MusPy: Symbolic Music Processing in Python

Hao-Wen Dong Ke Chen Julian McAuley Taylor Berg-Kirkpatrick

University of California San Diego




UC San Diego

About me

Hi, I'm Herman.
I do Music x AI research.
I love music and movies!

National Taiwan University
B.S. in Electrical Engineering
2013 – 2017

2017 – 2019
Academia Sinica
Research Assistant

Summer 2019
Yamaha
Research Intern

University of California San Diego
M.S. in Computer Science
2019 – 2021

Summer 2021
Dolby
Deep Learning Audio Intern

University of California San Diego
PhD Candidate in Computer Science
UC San Diego
2019 – present

(Summer 2022)
Sony
Student Intern

(Fall 2022)
Amazon
Applied Scientist Intern

Outline

- Intro to MusPy
- Experiments
- Case Study I – Automatic Instrumentation
- Case Study II – Music Performance Synthesis

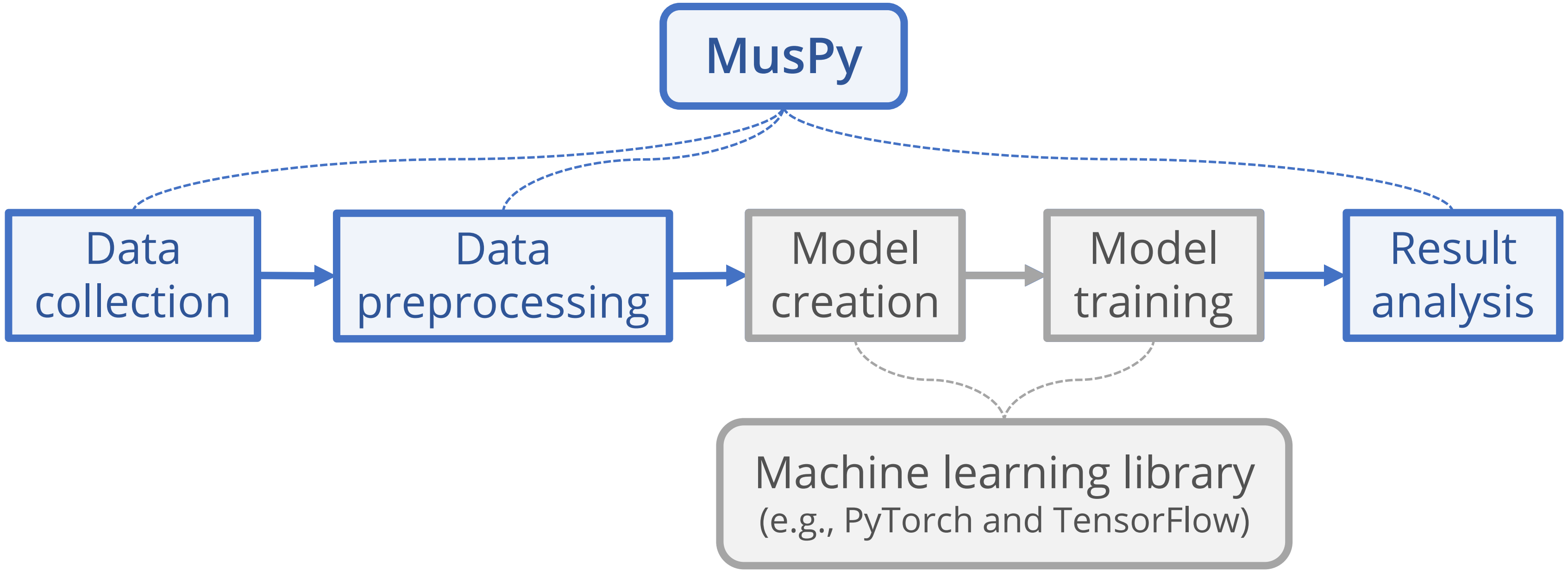
Intro to MusPy

"MusPy: A Toolkit for Symbolic Music Generation" (ISMIR 2020)

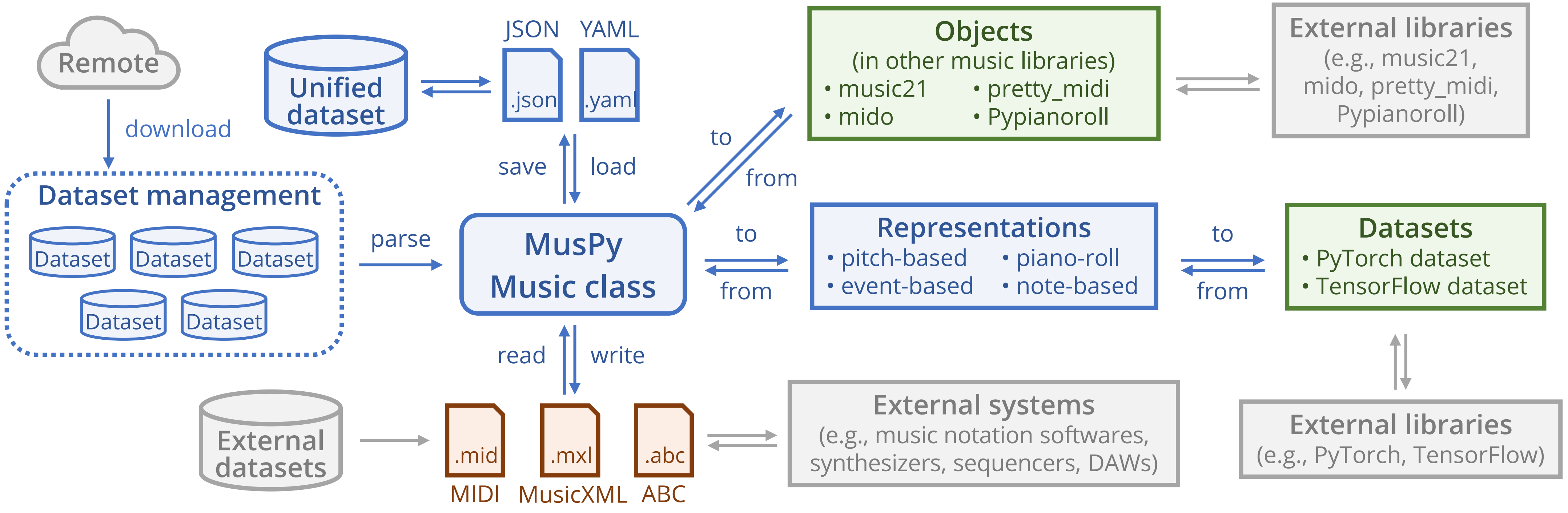
Hao-Wen Dong   Ke Chen   Julian McAuley   Taylor Berg-Kirkpatrick

Why MusPy?

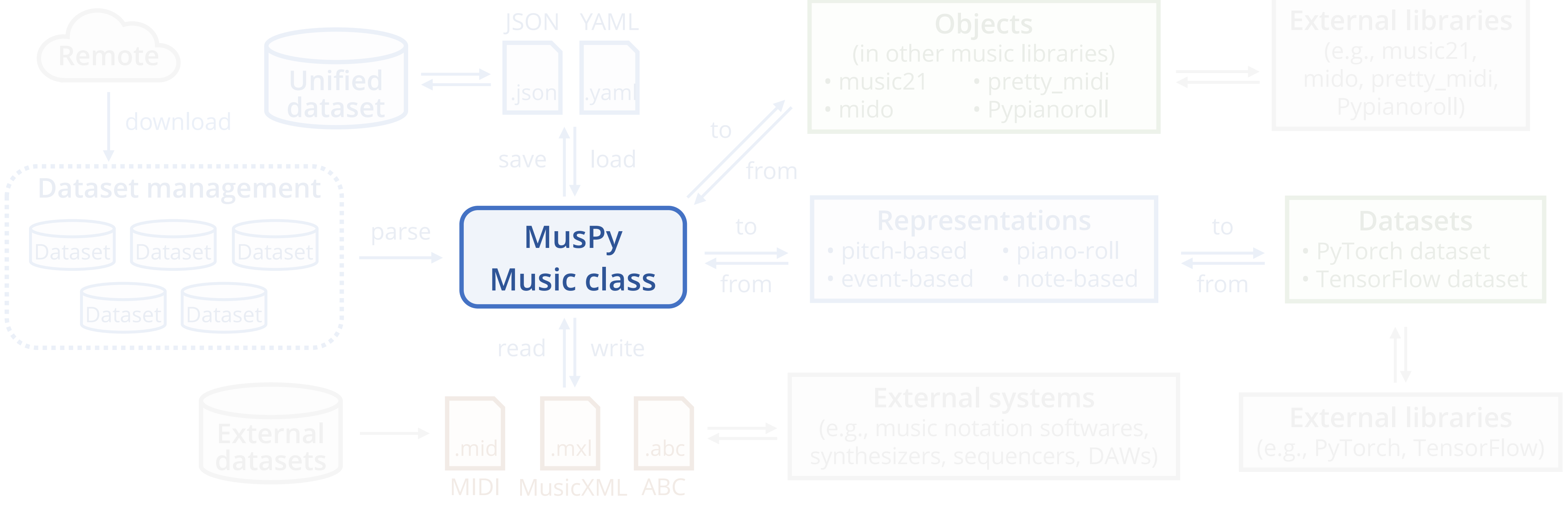

MusPy
Data collection
Data preprocessing
Model creation
Model training
Result analysis
Machine learning library
(e.g., PyTorch and TensorFlow)

Overview



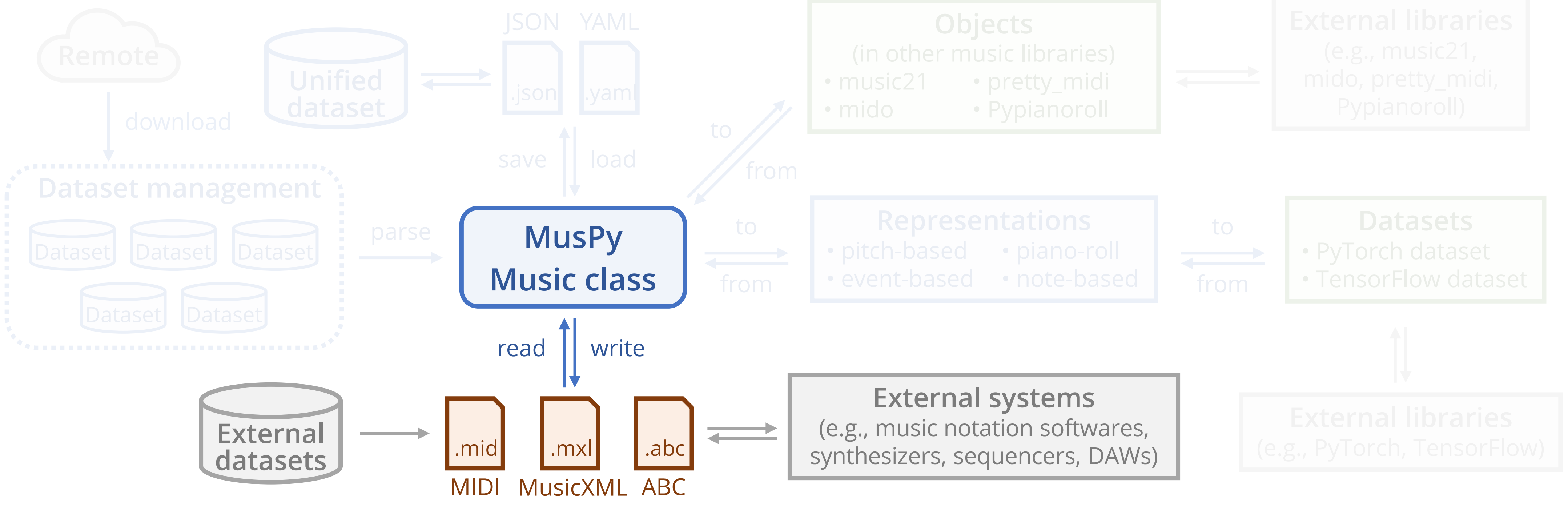
Remote
download
Dataset management
Dataset
Dataset
Dataset
Dataset
Dataset
Unified dataset
JSON
YAML
.json
.yaml
save
load
parse
MusPy Music class
read
write
External datasets
.mid
.mxl
.abc
MIDI
MusicXML
ABC
to
from
Objects
(in other music libraries)
• music21
• pretty_midi
• mido
• Pypianoroll
External libraries
(e.g., music21, mido, pretty_midi, Pypianoroll)
to
from
Representations
• pitch-based
• piano-roll
• event-based
• note-based
to
from
Datasets
• PyTorch dataset
• TensorFlow dataset
External systems
(e.g., music notation softwares, synthesizers, sequencers, DAWs)
External libraries
(e.g., PyTorch, TensorFlow)

Overview



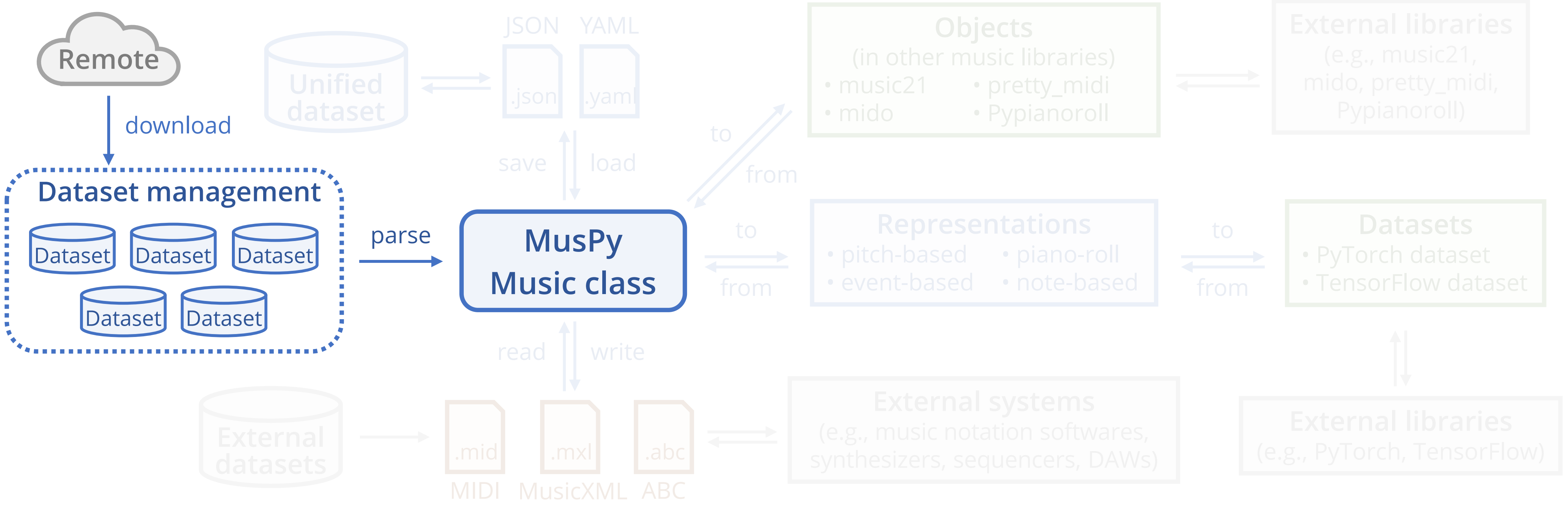
Remote
download
Unified dataset
JSON
YAML
.json
.yaml
save
load
Dataset management
Dataset
Dataset
Dataset
Dataset
Dataset
parse
MusPy Music class
read
write
External datasets
.mid
.mxl
.abc
MIDI
MusicXML
ABC
to
from
Objects (in other music libraries)
• music21
• mido
• pretty_midi
• Pypianoroll
External libraries (e.g., music21, mido, pretty_midi, Pypianoroll)
to
from
Representations
• pitch-based
• event-based
• piano-roll
• note-based
to
from
Datasets
• PyTorch dataset
• TensorFlow dataset
External systems (e.g., music notation softwares, synthesizers, sequencers, DAWs)
External libraries (e.g., PyTorch, TensorFlow)

Overview



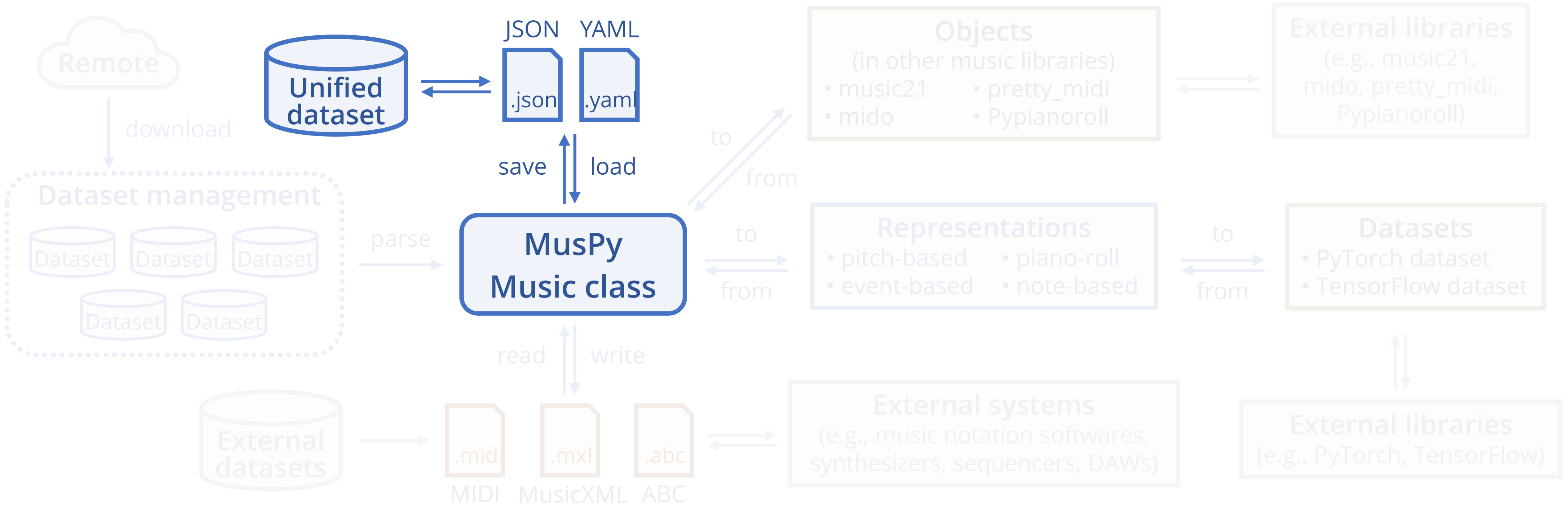
Remote
download
Unified dataset
JSON
YAML
.json
.yaml
save
load
Dataset management
Dataset
Dataset
Dataset
Dataset
Dataset
parse
MusPy Music class
to
from
Objects
(in other music libraries)
• music21
• pretty_midi
• mido
• Pypianoroll
External libraries
(e.g., music21, mido, pretty_midi, Pypianoroll)
to
from
Representations
• pitch-based
• piano-roll
• event-based
• note-based
to
from
Datasets
• PyTorch dataset
• TensorFlow dataset
read
write
External datasets
.mid
.mxl
.abc
MIDI
MusicXML
ABC
External systems
(e.g., music notation softwares, synthesizers, sequencers, DAWs)
External libraries
(e.g., PyTorch, TensorFlow)

Overview



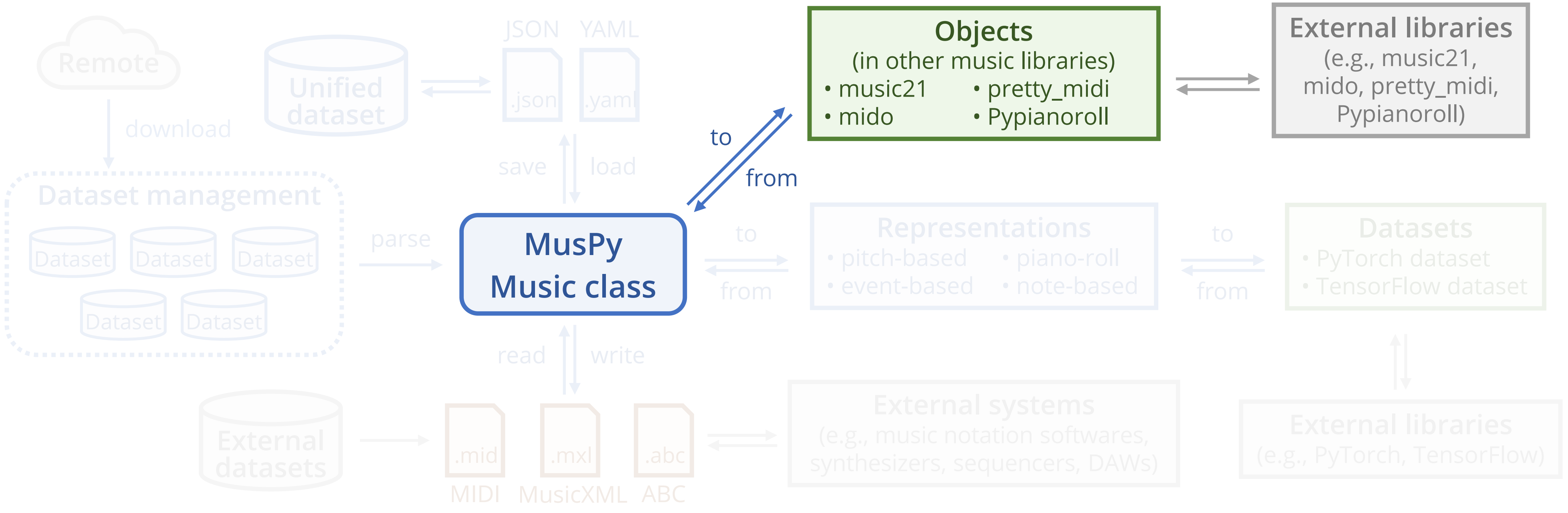
Remote
download
Dataset management
Dataset
Dataset
Dataset
Dataset
Dataset
parse
MusPy
Music class
Unified dataset
JSON
YAML
.json
.yaml
save
load
to
from
Objects
(in other music libraries)
• music21
• pretty_midi
• mido
• Pypianoroll
External libraries
(e.g., music21, mido, pretty_midi, Pypianoroll)
to
from
Representations
• pitch-based
• piano-roll
• event-based
• note-based
to
from
Datasets
• PyTorch dataset
• TensorFlow dataset
read
write
External datasets
.mid
.mxl
.abc
MIDI
MusicXML
ABC
External systems
(e.g., music notation softwares, synthesizers, sequencers, DAWs)
External libraries
(e.g., PyTorch, TensorFlow)

Overview

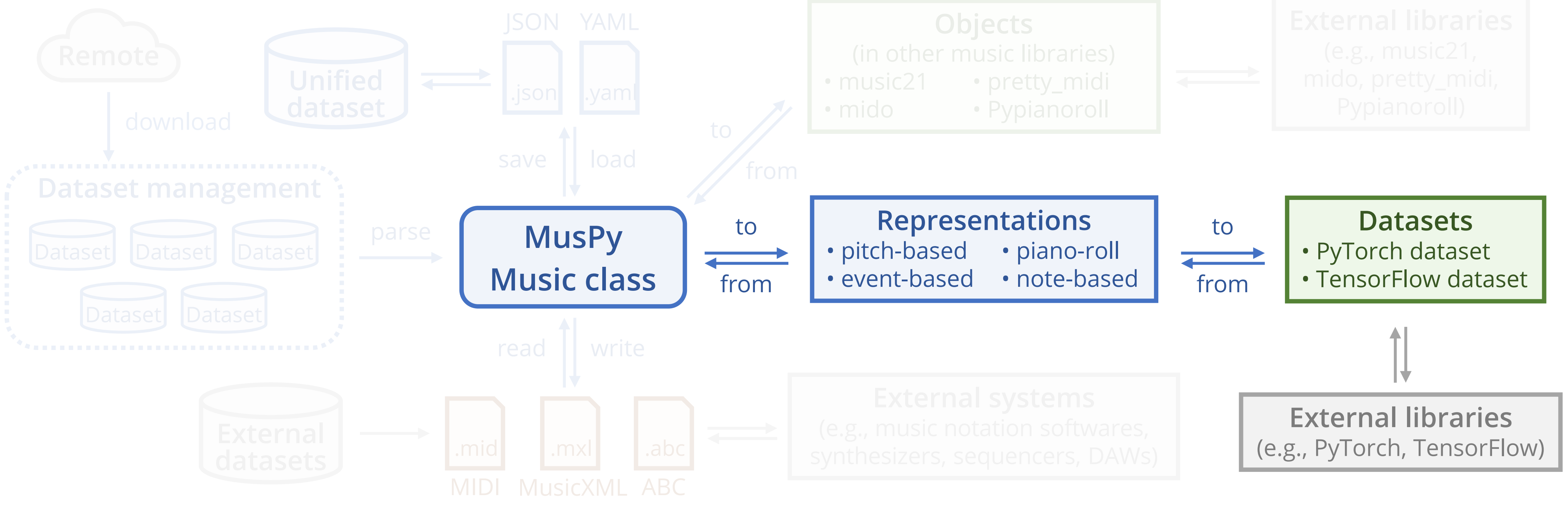


Remote
download
Dataset management
Dataset
Dataset
Dataset
Dataset
Dataset
parse
Unified
dataset
JSON
YAML
.json
.yaml
save
load
MusPy
Music class
read
write
External
datasets
.mid
.mxl
.abc
MIDI
MusicXML
ABC
to
from
Objects
(in other music libraries)
• music21
• pretty_midi
• mido
• Pypianoroll
External libraries
(e.g., music21,
mido, pretty_midi,
Pypianoroll)
to
from
Representations
• pitch-based
• piano-roll
• event-based
• note-based
to
from
Datasets
• PyTorch dataset
• TensorFlow dataset
External systems
(e.g., music notation softwares,
synthesizers, sequencers, DAWs)
External libraries
(e.g., PyTorch, TensorFlow)

Overview

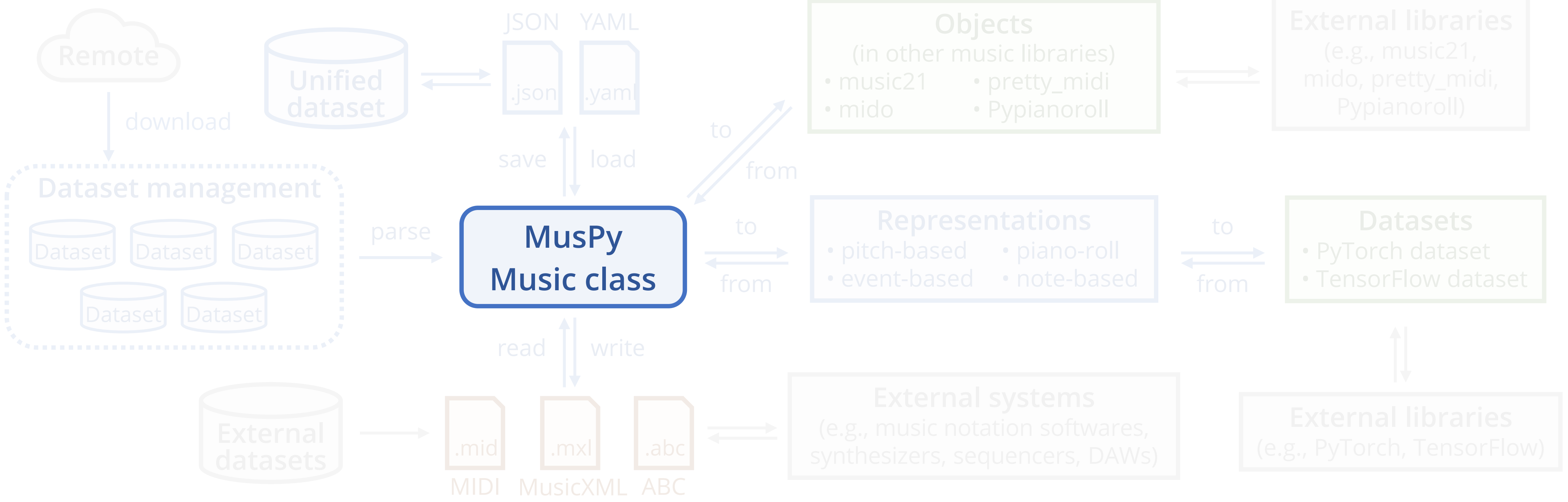

Remote
download
Unified dataset
JSON YAML
.json .yaml
save load
Dataset management
Dataset Dataset Dataset
Dataset Dataset
parse
MusPy Music class
read write
External datasets
.mid .mxl .abc
MIDI MusicXML ABC
to from
Objects
(in other music libraries)
• music21 • pretty_midi
• mido • Pypianoroll
External libraries
(e.g., music21, mido, pretty_midi, Pypianoroll)
Representations
• pitch-based • piano-roll
• event-based • note-based
Datasets
• PyTorch dataset
• TensorFlow dataset
External systems
(e.g., music notation softwares, synthesizers, sequencers, DAWs)
External libraries
(e.g., PyTorch, TensorFlow)

Overview

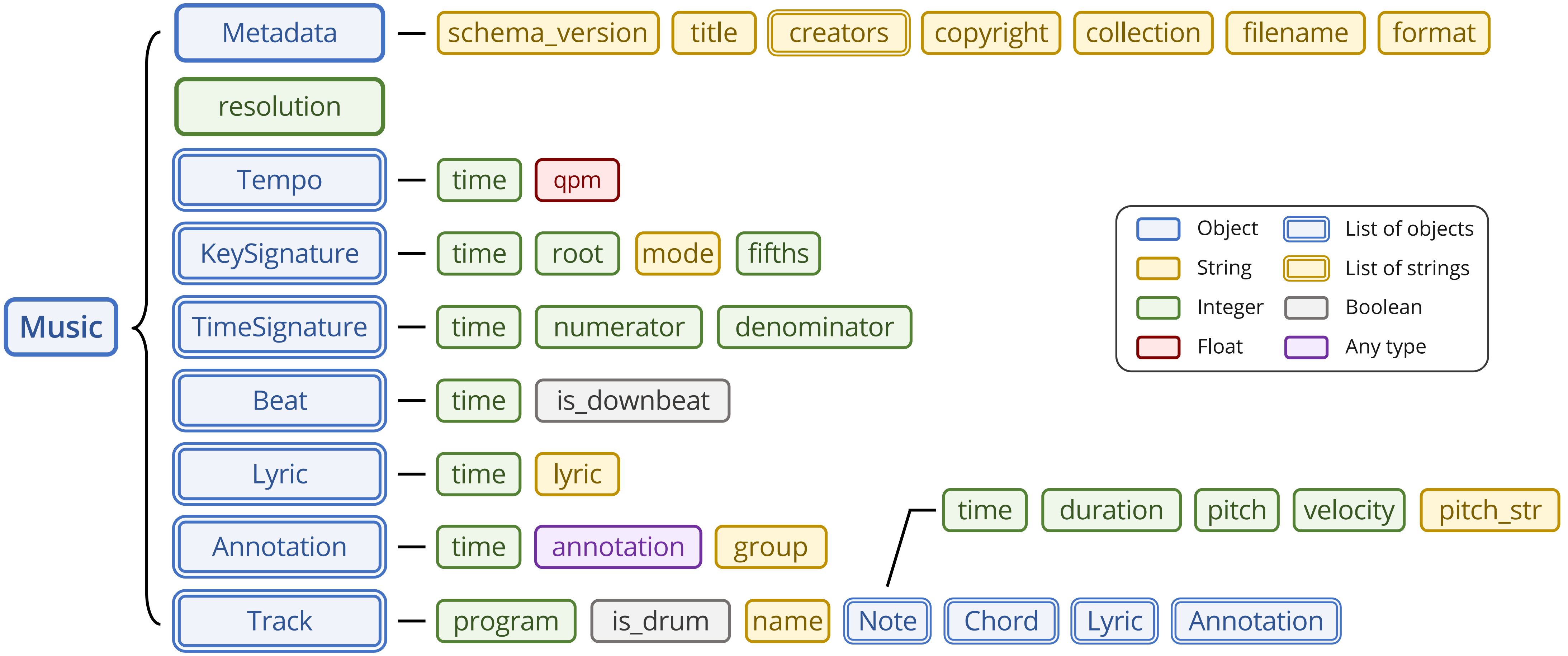


Remote
download
Unified dataset
JSON YAML
.json .yaml
save load
Dataset management
Dataset Dataset Dataset Dataset Dataset
parse
MusPy Music class
to from
Objects
(in other music libraries)
• music21 • pretty_midi
• mido • Pypianoroll
External libraries
(e.g., music21, mido, pretty_midi, Pypianoroll)
to from
Representations
• pitch-based • piano-roll
• event-based • note-based
to from
Datasets
• PyTorch dataset
• TensorFlow dataset
External libraries
(e.g., PyTorch, TensorFlow)
read write
External datasets
.mid .mxl .abc
MIDI MusicXML ABC
External systems
(e.g., music notation softwares, synthesizers, sequencers, DAWs)

MusPy Music class



MusPy Music class



MusPy native format

- A universal container for symbolic music
- Serializable to JSON/YAML
- Human-readable and machine-friendly


```yaml
metadata:
  schema_version: "0.0"
  title: Für Elise
  creators: [Ludwig van Beethoven]
  copyright: null
  collection: Example dataset
  source_filename: example.yaml
  source_format: yaml
resolution: 24
tempos:
  - {time: 0, qpm: 72}
key_signatures:
  - {time: 0, root: 9, mode: minor, fifths: 0}
time_signatures:
  - {time: 0, numerator: 3, denominator: 8}
beats:
  - {time: 0, is_downbeat: false}
  - {time: 12, is_downbeat: true}
  - {time: 24, is_downbeat: false}
  - {time: 36, is_downbeat: false}
  - {time: 48, is_downbeat: true}
lyrics:
  - {time: 0, lyric: Nothing but a lyric}
annotations:
  - {time: 0, annotation: Nothing but an annotation, group: null}
tracks:
  - program: 0
    is_drum: false
    name: Melody
    notes:
      - {time: 0, duration: 6, pitch: 76, velocity: 64}
      - {time: 6, duration: 6, pitch: 75, velocity: 64}
      - {time: 12, duration: 6, pitch: 76, velocity: 64}
      - {time: 18, duration: 6, pitch: 75, velocity: 64}
      - {time: 24, duration: 6, pitch: 76, velocity: 64
      - {time: 30, duration: 6, pitch: 71, velocity: 64}
      - {time: 36, duration: 6, pitch: 74, velocity: 64}
      - {time: 42, duration: 6, pitch: 72, velocity: 64}
      - {time: 48, duration: 6, pitch: 69, velocity: 64}
    chords: null
    lyrics:
      - {time: 0, lyric: Nothing but a lyric}
    annotations:
      - {time: 0, annotation: Nothing but an annotation, group: null}
```

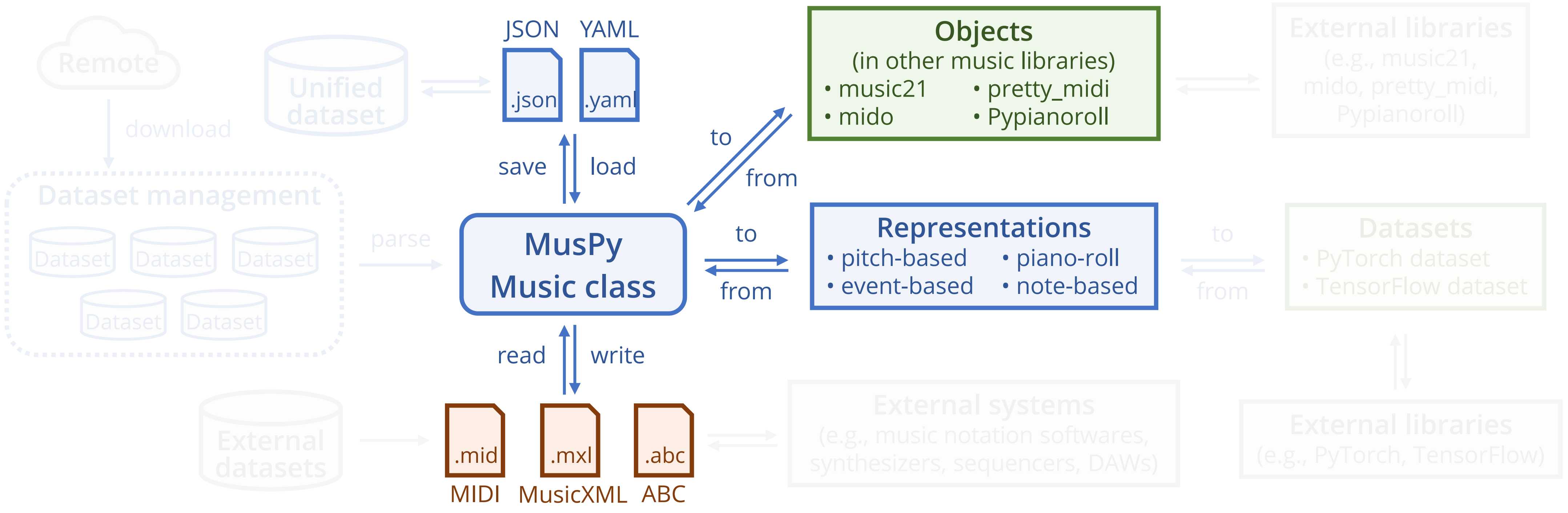
Comparisons to MIDI & MusicXML

	MIDI	MusicXML	MusPy
Sequential timing	✓		✓
Playback velocities	✓	△	✓
Program information	✓	△	✓
Layout information		✓	
Note beams and slurs		✓	
Song/source meta data	△	✓	✓
Track/part information	△	✓	✓
Dynamic/tempo markings		✓	✓
Concept of notes		✓	✓
Measure boundaries		✓	✓
Human readability		△	✓

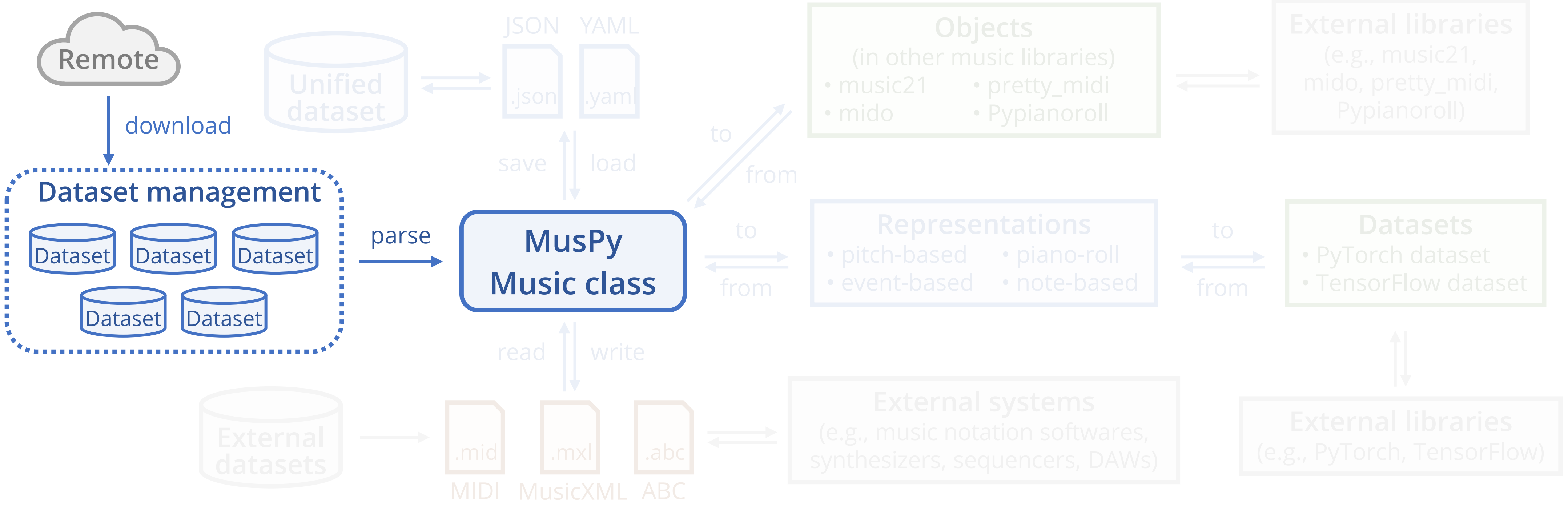
Comparisons to MIDI & MusicXML

	MIDI	MusicXML	MusPy
Sequential timing	✓		✓
Playback velocities	✓	△	✓
Program information	✓	△	✓
Layout information		✓	
Note beams and slurs		✓	
Song/source meta data	△	✓	✓
Track/part information	△	✓	✓
Dynamic/tempo markings		✓	✓
Concept of notes		✓	✓
Measure boundaries		✓	✓
Human readability		△	✓

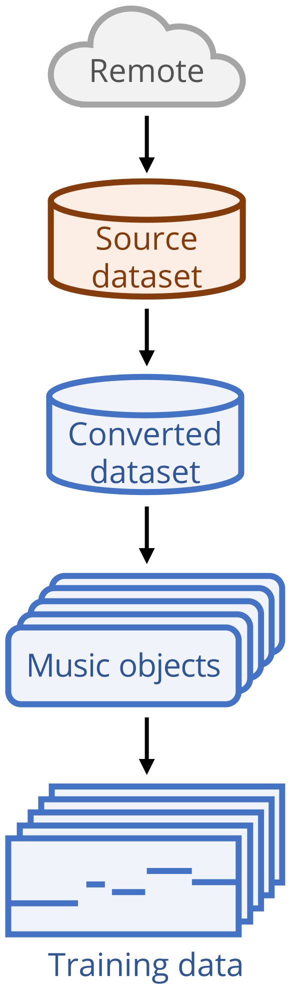
I/O interfaces



Dataset management



Dataset management – An example




```python
# Download and extract the dataset
nes = muspy.NESMusicDatabase(root="data/nes/",
                             download_and_extract=True)

# Convert the dataset to MusPy Music objects
nes.convert()

# Iterate over the dataset
for music in nes:
  do_something(music)

# Convert to a PyTorch dataset
dataset = nes.to_pytorch_dataset(representation="pianoroll")
```

Datasets supported

Dataset	**Format**	**Hours**	**Songs**	**Genre**	**Melody**	**Chords**	**Multitrack**
Lakh MIDI Dataset	MIDI	>5000	174,533	misc	△	△	△
MAESTRO Dataset	MIDI	201.21	1,282	classical			
Wikifonia Lead Sheet Dataset	MusicXML	198.40	6,405	misc	✓	✓	
Essen Folk Song Dataset	ABC	56.62	9,034	folk	✓	✓	
NES Music Database	MIDI	46.11	5,278	game	✓		✓
MusicNet Dataset	MIDI	30.36	323	classical			△
Hymnal Tune Dataset	MIDI	18.74	1,756	hymn	✓		
Hymnal Dataset	MIDI	17.50	1,723	hymn			
music21's Corpus	misc	16.86	613	misc	△		△
EMOPIA Dataset	MIDI	10.98	387	pop			
Nottingham Database	ABC	10.54	1,036	folk	✓	✓	
music21's JSBach Corpus	MusicXML	3.46	410	classical			✓
JSBach Chorale Dataset	MIDI	3.21	382	classical			✓
Haydn Op.20 Dataset	Humdrum	1.26	24	classical		✓	

Result analysis tools

Pitch-related metrics

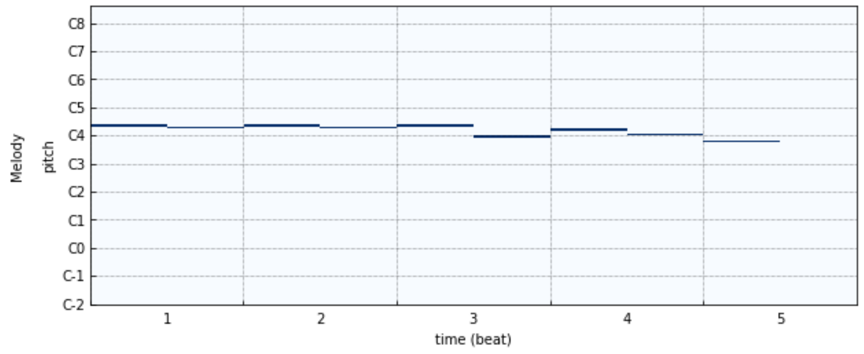
- pitch_range
- n_pitches_used
- n_pitch_classes_used
- polyphony
- polyphony_rate
- pitch_in_scale_rate
- scale_consistency
- pitch_entropy
- pitch_class_entropy

Rhythm-related metrics

- empty_beat_rate
- empty_measure_rate
- drum_in_pattern_rate
- drum_pattern_consistency
- groove_consistency

Audio rendering

Piano-roll visualization



Related work

- Magenta
  - Provides several model instances in TensorFlow
- music21 (Cuthbert and Ariza 2010)
  - Provides powerful tools for computational musicology
  - Comes with its own corpus
- jSymbolic (McKay and Fujinaga 2006)
  - Extracts statistical information from symbolic music data

Magenta, https://magenta.tensorflow.org/.
Cuthbert and Ariza, "A Toolkit for Computer-Aided Musicology and Symbolic Music Data," *Proc. ISMIR*, 2010.
McKay and Fujinaga, "jSymbolic: A Feature Extractor for MIDI Files," *Proc. ICMC*, 2006.

Summary

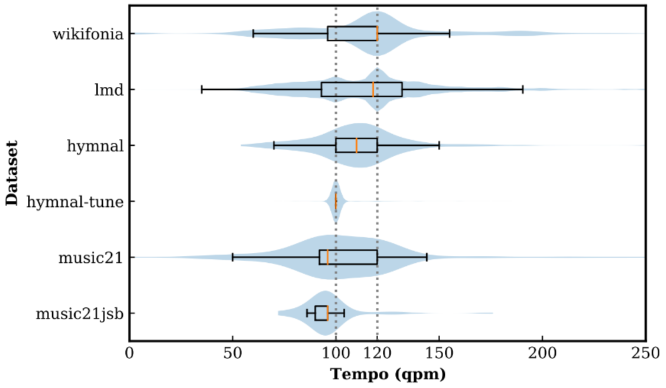
MusPy provides

- Dataset management
- Data I/O for common formats
- Interfaces to common music libraries
- Implementations of common music representations
- Result analysis tools

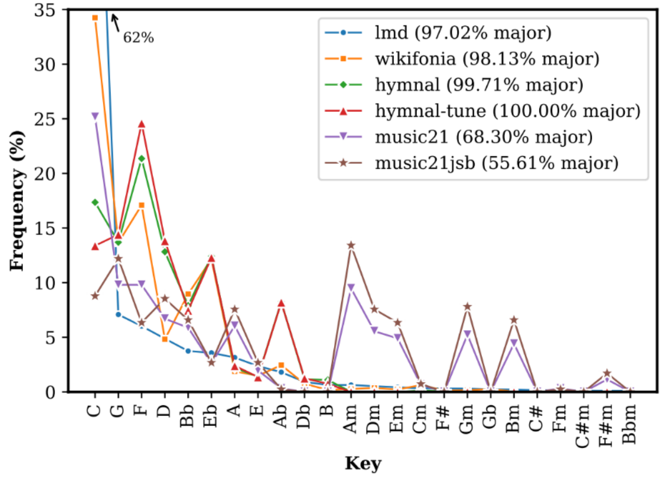
Experiments

Dataset analysis

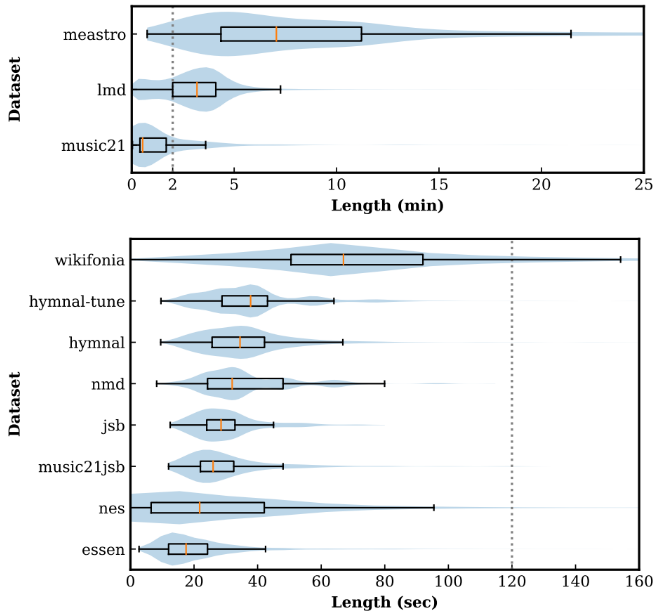
Tempo



Key



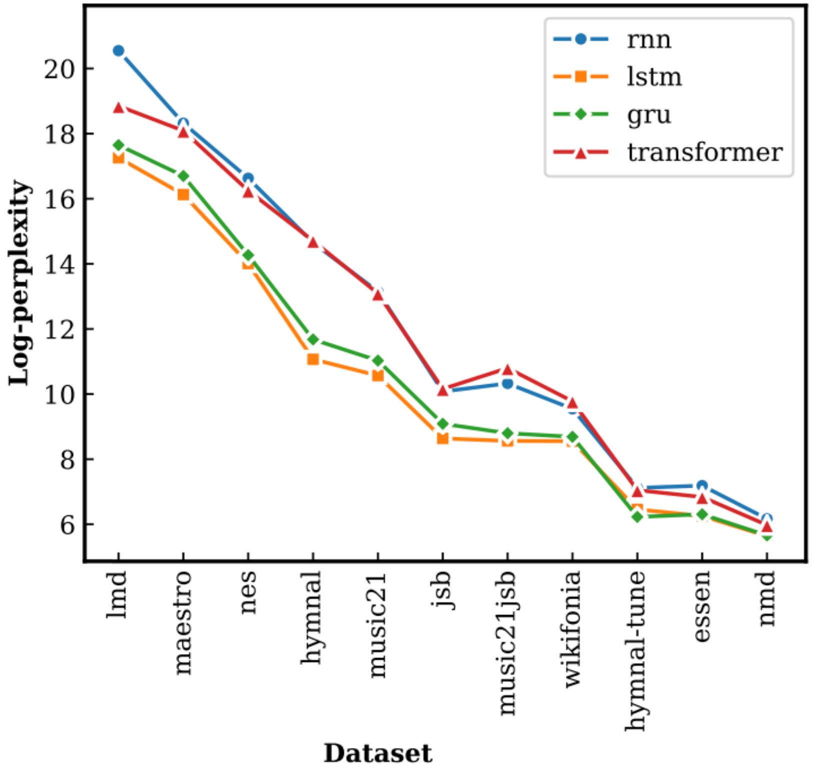
Length



Music language models

Settings

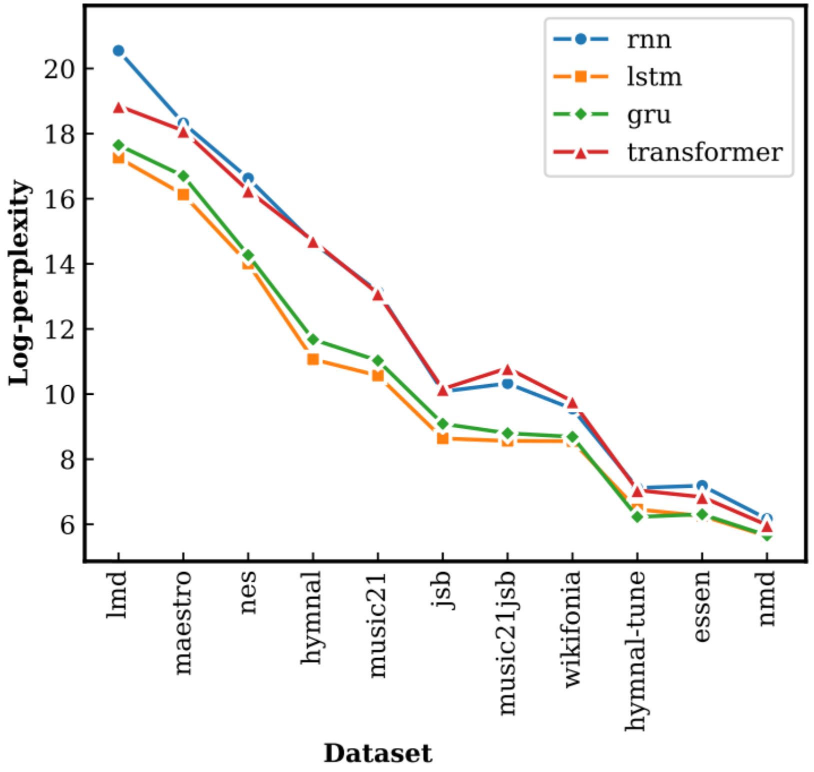
- Implement four autoregressive models
  - RNN, LSTM, GRU and Transformer
- Use a MIDI-like event representation
- Measure the perplexity of 1000 test samples



Music language models

Results

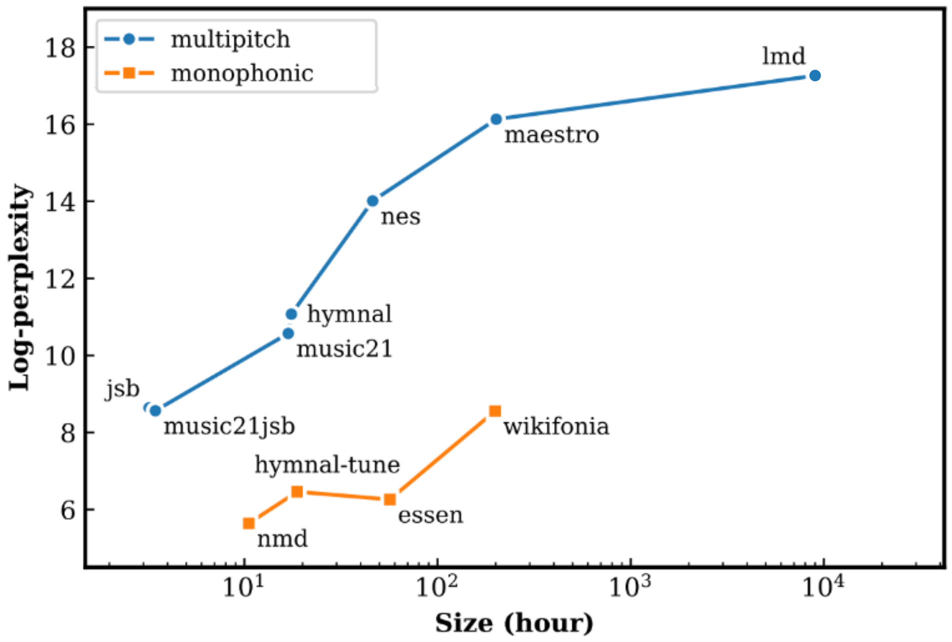
- All models have similar tendencies



Music language models

Results

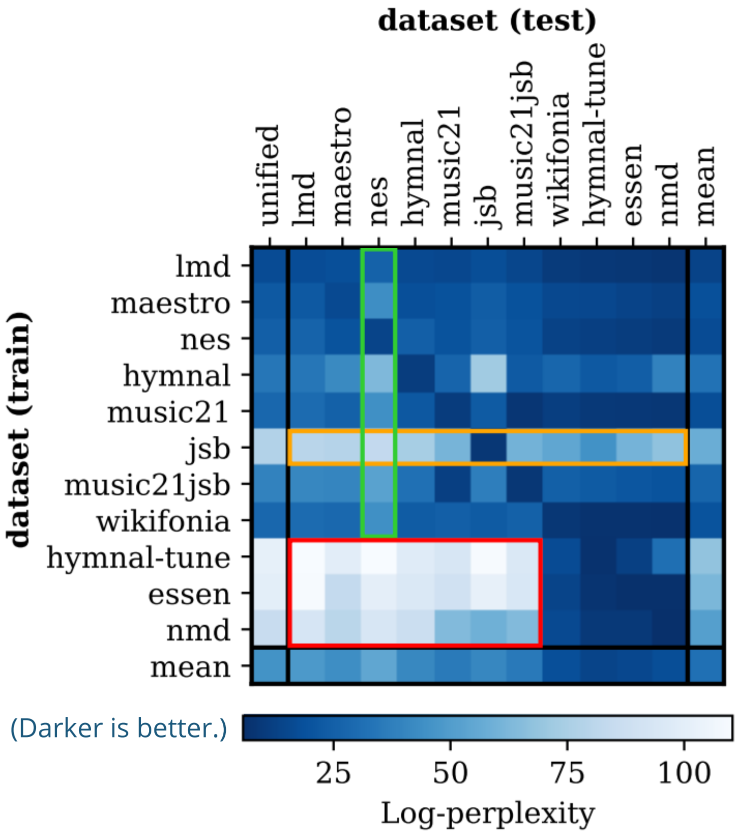
- All models have similar tendencies.
- Perplexity is positively correlated to dataset size.
  - Within each group (multipitch vs monophonic)



Measuring cross-dataset generalizability

Settings

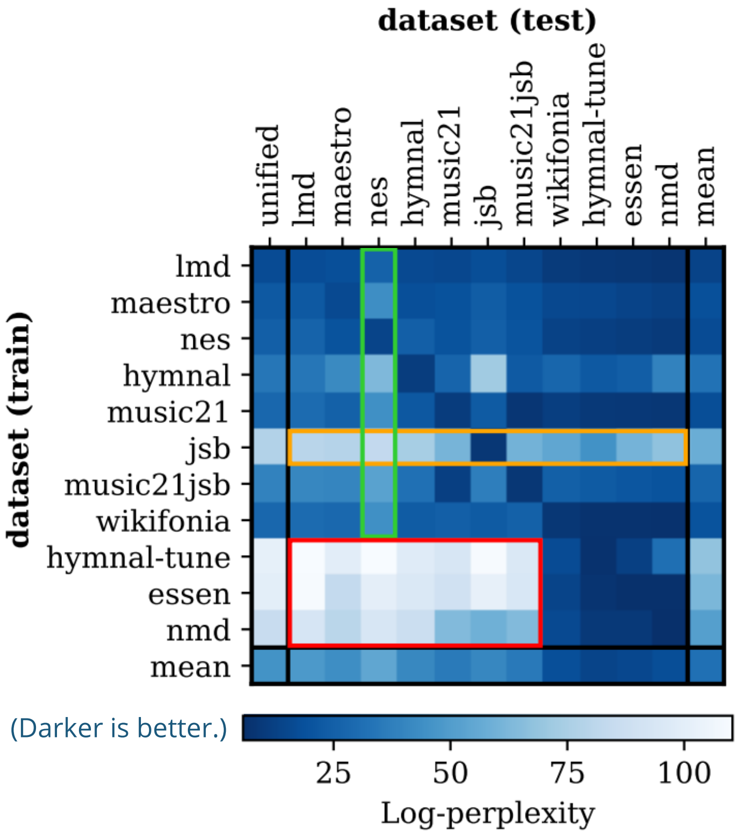
1. Train the model on a dataset $\mathcal{D}$
2. Test the trained model on dataset $\mathcal{D}'$
3. Repeat for all 11x11 pairs of $(\mathcal{D}, \mathcal{D}')$



Measuring cross-dataset generalizability

Results

- Cross-dataset generalizability is asymmetric.
- A model trained on a multi-pitch dataset generalizes well to a monophonic dataset.
  - Yet not the other way around (red block)



(Darker is better.)

Combining heterogeneous datasets

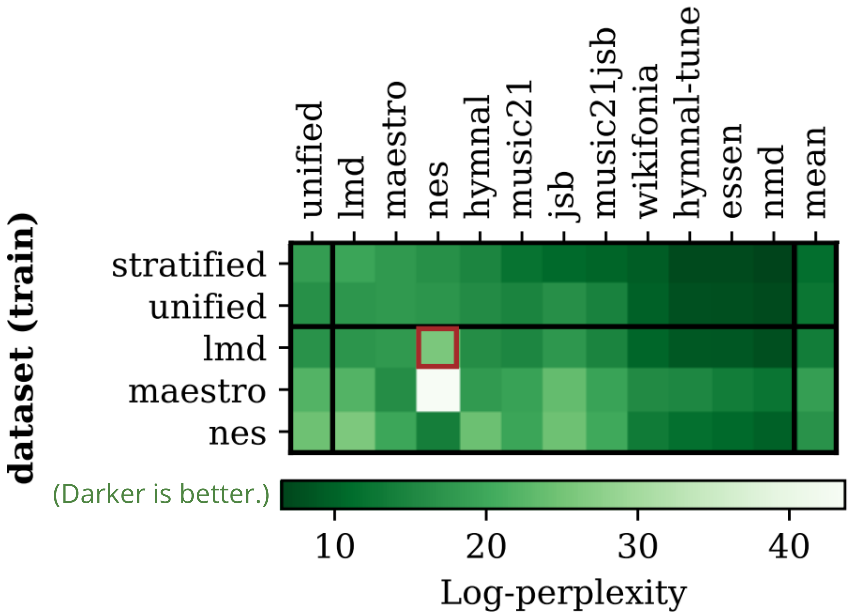
Settings

- **Unified**

  Sample uniformly from the pool of all data of different datasets

- **Stratified**

  Pick a dataset randomly and sample uniformly from that dataset (to alleviate data imbalance issue)



Combining heterogeneous datasets

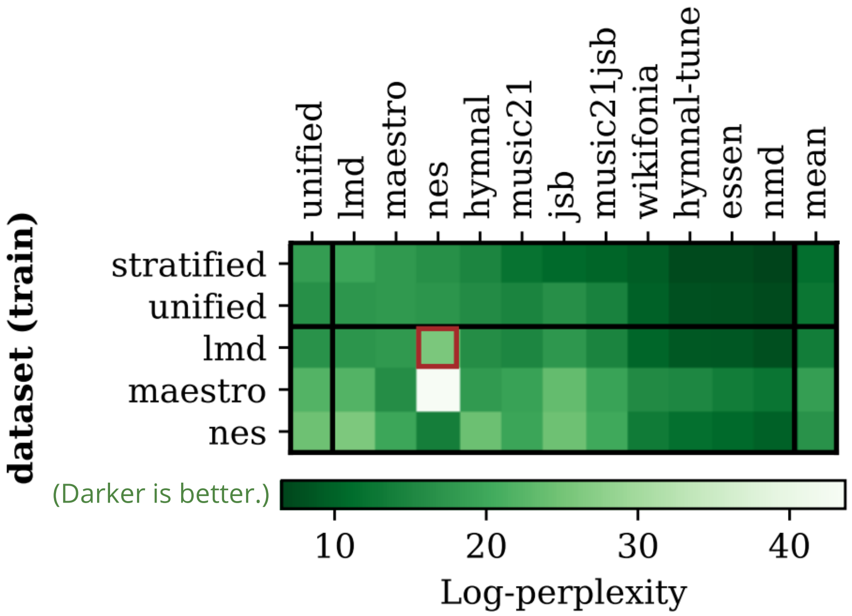
Results

- **Unified**

  The model trained on the unified dataset yields a lower perplexity on each dataset.

- **Stratified**

  Stratified sampling reduce perplexities on most datasets with a sacrifice of an increased perplexity on LMD. (LMD is the largest dataset.)



Summary

- Measured the relative diversities of 11 datasets
- Analyzed the cross-dataset generalizabilities of a music generation system
- Showed how combining heterogenous datasets can help improve generalizability

Case Study I – Automatic Instrumentation

"Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music" (ISMIR 2021)

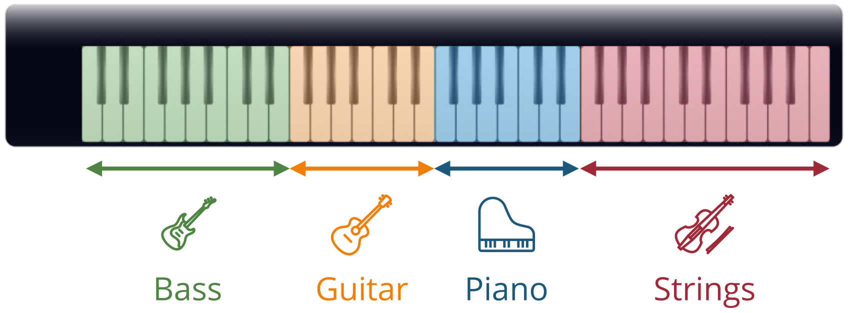
Hao-Wen Dong Chris Donahue Taylor Berg-Kirkpatrick Julian McAuley

Motivation

- Automatic instrumentation – Dynamically assign instruments to notes in solo music

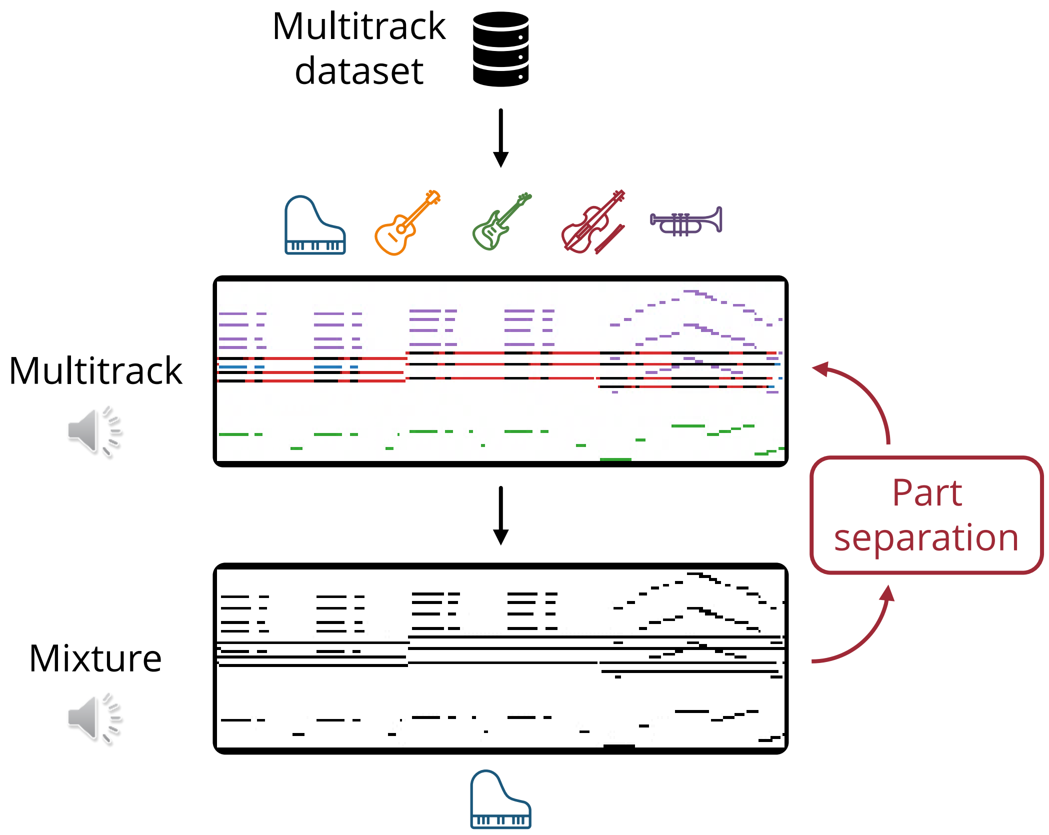
Intelligent musical instruments



Assistive composing tools

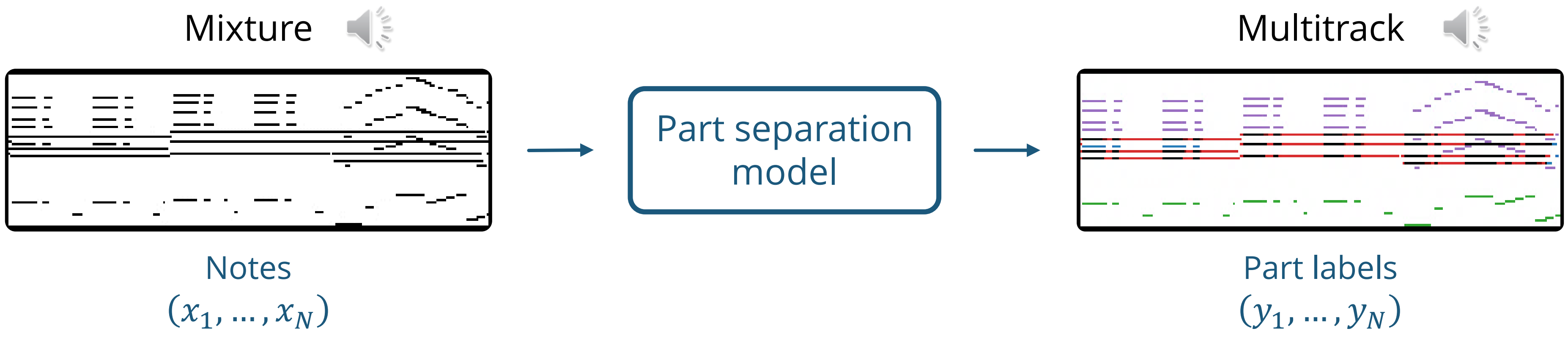
Overview

- Acquire paired data
- Train a part separation model
- Perform automatic instrumentation



Problem formulation

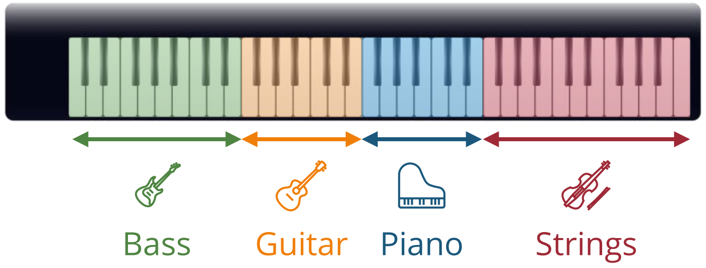
- Part separation – Separate parts from their mixture in multitrack music
- Frame as a sequential multiclass classification problem



Models

Online models

- LSTMs
- Transformer decoders



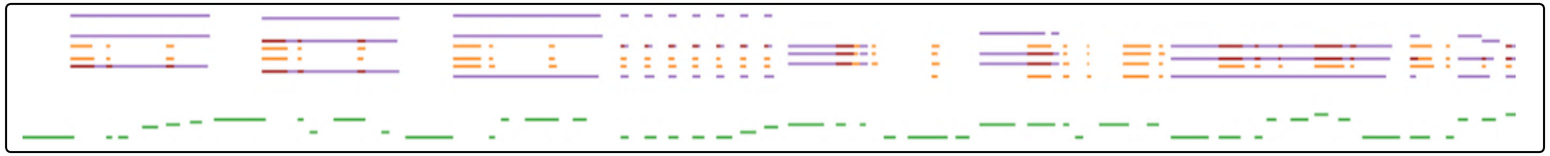
Offline models

- BiLSTMs
- Transformer encoders

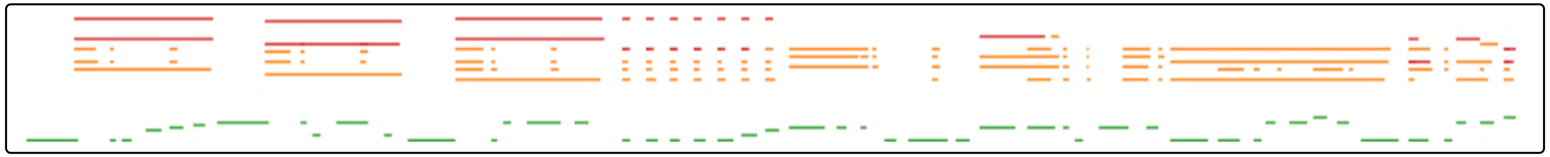
Demo

- Produce convincing alternative instrumentations for an existing arrangement

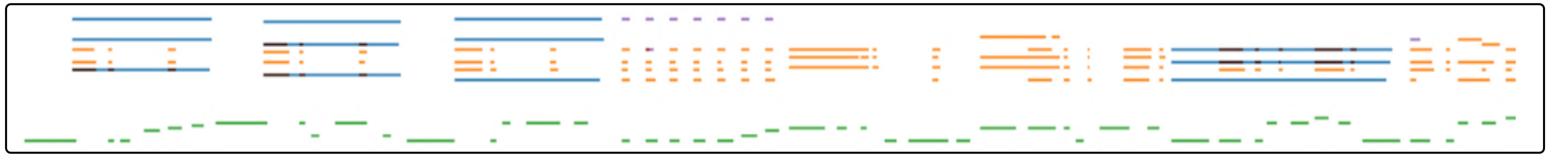
Original



LSTM
(without entry hints)

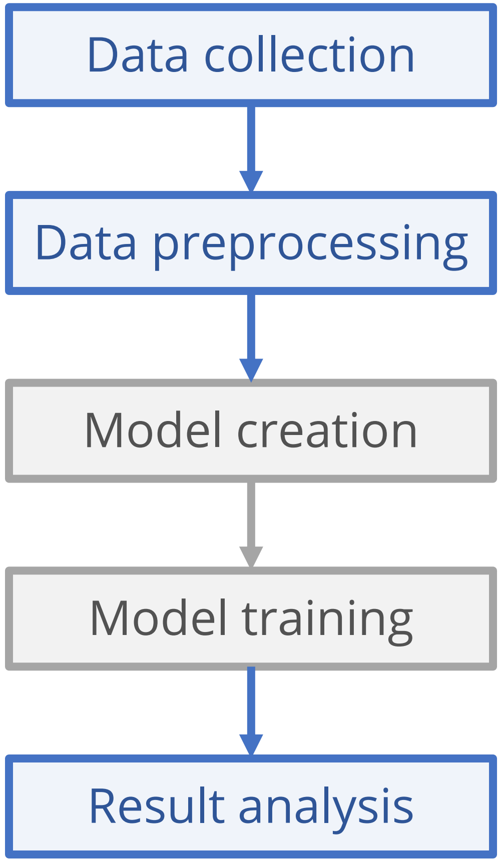


BiLSTM
(with entry hints)



More samples can be found at salu133445.github.io/arranger/.

MusPy in the pipeline



- Download datasets
- Convert data into MusPy JSON format

- Adjust temporal resolution
- Map instrument names
- Convert data into note representation

- Save the results as MIDI files
- Synthesize the results into audio
- Visualize the results in the piano roll form

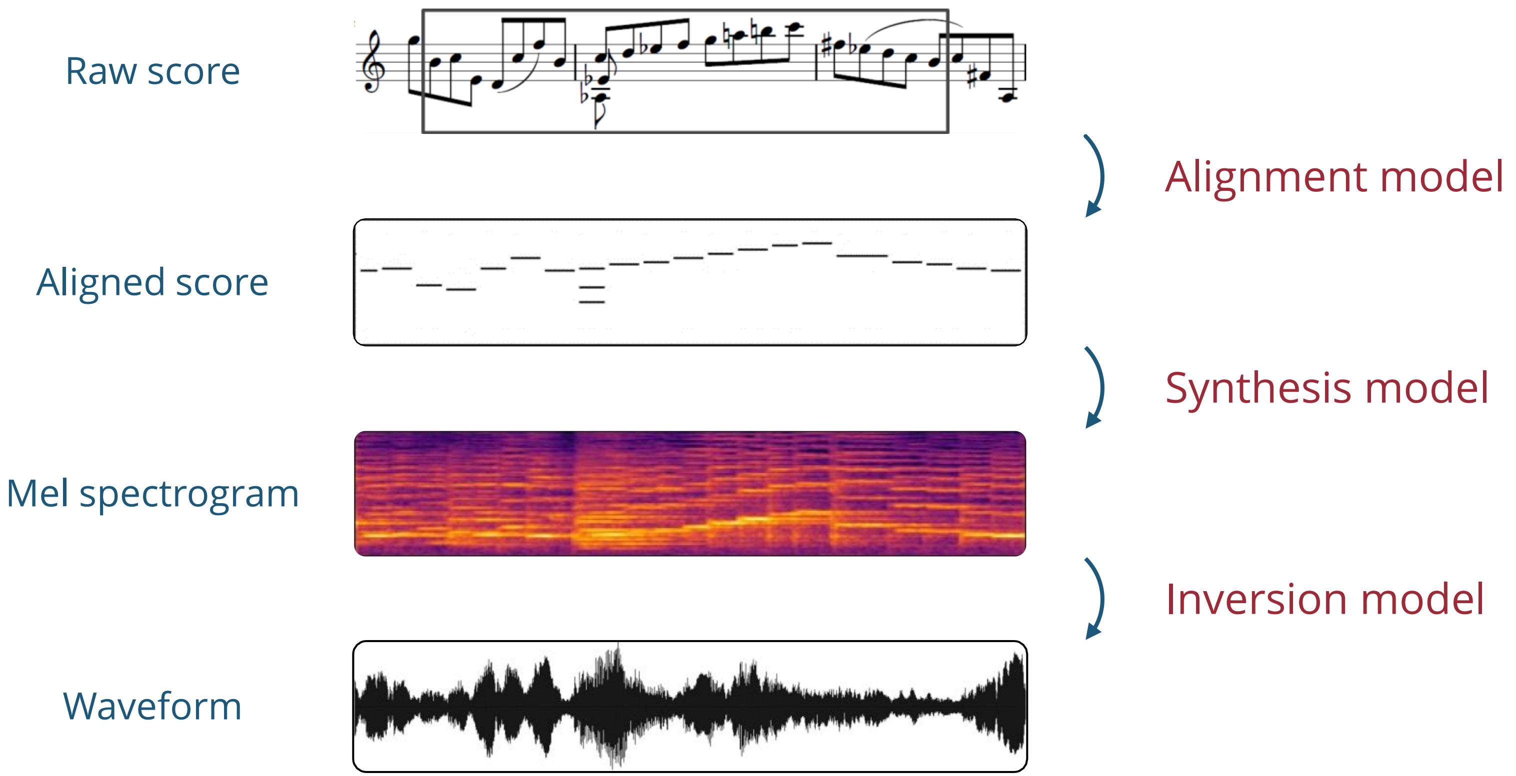
Case Study II – Music Performance Synthesis

"Deep Performer: Score-to-Audio Music Performance Synthesis" (ICASSP 2022)

Hao-Wen Dong Cong Zhou Taylor Berg-Kirkpatrick Julian McAuley




Overview


Raw score
Alignment model
Aligned score
Synthesis model
Mel spectrogram
Inversion model
Waveform

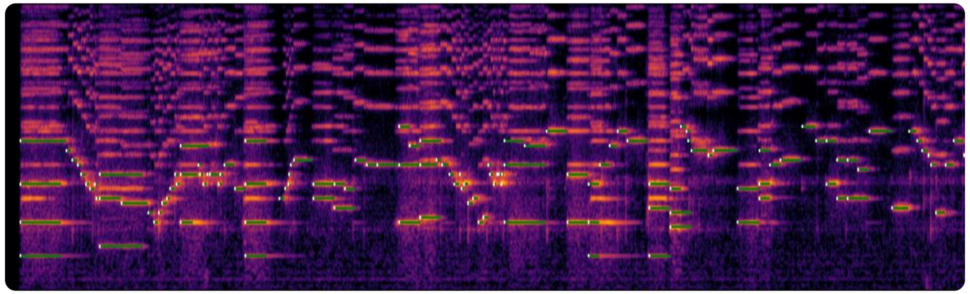
Bach Violin Dataset

- Bach's sonatas and partitas for solo violin (BWV 1001–1006)
- 6.7 hours, 17 violinists

Alignment derivation

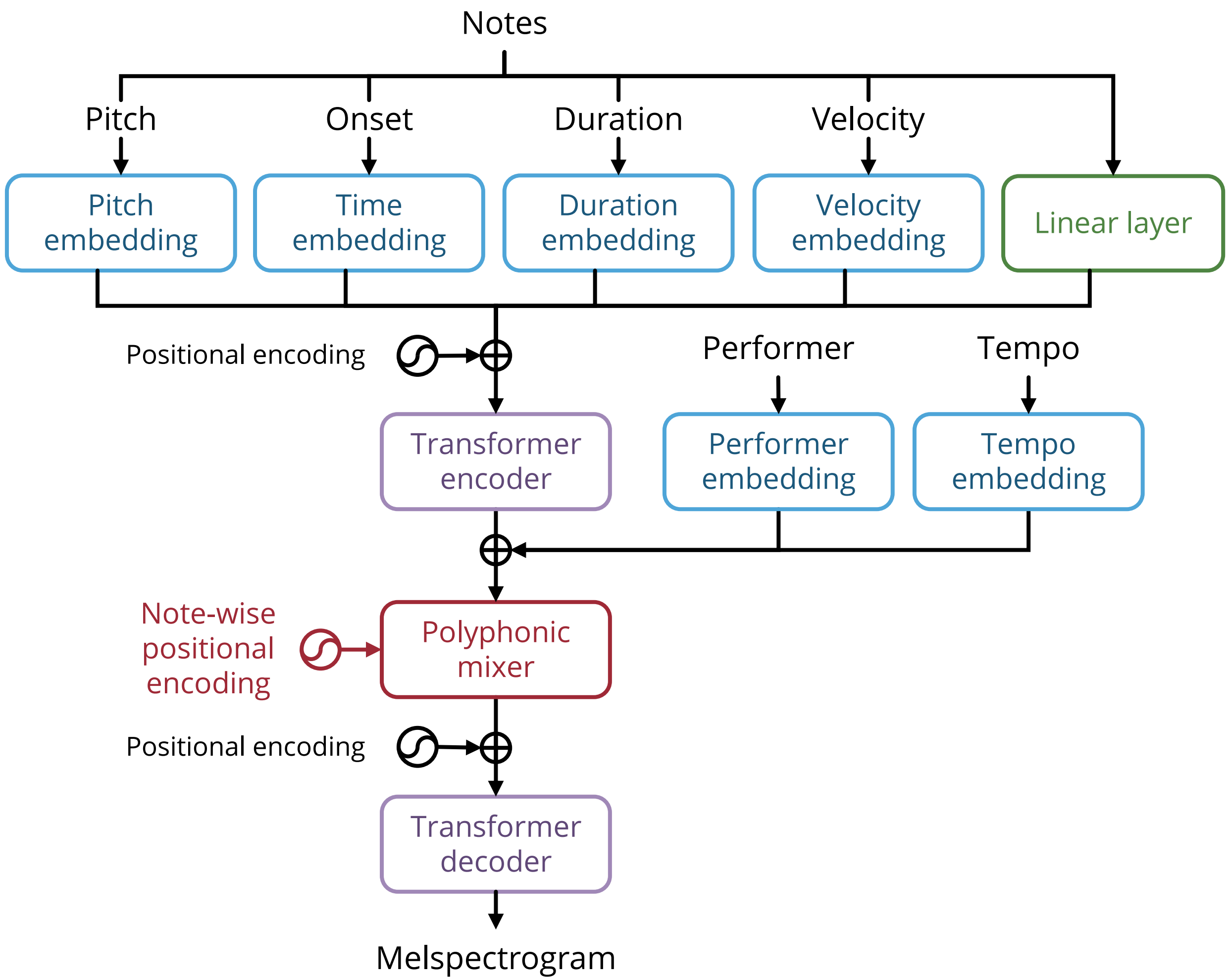
1. Synthesize the scores using FluidSynth (a free software synthesizer)
2. Run dynamic time warping on the spectrograms (of the recording & synthesized audio)

Alignment result



Synthesis model

- A transformer network
  based on FastSpeech (Ren et al. 2019)



Ren et al., "FastSpeech: Fast, robust and controllable text to speech," *Proc. NeurIPS*, 2019.

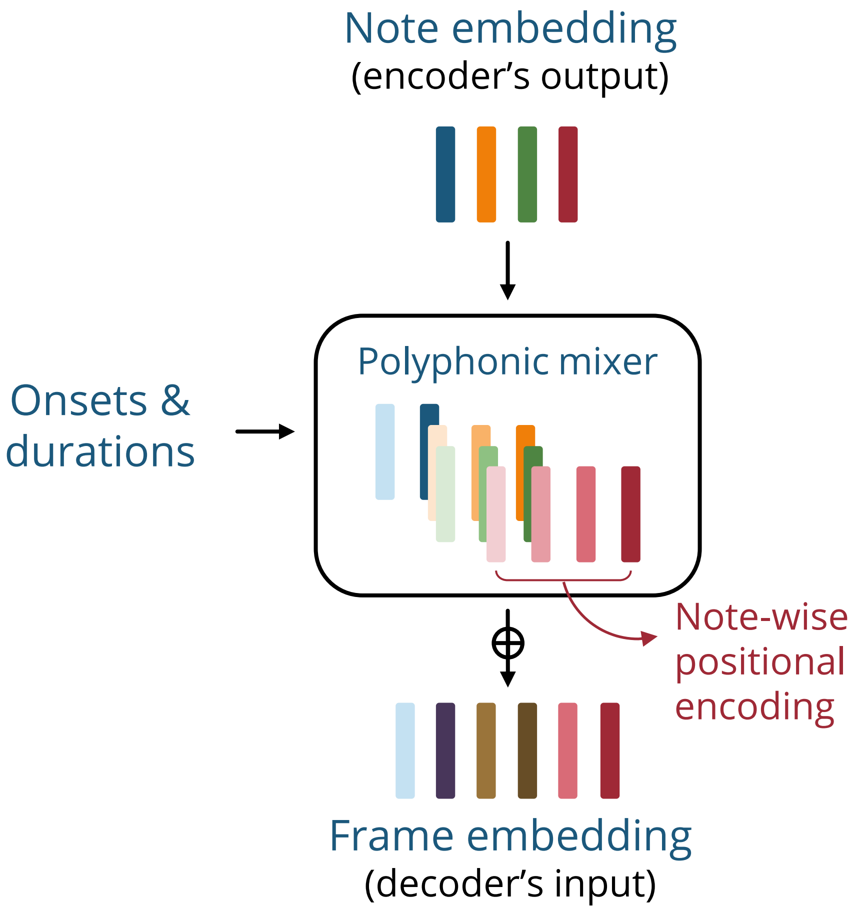
Methods

- Polyphonic mixer

  Extend the state expansion mechanism to handle polyphonic inputs

- Note-wise positional encoding

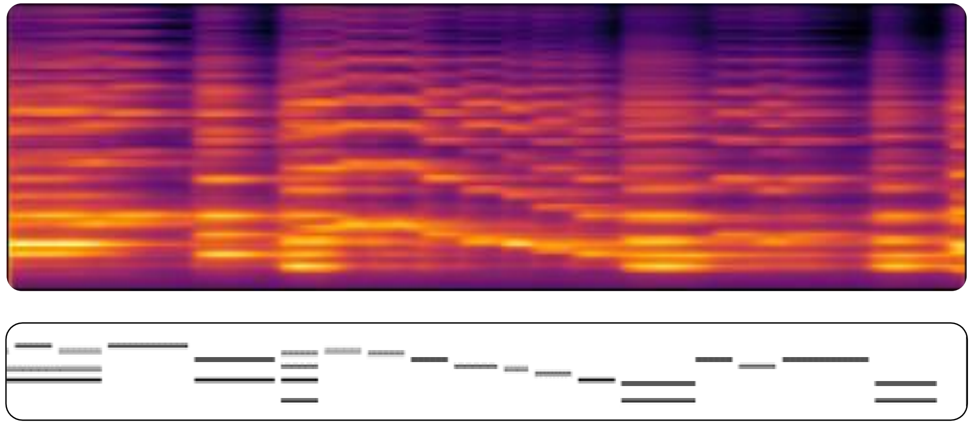
  Provide positional information within each note for a fine-grained conditioning



Demo

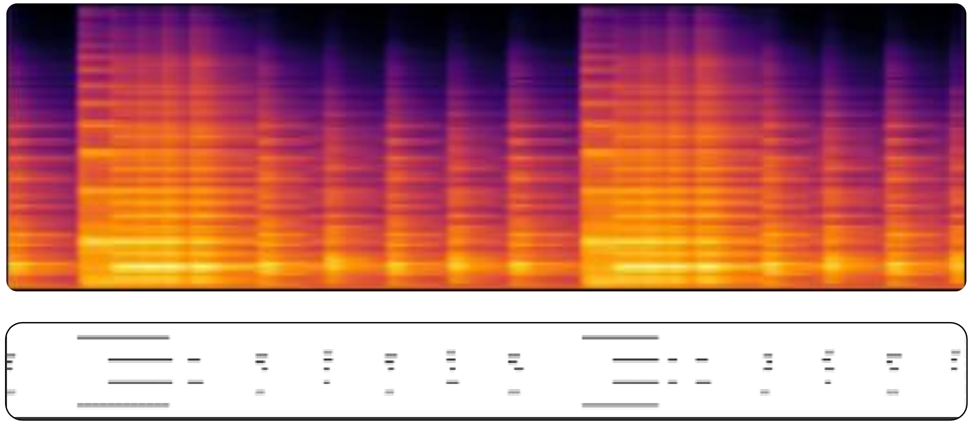
Violin

(trained on Bach Violin Dataset)



Piano

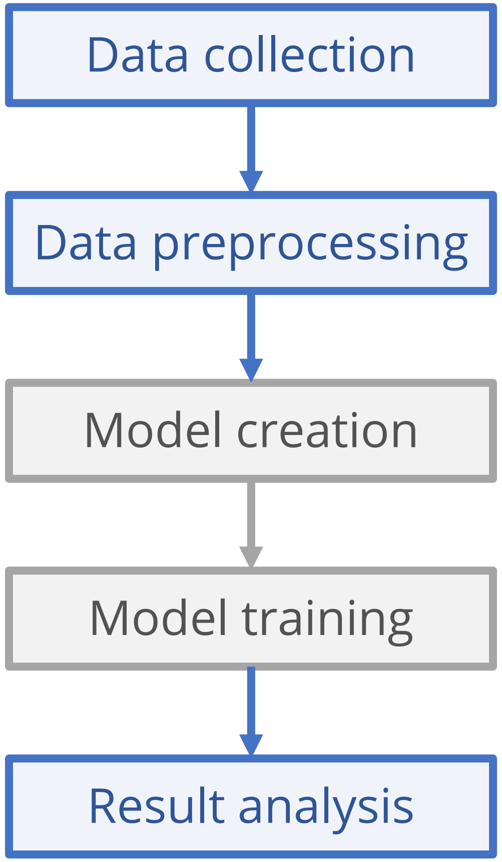
(trained on MAESTRO Dataset)



More samples can be found at salu133445.github.io/deepperformer/.

Hawthorne et al., "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," *Proc. ICLR*, 2019.

MusPy in the pipeline



- Convert data into MusPy JSON format
- Synthesize the data for alignment purpose
- Convert data into note representation
- Visualize the results in the piano roll form

Conclusion

Conclusion

- Presented a Python library for processing symbolic music
- Showcased how MusPy enabled large-scale cross-dataset analysis
  - Relative diversities of the 11 supported datasets
  - Cross-dataset generalizabilities of a music generation system
- Studied how we used MusPy in two recent projects
  - Automatic instrumentation
  - Music performance synthesis

Thank you!

```
pip install muspy
```

Learn more at salu133445.github.io/muspy/