duration, ...])

Magnitude scaling

- `amplitude_to_db` (S, *[, ref, amin, top_db])
- `db_to_amplitude` (S_db, *[, ref])
- `power_to_db` (S, *[, ref, amin, top_db])
- `db_to_power` (S_db, *[, ref])
- `perceptual_weighting` (S, frequencies, *[, kind])
- `frequency_weighting` (frequencies, *[, kind])
- `multi_frequency_weighting` (frequencies, *[, ...])
- `A_weighting` (frequencies, *[, min_db])
- `B_weighting` (frequencies, *[, min_db])

Library: LibROSA

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

Time unit conversion

- `frames_to_samples` (frames, *[, hop_length, n_fft])
- `frames_to_time` (frames, *[, sr, hop_length, ...])
- `samples_to_frames` (samples, *[, hop_length, ...])
- `samples_to_time` (samples, *[, sr])
- `time_to_frames` (times, *[, sr, hop_length, n_fft])
- `time_to_samples` (times, *[, sr])
- `blocks_to_frames` (blocks, *, block_length)
- `blocks_to_samples` (blocks, *, block_length, ...)
- `blocks_to_time` (blocks, *, block_length, ...)

Frequency unit conversion

- `hz_to_note` (frequencies, **kwargs)
- `hz_to_midi` (frequencies)
- `midi_to_hz` (notes)
- `midi_to_note` (midi, *[, octave, cents, key, ...])
- `note_to_hz` (note, **kwargs)
- `note_to_midi` (note, *[, round_midi])

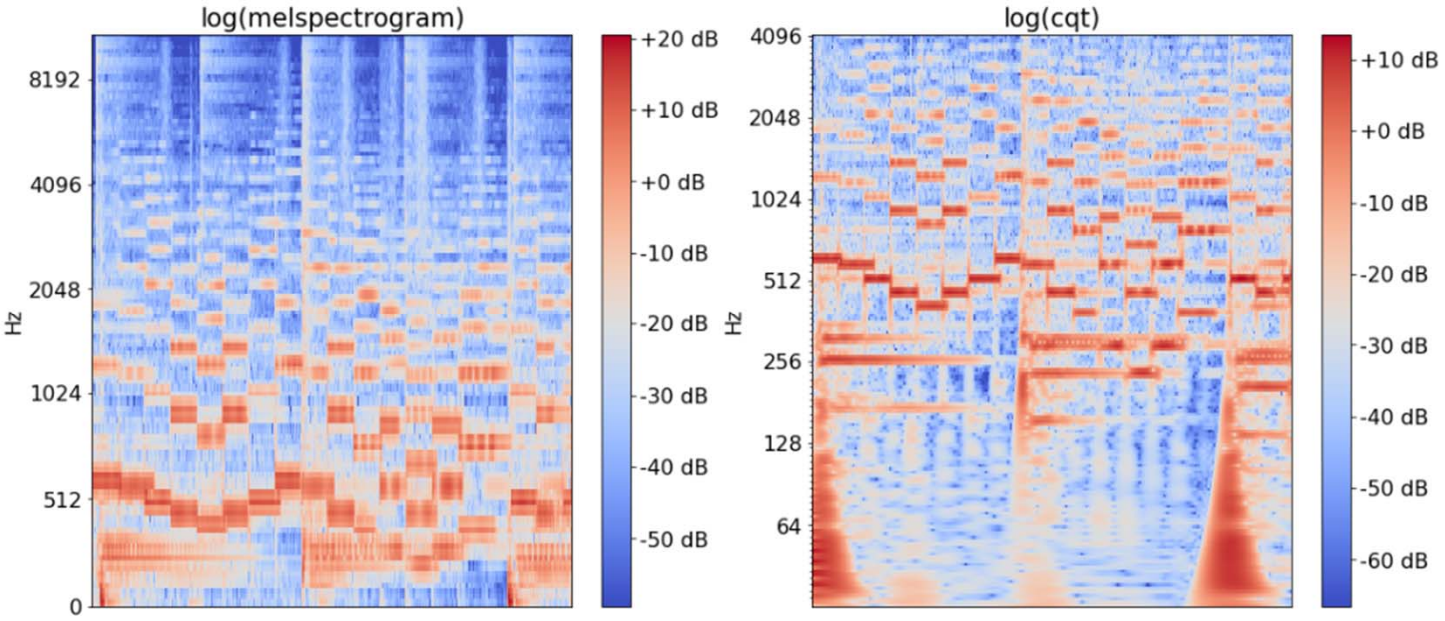
Music notation

- `key_to_notes` (key, *[, unicode])
- `key_to_degrees` (key)

Constant-Q Transform (CQT)

https://music-classification.github.io/tutorial/part2_basics/input-representations.html

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

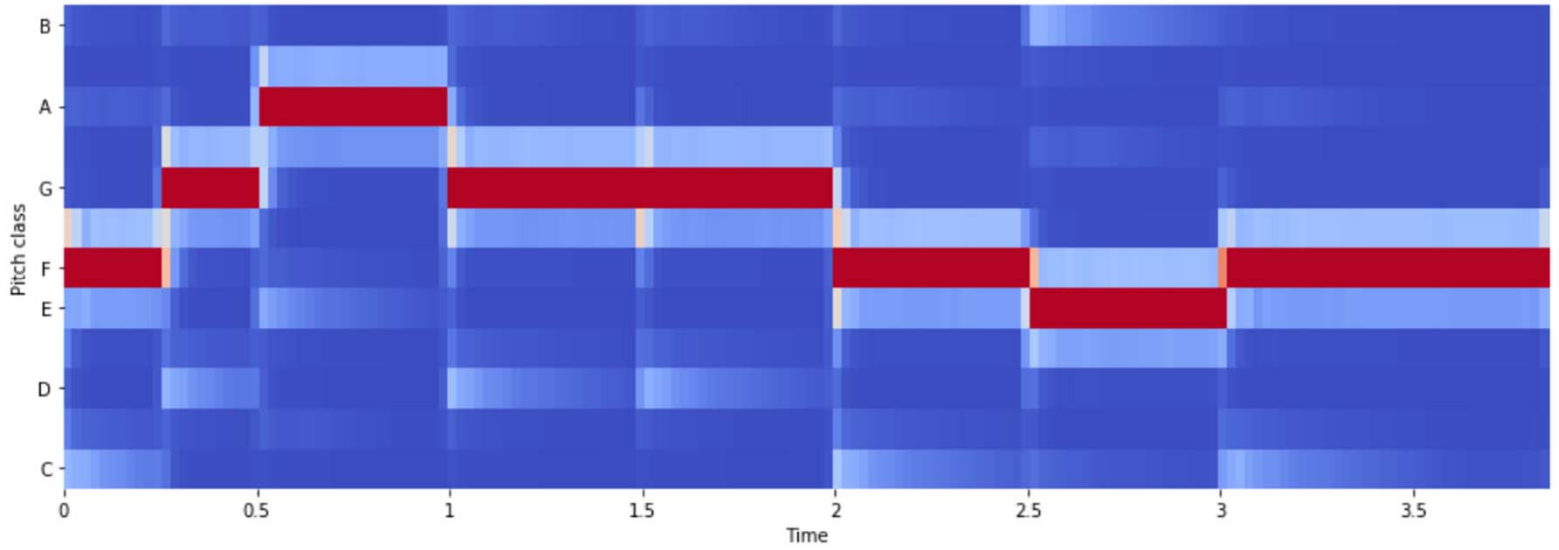


- **Logarithmic** frequency, **logarithmic** magnitude

Pitch Class Profile / Chromagram

https://musicinformationretrieval.com/chroma.html

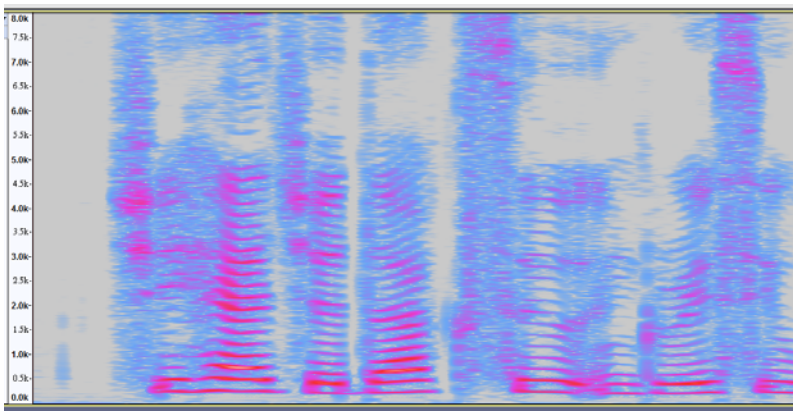
- "A **chroma vector** (Wikipedia) is a typically a 12-element feature vector indicating how much energy of each pitch class, {C, C#, D, D#, E, ..., B}, is present in the signal"
  - i.e., ignore octaves



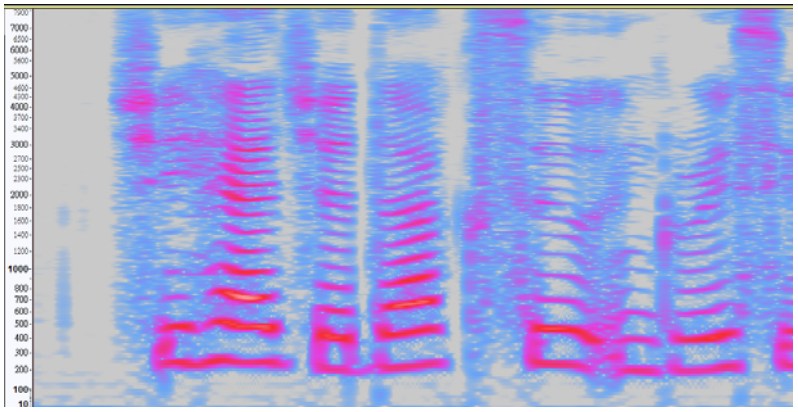
Mel-Spectrogram

- 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

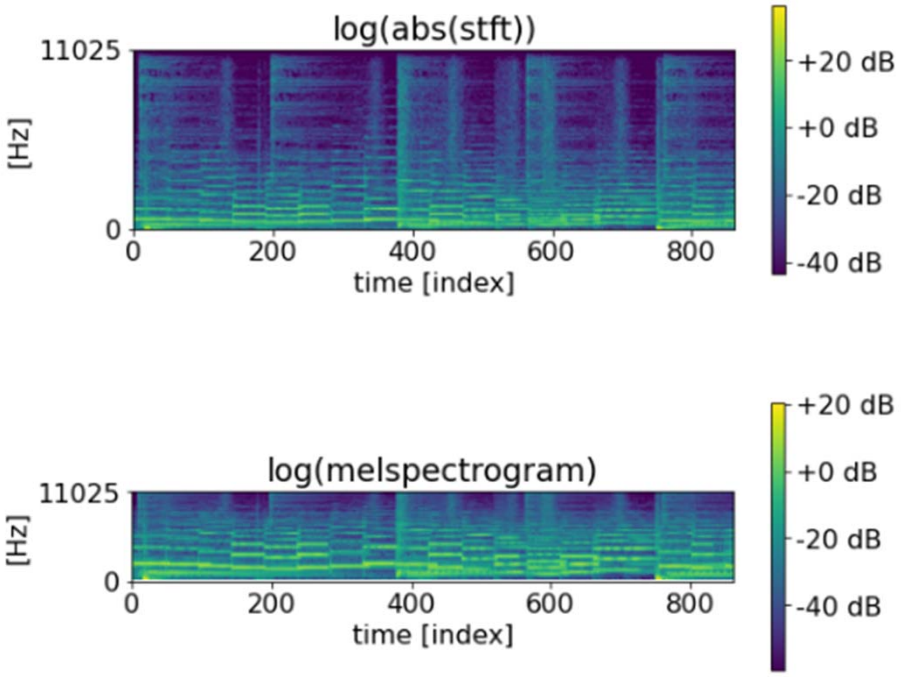


mel scale



Mel-Spectrogram

https://music-classification.github.io/tutorial/part2_basics/input-representations.html



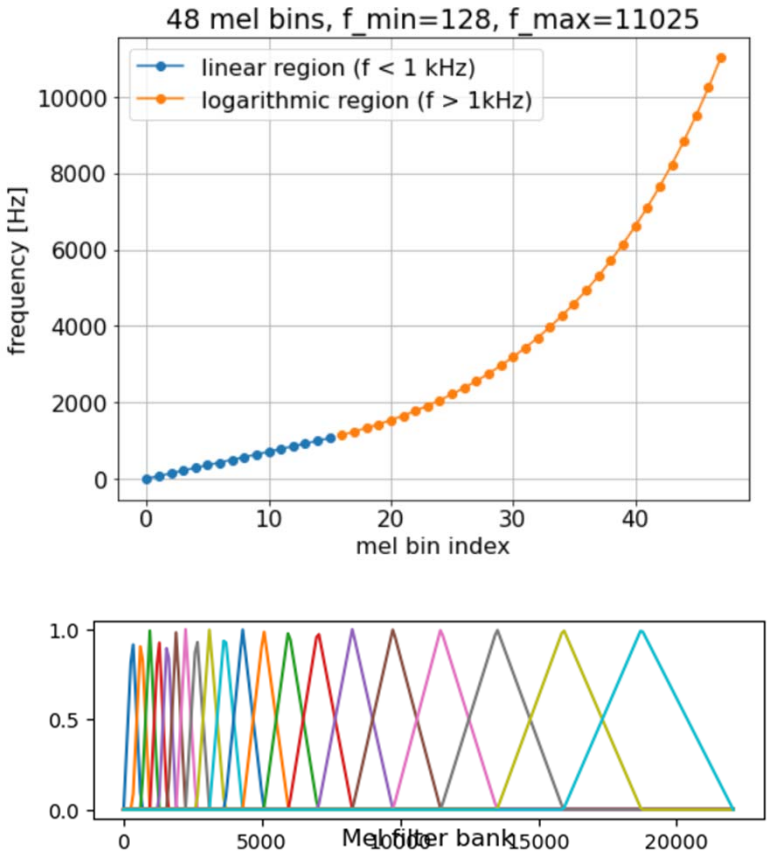


Figure from librosa

Library: LibROSA

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

Spectral representations

- `stft` (y, *[, n_fft, hop_length, win_length, ...])
- `istft` (stft_matrix, *[, hop_length, ...])
- `cqt` (y, *[, sr, hop_length, fmin, n_bins, ...])
- `icqt` (C, *[, sr, hop_length, fmin, ...])

Harmonics

- `interp_harmonics` (x, *, freqs, harmonics[, ...])
- `salience` (S, *, freqs, harmonics[, weights, ...])
- `f0_harmonics` (x, *, f0, freqs, harmonics[, ...])
- `phase_vocoder` (D, *, rate[, hop_length, n_fft])

Spectral features

- `chroma_stft` (*[, y, sr, S, norm, n_fft, ...])
- `chroma_cqt` (*[, y, sr, C, hop_length, fmin, ...])
- `chroma_cens` (*[, y, sr, C, hop_length, fmin, ...])
- `chroma_vqt` (*[, y, sr, V, hop_length, fmin, ...])
- `melspectrogram` (*[, y, sr, S, n_fft, ...])
- `mfcc` (*[, y, sr, S, n_mfcc, dct_type, norm, ...])
- `rms` (*[, y, S, frame_length, hop_length, ...])
- `spectral_centroid` (*[, y, sr, S, n_fft, ...])

Pitch and tuning

- `pyin` (y, *, fmin, fmax[, sr, frame_length, ...])
- `yin` (y, *, fmin, fmax[, sr, frame_length, ...])
- `estimate_tuning` (*[, y, sr, S, n_fft, ...])

Rhythm features

- `tempo` (*[, y, sr, onset_envelope, tg, ...])
- `tempogram` (*[, y, sr, onset_envelope, ...])

Feature Extraction

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

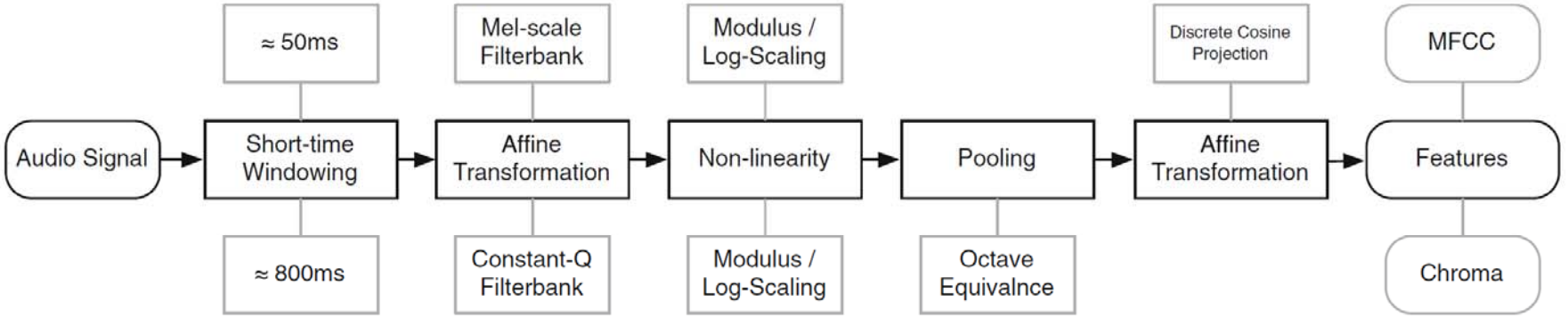


Fig. 3 *State of the art*: standard approaches to feature extraction proceed as the cascaded combination of a few simpler operations; on closer inspection, the main difference between chroma and MFCCs is the parameters used

Ref: Humphrey et al, "Feature learning and deep architectures: new directions for music informatics," JIIS 2013

Feature Learning by Convolutional Layers

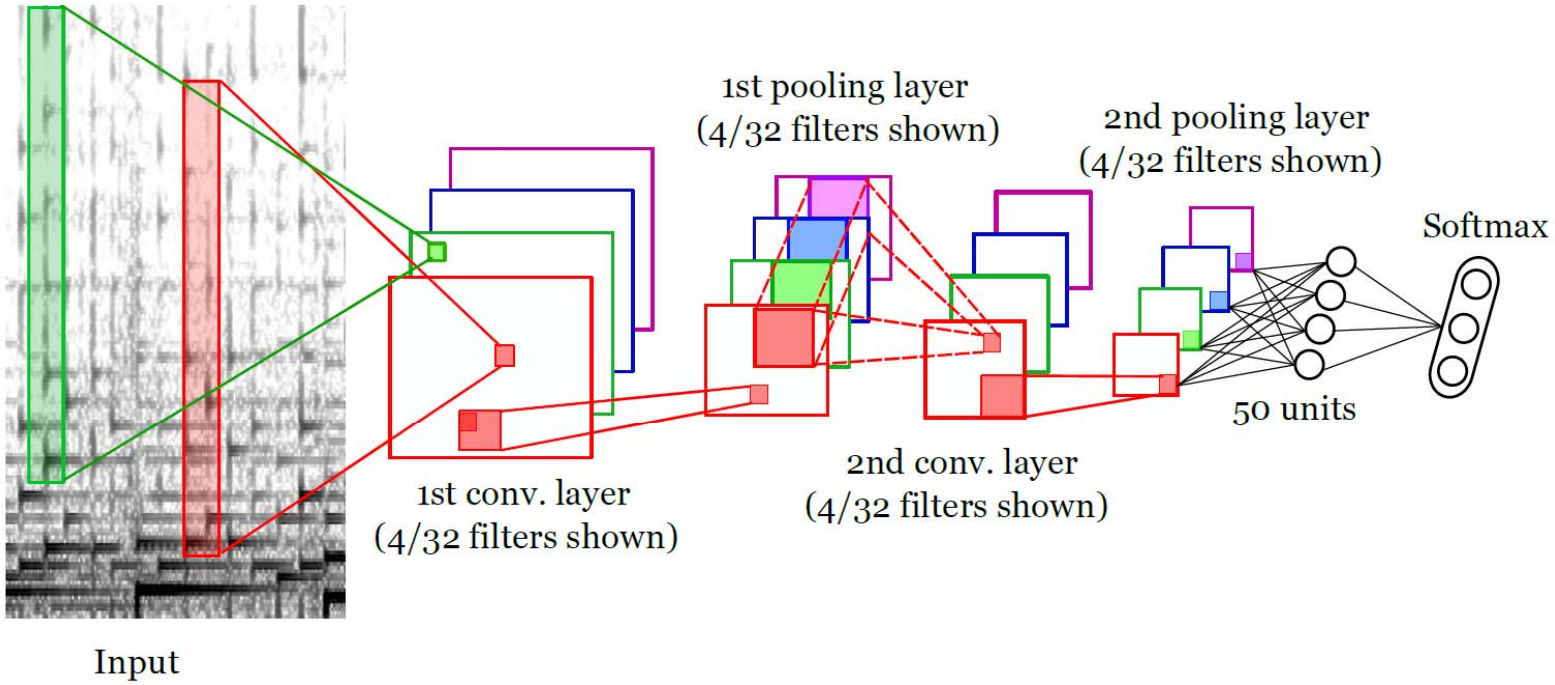


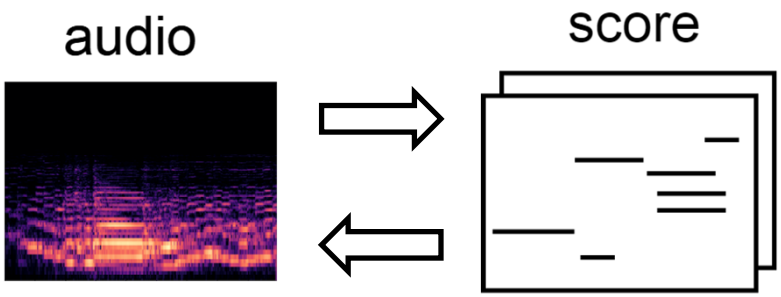
Fig. 1. Illustration of the CDNN architecture we use for our experiments. The CDNN first applies narrow vertical filters to the input sonogram (left) to capture harmonic structure. Then, it applies 32 different filters in the first convolutional layer (we show only 4). This is followed by the first max-pooling 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.

Ref: Kereliuk et al, "Deep learning and music adversaries," IEEE Trans. Multimedia, 2015

Summary

- Audio representation
  - Waveforms
  - Spectrograms

Music AI Research



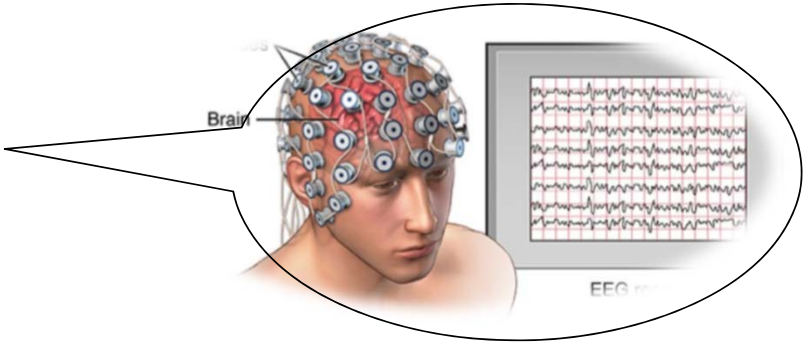
- Topics
  - **audio → audio**: signal processing
  - **audio → score**: transcription (similar to automatic speech recognition)
  - **score → score**: composition
  - **score → audio**: synthesis (similar to text-to-speech)
  - **audio → knowledge**: audio analysis
  - **score → knowledge**: symbolic-domain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

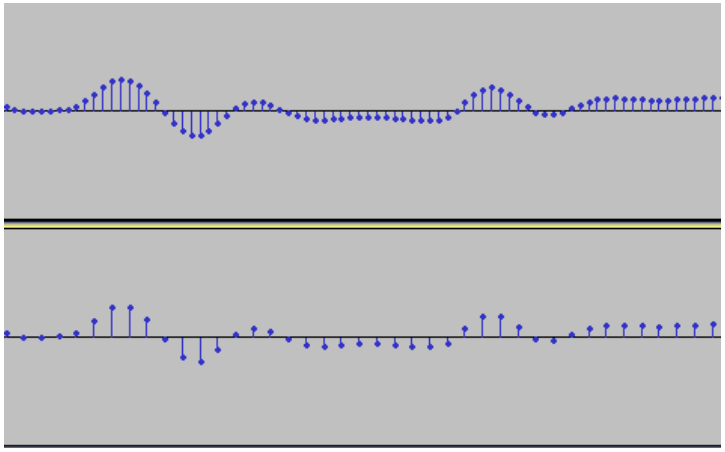


https://www.brightbraincentre.co.uk/electroencephalogram-eeg-brainwaves/

- 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'

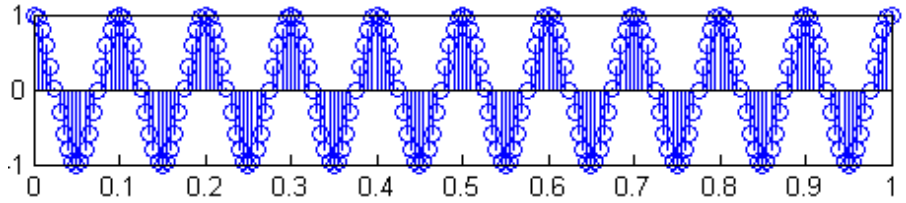
Sampling Rate (Cont')

- 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;
  - f0 = 10; % frequency
  - a = 1; % amplitude
  - **y = a*sin(2*pi*f0*t + pi/2);**
  - stem(t,y)



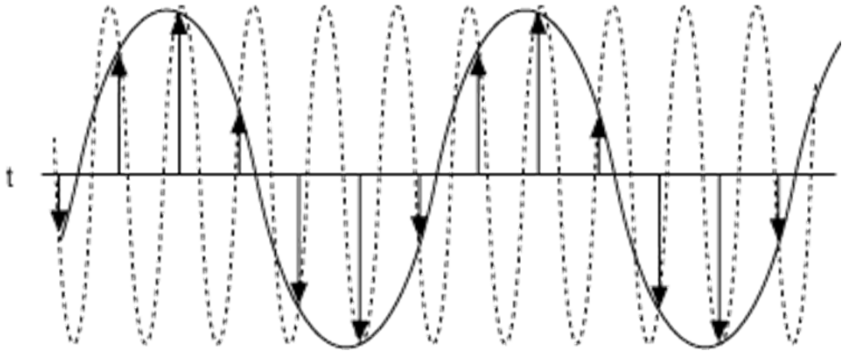
- Why?

  - sin(2*pi*f0*t + pi/2) = 1, when t = 1/f0, 2/f0, 3/f0, 4/f0, …

  - frequency = inverse of the period

Nyquist–Shannon Sampling Theorem

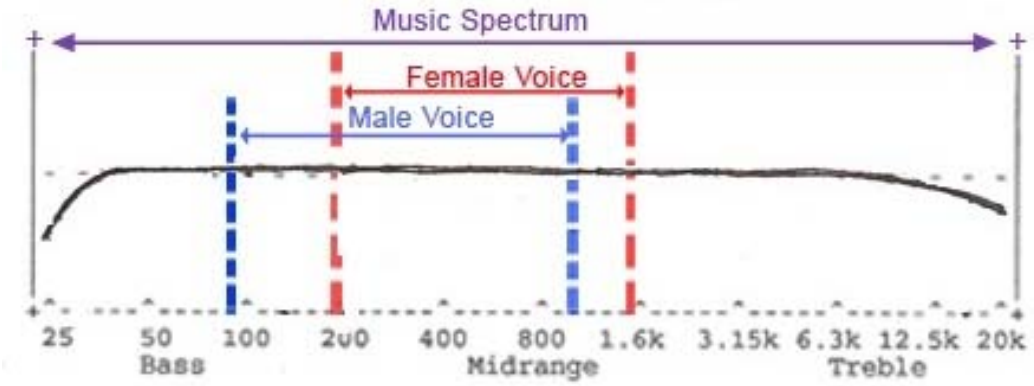
- A signal must be sampled at least **twice** as fast as the bandwidth of the signal to accurately reconstruct the waveform; otherwise, the high-frequency content will alias at a frequency inside the spectrum of interest
- **Sampling freq > 2* the highest freq in the signal**



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 sr=128 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

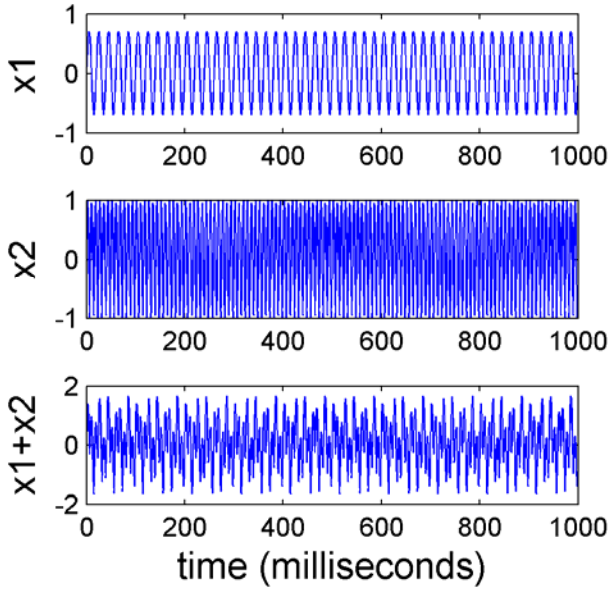
Fourier Transform

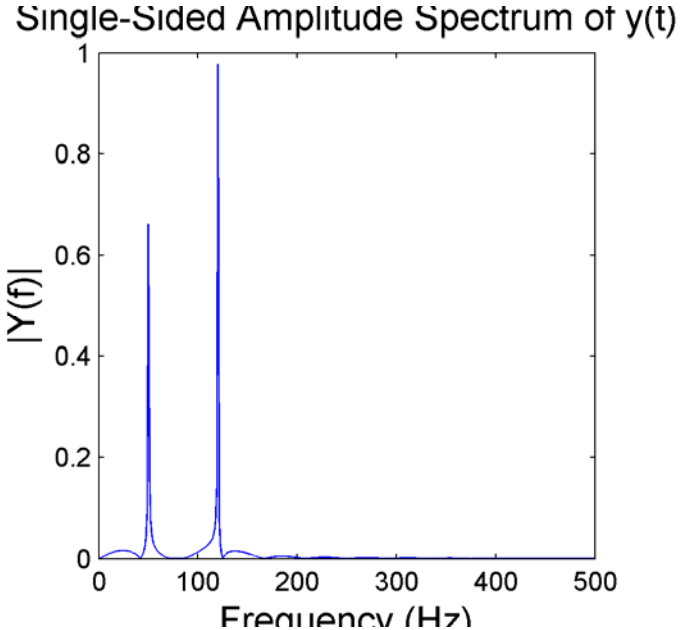
- To get the spectrum of a signal

- MATLAB code
  - https://www.mathworks.com/help/matlab/ref/fft.html
  - doc fft
  - Y = abs(fft(y));

Fourier Transform

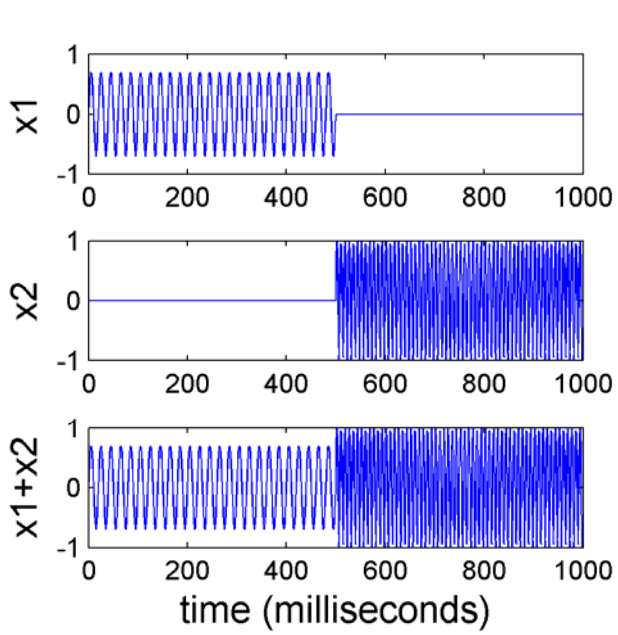
- MATLAB code
  - x1 = 0.7*sin(2*pi*50*t);
  - x2 = sin(2*pi*120*t);

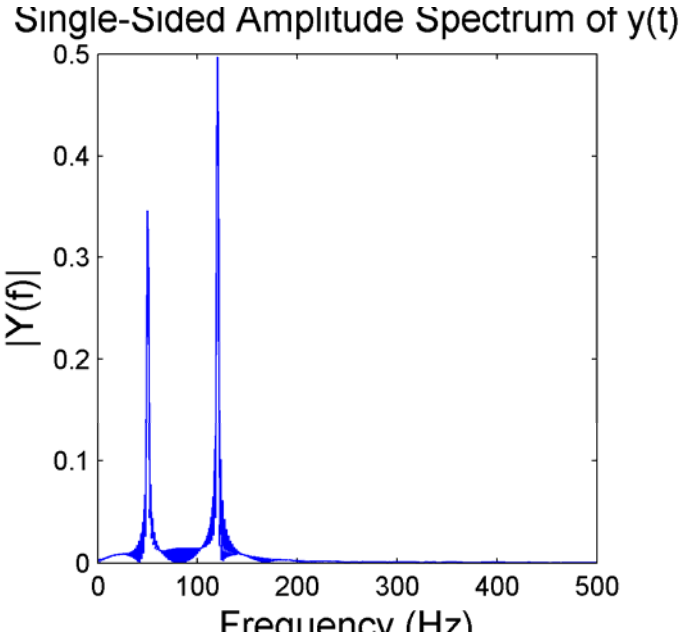




Fourier Transform

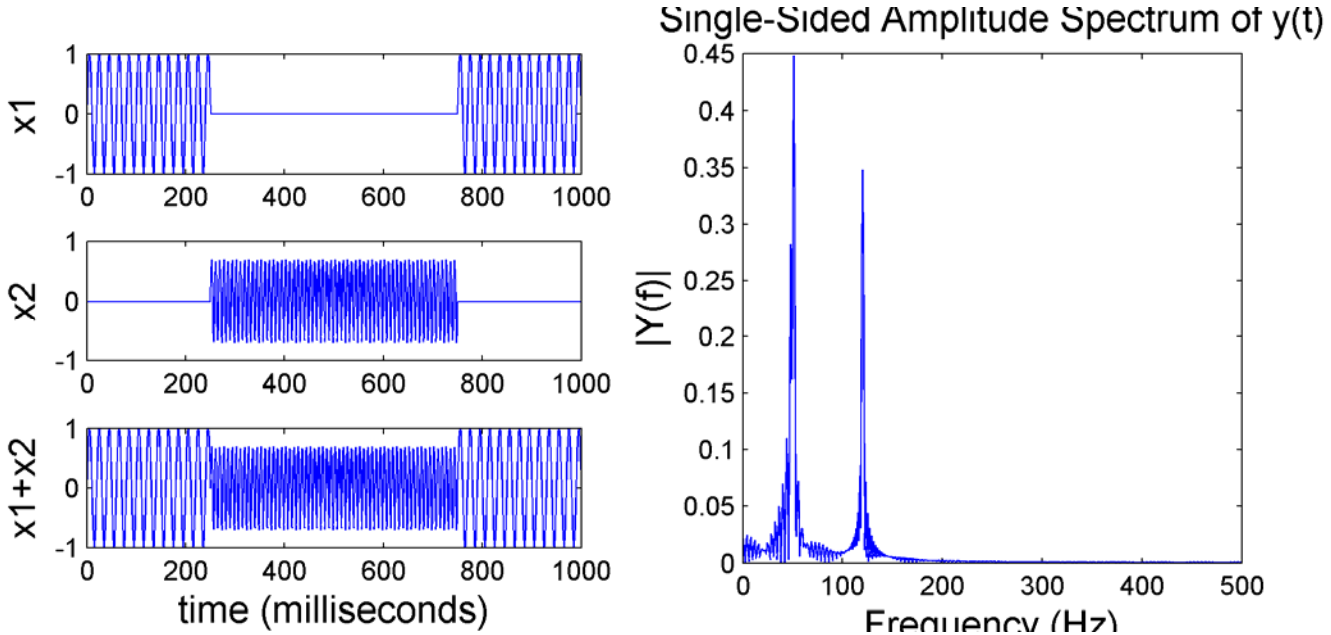
- Problem: cannot “localize” signal of interest





Fourier Transform

- Problem: cannot “localize” signal of interest

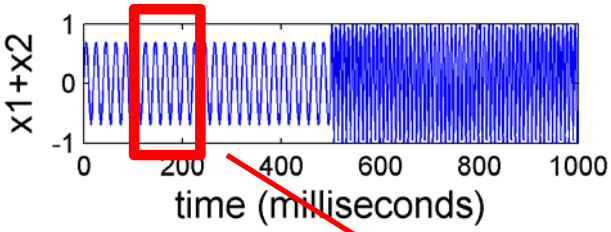


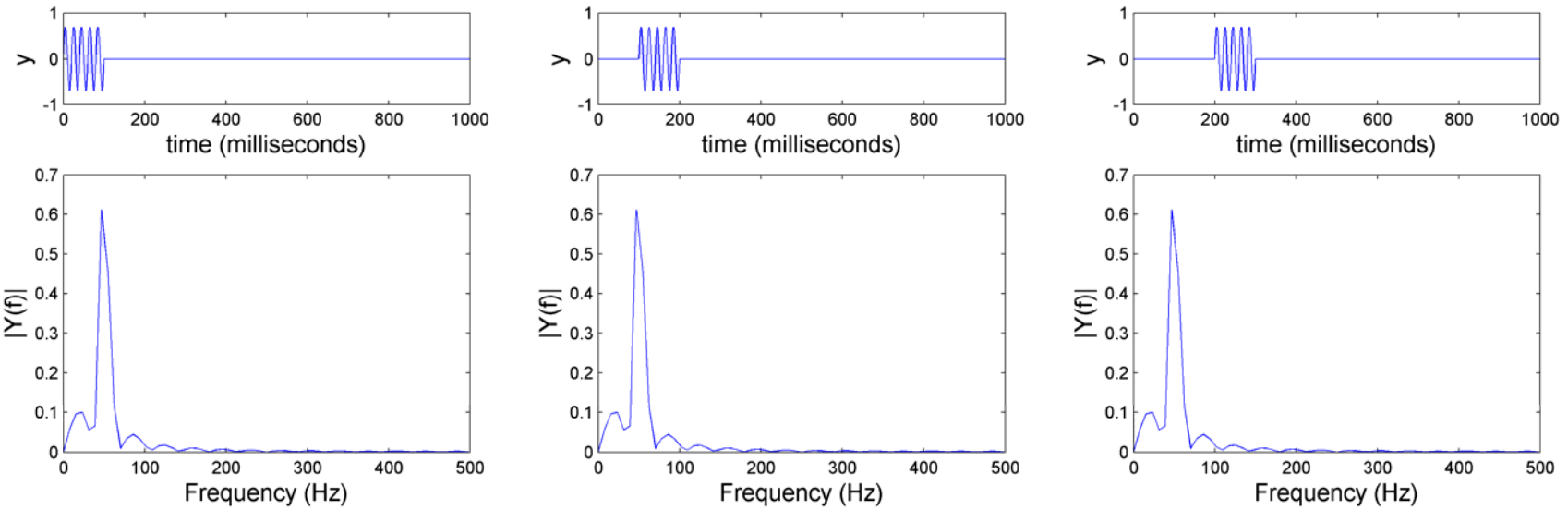
Short Time Fourier Transform (STFT)

- "**Windowed**" version of the Fourier Transform
- Output: a **time-frequency representation**

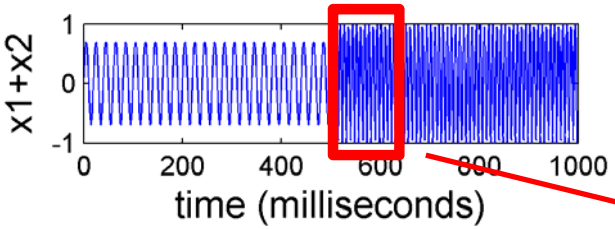
- MATLAB code
  - https://www.mathworks.com/help/signal/ref/spectrogram.html
  - doc spectrogram
  - spectrogram(y,window,noverlap,nfft)
  - spectrogram(y,100,50,100,sr,'yaxis')

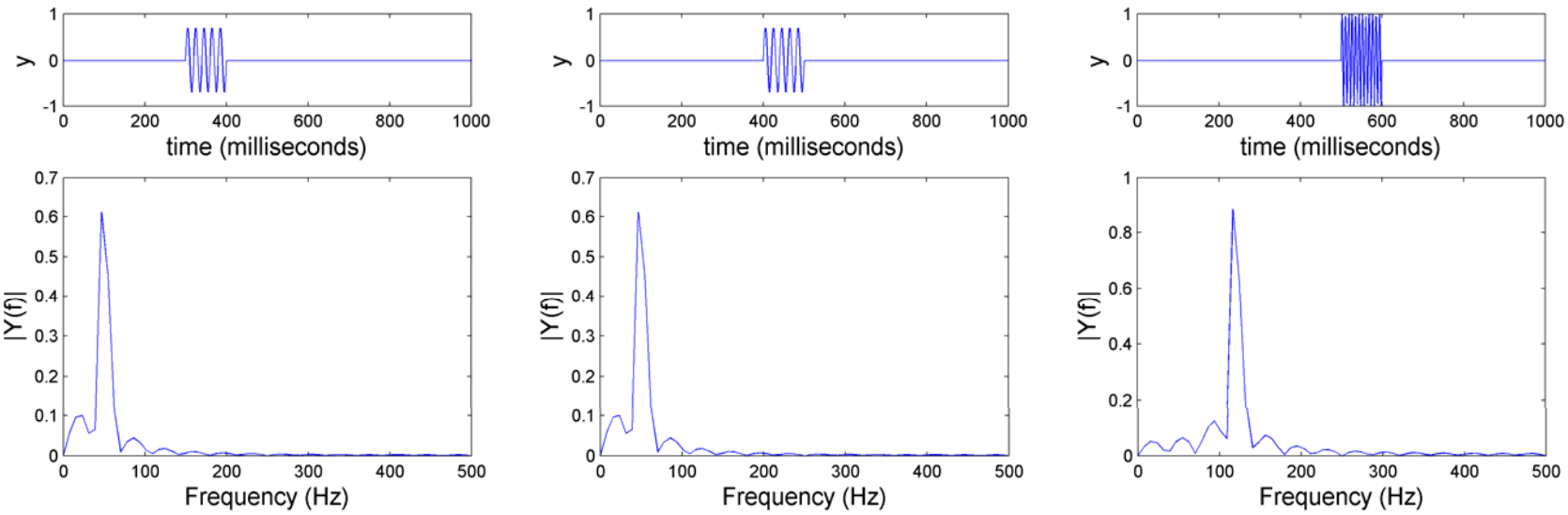
Short Time Fourier Transform (STFT)





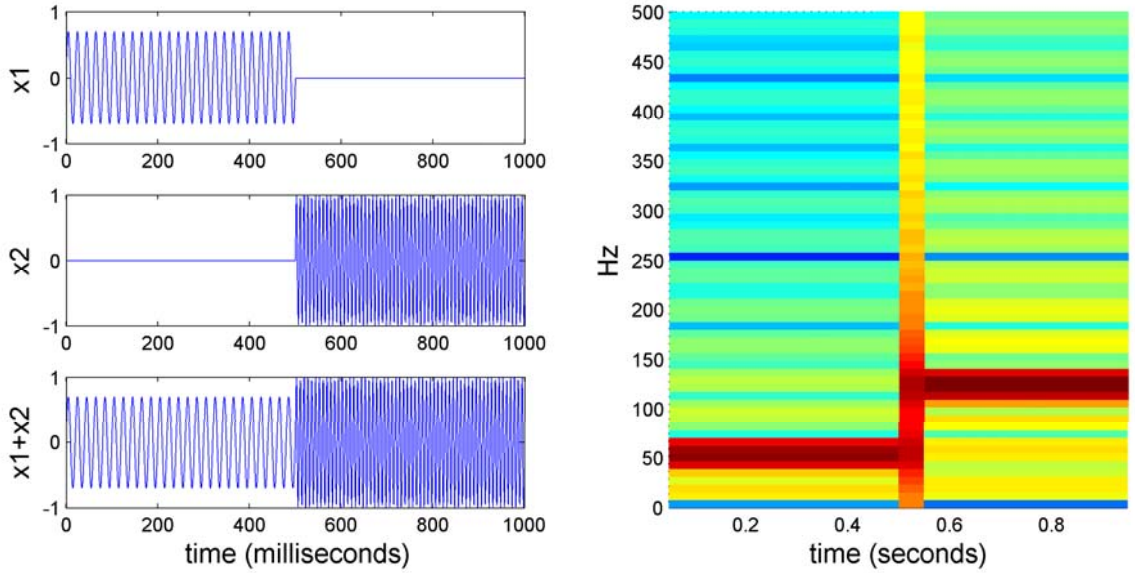
Short Time Fourier Transform (STFT)





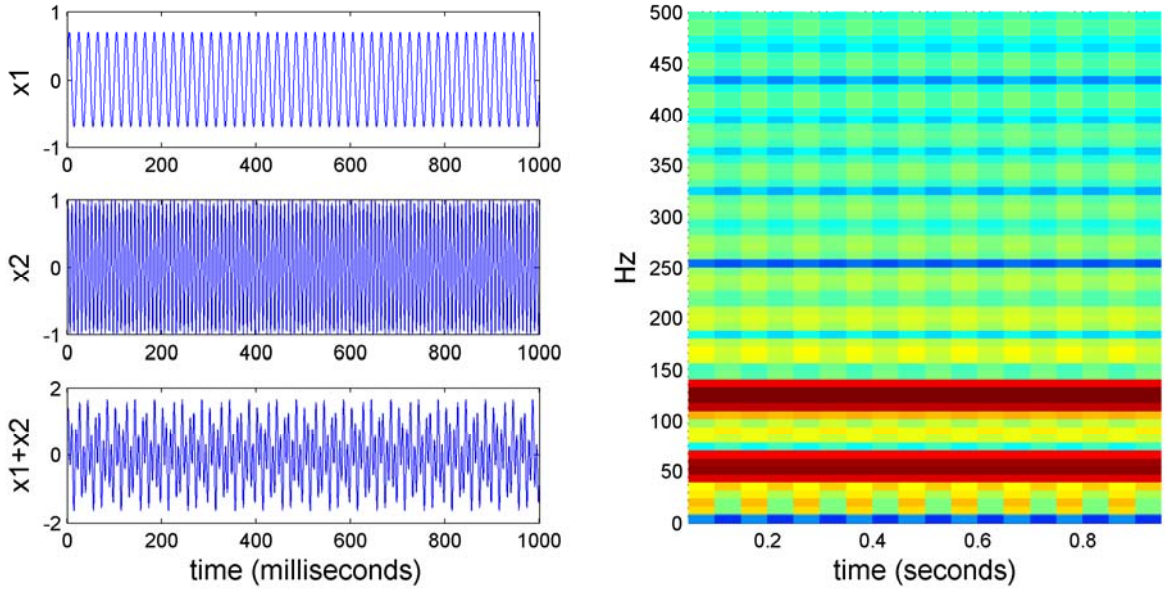
Short Time Fourier Transform (STFT)

- window size = 100



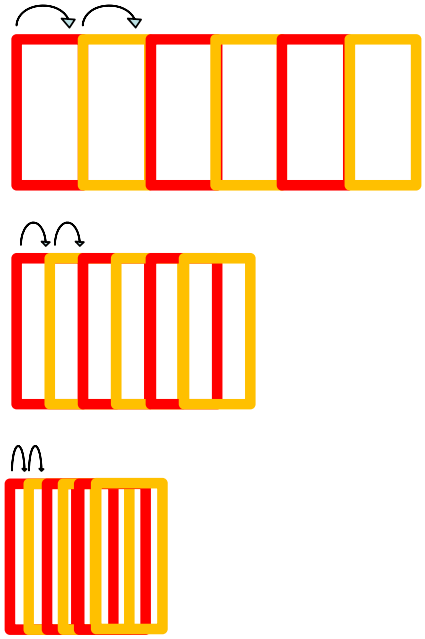
Short Time Fourier Transform (STFT)

- window size = 100



Short Time Fourier Transform (STFT)

- Hop size
  - Can be proportional to the window size
    - hop_size = win_size
    - hop_size = 0.5*win_size
    - hop_size = 0.1*win_size

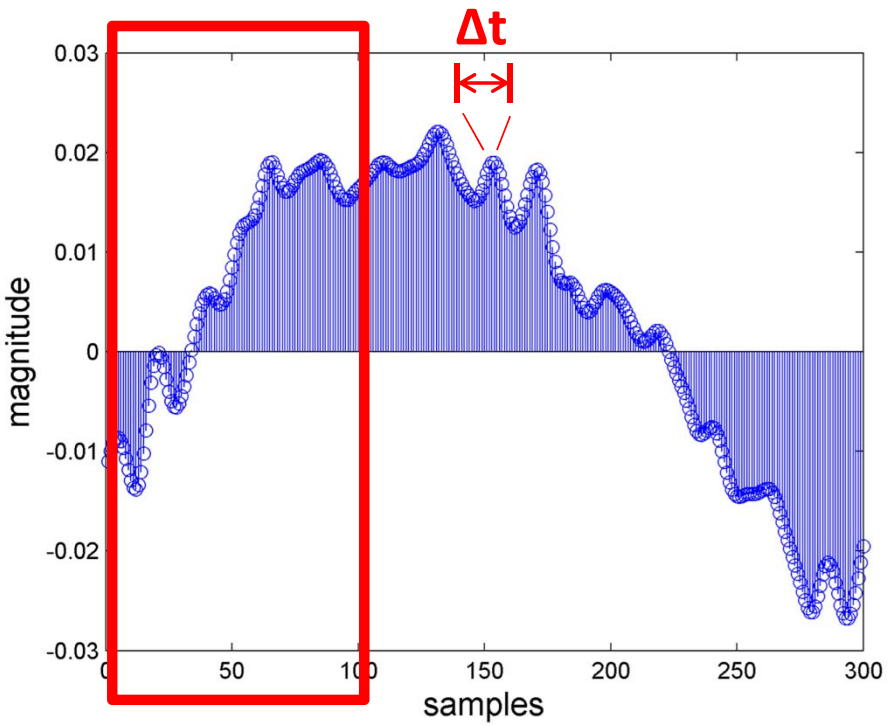


  - Can also be not proportional to the window size

Quiz Time

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

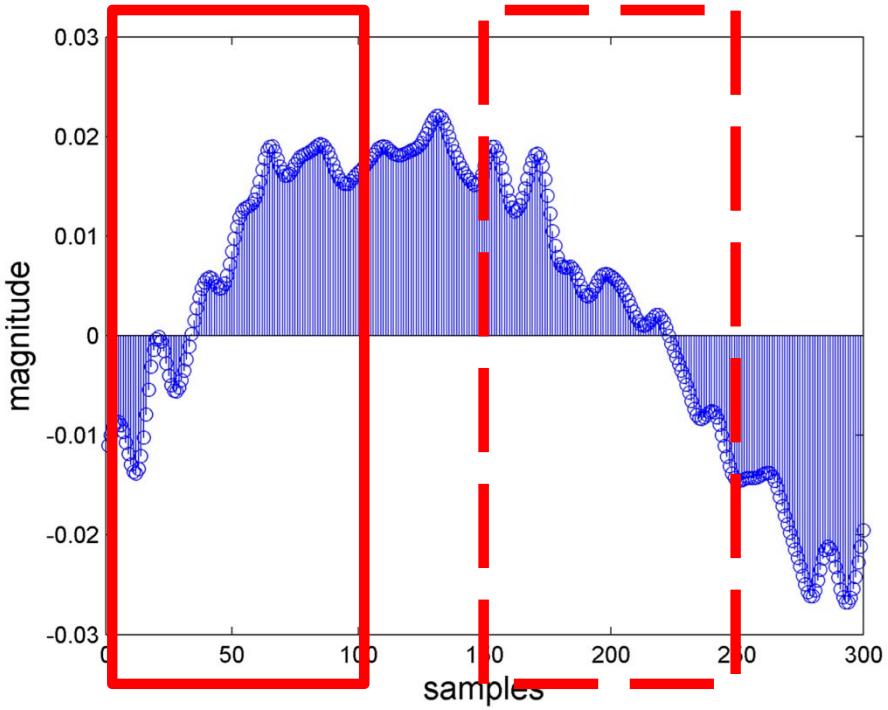
2. When the sr=1k Hz, if we use a window size of 100 samples for the STFT, what is the actual duration of the window (in seconds)?



Quiz Time

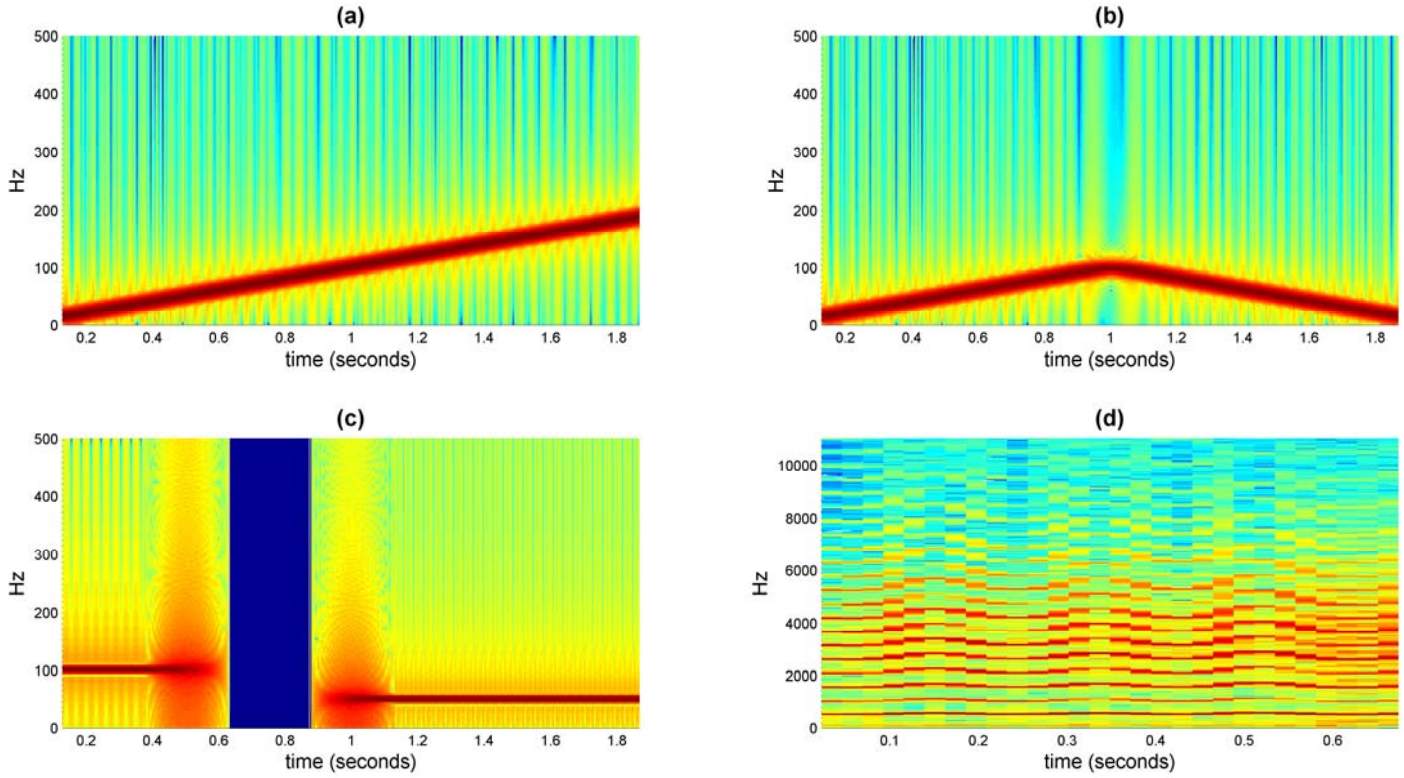
3. When the sr=1k Hz and we use a STFT window size of 100 samples with no overlaps between consecutive windows, how many times do we need to move the window to cover a signal with 300 samples?

4. And, if there is 50% over-laps between windows, how many times do we need to move the window?



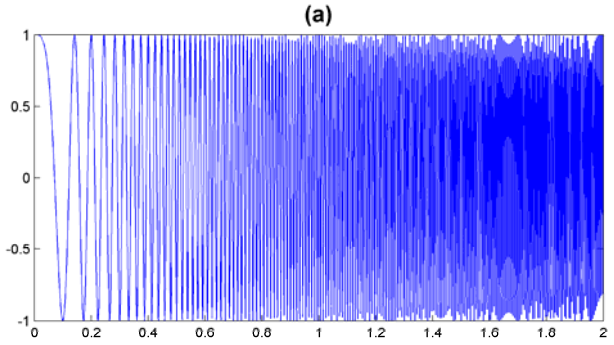
Quiz Time

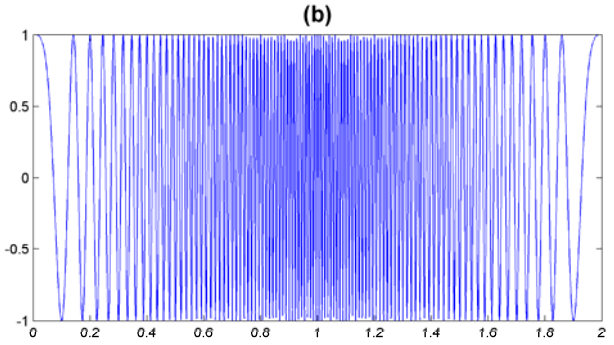
5. Given the following spectrogrms, try to draw the corresponding waveforms

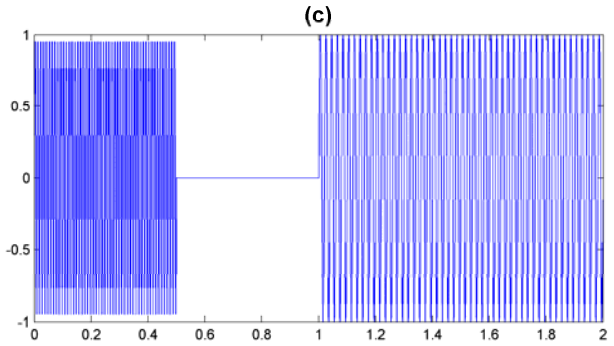


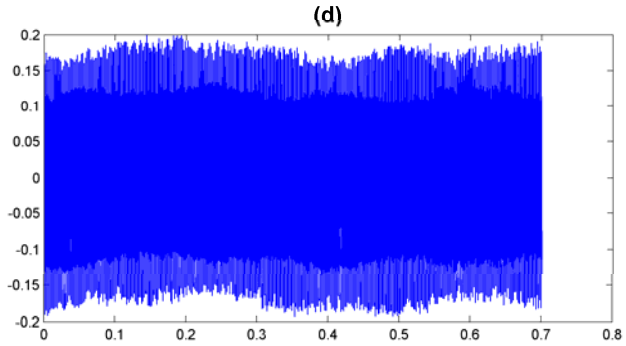
Quiz Time

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



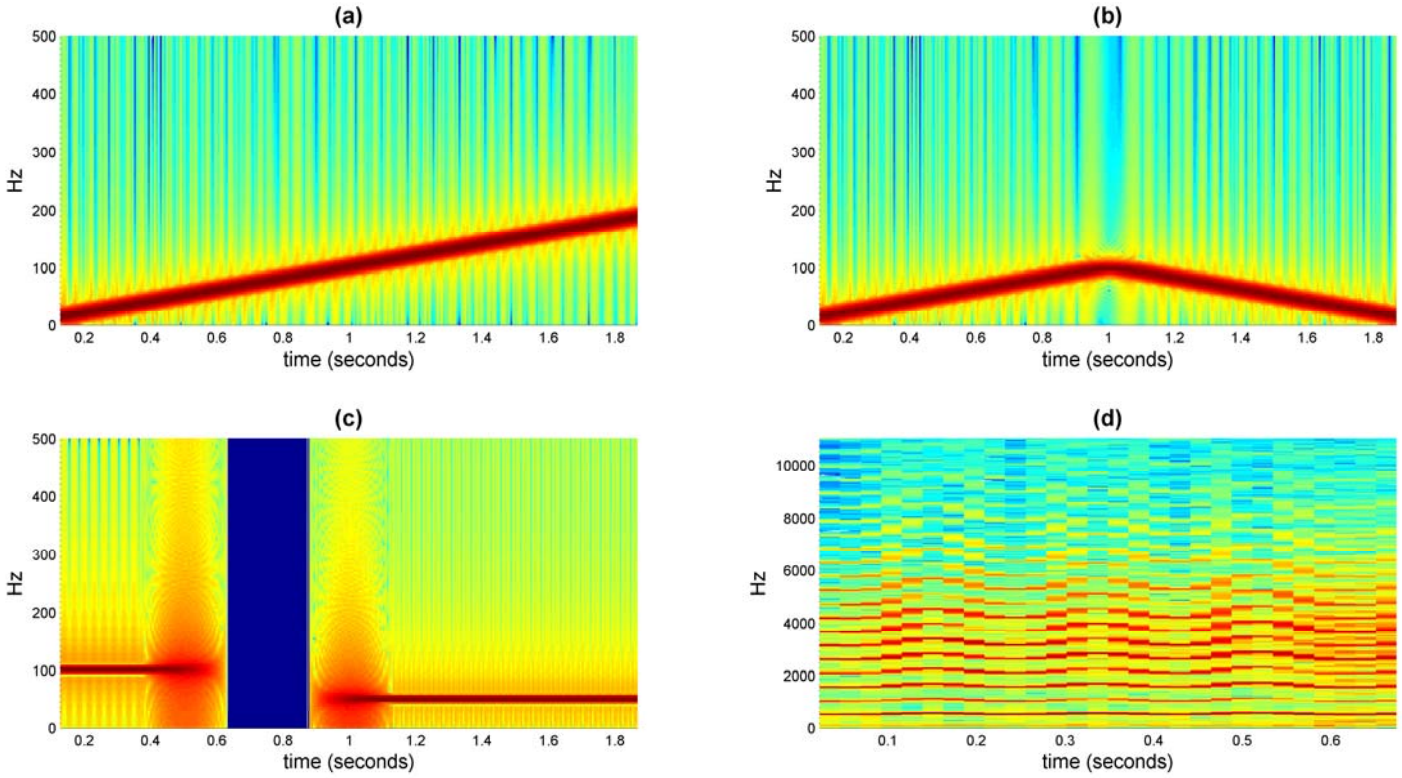






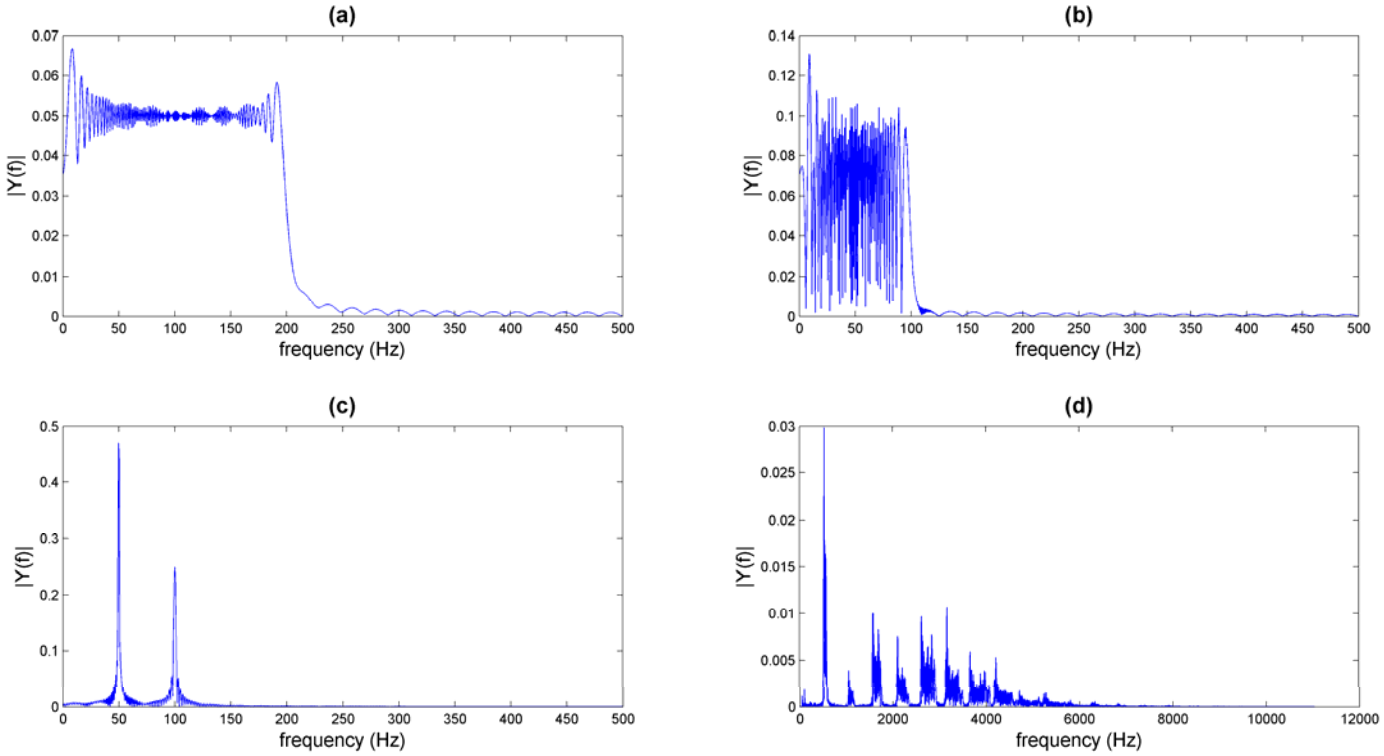
Quiz Time

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



Quiz Time

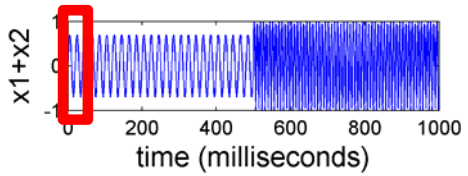
6. Given the following spectrogrms, try to draw the corresponding the spectra computed by Fourier Transform (SOLUTION)

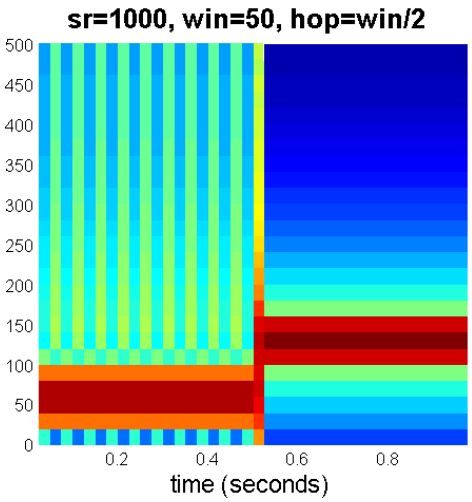


Understanding STFT

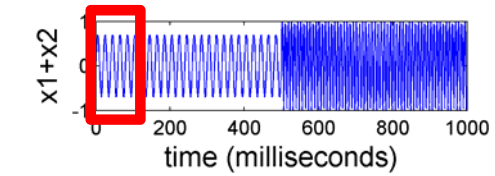
(assuming the hop size is proportional to the window size)

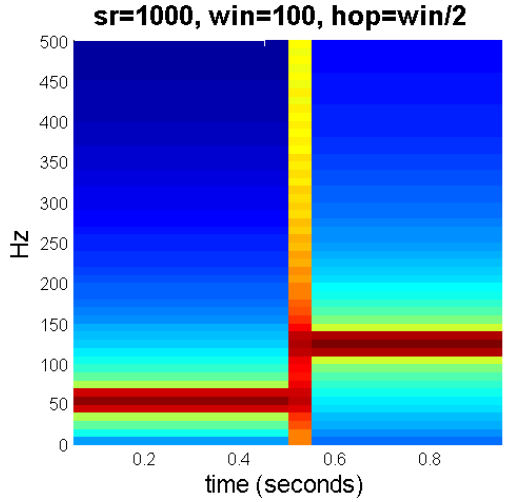
- Different window size (win_size = 50, 100, 150)



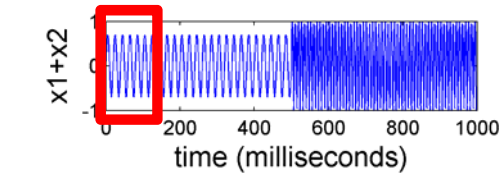


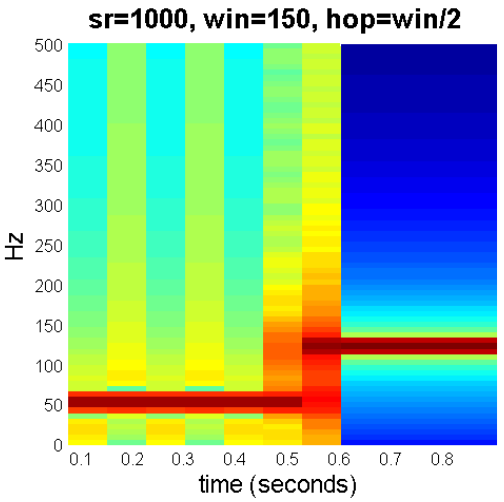
size: 26 x 39





size: 51 x 19



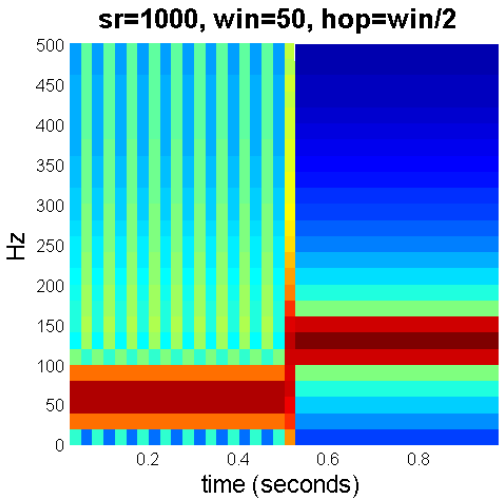


size: 76 x 12

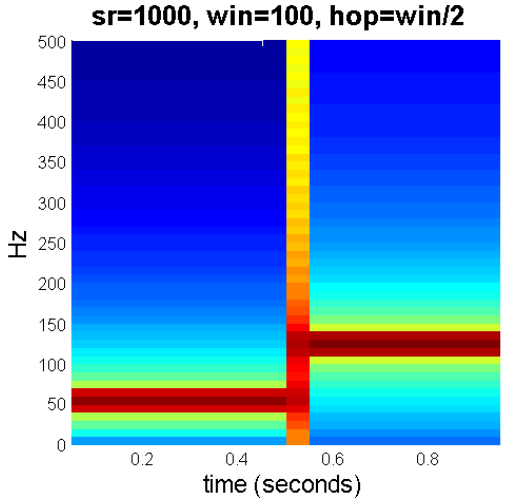
Understanding STFT

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

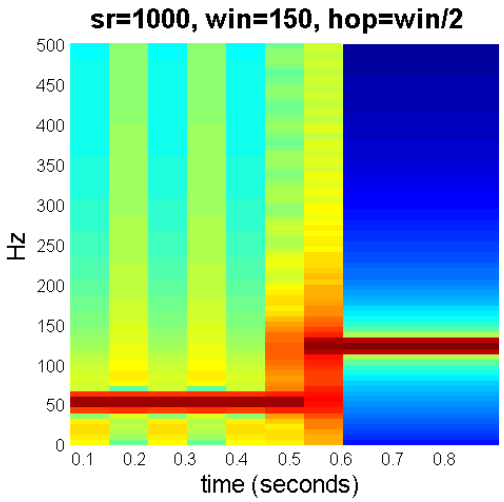
- Shorter window → worse **frequency resolution**
- Longer window → worse **temporal resolution**



size: 26 x 39



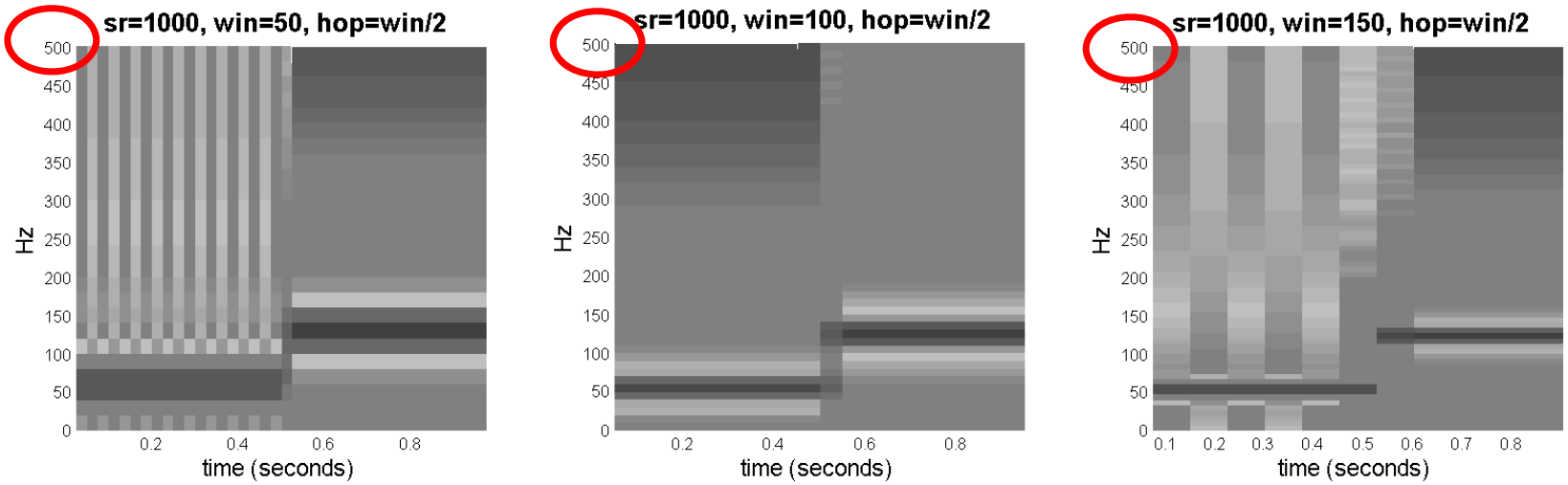
size: 51 x 19



size: 76 x 12

Understanding STFT

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



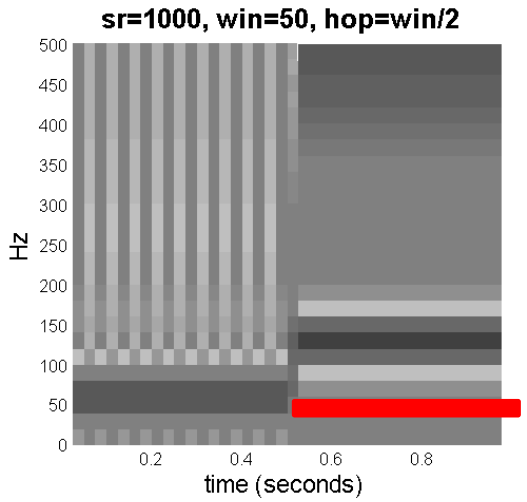
size: 26 x 39

size: 51 x 19

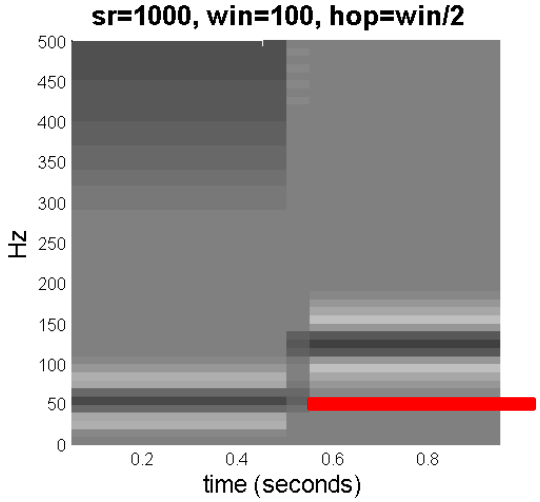
size: 76 x 12

Understanding STFT

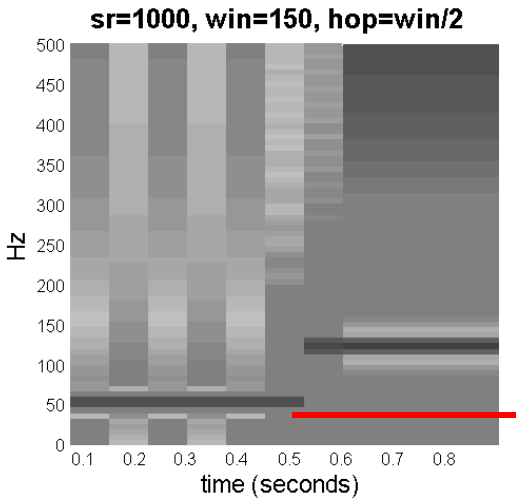
- **freq_resolution = sr/win_size**
  - longer window → better frequency resolution
  - freq_resolution = 20, 10, 6.6667 (Hz), respectively



size: 26 x 39



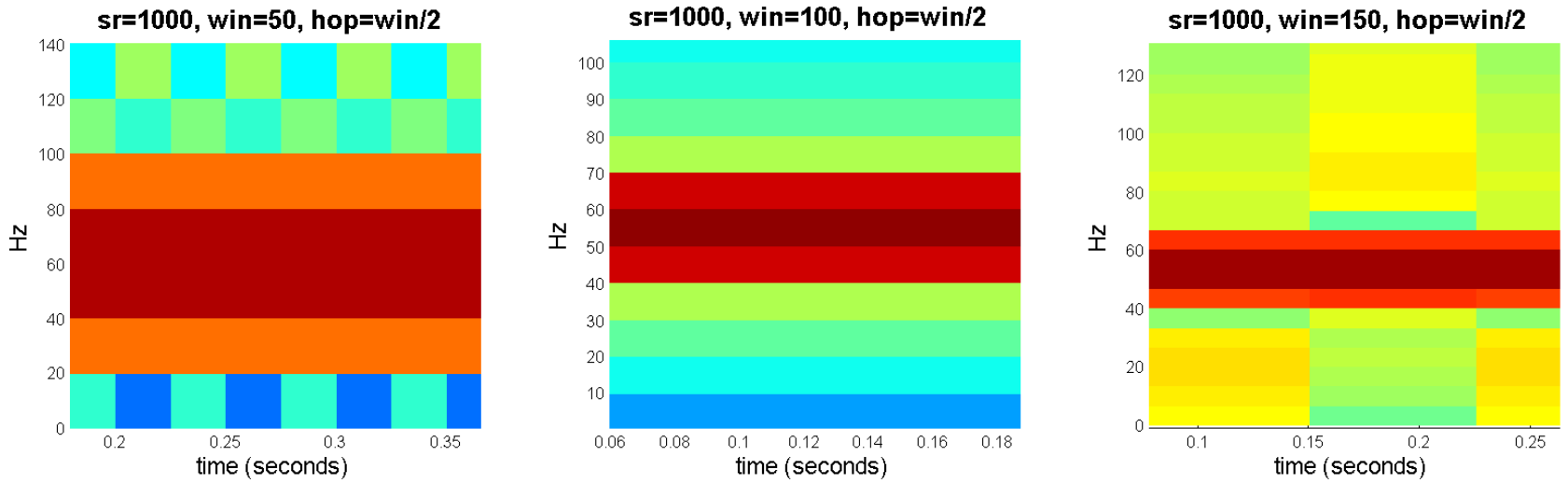
size: 51 x 19



size: 76 x 12

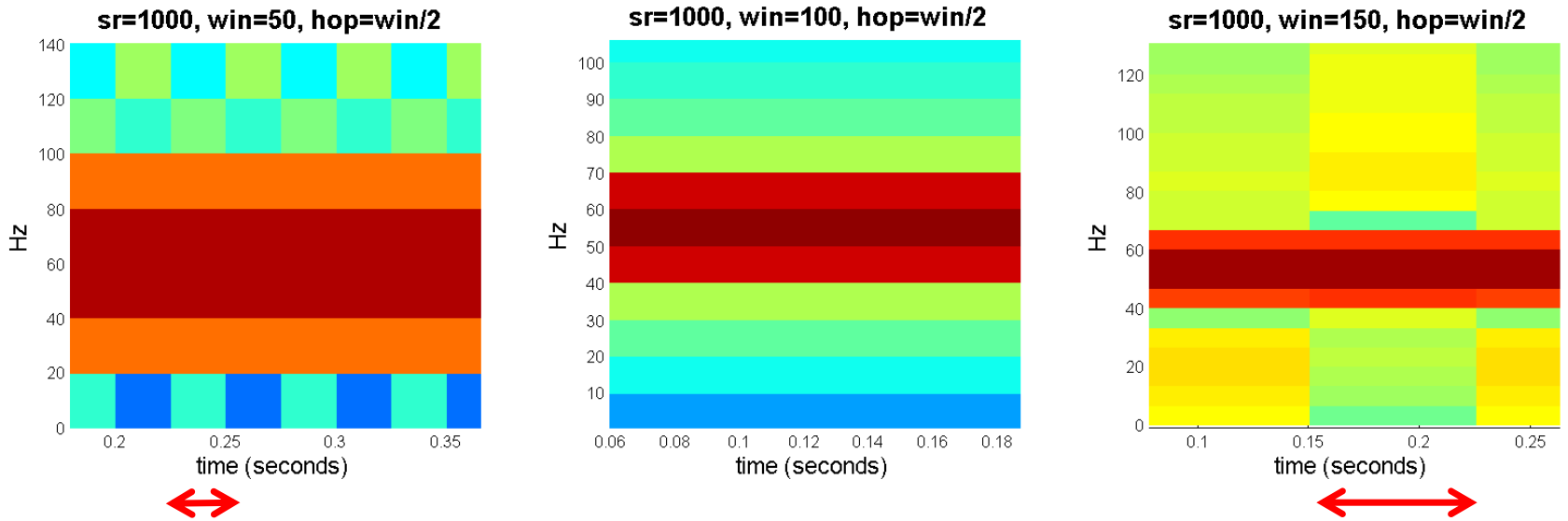
Understanding STFT

- **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
  - temp_resolution = 25, 50, 75 (ms), respectively

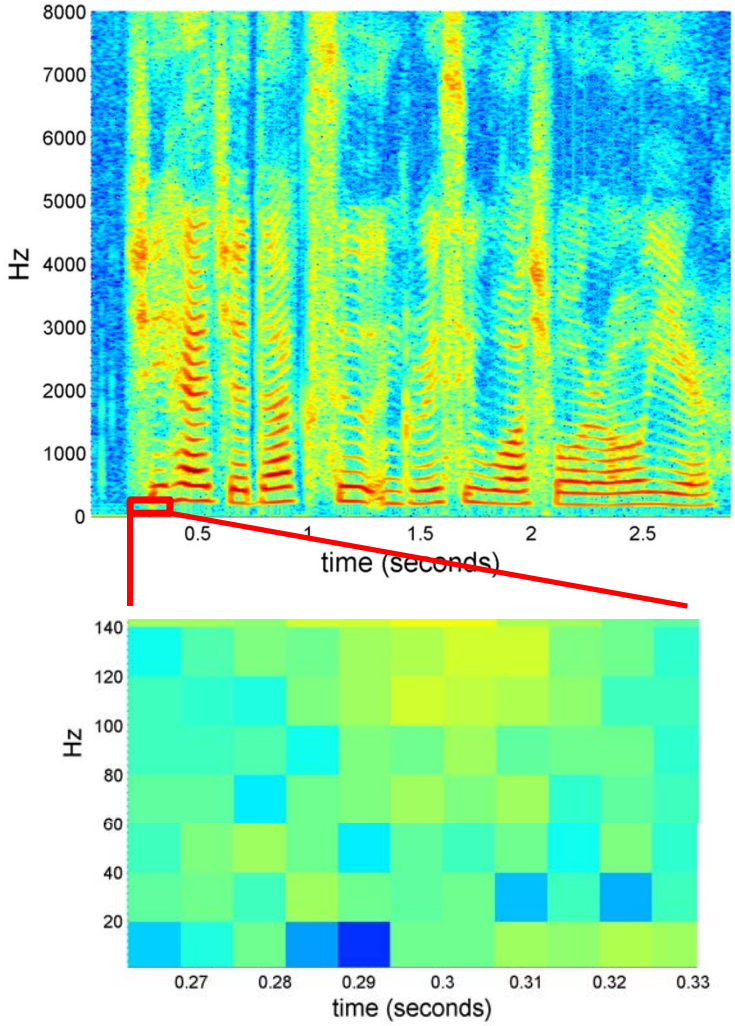


Quiz Time

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

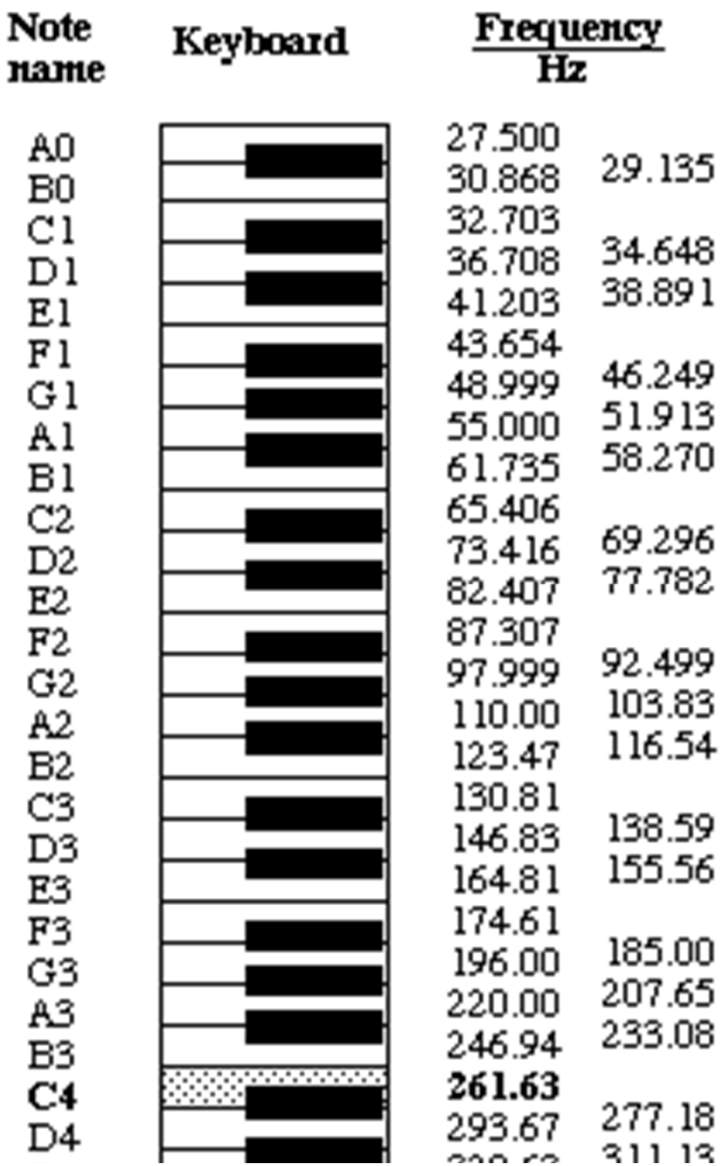
Unconscious		Conscious		
Delta	**Theta**	**Alpha**	**Beta**	**Gamma**
0,5 – 4 Hz	4 – 8 Hz	8 – 13 Hz	13 – 30 Hz	30-42 Hz
Instinct	**Emotion**	**Consciousness**	**Thought**	**Will**

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 (f0) 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?

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

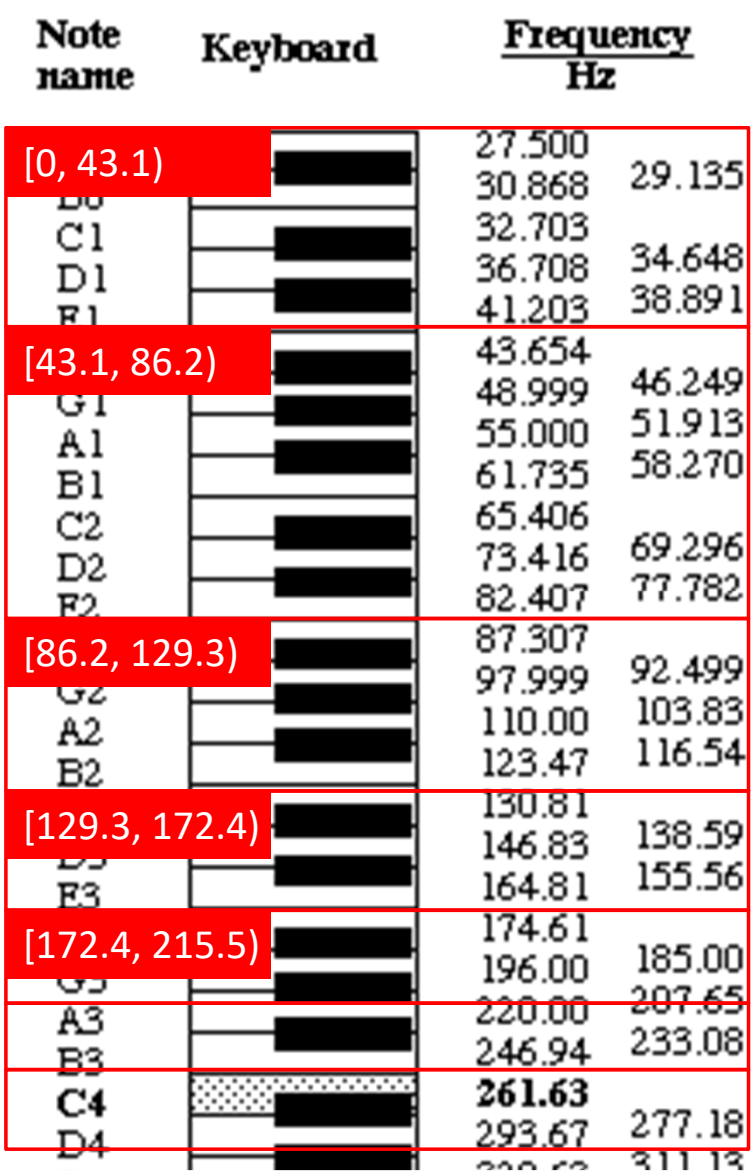


Quiz Time

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 (f0) 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?

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



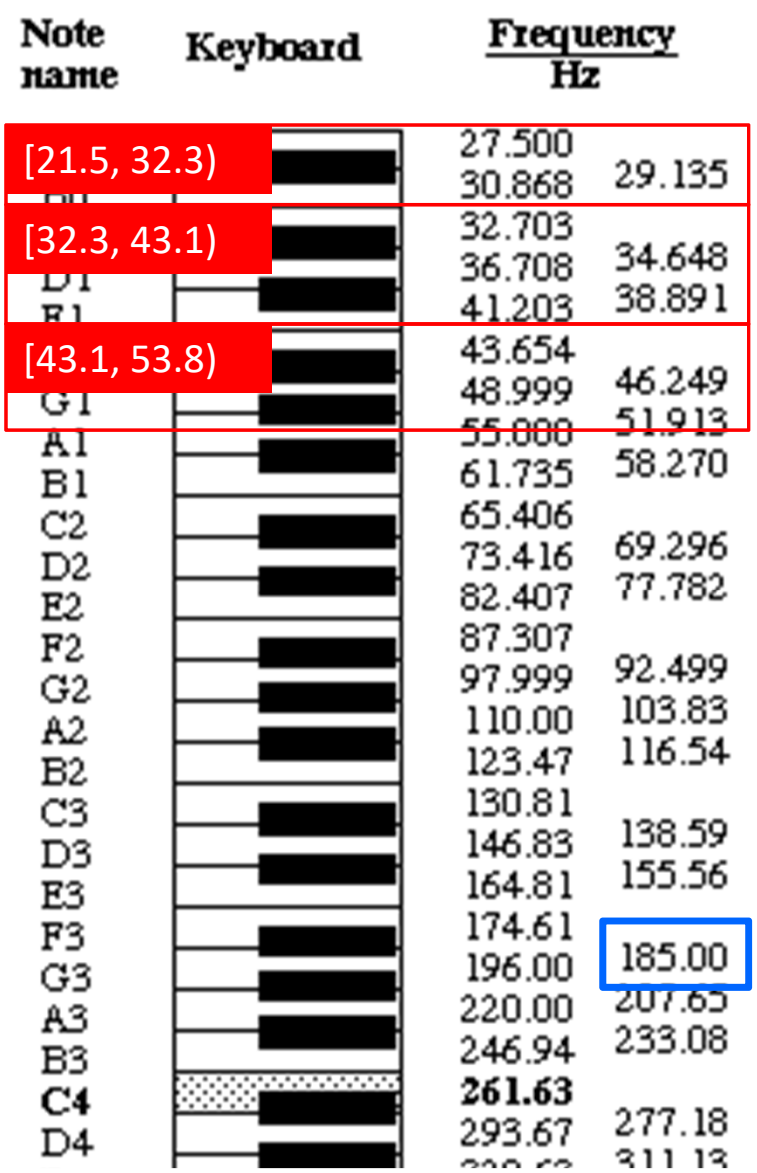
Quiz Time

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)

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



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)
```

Useful in audio generation research (only known to seasoned researchers)