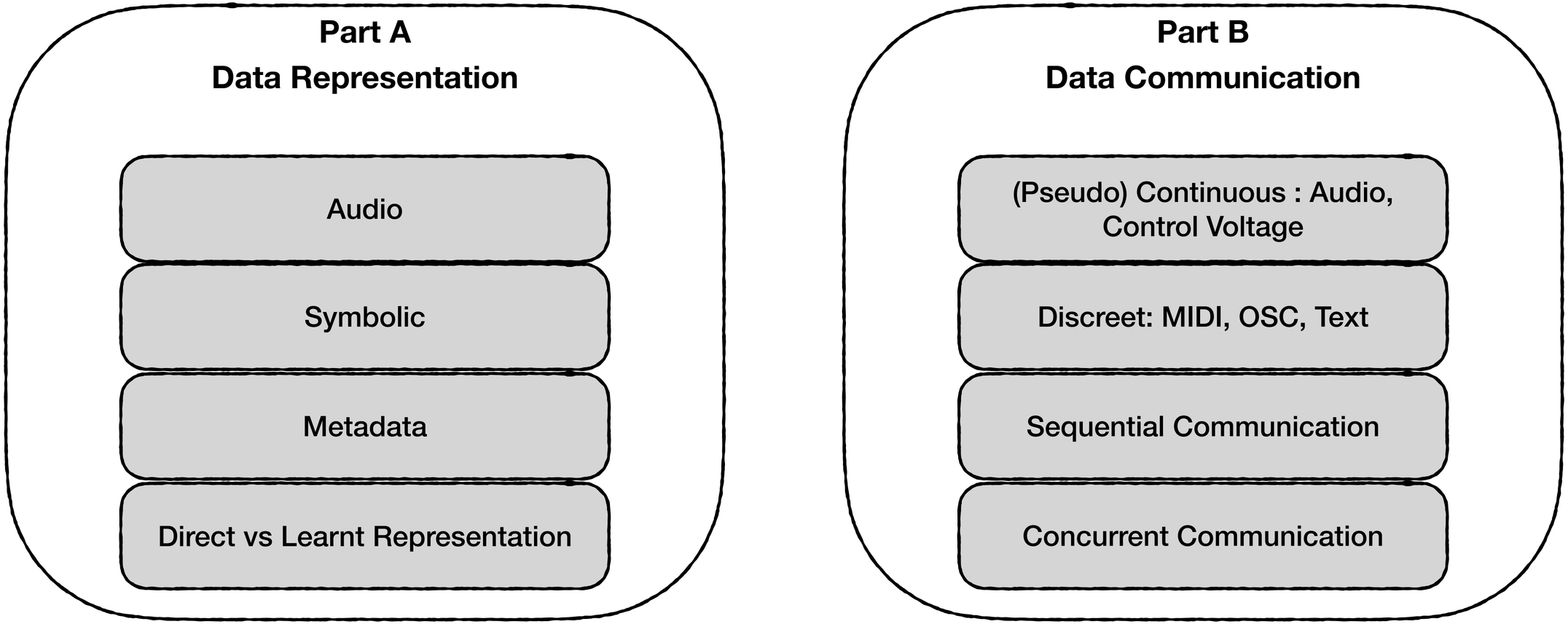
Data Representation and Communication

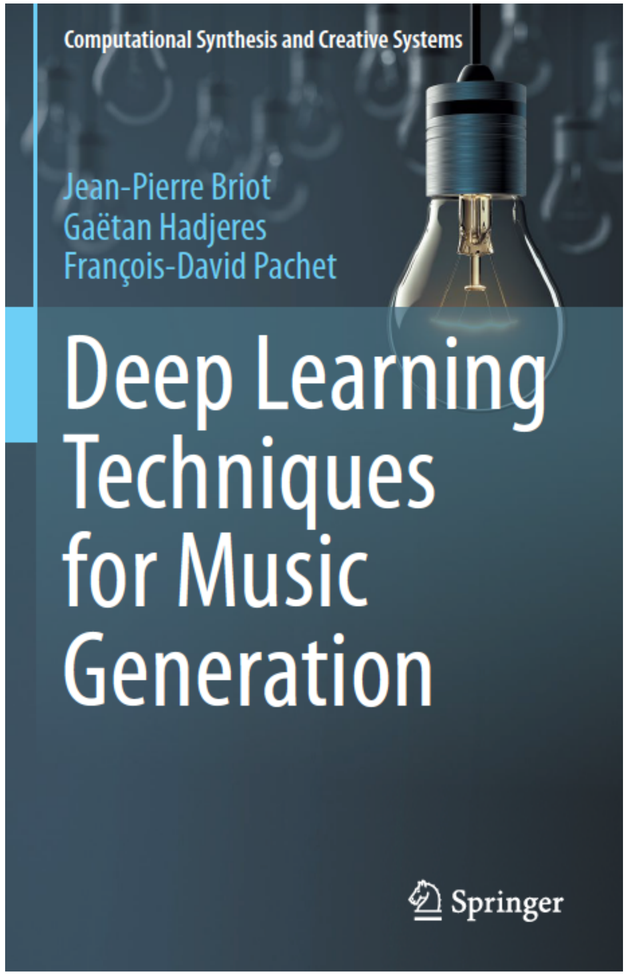
Computational Music Creativity, SMC (2020/21)

Prepared by Behzad Haki, February 2021

Overview


Part A
Data Representation
Audio
Symbolic
Metadata
Direct vs Learnt Representation
Part B
Data Communication
(Pseudo) Continuous : Audio, Control Voltage
Discreet: MIDI, OSC, Text
Sequential Communication
Concurrent Communication

Part A: Data Representation



A pre-print version here:

https://arxiv.org/abs/1709.01620

Tasks and Datatypes

Field Overview

Tasks

Classification / Identification

Retrieval

Generation

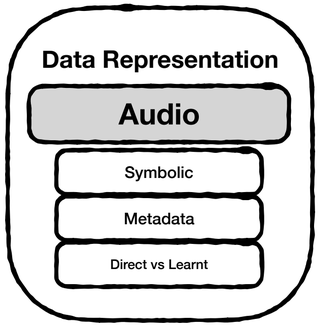
Data Types

Symbolic

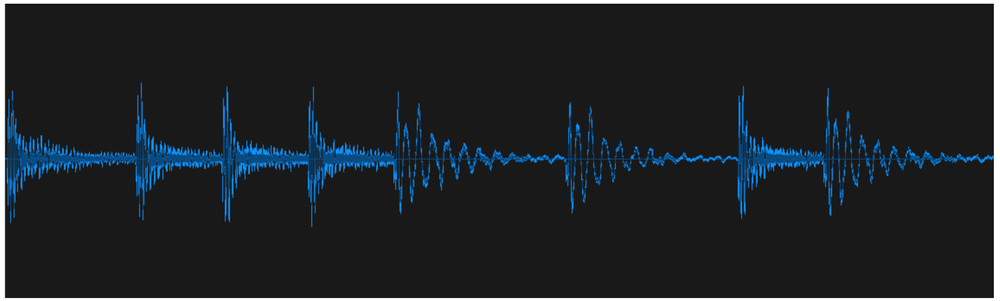
Audio / Audio-Derived

Metadata

Audio

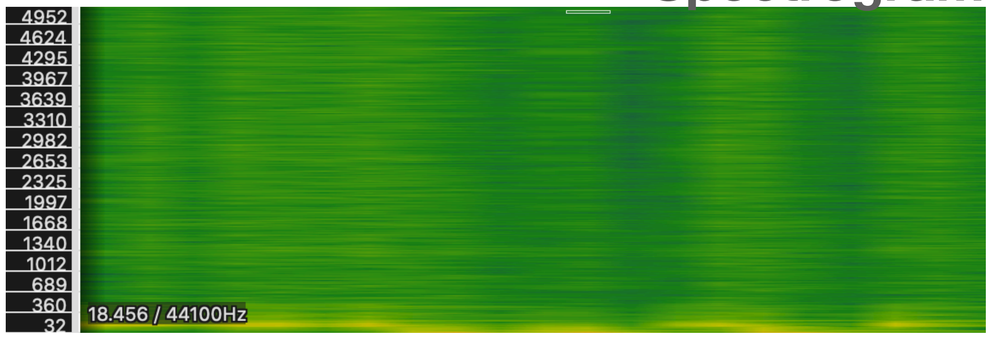


Raw Form



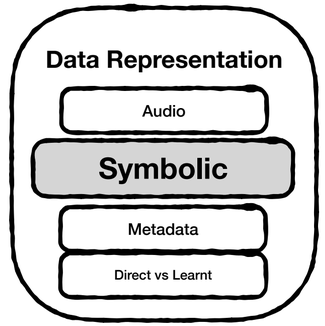
Transformed

Spectrogram



Symbolic Representation

Main Concepts [1]

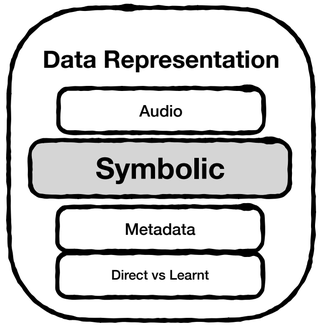


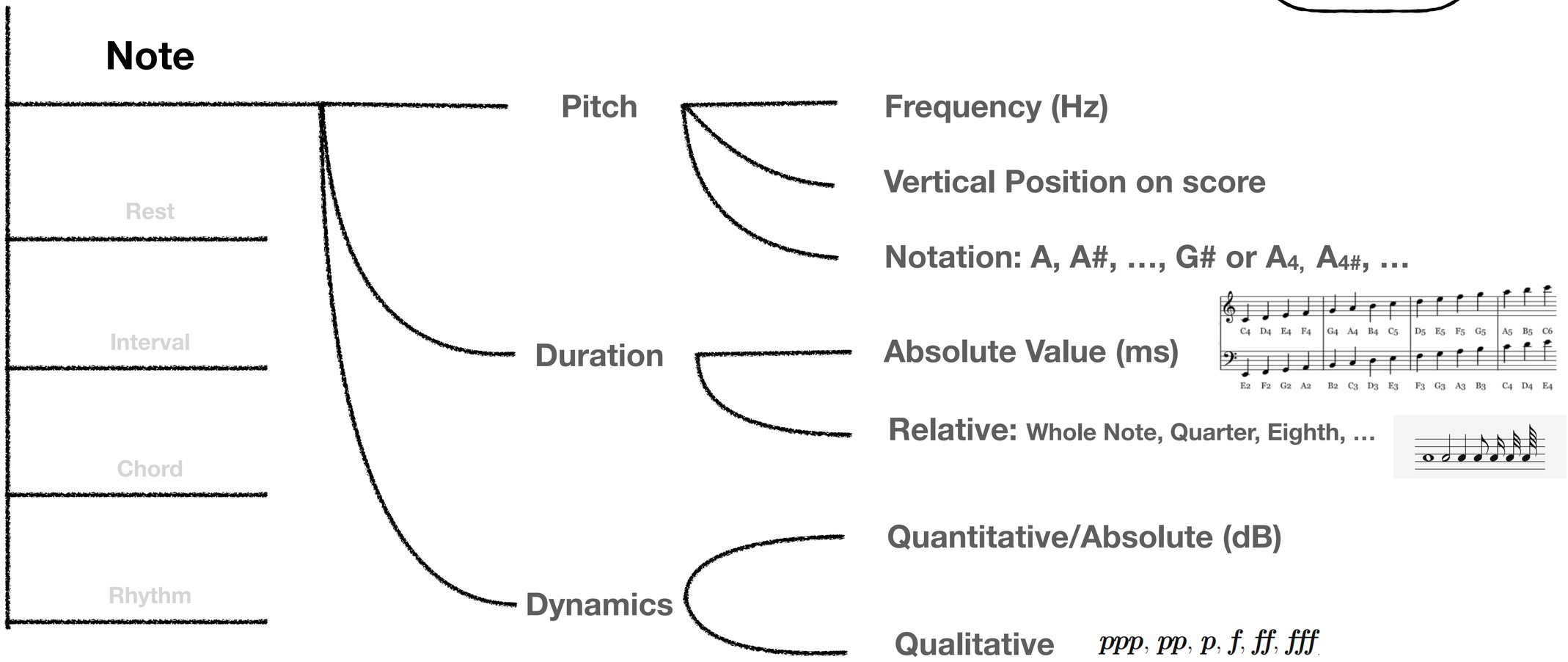
- Note
- Rest
- Interval
- Chord
- Rhythm

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 25)

Symbolic Representation

Main Concepts [1]

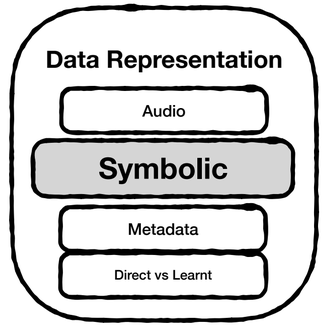




[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 25)

Symbolic Representation

Main Concepts [1]



Note

Rest

Duration

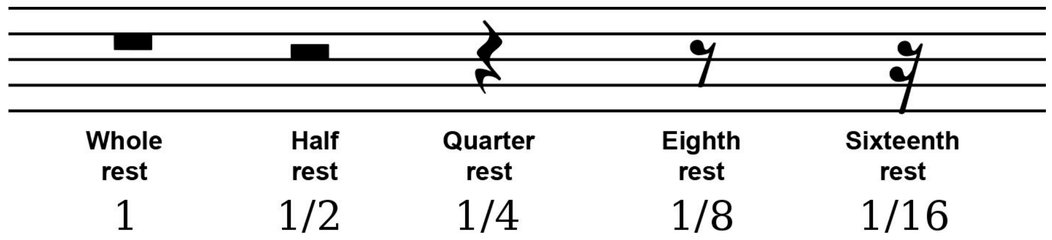
Absolute Value (ms)

Relative: Whole Note, Quarter, Eighth, …

Interval

Chord

Rhythm

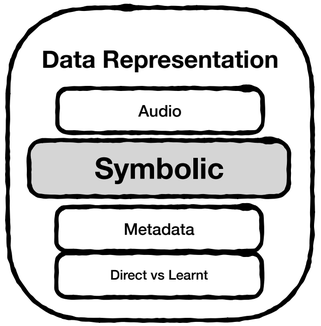


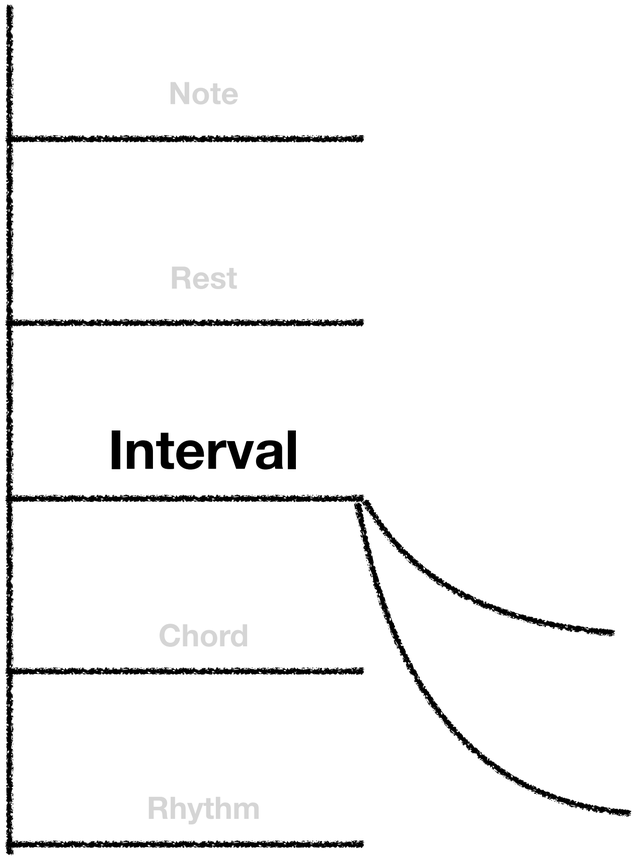
[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 25)

** Image from https://pianolit.com/blog/learn-to-read-music-fast

Symbolic Representation

Main Concepts [1]





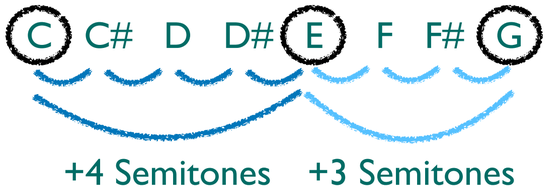
Melodic Interval	Semitones	Frequency Ratio	Example
Unison	0	1:1	C_4–C_4
Minor second	1	16:15	C_4–$C\sharp_4$/C_4–B_3
Major second	2	9:8	C_4–D_4/C_4–$A\sharp_3$
Minor third	3	6:5	C_4–$D\sharp_4$/C_4–A_3
Major third	4	5:4	C_4–E_4/C_4–$G\sharp_3$
Perfect fourth	5	4:3	C_4–F_4/C_4–G_3
Tritone	6	45:32	C_4–$F\sharp_4$/C_4–$F\sharp_3$
Perfect fifth	7	3:2	C_4–G_4/C_4–F_3
Minor sixth	8	8:5	C_4–$G\sharp_4$/C_4–E_3
Major sixth	9	5:3	C_4–G_4/C_4–$D\sharp_3$
Minor seventh	10	9:5	C_4–A_4/C_4–D
Major seventh	11	15:8	C_4–B_4/C_4–$C\sharp_3$
Octave	12	2:1	C_4–C_5/C_4–C_3

Absolute Interval Melody Representation

C_4, E_4, G_4

Relative Interval Melody Representation

$C_4, +4, +3$

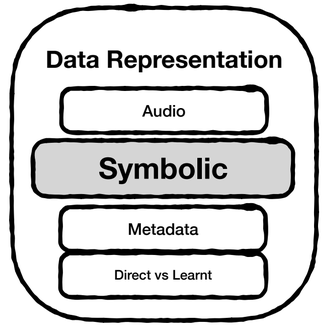


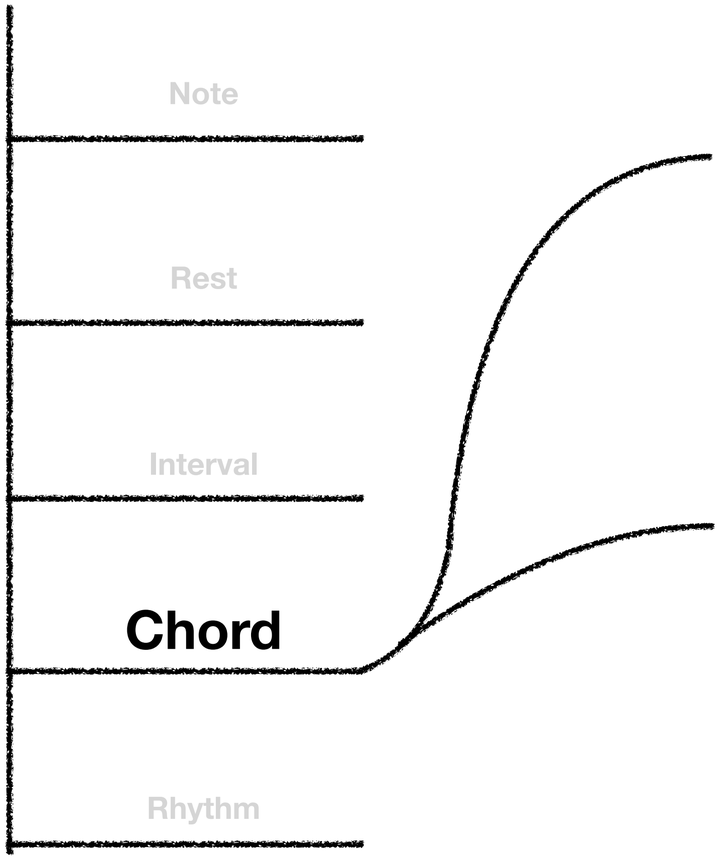
[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 26)

** Table from https://www.researchgate.net/figure/Survey-of-the-13-Within-Octave-Intervals-Their-Size-in-Semitones-the-Ratio-of-the_tbl1_11248565

Symbolic Representation

Main Concepts [1]

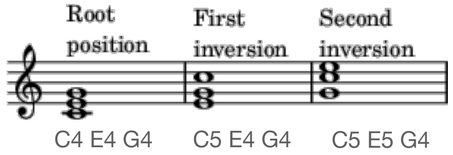




Implicit and Extensional

Explicitly mention the exact notes

Can specify the exact octave and voicing (i.e. inversions)



Explicit and Intensional

Chord symbol combining root note and the type

(e.g., major, minor, dominant seventh, or diminished)

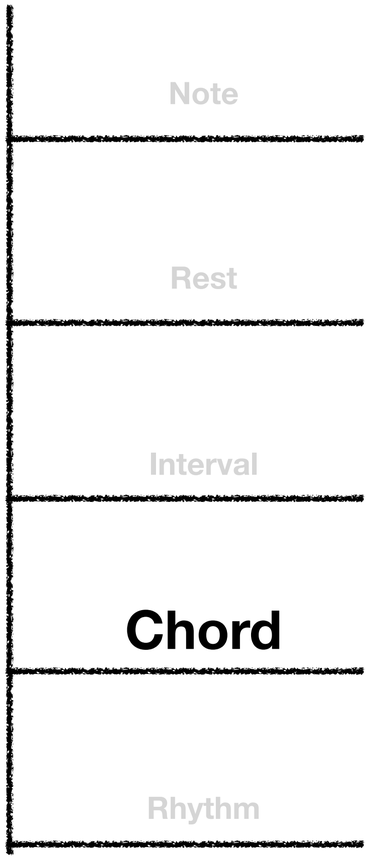
CMaj

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 26-27)

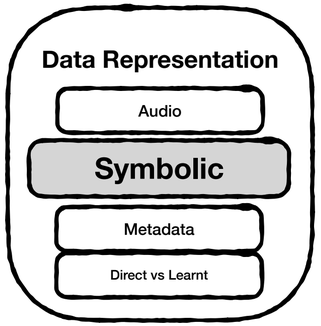
** Image from https://musictheoryblog.blogspot.com/2007/02/chord-roots-and-chord-inversion.html

Symbolic Representation

Main Concepts [1]



Chord			Components												
Name	Symbol (on C)		Interval	P1	m2	M2	m3	M3	P4	d5	P5	A5	M6/d7	m7	M7
	Short	Long	Semitones	0	1	2	3	4	5	6	7	8	9	10	11
Major triad	C CΔ			P1				M3			P5				
Major sixth chord	C^6 CM^6	Cmaj6		P1				M3			P5		M6		
Dominant seventh chord	C^7	Cdom7		P1				M3			P5			m7	
Major seventh chord	CM^7 $C^{\Delta 7}$	$Cmaj^7$		P1				M3			P5				M7
Augmented triad	C+	Caug		P1				M3				A5			
Augmented seventh chord	$C+^7$	$Caug^7$		P1				M3				A5		m7	
Minor triad	Cm	Cmin		P1			m3				P5				
Minor sixth chord	Cm^6	$Cmin^6$		P1			m3				P5		M6		
Minor seventh chord	Cm^7	$Cmin^7$		P1			m3				P5			m7	
Minor-major seventh chord	Cm^{M7} Cm/M7 Cm(M7)	$Cmin^{maj7}$ Cmin/maj7 Cmin(maj7)		P1			m3				P5				M7
Diminished triad	C^o	Cdim		P1			m3			d5					
Diminished seventh chord	C^{o7}	$Cdim^7$		P1			m3			d5			d7		
Half-diminished seventh chord	$C^{ø}$ $C^{ø7}$			P1			m3			d5				m7	

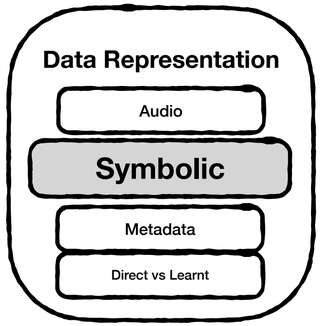


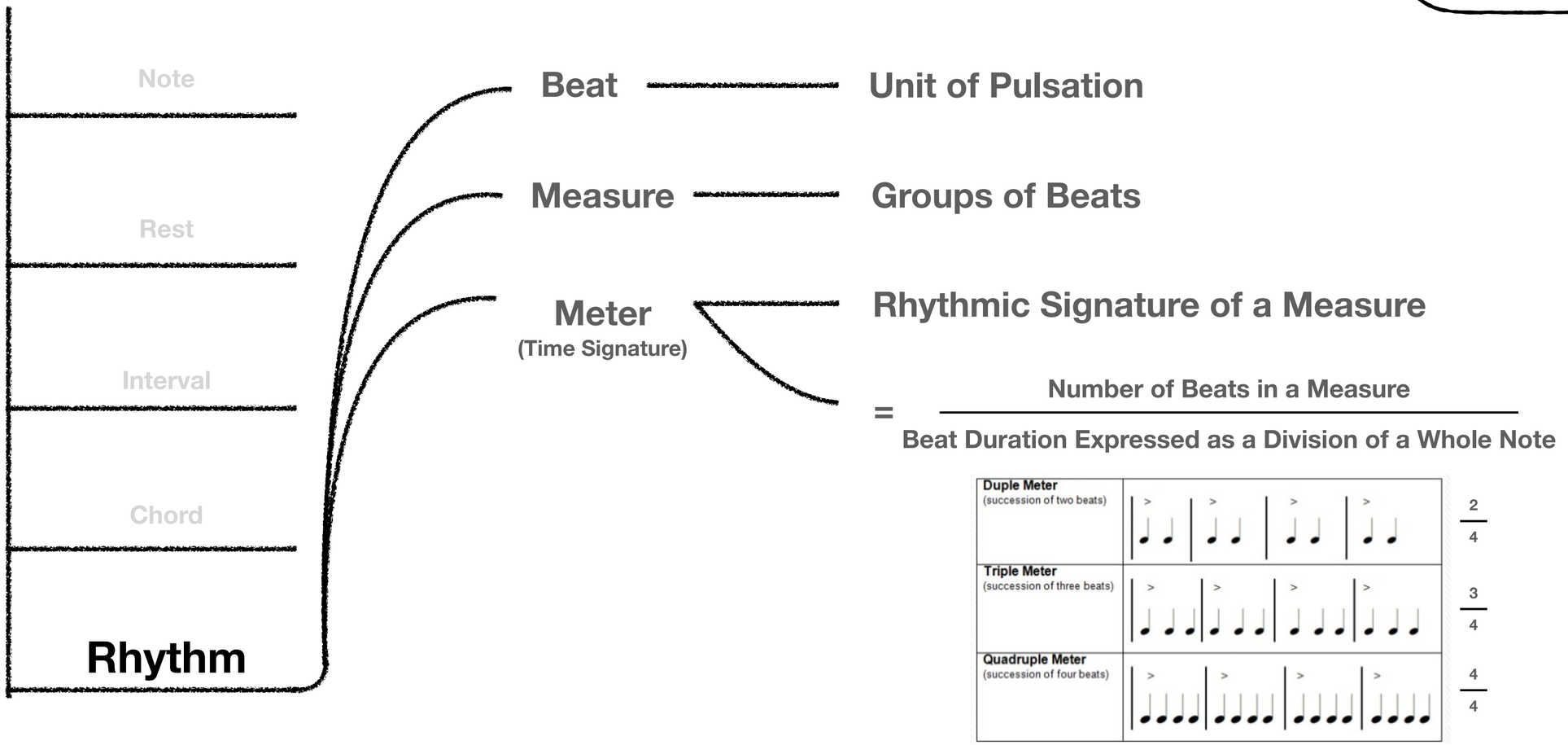
[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 26-27)

** Table from https://en.wikipedia.org/wiki/Chord_(music)

Symbolic Representation

Main Concepts [1]



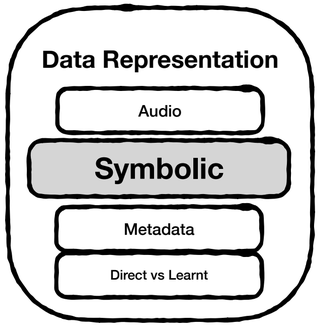


[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 26-27)

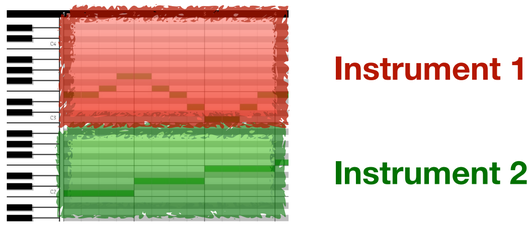
** Table from https://form1ocs2016-2017.blogspot.com/2016/10/rhythm-meter-and-time-signature.html

Symbolic Representation

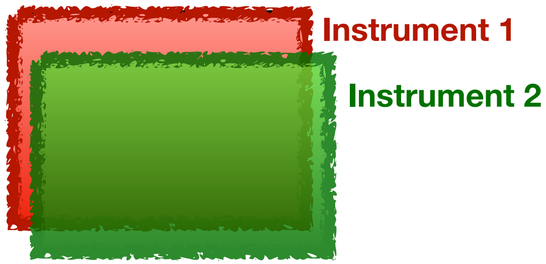
Multivoice/Multitrack



- Either in one track assigned to different midi ranges



- Or, more commonly stored in separate tracks

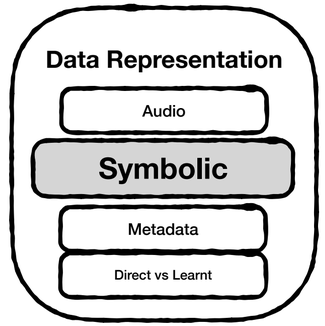


- Different voices of drums can be considered as different notes of a single voice

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 28)

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]

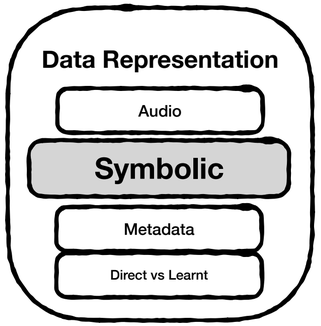


- MIDI
- Piano Roll
- Text
- Markup Language
- Lead Sheet

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Pages 29-36)

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]



MIDI

2 Byte Messages
(0x00 - 0xFF)

0x00 - 0x7F : Data Bytes
(contains note number or other parameters

0x80 - 0xFF : Status/Command Bytes
(contains note number or other parameters

Piano Roll

Text

Markup Language

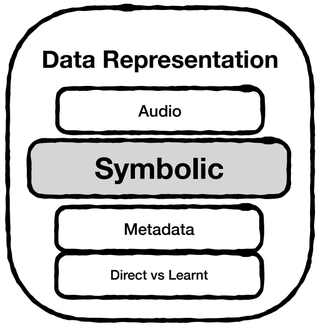
Lead Sheet

*0x*8n : Note-On
*0x*9n : Note-Off
*0x*An : Key Pressure / Aftertouch (polyphonic - per Key)
*0x*Bn : Control Change (eg Mod Wheel)
*0x*Cn : Program Change (select instrument)
*0x*Dn : Key Pressure / Aftertouch (per channel)
*0x*En : Pitch Bend

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Pages 29-30)
[2] https://sound.stackexchange.com/questions/24231/difference-between-channel-and-track-in-midi-files

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]



MIDI

- Standard Midi Files
  - SMF0 — All messages stored in a single track
  - SMF1 — Can have more than one track.

Example:

- Track 1 —> Notes played on the right hand of piano, on channel 0
- Track 2 —> Notes played on the left hand of piano, on channel 0
- Track 3 —> Bass Voice on channel 1
- Track 4 —> Vocals on Channel 3
- Track 5 —> Drums on Channel 9

Piano Roll

Text

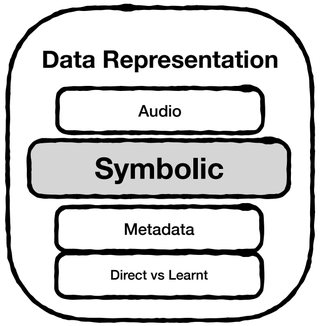
Markup Language

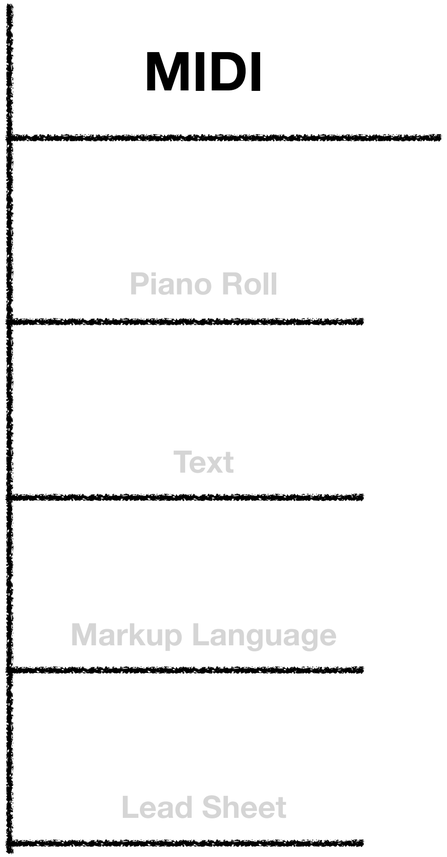
Lead Sheet

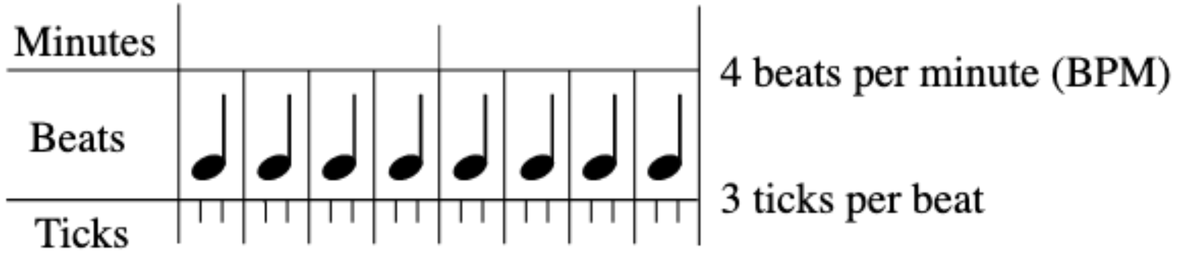
[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Pages 29-30)
[2] https://sound.stackexchange.com/questions/24231/difference-between-channel-and-track-in-midi-files

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]







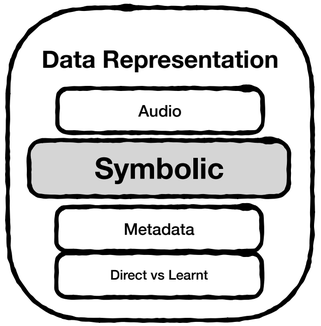
PPQ
(parts per quarter,
aka TPQN - Ticks per Quarter Note)

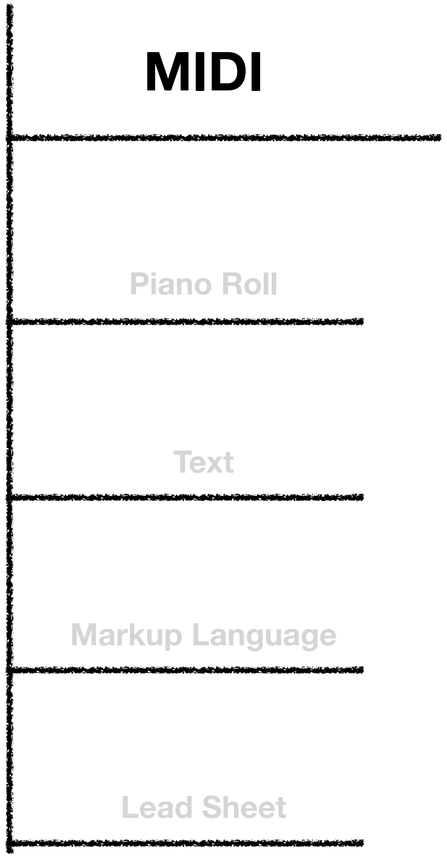
$$\text{Time Resolution (ms)} = \frac{60000}{\text{PPQ} \times \text{BPM}}$$

[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Pages 29-30)
[2] https://mido.readthedocs.io/en/latest/midi_files.html

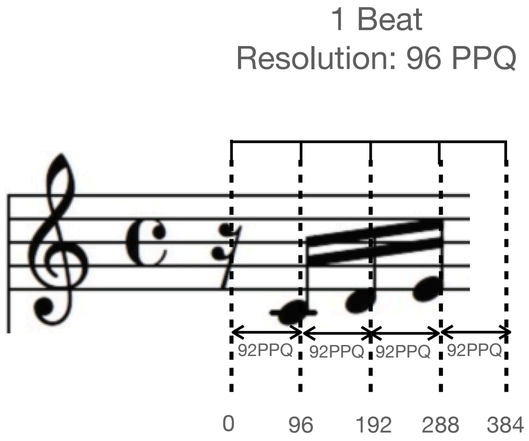
Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]





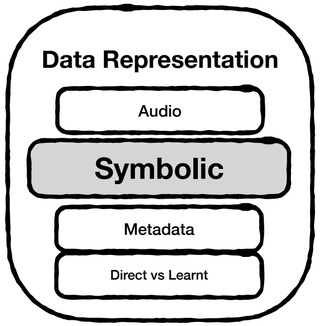
Midi Track	Time (ppqs rel. to start)	On/Off	Midi Channel	Pitch	Velocity
2,	96,	Note_on,	0,	60,	90
2,	192,	Note_off,	0,	60,	0
2,	192,	Note_on,	0,	62,	90
2,	288,	Note_off,	0,	62,	0
2,	288,	Note_on,	0,	64,	90
2,	384,	Note_off,	0,	64,	0



[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 29-30)

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]



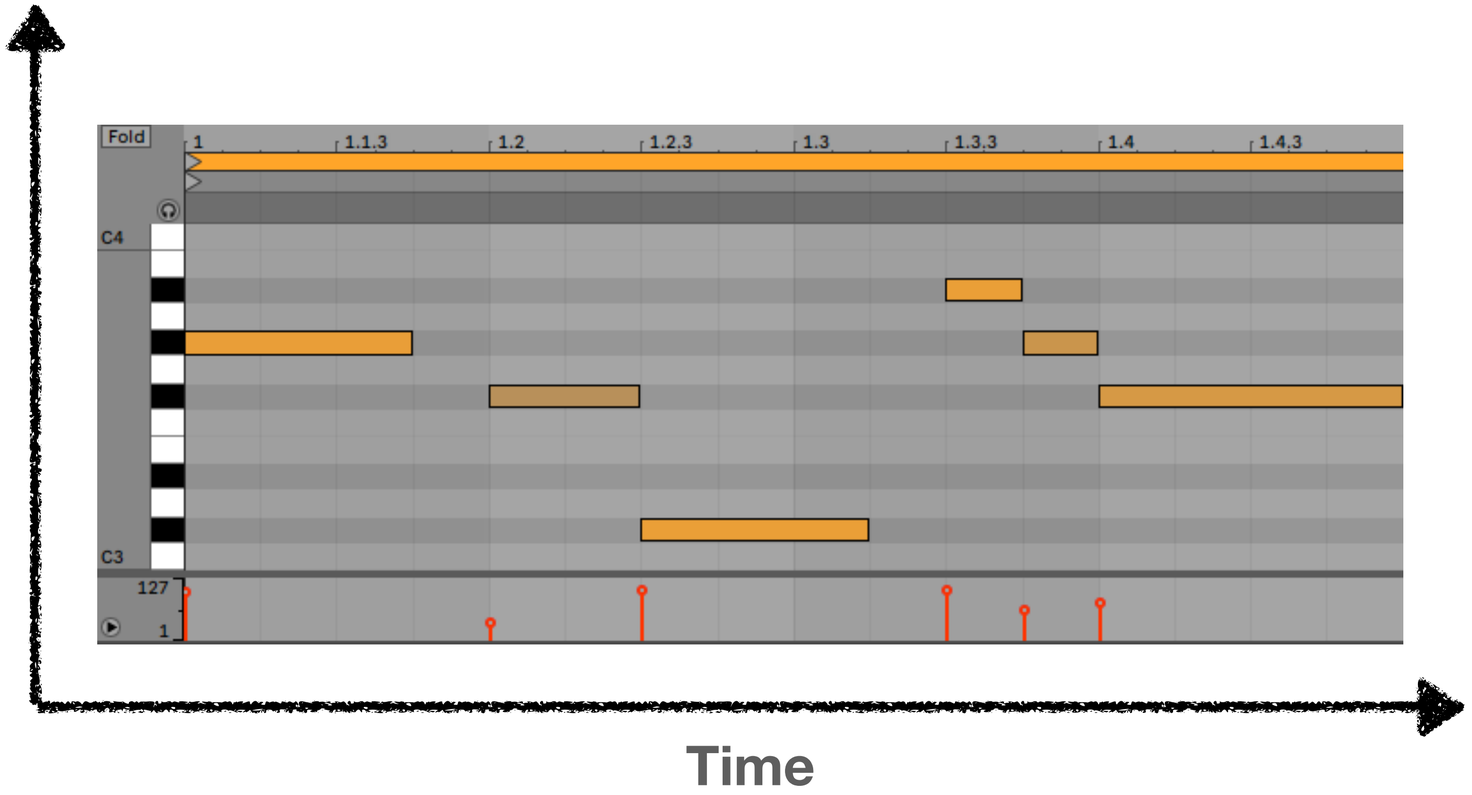
MIDI

Piano Roll

Text

Markup Language

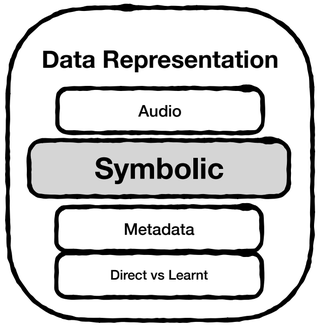
Lead Sheet

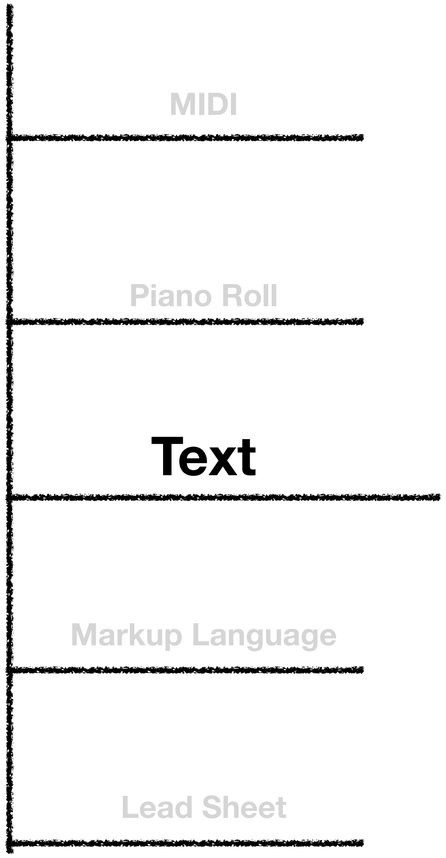


[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 29-30)

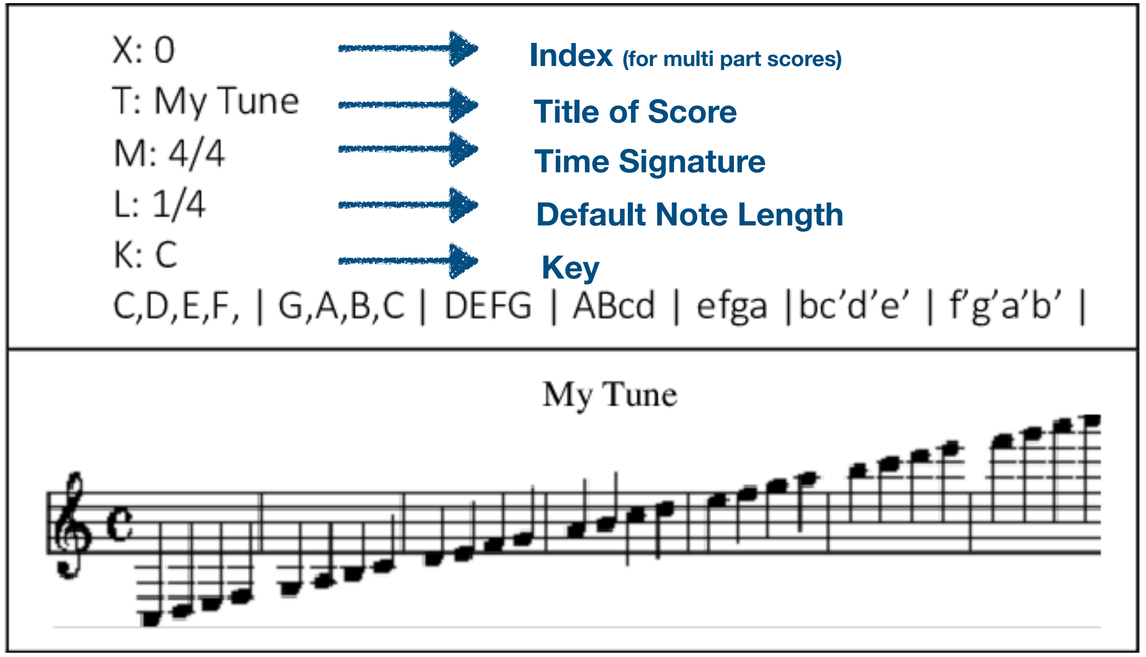
Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]





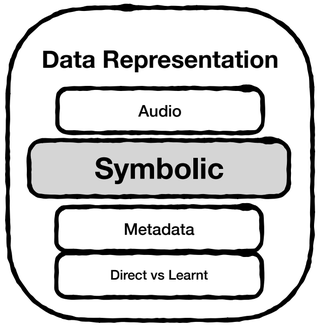
ABC Notation



[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 32-33)
[2] https://www.researchgate.net/figure/An-example-of-ABC-notation-language_fig1_340541536

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]



MIDI

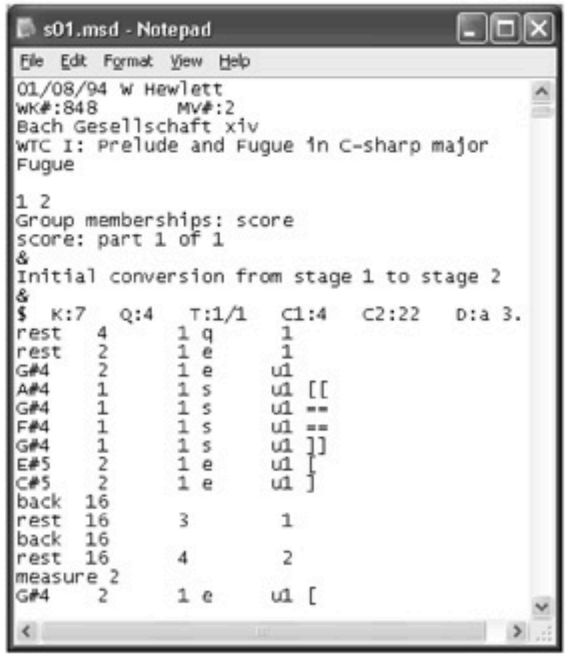
Piano Roll

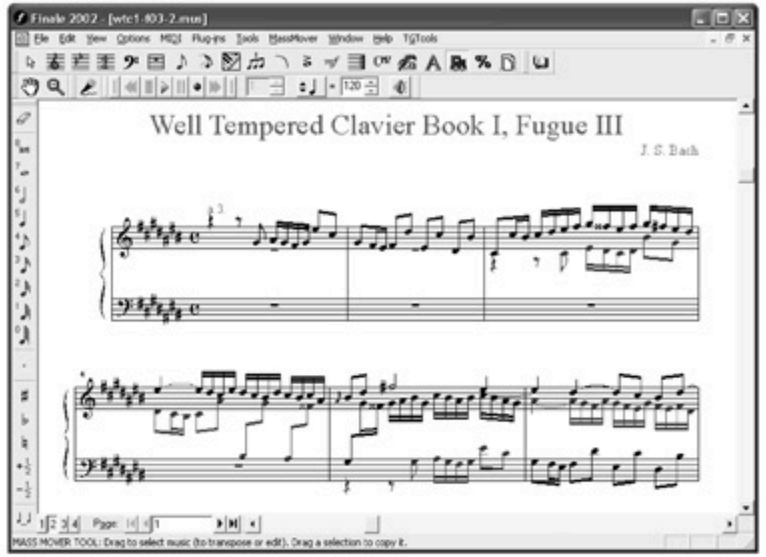
Text

Markup Language

Lead Sheet

HTML / MusicXML

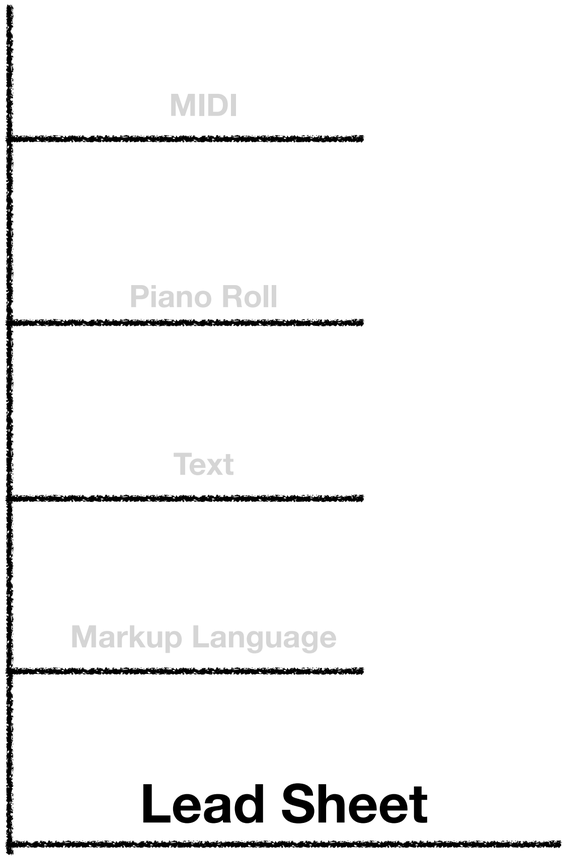


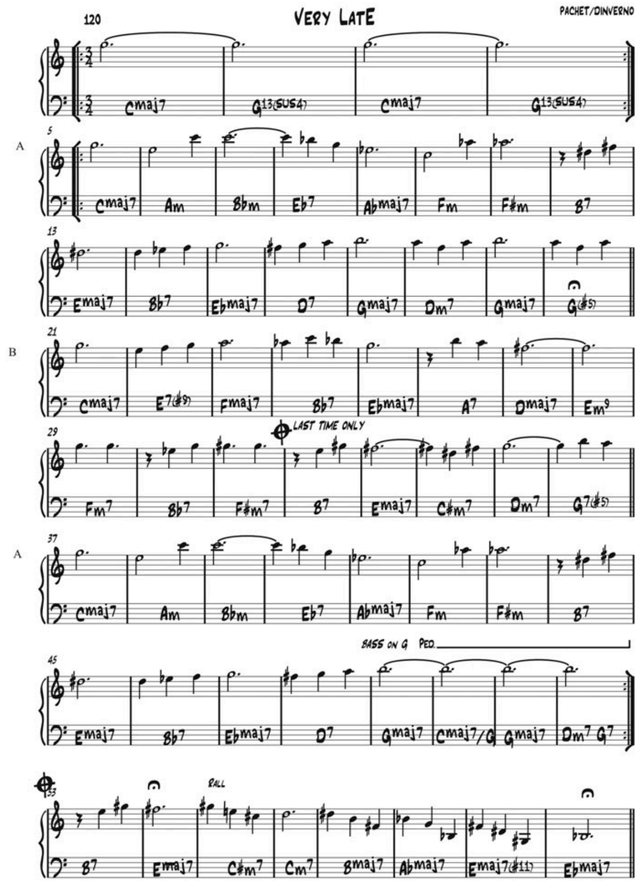


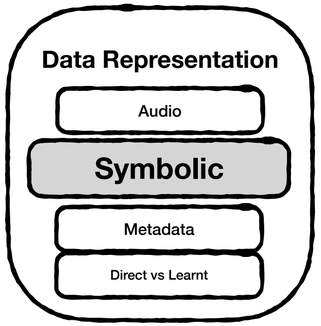
[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 32-33)
[2] https://www.researchgate.net/figure/An-example-of-ABC-notation-language_fig1_340541536

Symbolic Representation

Formats (Grammar / Syntax used for Representation) [1]

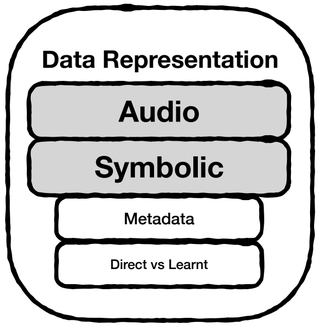






[1] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 32-33)
[2] https://www.researchgate.net/figure/An-example-of-ABC-notation-language_fig1_340541536

Audio Vs. Symbolic



Can be the synthesized score OR
Can be a performance (or rather interpretation) of the score

Captures every sonic aspect of performance

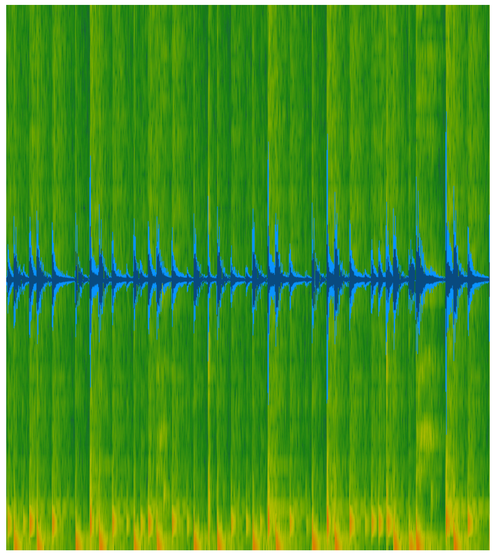
Captures the timbral aspects of music

Computationally more expensive

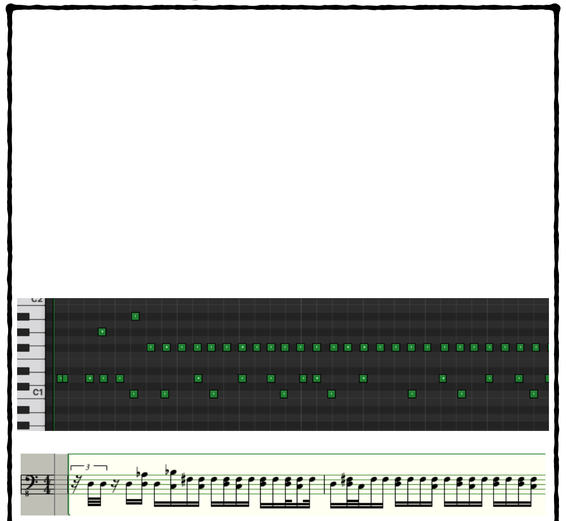
Hard to visually interpret

Harder to investigate interdependencies/interlocking of voices

Audio



Symbolic



Captures the essence of a composition!

Potentially captures some aspects of performance

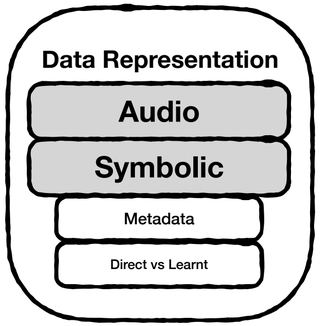
Captures very limited timbral aspects of music

Computationally less expensive

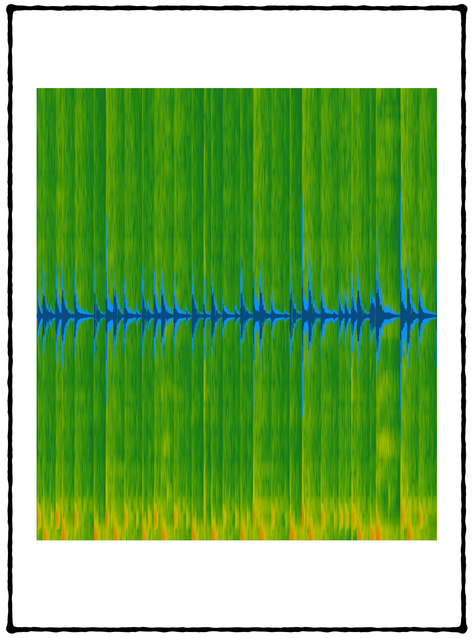
Easier to visually interpret

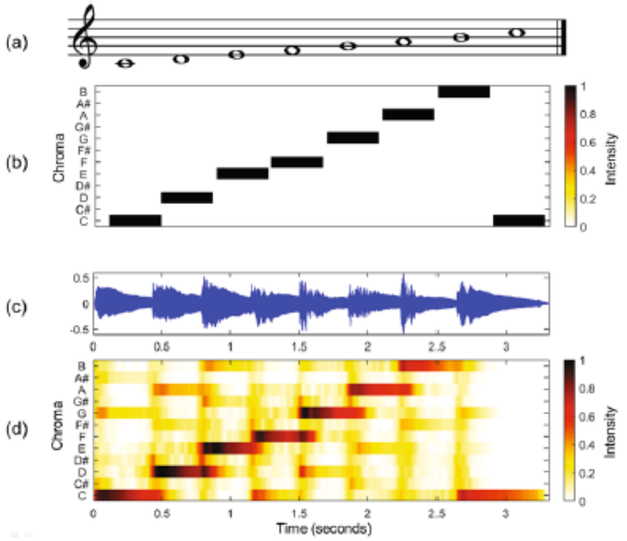
Easier to investigate interdependencies/interlocking of voices

Audio Vs. Symbolic

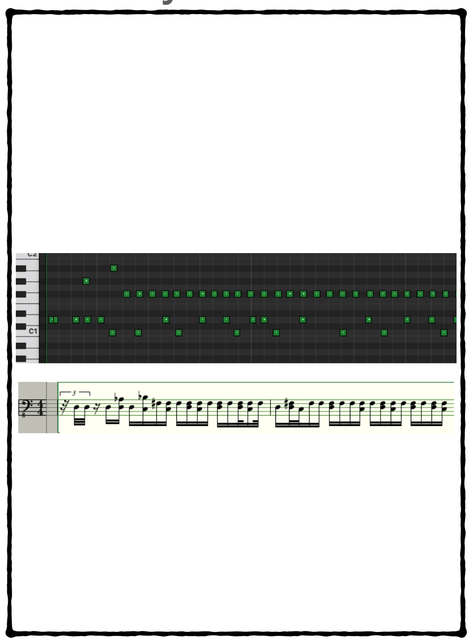


Audio





Symbolic

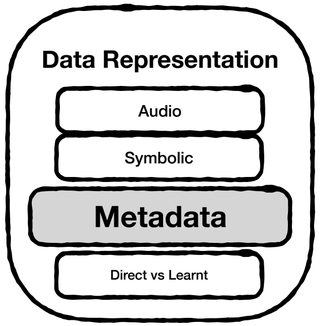


"… the symbolic and the auditory aspects of music representation aren't separate categories but rather the two ends of a continuum. A good example for a commonly used representation that lies somewhere in between these two ends is that of chroma features …" [1]

[1] Liebman, Elad, and Peter Stone. "Artificial Musical Intelligence: A Survey." *arXiv preprint arXiv:2006.10553* (2020)
[2] Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2020 (Page 32-33)

Metadata

Data providing information about other data



Music metadata provide information about aspects of music that do not exist in a recording nor the representations derived from the recording

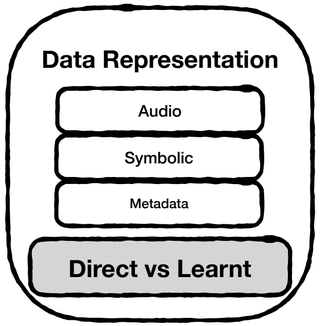
Most common formats are ID3v1 and ID3v2

Some fields in metadata:

1. **Title**
2. **Artist**
3. **Album**
4. **Year**
5. **Track**
6. **Genre**
7. **Location**
8. **Publisher**
9. **Composer**

Direct vs Learnt

Symbolic Data as Text

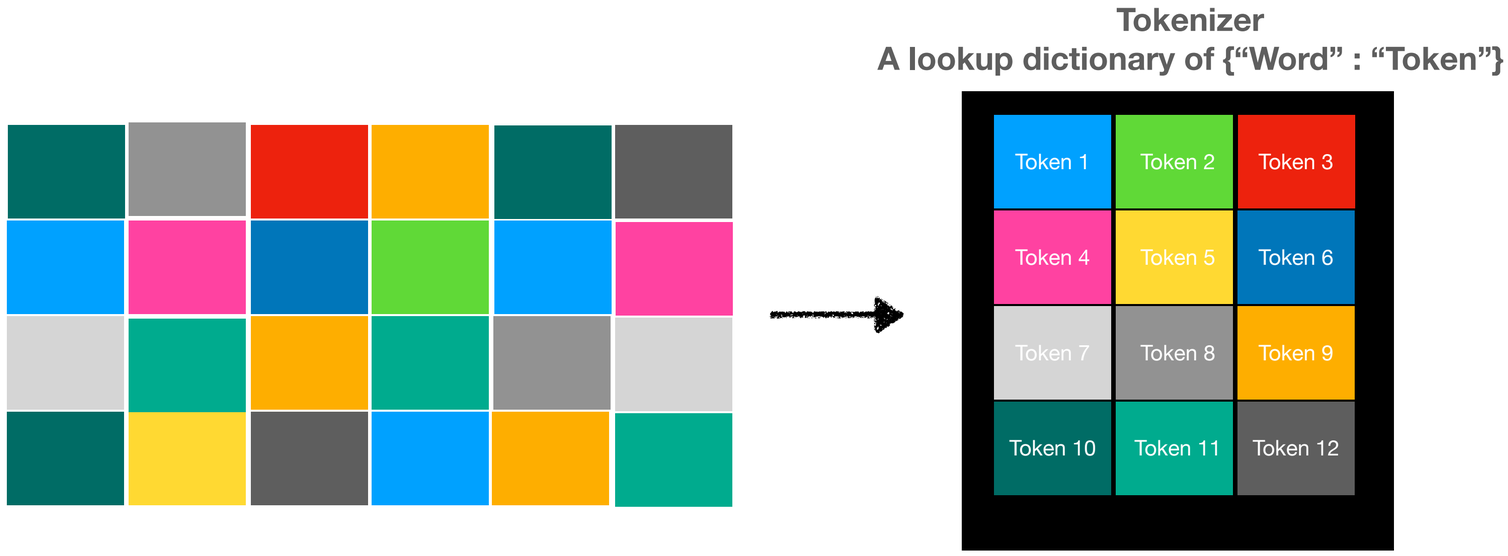


Many generative music models are adapted from NLP.

In NLP words are not numerical (unless you use their character by character ASCII values)

Hence, we need to transform works into numerical data/vectors that can be fed into a NLP model

Let's See How!

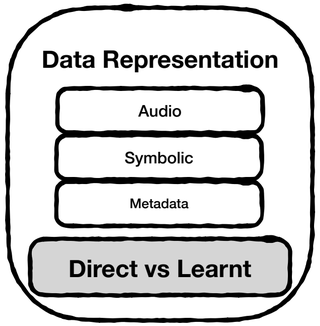


Step 1: Create a set of all unique words existing in the dataset
And then assign a single value to each unique word

In other words, we create a dictionary containing all possible words and assign tokens (numerical identifiers) to each word

Direct vs Learnt

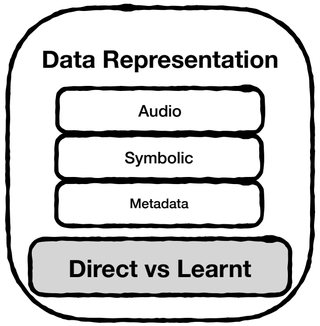
Symbolic Data as Text



Step 2: Use the tokenizer, to get the numerical identifier (token) associated with the input word

Direct vs Learnt

Symbolic Data as Text



An Input Word

Tokenizer

Token 1	Token 2	Token 3
Token 4	Token 5	Token 6
Token 7	Token 8	Token 9
Token 10	Token 11	Token 12

Token = 9

Embedding
A look-up table of {“Token N” : “Representation Vector N”}

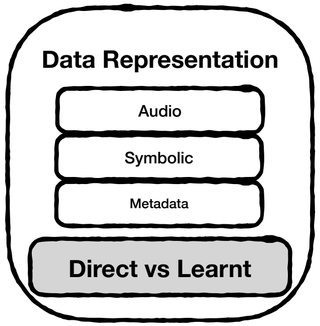
Token 1				
Token 2				
Token 3				
Token 4				
Token 5				
Token 6				
Token 7				
Token 8				
Token 9	0.125	-0.814	0.992	0.002
Token 10				
Token 11				
Token 12				

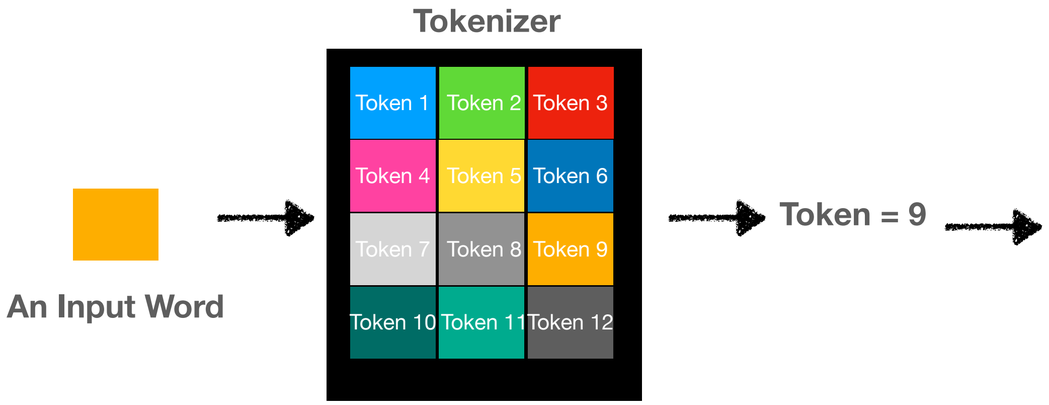
Step 3: Assign a vector to a given token
These vectors can either be **learnt** during training or be re-used from a pre-trained model

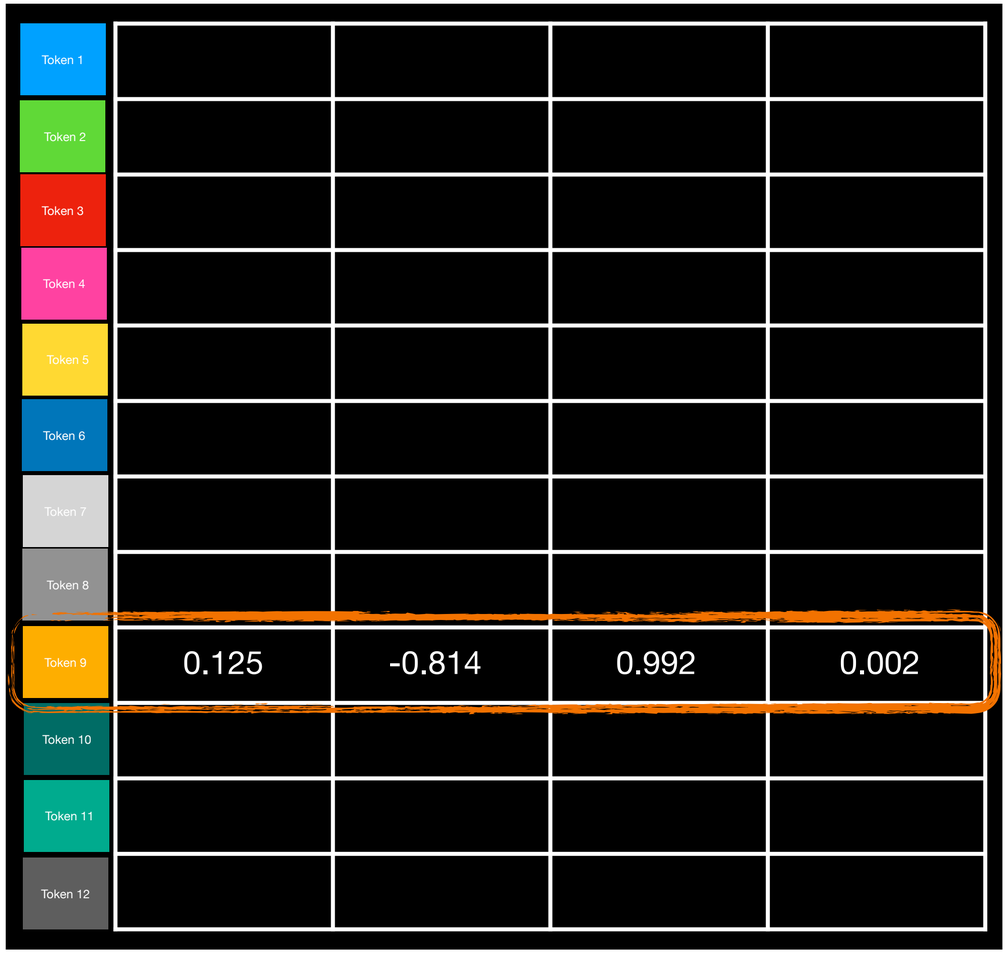
Direct vs Learnt

Symbolic Data as Text

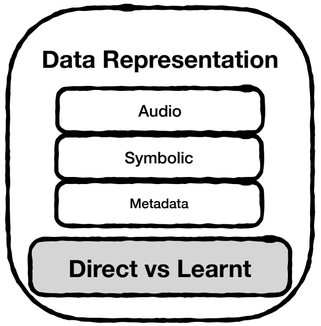
Why do we do this? Why don't we just assign equally distanced vectors to the tokens?







Direct vs Learnt



Symbolic Data as Text

Why do we do this? Why don’t we just assign equally distanced vectors to the tokens?

1. Similar words will end up with closer embeddings!

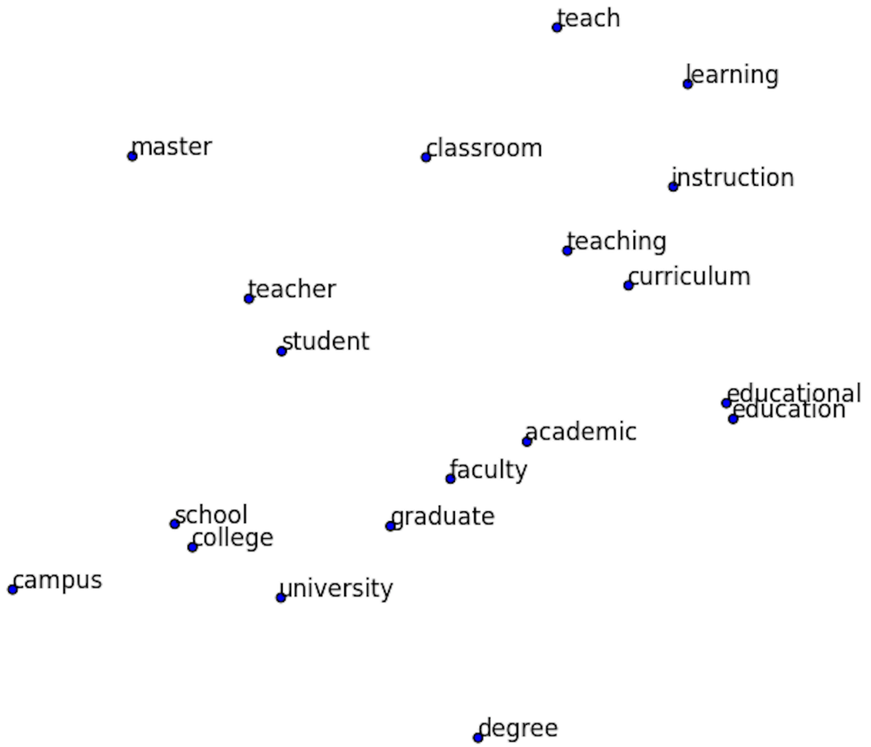
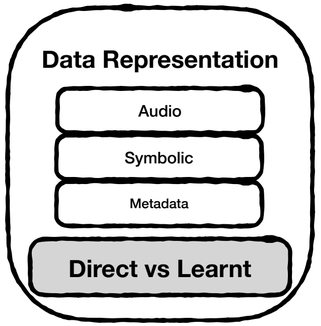


Image from https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12

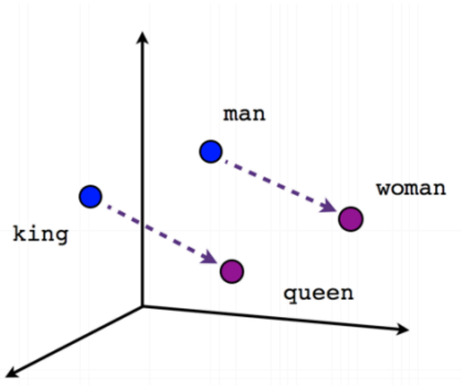
Direct vs Learnt



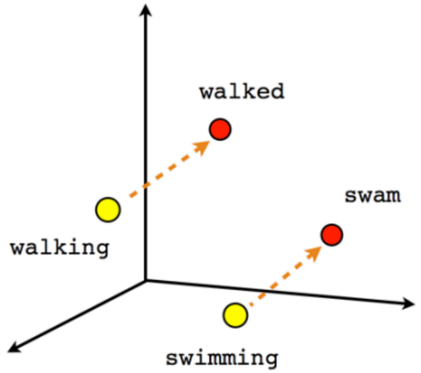
Symbolic Data as Text

Why do we do this? Why don't we just assign equally distanced vectors to the tokens?

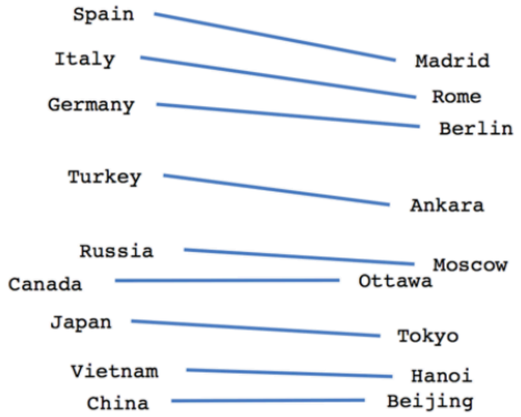
2. The distances between the embeddings are semantically meaningful!



Male-Female



Verb tense

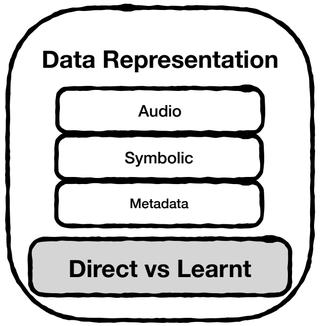


Country-Capital

Plots from https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12

Direct vs Learnt

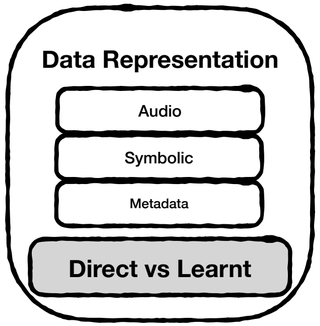
Symbolic Data as Text



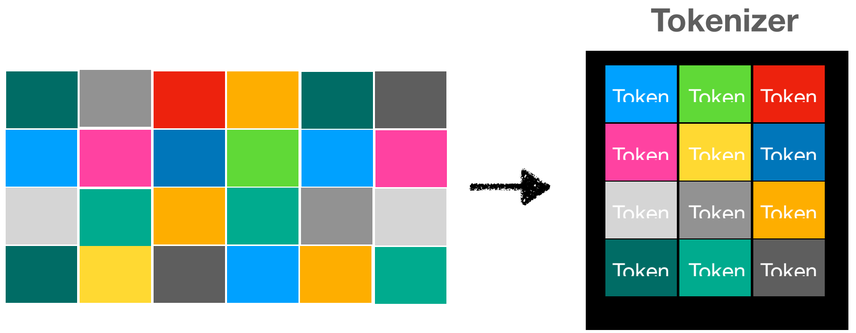
How can we apply the same concept to music?

Direct vs Learnt

Symbolic Data as Text

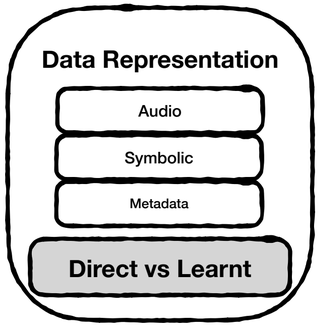


The first intuition is to treat musical events in a score as words or characters!
(This has shown to be quite effective! But …)

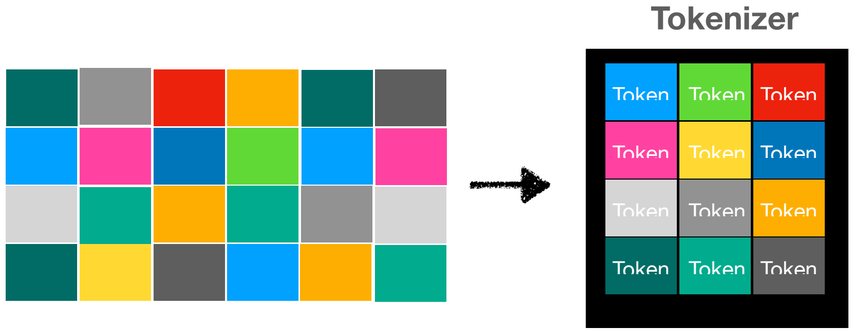


Direct vs Learnt

Symbolic Data as Text



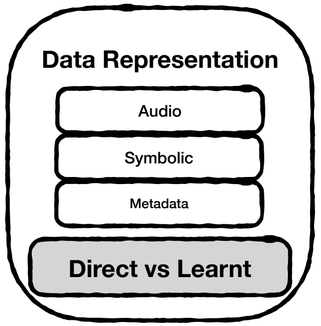
The first intuition is to treat musical events in a score as words or characters!
(This has shown to be quite effective! But …)



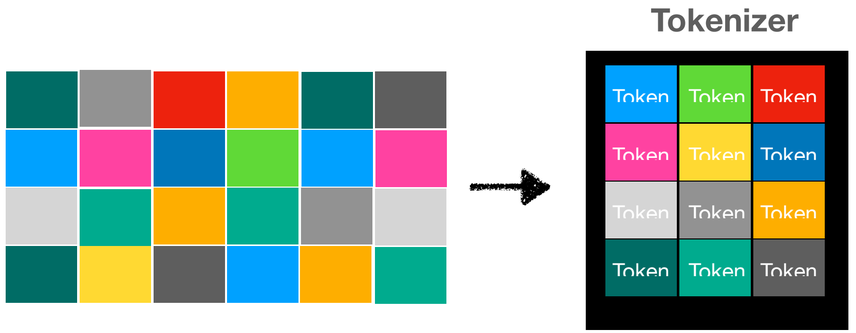
In language, order of words matter. However, in music, not only the order matters but also the timing of the words (or events) matters

Direct vs Learnt

Symbolic Data as Text



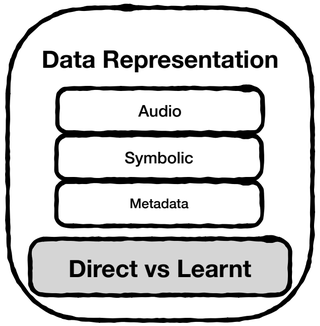
The first intuition is to treat musical events in a score as words or characters!
(This has shown to be quite effective! But …)



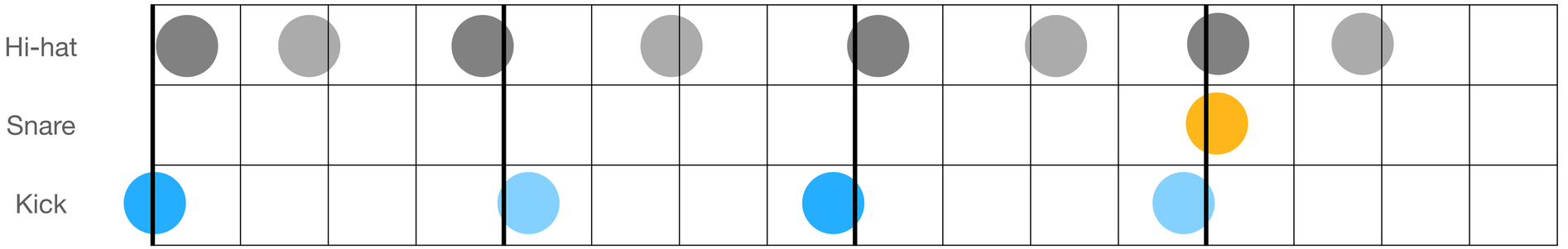
As opposed to language, the “semantic” meaning of musical events is highly dependant on the context in which their used

Direct vs Learnt

Symbolic Data as Text

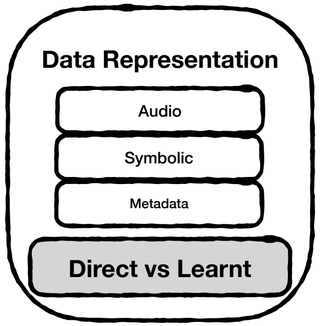


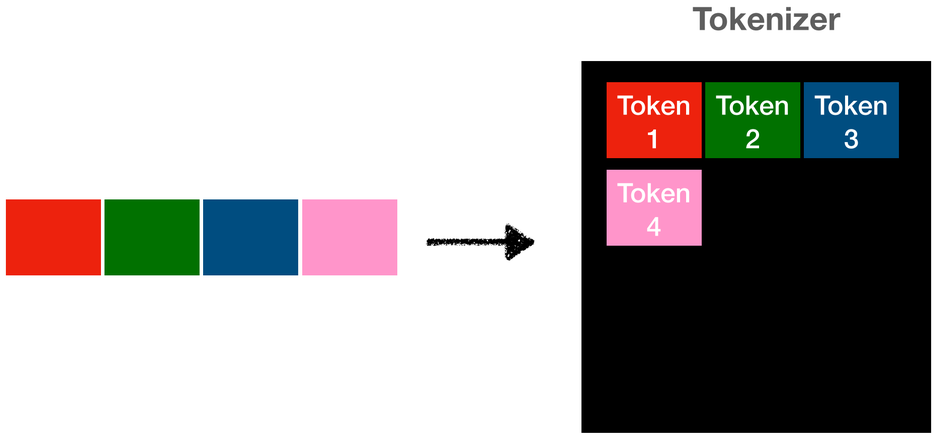
How can we represent this pattern?

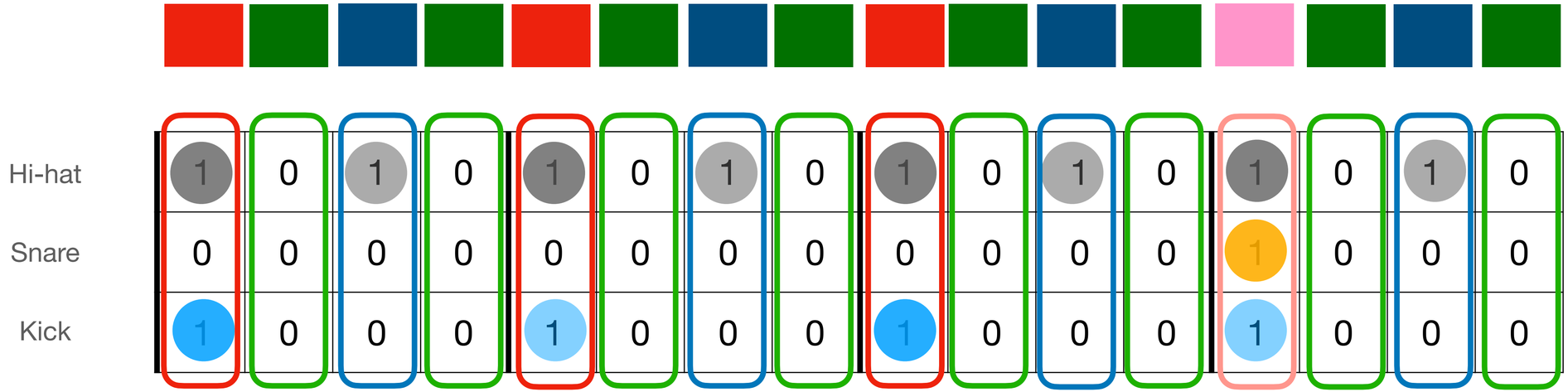


Direct vs Learnt

Symbolic Data as Text

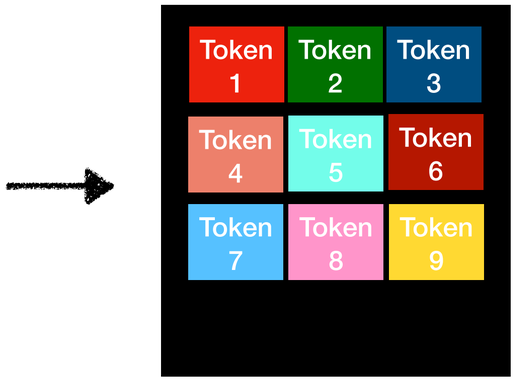


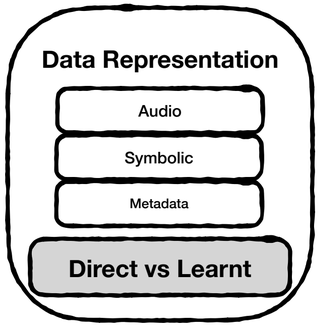




Direct vs Learnt

Symbolic Data as Text

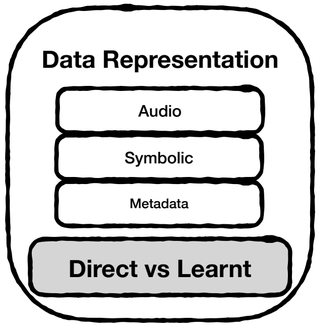




Hi-hat Velocity	1	0	0.7	0	0.78	0	0.62	0	0.73	0	0.54	0	0.89	0	0.51	0
Snare Velocity	0	0	0	0	0	0	0	0	0	0	0	0	0.64	0	0	0
Kick Velocity	1	0	0	0	0.4	0	0	0	0.9	0	0	0	0.6	0	0	0
Hi-hat	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Snare	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Kick	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Direct vs Learnt

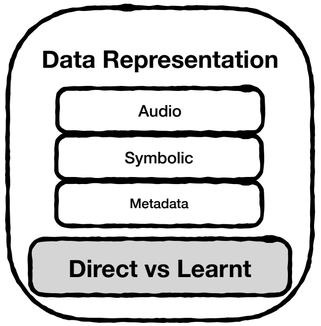
Symbolic Data as Text



Hi-hat uTiming	0.2	0	0.42	0	-0.31	0	0.2	0	0.4	0	0	0	0.23	0	0.18	0
Snare uTiming	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kick uTiming	0.4	0	-0.2	0	0.23	0	0	0	0	0	0	0	-0.35	0	0	0
Hi-hat Velocity	1	0	0.7	0	0.78	0	0.62	0	0.73	0	0.54	0	0.89	0	0.51	0
Snare Velocity	0	0	0	0	0	0	0	0	0	0	0	0	0.64	0	0	0
Kick Velocity	1	0	0	0	0.4	0	0	0	0.9	0	0	0	0.6	0	0	0
Hi-hat	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Snare	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Kick	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Direct vs Learnt

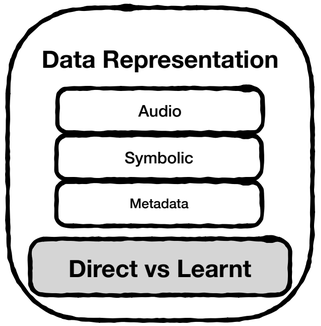
Symbolic Data as Text



Hi-hat uTiming	0.2	0	0.42	0	-0.31	0	0.2	0	0.4	0	0	0	0.23	0	0.18	0
Snare uTiming	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kick uTiming	0.4	0	-0.2	0	0.23	0	0	0	0	0	0	0	-0.35	0	0	0
Hi-hat Velocity	1	0	0.7	0	0.78	0	0.62	0	0.73	0	0.54	0	0.89	0	0.51	0
Snare Velocity	0	0	0	0	0	0	0	0	0	0	0	0	0.64	0	0	0
Kick Velocity	1	0	0	0	0.4	0	0	0	0.9	0	0	0	0.6	0	0	0
Hi-hat	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Snare	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Kick	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Direct vs Learnt

Symbolic Data as Text

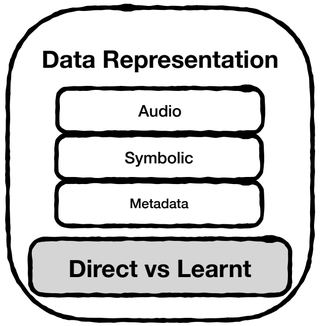


Tokenization requires quantizing time and dynamics

Quantizing certain aspects like time (specially in a performance) is quite hard and sometime even impractical.

Direct vs Learnt

Symbolic Data as Text



If we decide to quantize timing, how would we do it here?

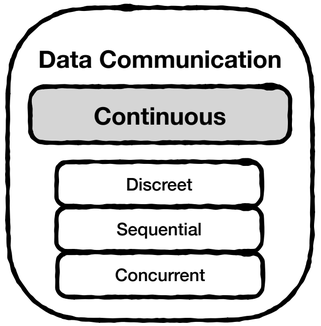
https://youtube.com/watch?v=E9oW5qojs_g

Part B: Data Communication*

* Mainly Focused on Pd & Python Communication

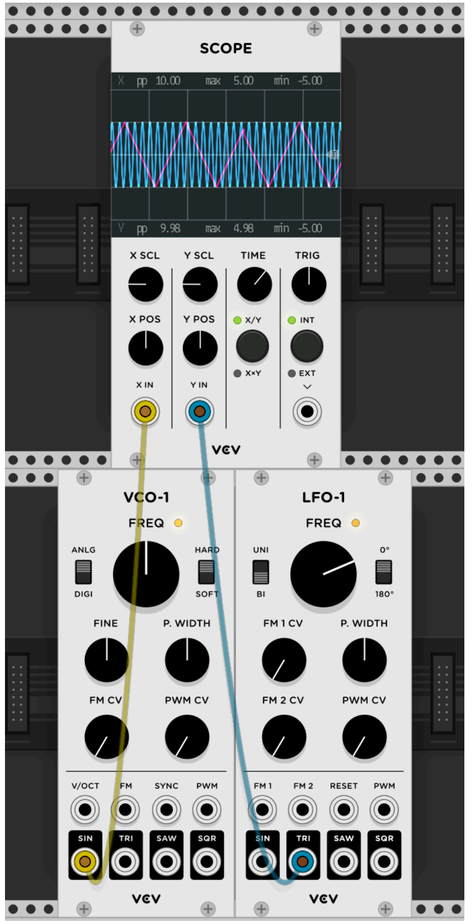
(Pseudo*) Continuous Signals

Audio, CV (Control Voltage)



Control Voltages were the first (sort-of) standardized protocol for inter-device communication

CV can be used for controlling, modulating (or automating) processes



<u>Common Usage:</u>

Trigger —> A voltage burst (or a short pulse) triggering an action (similar to pd bang objects) - Think of it as triggering a drum pad!

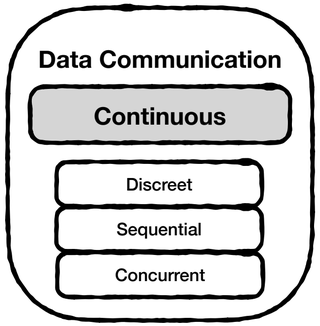
Gate —> A sustained trigger (Note-on, Note-off indicator - think of it as holding down a key)

Pitch —> Voltage level the amount of pitch variation (either in Volts/Octave or Hz/Volts)

* In the digital domain, there are no continuous signals. Hence, in the digital domain, we emulate continuous streams of data through audio rate streams of discreet data

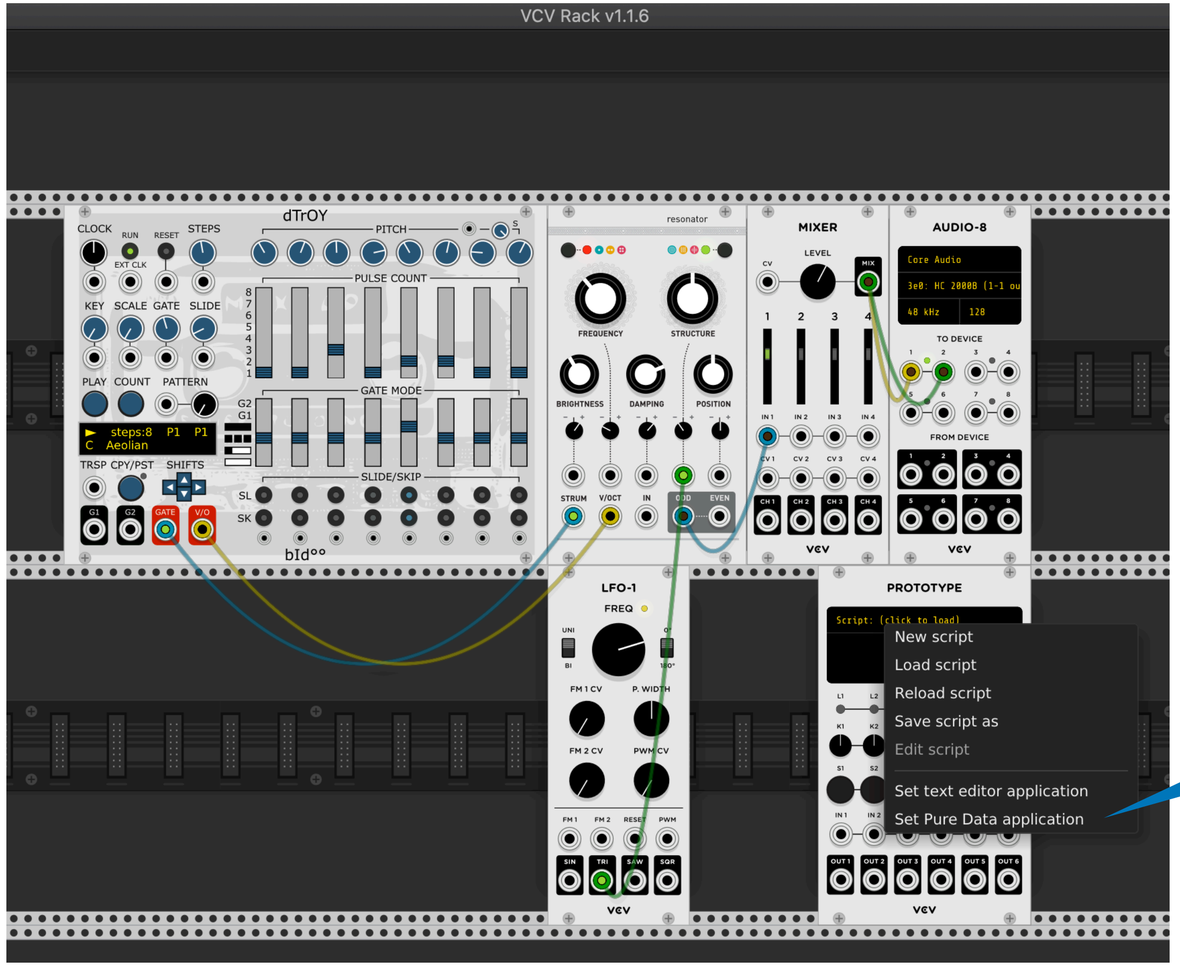
(Pseudo) Continuous Signals

Audio, CV (Control Voltage)



VCV Rack
(A Free, Open-Source, Cross-Platform Virtual Modular Synth Environment)

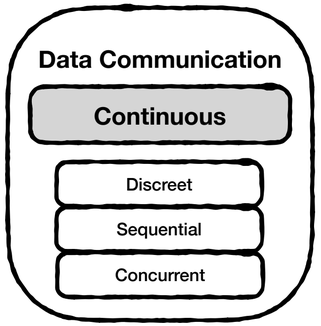
Most communications are through Audio Signals or Control Voltages



Latest updates allow for integrating pd patches inside VCV environment

(Pseudo) Continuous Signals

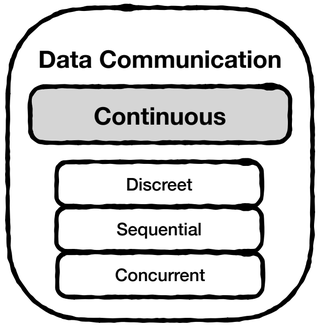
Audio, CV (Control Voltage)



AUDIO	CV
AC Voltages	Can be DC
Fast Oscillation Rates	Lower Oscillation Rates
Audible	Not Necessarily Audible

(Pseudo) Continuous Signals

Audio, CV (Control Voltage)



AUDIO	CV
AC Voltages	Can be DC
Fast Oscillation Rates	Lower Oscillation Rates
Audible	Not Necessarily Audible

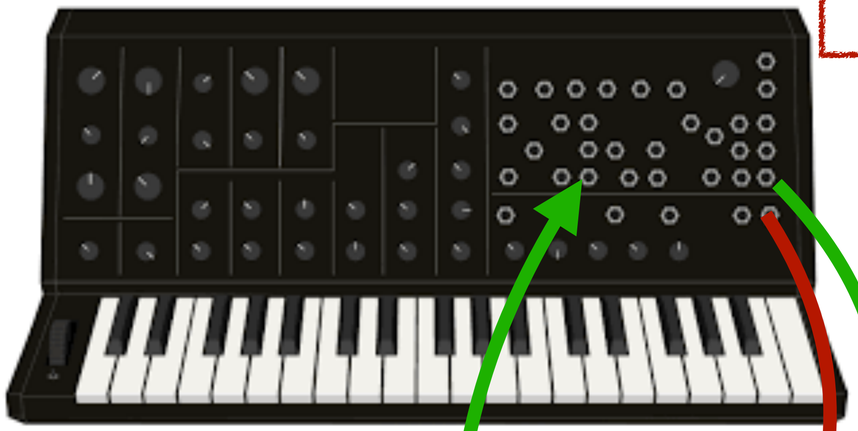
Most Audio Interfaces filter out DC voltages (or very low frequency signals)

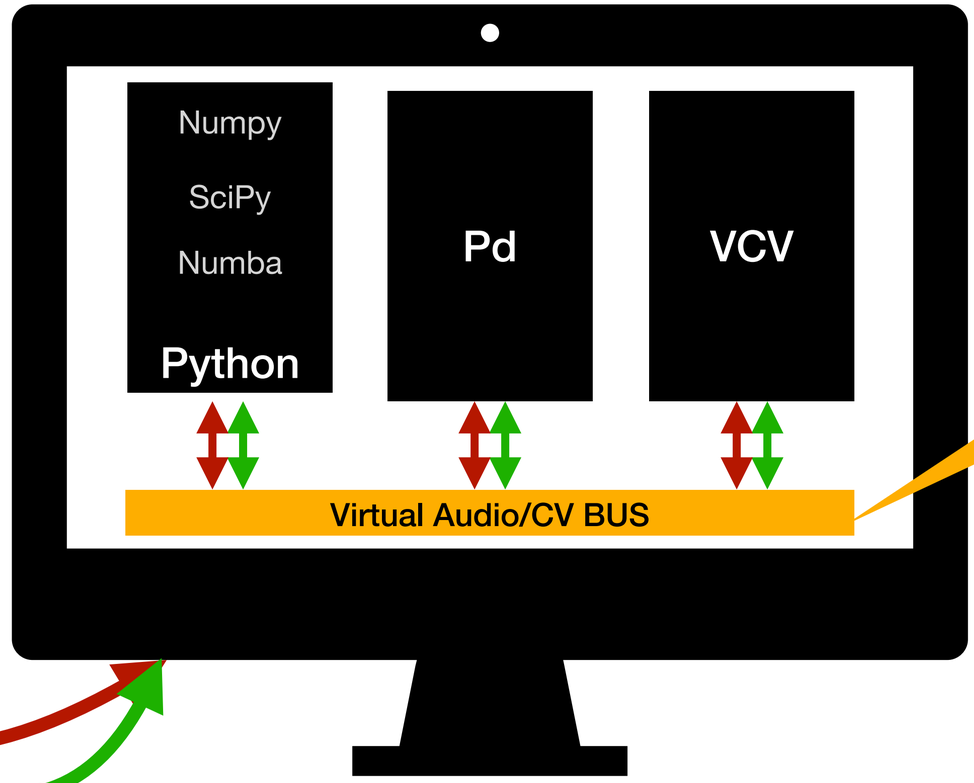
(Pseudo) Continuous Signals

Audio, CV (Control Voltage)

A very interesting watch on live coding in Python:
Drop the DAW - Sound Design in Python by Isaac Roberts at ADC20

https://www.youtube.com/watch?v=Q40qEg8Yq5c&list=LL&index=7&t=2376s





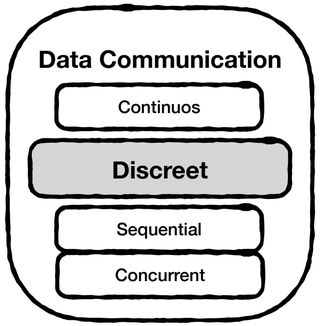
LINUX
Pavucontrol

Mac OS
Soundflower

Windows
Voicemeeter Banana

Audio CV

Discreet Signals



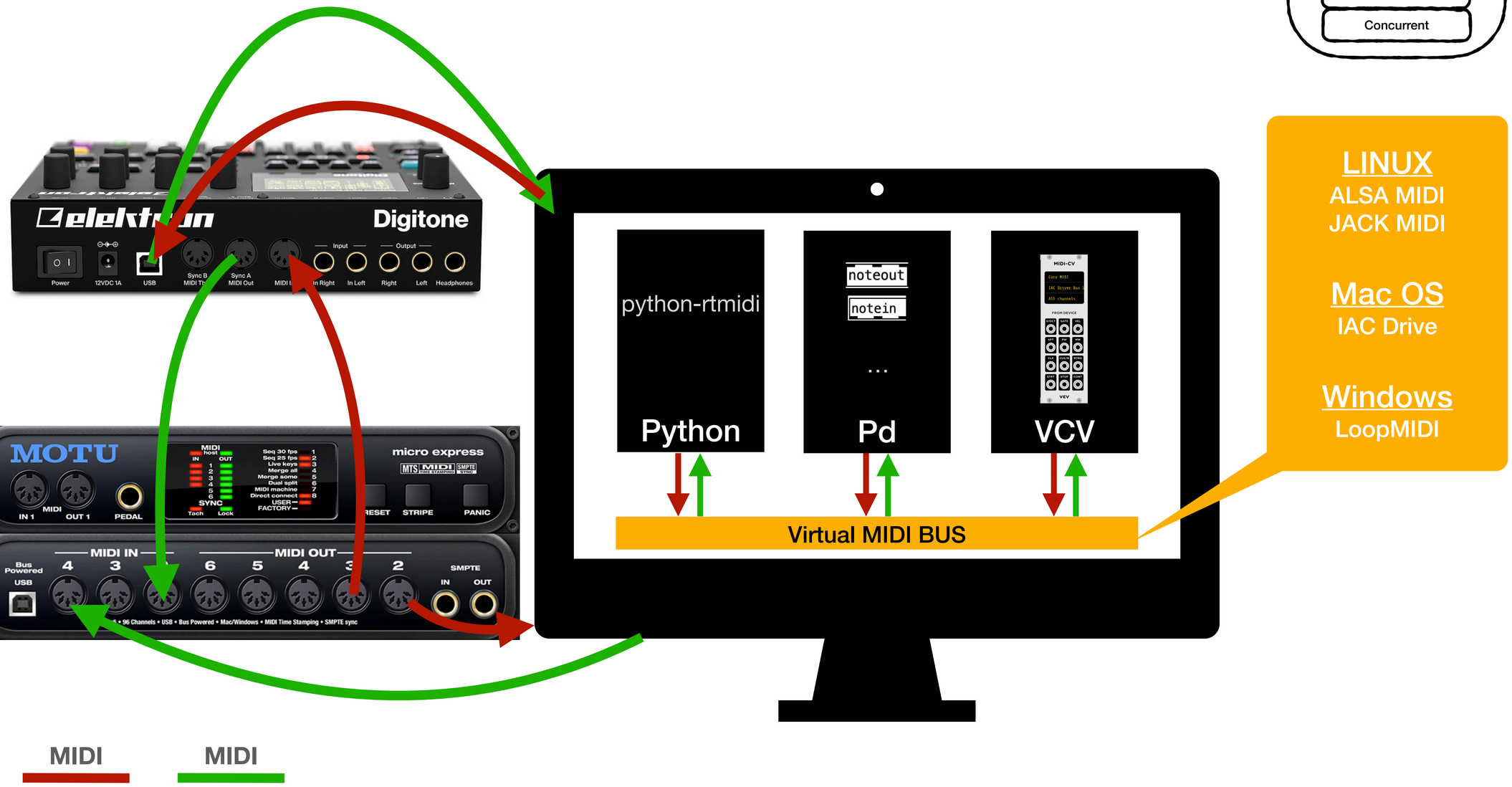
Three common protocols for inter-device, inter-process communication

1. **MIDI**
2. **Text**
3. **OSC**

* In the digital domain, there are no continuous signals. Hence, in the digital domain, we emulate continuous streams of data through audio rate streams of discreet data

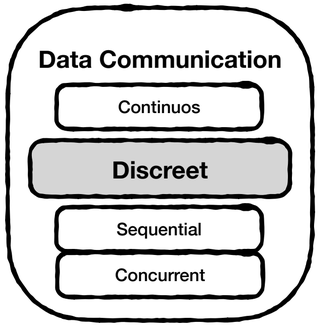
Discreet Signals

MIDI



Discreet Signals

Text



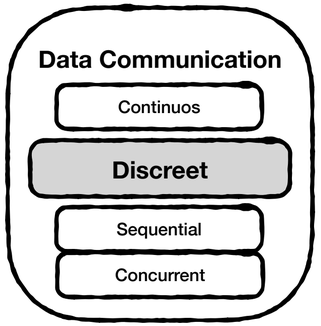
Text can be used as a way of communicating between processes:

1. **Write/Read to/from file —> Extremely slow**
2. **Use UDP/TCP protocols**

* In the digital domain, there are no continuous signals. Hence, in the digital domain, we emulate continuous streams of data through audio rate streams of discreet data

Discreet Signals

Open Sound Control



“The basic unit of OSC is a message, consisting of an Address Pattern (AP), a Type Tag String (TTS), an optional time tag, and arguments.” [1]

Accepted Type Tags:

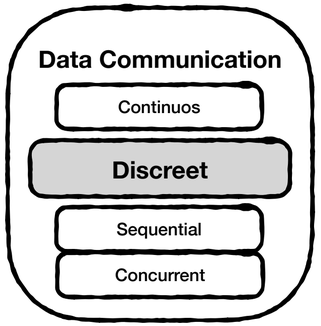
- **Integer (tt: ‘i’)**
- **Floating Number (tt: ‘f’)**
- **Strings of Text (tt: ’s’)**

The unit of transmission of OSC is an *OSC Packet*. Any application that sends OSC Packets is an *OSC Client*; any application that receives OSC Packets is an *OSC Server*.

[1] https://thewizardofosc.com/more-on-osc/
[2] https://web.archive.org/web/20030914224904/http://cnmat.berkeley.edu/OSC/OSC-spec.html

Discreet Signals

Open Sound Control (Address Space)



/body/arm/left/hand/position (0.2, 0.5, 0.9)

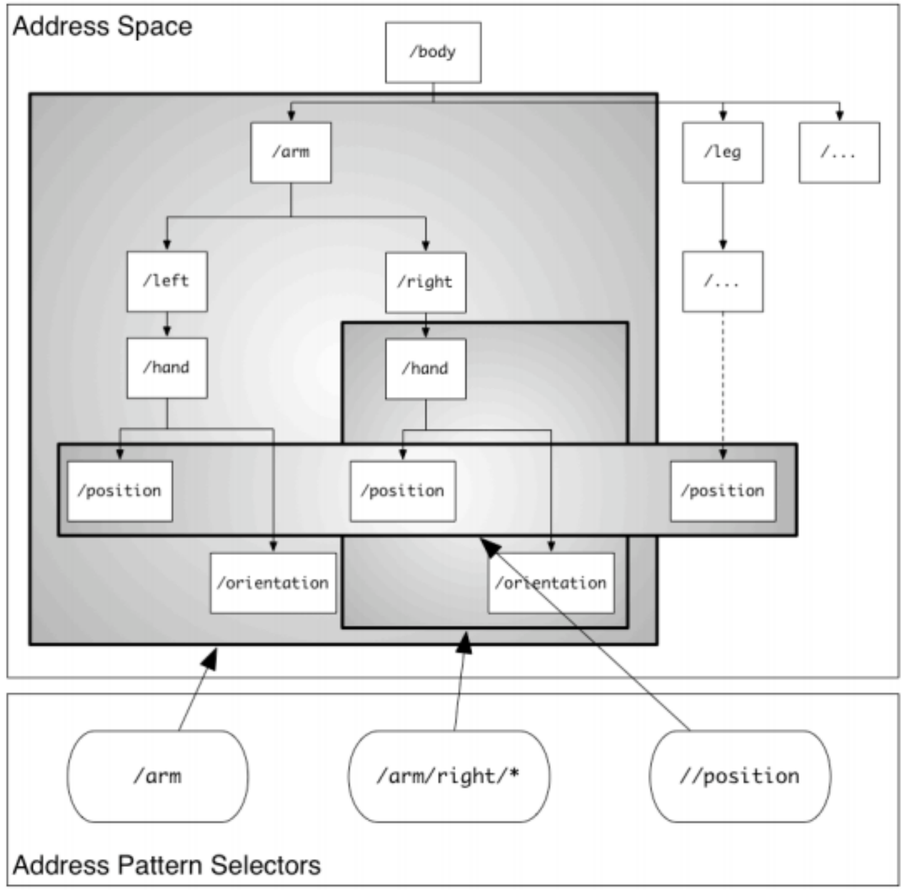


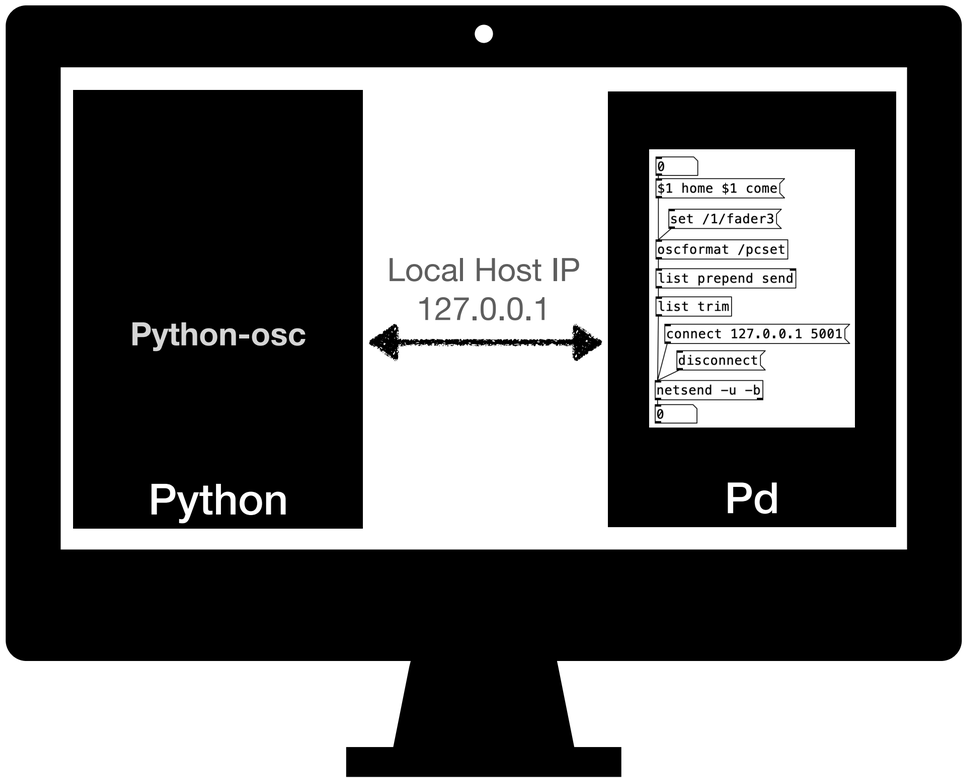
Image from: [1] Freed, Adrian, and Andrew Schmeder. "Features and Future of Open Sound Control version 1.1 for NIME." *NIME*. Vol. 4. No. 06. 2009.

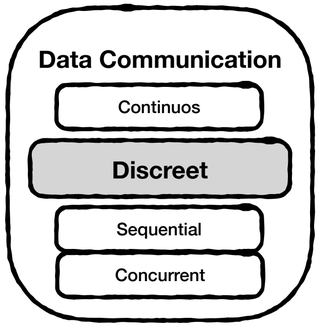
Discreet Signals

Open Sound Control

LOCAL_IP_ADDRESS_ORGANELLE

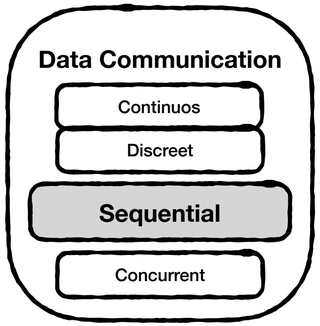
LOCAL_IP_ADDRESS_PC

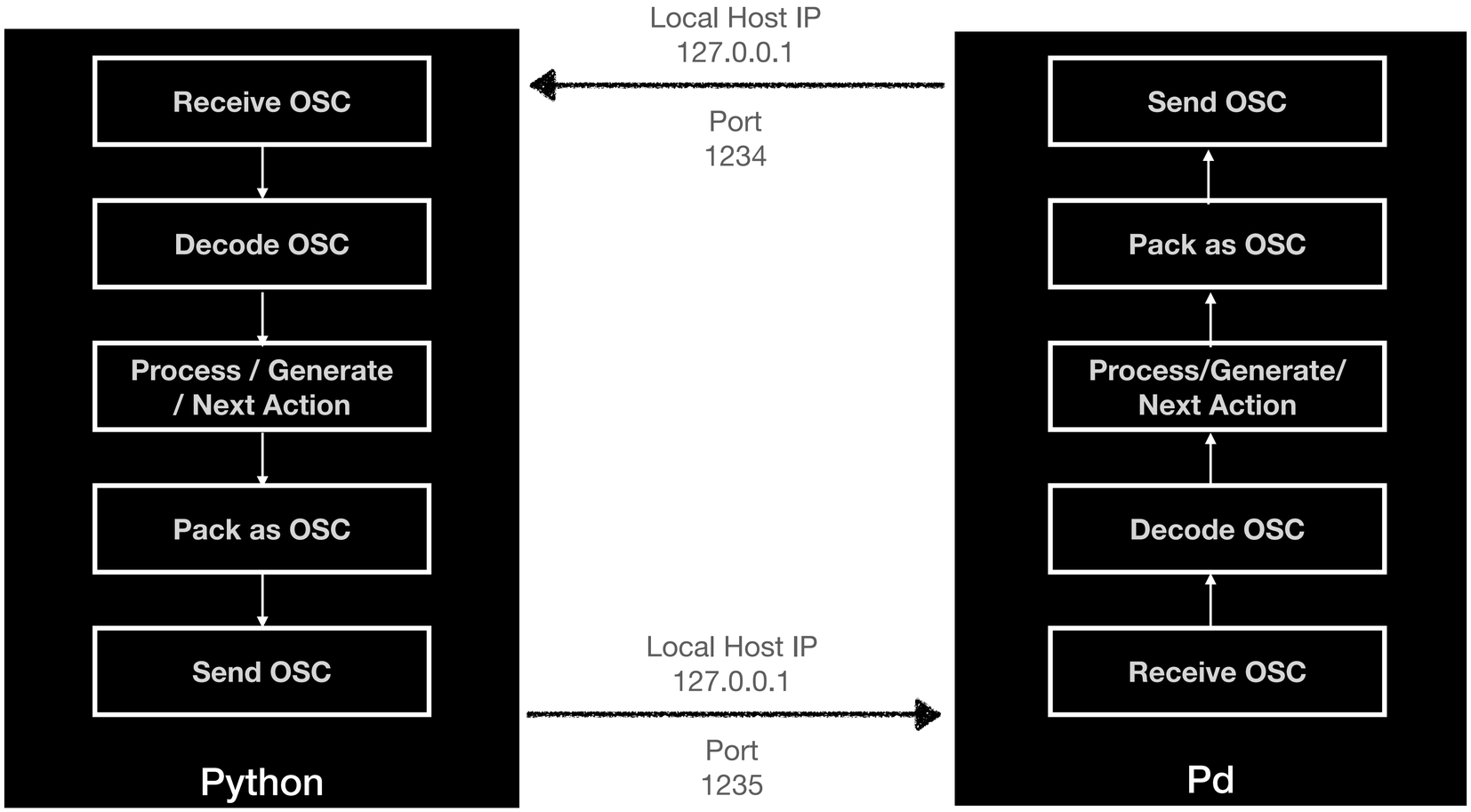




Processes/Devices on the same network can communicate OSC messages via their IP address and an available Port

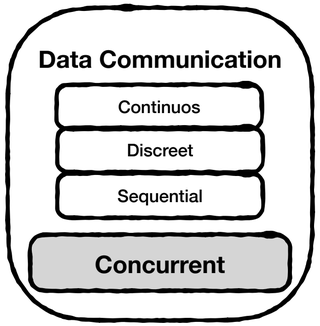
Sequential Communication

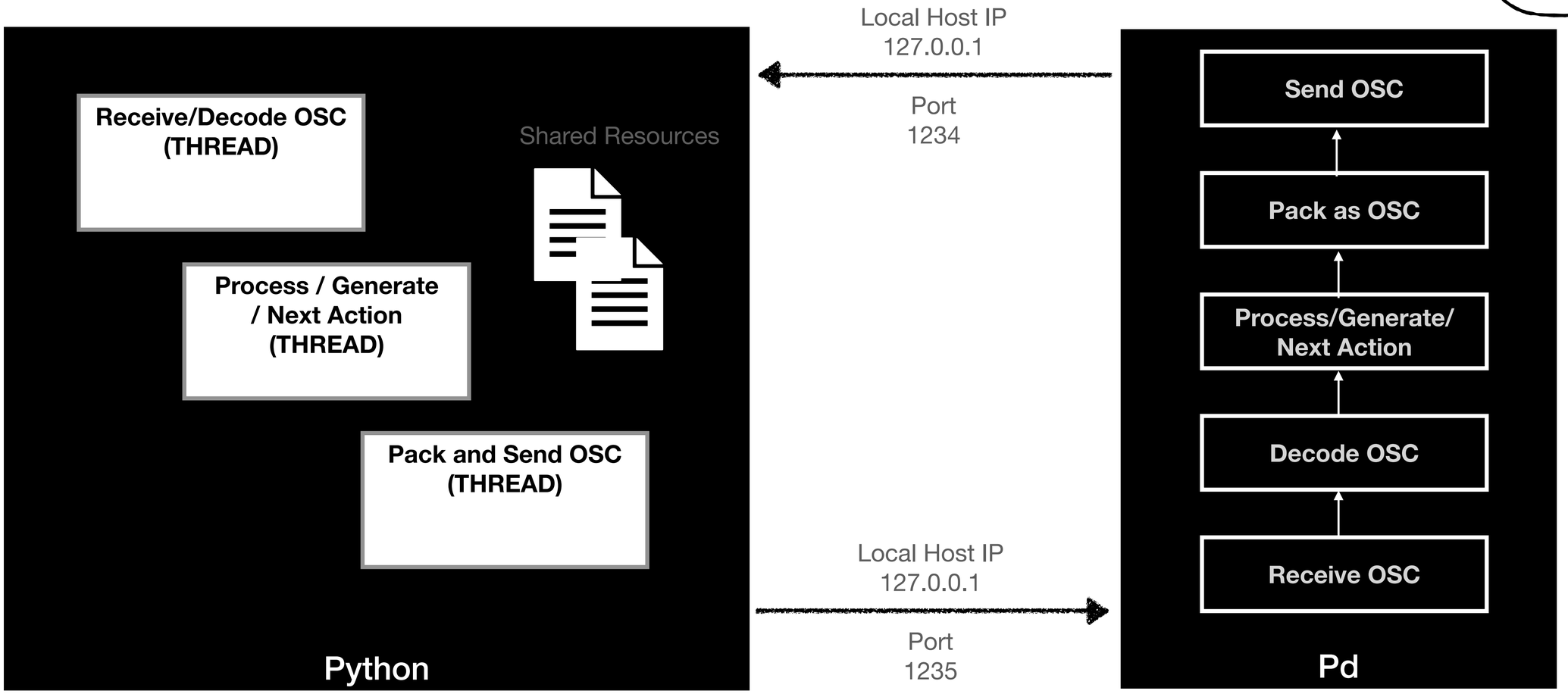




Works well in non-realtime use cases! But quite unreliable for realtime implementation

Concurrent Communication





Update Resources concurrently whenever new data is available,
process / stream the already available data in separate parallel threads

Questions?