Towards Automatic Instrumentation by Learning to Separate Parts in Multitrack Music

Hao-Wen Dong

Advisors: Julian McAuley and Taylor Berg-Kirkpatrick

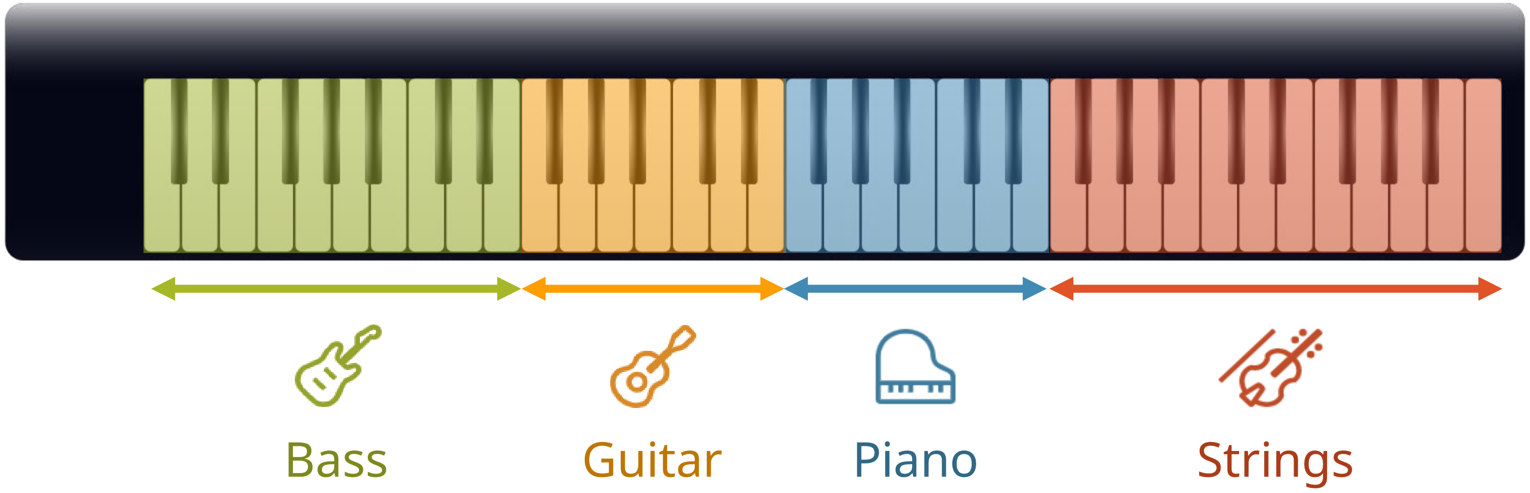
Outline

- Introduction
- Prior Work
- Problem Formulation
- Models
- Data
- Experiments & Results
- Discussion & Conclusion

Introduction

Keyboard zoning

- Modern keyboards allow a musician to play multiple instruments at the same time by assigning **zones**—*fixed pitch ranges of the keyboard*—to different instruments.

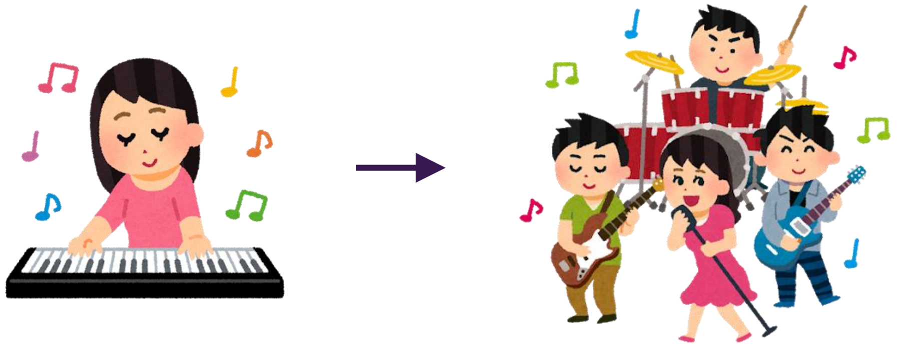


Automatic instrumentation

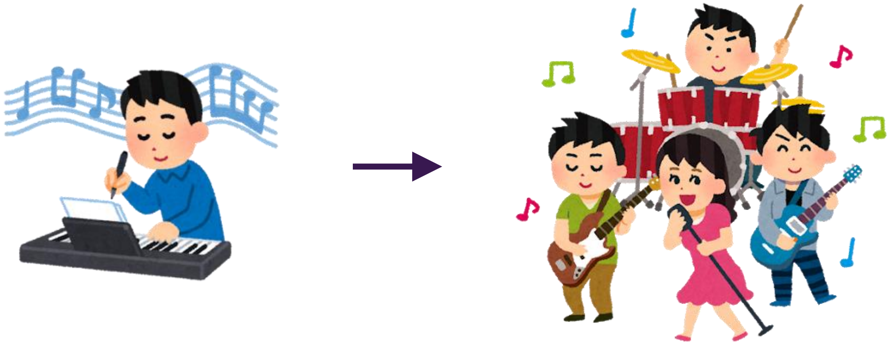
- **Goal**—Dynamically assign instruments to notes in solo music

Automatic instrumentation – Use cases

- Intelligent musical instruments



- Assistive composing tools



Challenges & proposed solutions

- Lack paired data of solo music and its instrumentation
  - Easy to obtain multitrack datasets

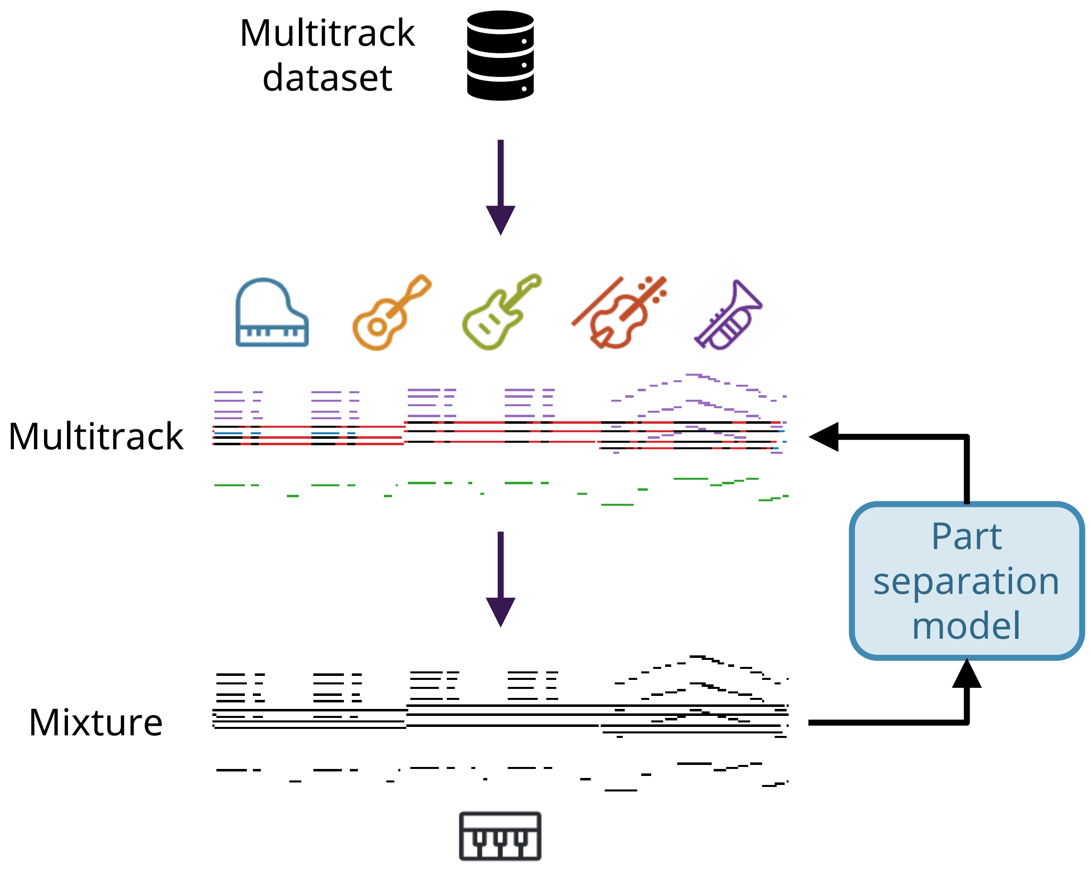
  → Downmix multitracks to mixtures to acquire paired data

- Require domain knowledge of each target instrument
  - Know which pitches, rhythms, chords and sequences are playable
  - Hard to specify as some fixed set of rules

  → Adopt a data-driven approach with machine learning

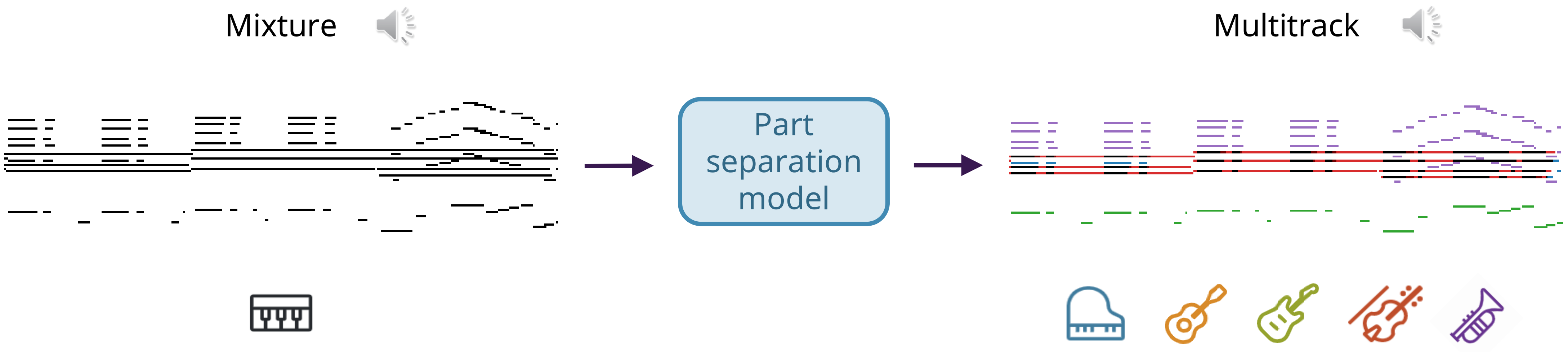
Proposed pipeline – Overview

- Acquire paired data of solo music and its instrumentation
  - Downmix multitracks into single-track mixtures
- Train a part separation model
  - Learn to infer the part label for each note in a mixture
- Approach automatic instrumentation
  - Treat input from a keyboard player as a downmixed mixture
  - Separate out the relevant parts



Part separation

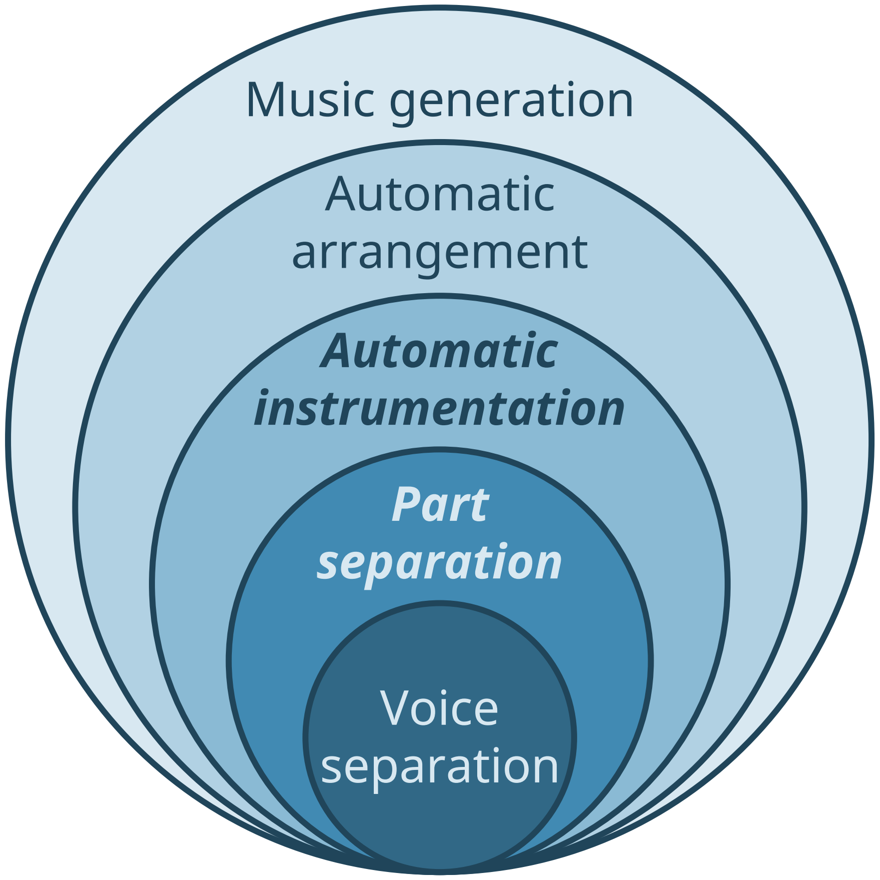
- **Goal**—Separate parts from their mixture in multitrack music
- A part can be a voice, an instrument, a track, etc.



Prior Work

Relevant topics

- **Voice separation**
  - A subset of part separation where all parts are monophonic
- **Automatic arrangement**
  - A more general task that involves instrumentation, reharmonization, melody paraphrasing or orchestration
- **Music generation**
  - A more general field that covers each stage in the music creation workflow



Prior work – Voice separation

- Some worked on small, carefully-annotated pop music datasets [1,2]
- Some allowed synchronous or overlapping notes in a voice [3–6]
  - Reported results only on small test sets in certain genres
- Some used machine learning models with hand-crafted input features
  - Multilayer perceptron (MLP) [1,7]
  - Convolutional neural network (CNN) [8]
  - Long short-term memory (LSTM) [9]

[1] P. Gray and R. Bunescu, "A neural greedy model for voice separation in symbolic music," *ISMIR*, 2016.
[2] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, "Comparing voice and stream segmentation algorithms," *ISMIR*, 2015.
[3] J. Kilian and H. H. Hoos, "Voice separation — a local optimisation approach," *ISMIR*, 2002.
[4] E. Cambouropoulos, "'voice' separation: theoretical, perceptual and computational perspectives," *ICMPC*, 2006.
[5] I. Karydis, A. Nanopoulos, A. Papadopoulos, and E. Cambouropoulos, "Visa: The voice integration/segregation algorithm," *ISMIR*, 2007.
[6] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, 26(1), 2008.
[7] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," *ISMIR*, 2019.
[8] P. Gray and R. Bunescu, "From note-level to chord-level neural network models for voice separation in symbolic music," *arXiv preprint arXiv:2011.03028*, 2020.
[9] A. Hadjakos, S. Waloschek, and A. Leemhuis, "Detecting hands from piano midi data," in Mensch und Computer 2019- Workshopband, 2019.

Prior work – Automatic arrangement

- Some worked on reduction
  - Mapped musical scores for large ensembles to parts playable by a specific instrument such as piano [1–6], guitar [7–9] and bass [10]
  - Focused on identifying the least important notes to remove
- Some arranged orchestral music from piano [11]
  - Did not guarantee that all notes in the input piano map to parts in the output

[1] S.-C. Chiu, M.-K. Shan, and J.-L. Huang, "Automatic system for the arrangement of piano reductions," *ISM*, 2009.
[2] S. Onuma and M. Hamanaka, "Piano arrangement system based on composers' arrangement processes." *ICMC*, 2010.
[3] J.-L. Huang, S.-C. Chiu, and M.-K. Shan, "Towards an automatic music arrangement framework using score reduction," TOMM, 8(1):1–23, 2012.
[4] E. Nakamura and S. Sagayama, "Automatic piano reduction from ensemble scores based on merged-output hidden markov model," *ICMC*, 2015.
[5] H. Takamori, H. Sato, T. Nakatsuka, and S. Morishima, "Automatic arranging musical score for piano using important musical elements," *SMC*, 2017.
[6] E. Nakamura and K. Yoshii, "Statistical piano reduction controlling performance difficulty," APSIPA Transactions on Signal and Information Processing, 7, 2018.
[7] D. R. Tuohy and W. D. Potter, "A genetic algorithm for the automatic generation of playable guitar tablature," *ICMC*, 2005.
[8] G. Hori, Y. Yoshinaga, S. Fukayama, H. Kameoka, and S. Sagayama, "Automatic arrangement for guitars using hidden markov model," *SMC*, 2012.
[9] G. Hori, H. Kameoka, and S. Sagayama, "Input-output hmm applied to automatic arrangement for guitars," Information and Media Technologies, 8(2):477–484, 2013.
[10] Y. Abe, Y. Murakami, and M. Miura, "Automatic arrangement for the bass guitar in popular music using principle component analysis," Acoustical Science and Technology, 33(4):229–238, 2012.
[11] L. Crestel and P. Esling, "Live orchestral piano, a system for real-time orchestral music generation," *arXiv preprint arXiv:1609.01203*, 2016.

Prior work – Music generation

- Some used RNNs with an event-based representation [1]
- Some used Transformers with event-based representations [2–8]
- Some generated multitrack music [4,7]

[1] I. Simon and S. Oore, "Performance RNN: Generating music with expressive timing and dynamics," Magenta Blog, 2017. [Online]. Available: https://magenta.tensorflow.org/performance-rnn
[2] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," *arXiv preprint arXiv:2101.02402*, 2021.
[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," *ICLR*, 2019.
[4] C. Payne, "MuseNet," OpenAI, 2019. [Online]. Available: https://openai.com/blog/musenet/
[5] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," *ISMIR*, 2019.
[6] Y.-S. Huang and Y.-H. Yang, "Pop music Transformer: Generating music with rhythm and harmony," *arXiv preprint arXiv:2002.00212*, 2020.
[7] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the Transformer," *arXiv preprint arXiv:2008.06048*, 2020.
[8] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, "Symbolic music generation with Transformer-GANs," AAAI, 2021.

Problem Formulation

Sequential representation of music

- A piece of music is a sequence of notes.

$$x = (x_1, \dots, x_N)$$

- A note is a tuple of its time, pitch and (optionally) duration.

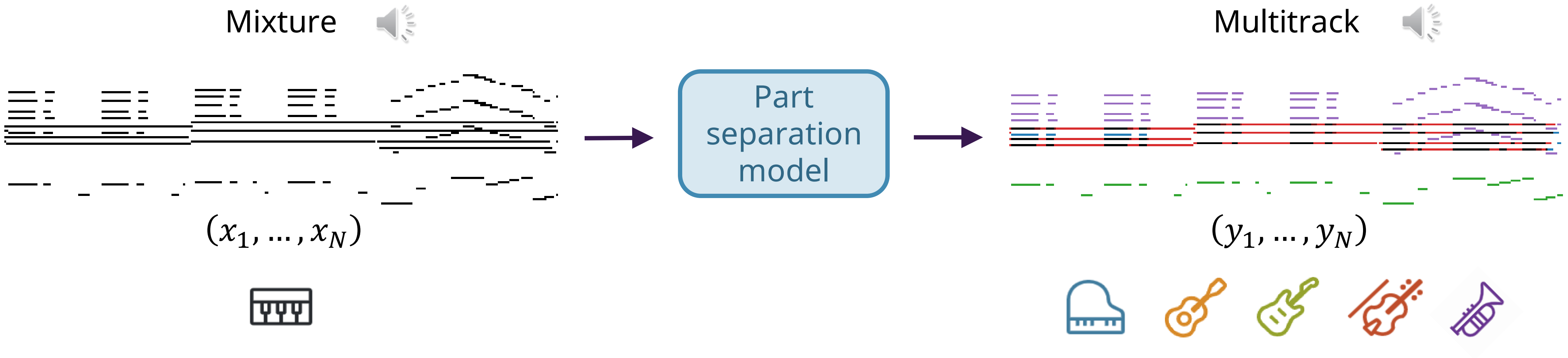
$$x_i = (t_i, p_i) \text{ or } x_i = (t_i, p_i, d_i)$$

- Each note is associated with a part label.

$$y_i \in \{1, \dots, K\}$$

Part separation

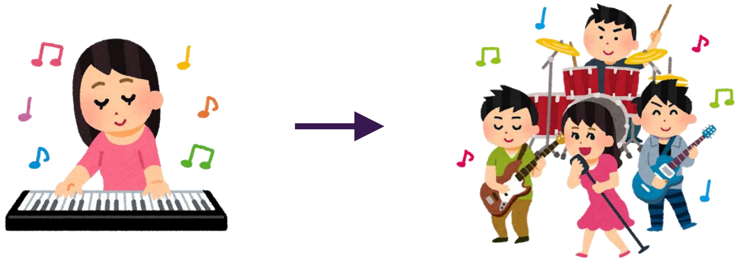
- **Goal**—Learn the mapping between notes and their part labels
- We frame the task of part separation as a **sequential multiclass classification problem**.



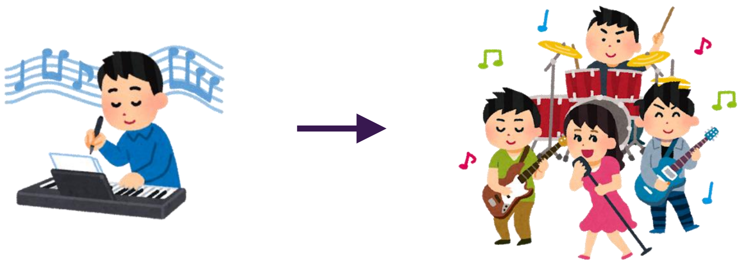
Three classes of part separation models

Context given to predict the label of the current note x_i:

- **Independent model**—without any context.
- **Online model**—*only* the past $(x_1, \ldots, x_{\boldsymbol{i-1}})$
  - Preferable for **live performance** and other use cases that require real-time outputs



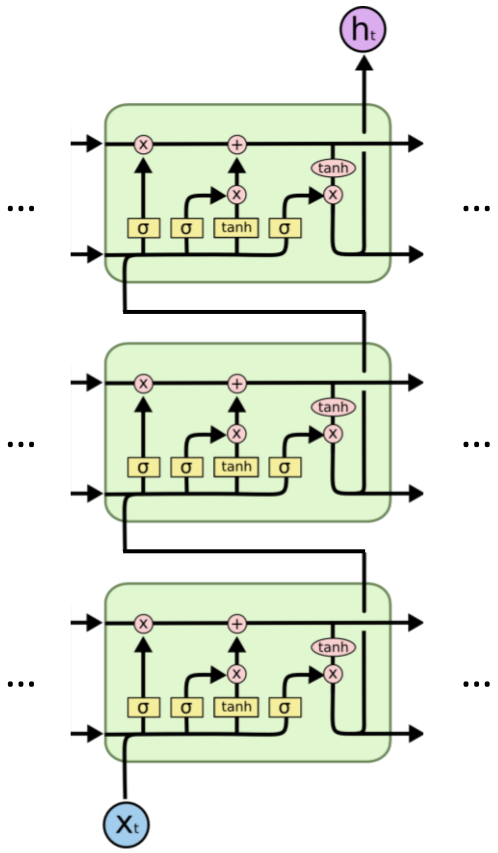
- **Offline model**—the past and the future $(x_1, \ldots, x_{\boldsymbol{N}})$
  - Can find applications in **assistive composing tools**



Models

Proposed models – LSTMs

- LSTM [1]
  - An online model
  - 3 layers
    - 128 hidden units per layer
- BiLSTM [2]
  - An offline model
  - 3 layers
    - 64 hidden units per layer

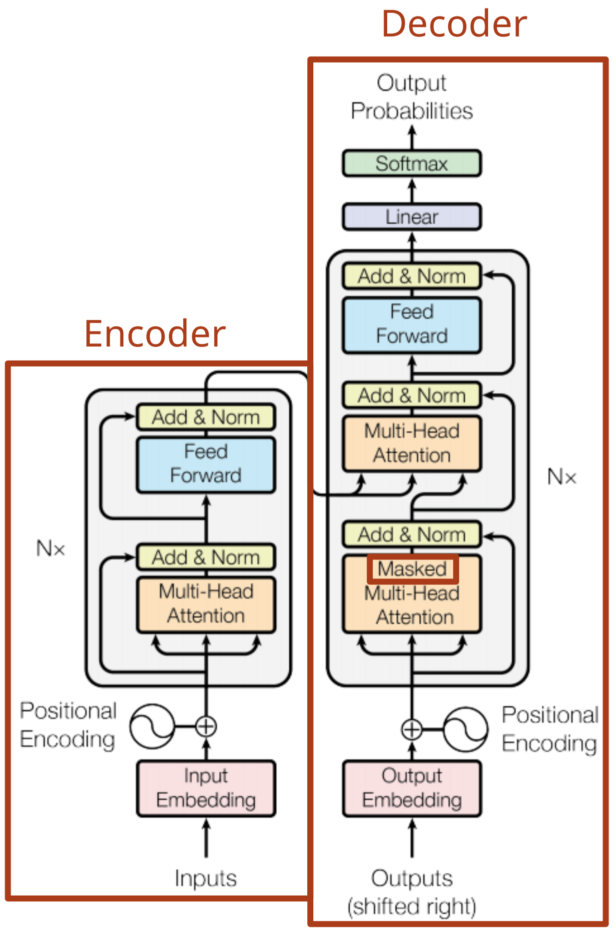


(Source: [3])

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 9(8):1735–1780, 1997.
[2] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
[3] C. Olah, "Understanding LSTM Networks," Colah's Blog, 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Proposed models – Transformers

- Transformer-Enc [1]
  - An offline model
  - 3 multi-head self-attention blocks
    - 128 hidden units in multi-head self-attention with 8 heads
    - 256 hidden units in feedforward network
- Transformer-Dec [1]
  - An online model that uses the lookahead mask
  - 3 multi-head self-attention blocks
    - 128 hidden units in multi-head self-attention with 8 heads
    - 256 hidden units in feedforward network

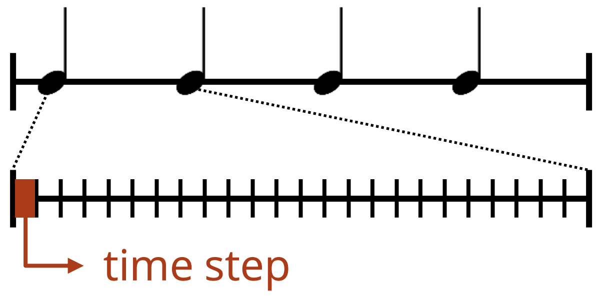


(Source: [1])

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

Input features

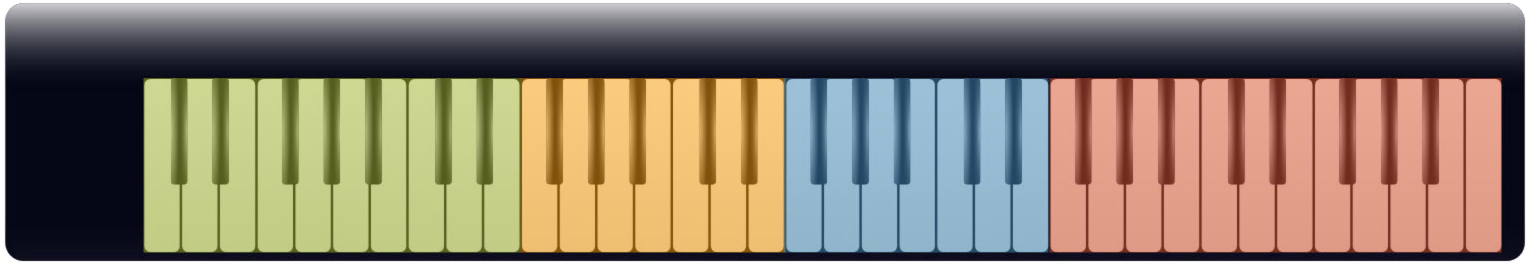
- **time**—onset time (in time step)
- **pitch**—pitch as a MIDI note number
- **duration**—note length (in time step)
- **frequency**—frequency of the pitch (in Hz; formula: $f = 440 \times 2^{(p-69)/12}$)



- **beat**—onset time (in beat)
- **position**—position within a beat (in time step)

Baseline models – Zone-based algorithm

- Simulate a common feature in modern keyboards
  - A pitch range (i.e., the *zone*) is preassigned for each instrument.
  - Notes will automatically be assigned to the corresponding instrument.
- Learn the optimal zones for the whole training data and use the learnt zones at test time
- *Oracle case*—Compute the optimal zones for each sample and use them at test time



Baseline models – Closest-pitch algorithm

- A heuristic algorithm assuming that "*the closer the pitches are, the more likely they belong to the same part*"
- Require the onset time of each track for initialization

$M = 0$ allows polyphonic parts
$M \gg 0$ assumes each part is monophonic

$$\hat{y}_i = \begin{cases} y_i, & \text{if } x_i \text{ is an onset} \\ \underset{j \in \{1,\ldots,k\}}{\arg\min} \left(p_i - \boxed{p'_j}\right)^2 & \text{otherwise} \end{cases}$$

last active pitches of track j

whether track i is active

Baseline models – Multilayer perceptrons (MLPs)

- Use multilayer perceptrons with hand-crafted features that encode the context [1]
- Use 3 fully-connected layers with 128 hidden units each
- Modifications:
  - Remove `interval' features as there is no upper bound for the number of concurrent notes
  - Change the proximity function to L1 distance
- *Oracle case*—Replace prior predictions with ground truth history labels

Index	Feature	Description
0	*pitch*	pitch, as a MIDI number
1	*duration*	duration, in whole notes
2	*isOrnamentation*	true (1) if a 16th note or shorter, false (0) if not
3	*indexInChord*	index (pitch-based) in the chord
4	*pitchDistBelow*	distance to note below
5	*pitchDistAbove*	distance to note above
6	*chordSize*	number of chord notes
7	*metricPosition*	metric position in the bar
8	*numNotesNext*	number of notes (onsets) in the next chord
~~9-12~~	~~*intervals*~~	~~intervals in the chord~~
13-17	*pitchProx*	for each voice v, the pitch proximity to the adjacent left note in v
18-22	*interOnsetProx*	idem, inter-onset
23-27	*offsetOnsetProx*	idem, offset-onset
28-32	*voicesOccupied*	for each voice v, whether it is currently occupied (1) or not (0)

(Source: [1])

[1] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," *ISMIR*, 2019.

Data

Datasets

- To examine the effectiveness of the proposed models, we consider four datasets that are diverse in their genres, sizes and ensembles.

Dataset	Hours	Files	Notes	Parts	Ensemble	Most common label
Bach chorales [31]	3.23	409	96.6K	4	soprano, alto, tenor, bass	bass (27.05%)
String quartets [32]	6.31	57	226K	4	first violin, second violin, viola, cello	first violin (38.72%)
Game music [33]	45.05	4.61K	2.46M	3	pulse wave I, pulse wave II, triangle wave	pulse wave II (39.35%)
Pop music [34]	1.02K	16.2K	63.6M	5	piano, guitar, bass, strings, brass	guitar (42.50%)

Table 1. Statistics of the four datasets considered in this paper.

[31] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," *ISMIR*, 2010.
[32] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," *ICLR*, 2017.
[33] C. Donahue, H. H. Mao, and J. McAuley, "The NES music database: A multi-instrumental dataset with expressive performance attributes," *ISMIR*, 2018.
[34] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching," Ph.D. dissertation, Columbia University, 2016.

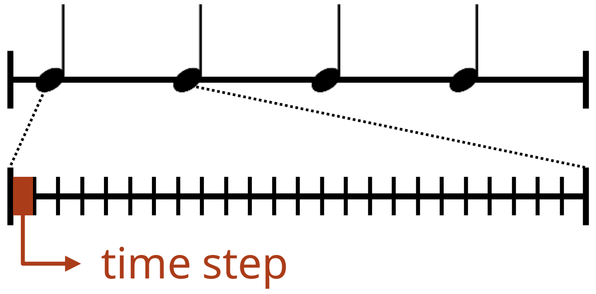
Data cleansing

- **Game music dataset**
  - Discard the percussive track as it does not follow the standard 128-pitch system

- **Pop music dataset**
  - Use a cleansed pop music subset [1] of Lakh MIDI Dataset
  - Map the instruments to the five most common instrument families—piano, guitar, bass, strings and brass—according to General MIDI 1 specification
  - Discard other instruments (might sometimes be the melody track)

[1] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," *AAAI*, 2018.

Data preprocessing

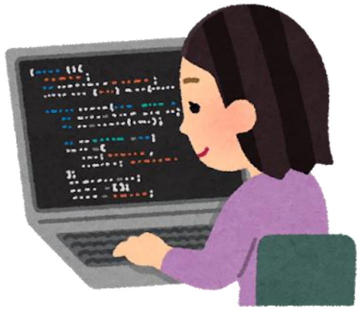


- Discard songs with only one active track
- **Bach chorales**, **string quartets** and **pop music**:
  - Use metric timing (a time step corresponds to some fraction of a quarter note)
  - Downsample to 24 time steps per quarter note (covering 32nd notes and triplets)
  - Split each dataset into train–test–validation sets with a ratio of $8 : 1 : 1$
- **Game music**:
  - Use absolute timing
  - Downsample to a temporal resolution equivalent to 24 time steps per quarter note in a tempo of 125 quarter notes per minute (qpm)
  - Use the original splits provided along with the dataset

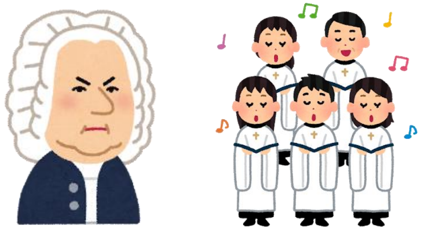
Experiments & Results

Implementation details

- Clip the time by 4096 time steps, the beat by 4096 beats and the duration by 192 time steps
- Use dropout and layer normalization
- Settings:
  - Batch size—16
  - Sequence length—500 for training and (up to) 2000 for validation/testing
  - Loss—cross entropy loss
  - Optimizer—Adam with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$
  - Framework—TensorFlow
  - Hardware—NVIDIA GeForce RTX 2070



Error analysis – Bach chorales



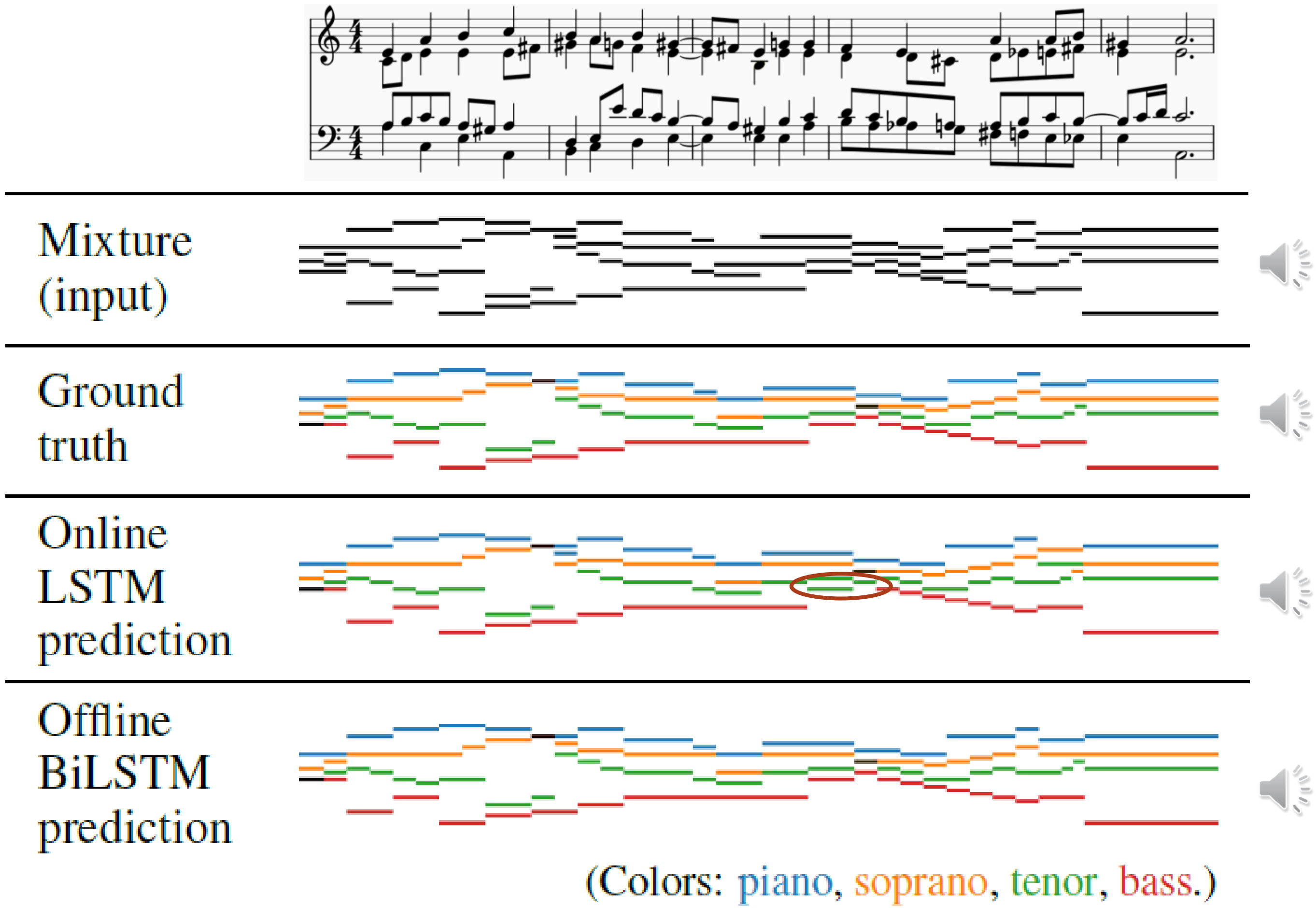


Figure 3. Example of the Bach chorales dataset—*Wer nur den lieben Gott läßt walten*, BWV 434, measures 1–5. The LSTM model makes two errors for the bass, as indicated by the arrow. The BiLSTM model gives a perfect prediction. Audio can be found in the Supplementary Material.

Error analysis – String quartets

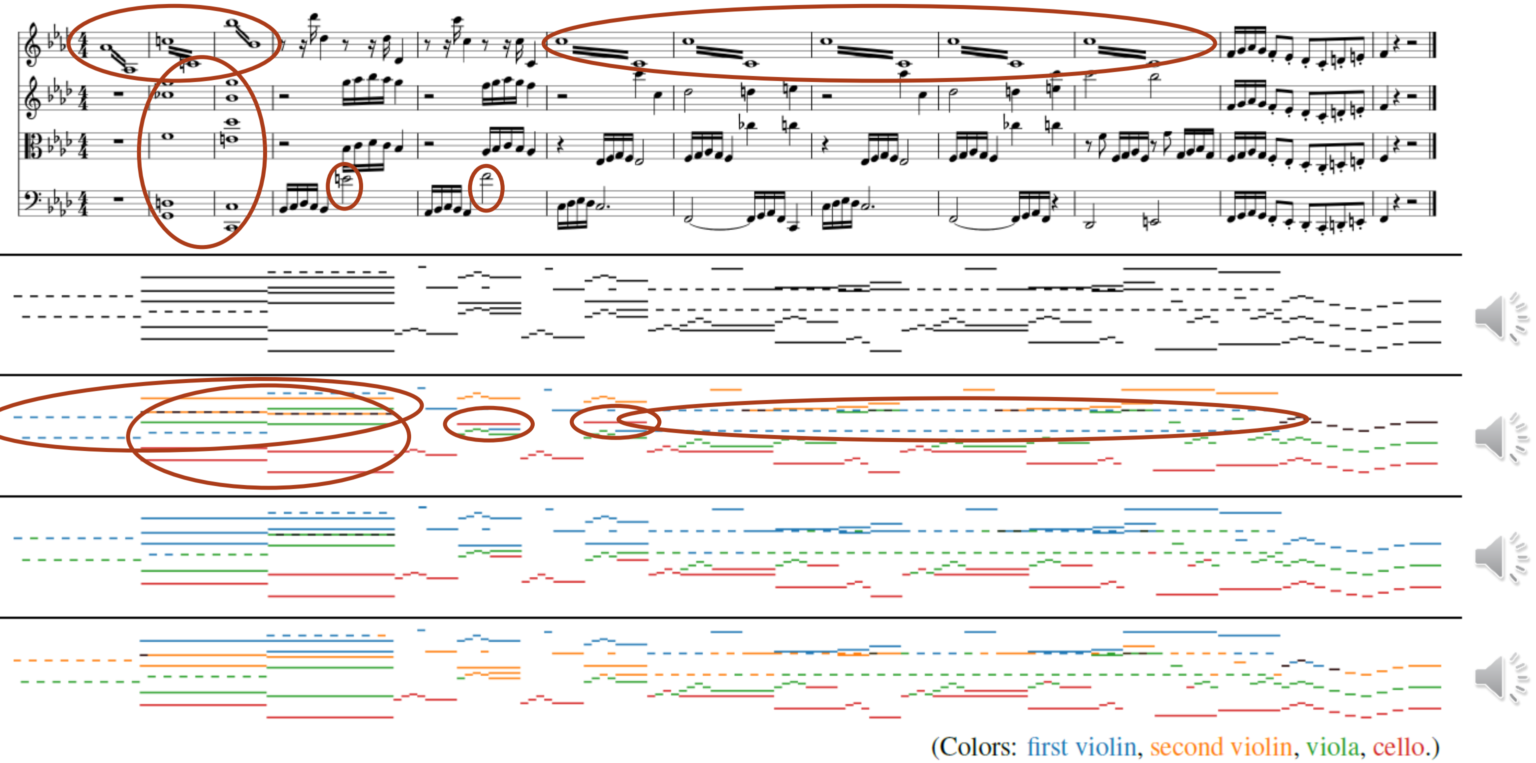


Figure 2. Hard excerpt in the string quartets dataset—Beethoven's *String Quartet No. 11 in F minor, Op. 95,* movement 1, measures 72–83. The tremolos of the first violin (measures 1–3 and 6–10), the double stops (measures 2–3) and the overlapping pitch ranges (measures 2–5) together compose a complex texture. Both models fail to handle the violins and viola properly, especially the second violin. Audio can be found in the Supplementary Material.

Error analysis – Game music

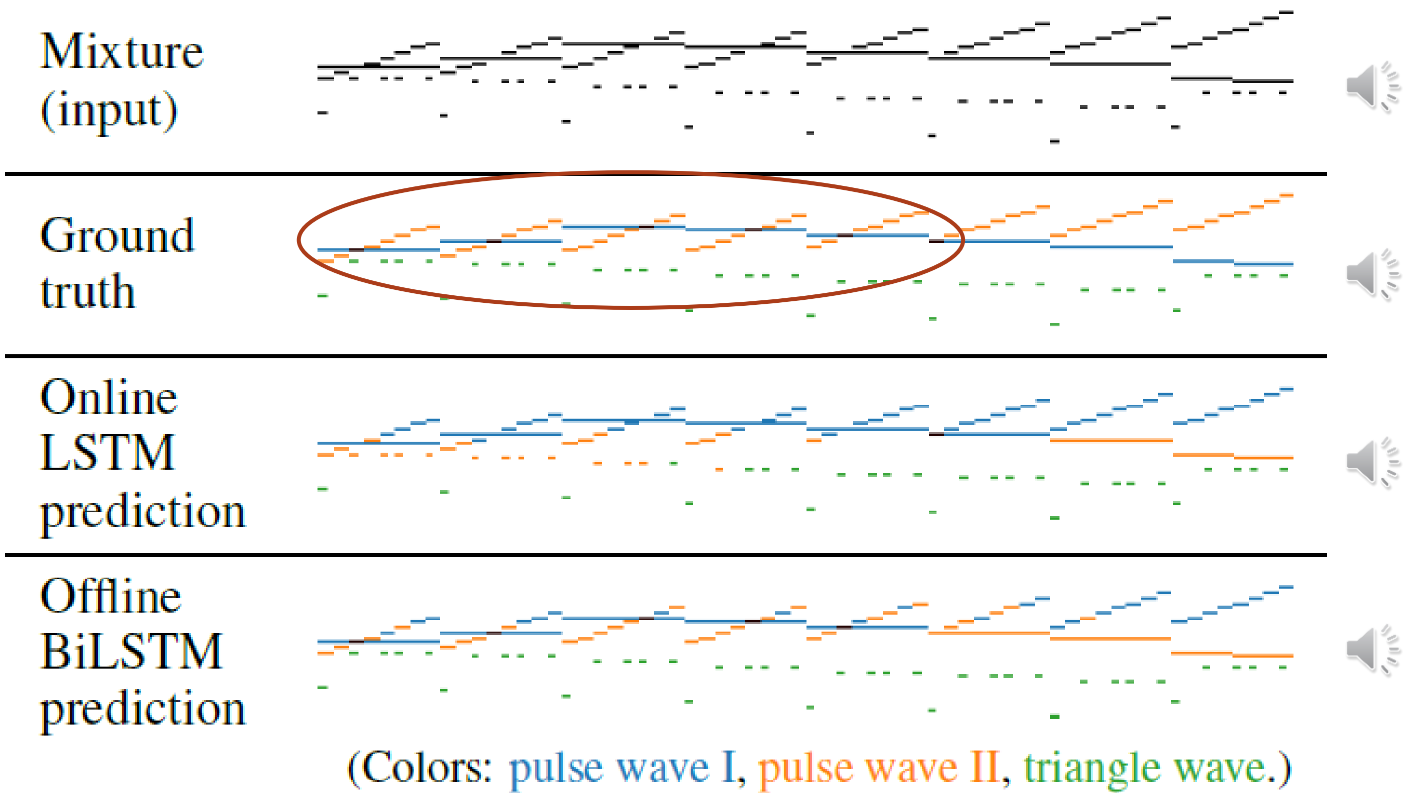


Figure 4. Hard excerpt in the game music dataset—*Theme of Universe* from *Miracle Ropit's Adventure in 2100*. Both models perform poorly when there is a sequence of short notes crossing a single long note. Audio can be found in the Supplementary Material.

Error analysis – Pop music

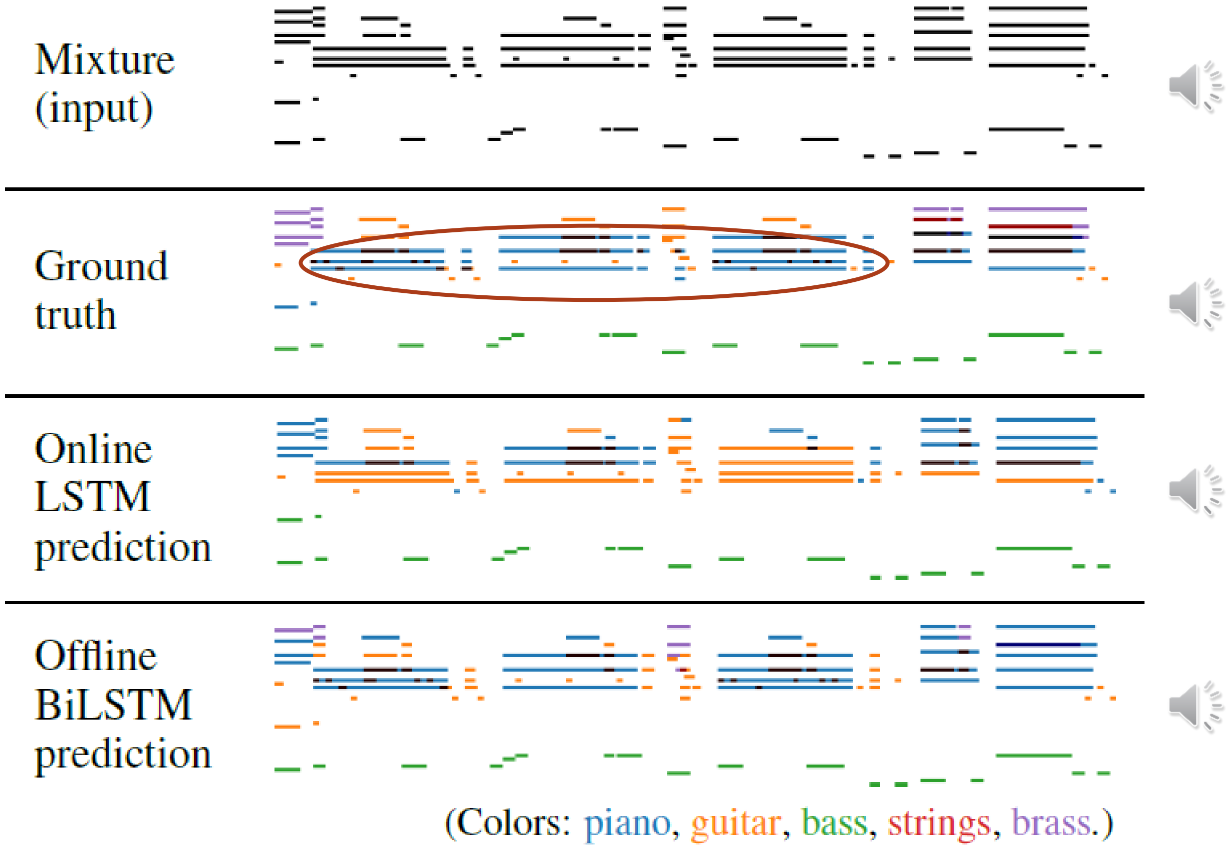
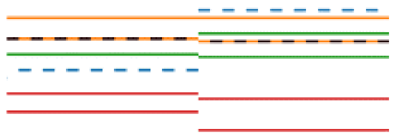


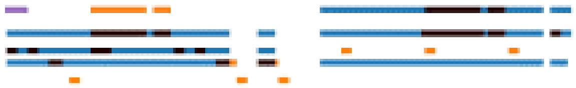
Figure 5. Hard excerpt in the pop music dataset—*Blame It On the Boogie* by The Jacksons. The BiLSTM model correctly identify and separate the overlapping guitar melody and piano chords, while the LSTM model fails in this case. Audio can be found in the Supplementary Material.

Representative error cases

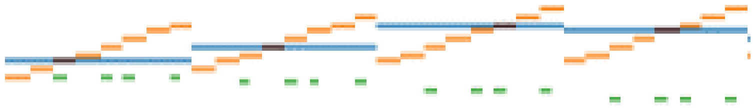
- Overlapping pitch ranges or chords for two polyphonic instruments



- Overlapping melodies and chords



- A sequence of short notes crossing a single long note



Quantitative Results

- Proposed models outperform baselines.
- BiLSTM outperforms LSTM.
  - BiLSTM have access to future information.
- LSTM models outperform Transformer models.
  - LSTM outperforms Transformer-Dec.
  - BiLSTM outperforms Transformer-Enc.
  - However, Transformer models benefit from faster inference speed at test time.

Model	Bach	String	Game	Pop
Online models				
Zone-based	73.14	58.85	43.67	57.07
MLP [9]	81.63	29.85	43.08*	33.50*
LSTM	**93.02**	**67.43**	**50.22**	**74.14**
Transformer-Dec	91.51	57.03	45.82	62.14
Zone-based (oracle)	78.33	66.89	79.54*	†
MLP [9] (oracle)	97.59	58.16	65.30	44.62
Offline models				
BiLSTM	**97.13**	**74.38**	**52.93**	**77.23**
Transformer-Enc	96.81	58.86	49.14	66.57
Online models (+entry hints)				
Closest-pitch	68.87	50.69	57.14	47.45
Closest-pitch (mono)	89.76	42.82	49.91	32.28
LSTM	**92.70**	**62.64**	**62.11**	**74.19**
Transformer-Dec	91.17	62.12	56.73	67.19
Offline models (+entry hints)				
BiLSTM	**97.39**	**71.51**	**64.79**	**75.59**
Transformer-Enc	93.81	56.72	54.67	67.23

*Reported on a subset of 100 test samples due to high computation cost.
†Omitted due to high computation cost.

Table 2. Comparison of our proposed models and baseline algorithms. Performance is measured in accuracy (%).

Quantitative Results

String quartets: **first violin**, **second violin**, viola, cello
Game music: **pulse wave I**, **pulse wave II**, triangle wave

- MLP baseline improves significantly when ground truth history labels are provided.
  - Errors can propagate over time as it predicts the label for each note independently
  - Emphasize the need to incorporate sequential models for this task

- Proposed models perform relatively poorly on string quartets & game music.
  - The two violins and two pulse waves are sometimes used interchangeably.
  - Motivate us to examine the use of pitch hints

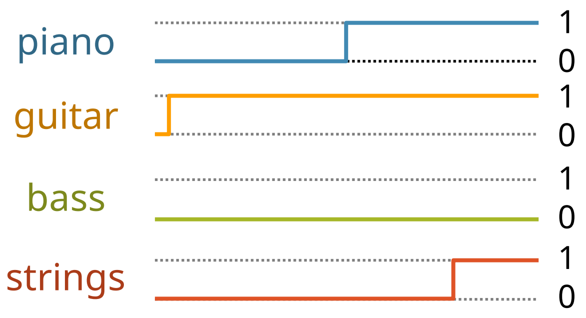
Model	Bach	String	Game	Pop
Online models				
Zone-based	73.14	58.85	43.67	57.07
MLP [9]	81.63	29.85	43.08*	33.50*
LSTM	**93.02**	**67.43**	**50.22**	**74.14**
Transformer-Dec	91.51	57.03	45.82	62.14
Zone-based (oracle)	78.33	66.89	79.54*	†
MLP [9] (oracle)	97.59	58.16	65.30	44.62
Offline models				
BiLSTM	**97.13**	**74.38**	**52.93**	**77.23**
Transformer-Enc	96.81	58.86	49.14	66.57
Online models (+entry hints)				
Closest-pitch	68.87	50.69	57.14	47.45
Closest-pitch (mono)	89.76	42.82	49.91	32.28
LSTM	**92.70**	**62.64**	**62.11**	**74.19**
Transformer-Dec	91.17	62.12	56.73	67.19
Offline models (+entry hints)				
BiLSTM	**97.39**	**71.51**	**64.79**	**75.59**
Transformer-Enc	93.81	56.72	54.67	67.23

*Reported on a subset of 100 test samples due to high computation cost.
†Omitted due to high computation cost.

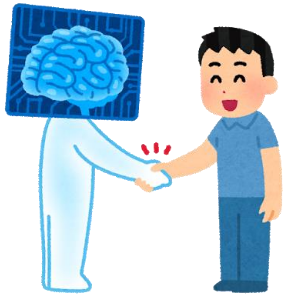
Table 2. Comparison of our proposed models and baseline algorithms. Performance is measured in accuracy (%).

Hints for controllability

- **Pitch hints**—average pitch of each part
  - Helpful in differentiating instruments that are used interchangeably
- **Entry hints**—onset position for each instrument
  - Encoded as a unit step function centered at its onset time
  - Also used in the closest-pitch algorithm



- Allow the musician to use interactively to make the instrumentation process more controllable



Quantitative Results

- Proposed models outperform baselines.
- BiLSTM outperforms LSTM.
- LSTM models outperform Transformer models.
- Closest-pitch algorithm with the monophonic assumption achieves a surprisingly high accuracy on Bach chorales.

Model	Bach	String	Game	Pop
Online models				
Zone-based	73.14	58.85	43.67	57.07
MLP [9]	81.63	29.85	43.08*	33.50*
LSTM	**93.02**	**67.43**	**50.22**	**74.14**
Transformer-Dec	91.51	57.03	45.82	62.14
Zone-based (oracle)	78.33	66.89	79.54*	†
MLP [9] (oracle)	97.59	58.16	65.30	44.62
Offline models				
BiLSTM	**97.13**	**74.38**	**52.93**	**77.23**
Transformer-Enc	96.81	58.86	49.14	66.57
Online models (+entry hints)				
Closest-pitch	68.87	50.69	57.14	47.45
Closest-pitch (mono)	89.76	42.82	49.91	32.28
LSTM	**92.70**	**62.64**	**62.11**	**74.19**
Transformer-Dec	91.17	62.12	56.73	67.19
Offline models (+entry hints)				
BiLSTM	**97.39**	**71.51**	**64.79**	**75.59**
Transformer-Enc	93.81	56.72	54.67	67.23

*Reported on a subset of 100 test samples due to high computation cost.
†Omitted due to high computation cost.

Table 2. Comparison of our proposed models and baseline algorithms. Performance is measured in accuracy (%).

Effects of input features

Four input features:

- Pitch, beat and position embedding (*Emb*)
- Duration (*Dur*)
- Entry hints (*EH*)
- Pitch hints (*PH*)

String quartets: **first violin**, **second violin**, viola, cello
Game music: **pulse wave I**, **pulse wave II**, triangle wave

Emb	Dur	EH	PH	Bach	String	Game	Pop
				92.10	37.29	43.89	58.78
✓				93.02	67.43	50.22	74.14
✓	✓			**96.17**	66.96	51.38	**78.17**
✓		✓		92.70	62.64	62.11	74.19
✓	✓	✓		95.95	68.17	63.35	74.74
✓			✓	92.87	**70.20**	**67.45**	75.89

Table 3. Effects of input features to the online LSTM model. Performance is measured in accuracy (%). Abbreviations: 'Emb'—pitch, beat and position embedding, 'Dur'—duration, 'EH'—entry hints, 'PH'—pitch hints.

Effects of time encoding

Four strategies:

- Raw time as a number
- Raw beat and position as two numbers
- Time embedding
- Beat and position embedding

Strategy	Bach	String	Game	Pop
Time encoding				
Raw time	91.97	37.26	44.10	37.92
Raw beat and position	**93.13**	66.72	48.60	68.42
Time embedding	92.21	**68.31**	49.32	70.64
Beat and position emb.	93.02	67.43	**50.22**	**74.14**
Data augmentation				
No augmentation	**93.03**	**69.36**	49.03	70.73
Light augmentation	92.85	68.66	46.38	71.10
Strong augmentation	93.02	67.43	**50.22**	**74.14**

Table 4. Comparisons of time encoding and data augmentation strategies for the online LSTM model. Performance is measured in accuracy (%).

Effects of data augmentation

Three strategies:

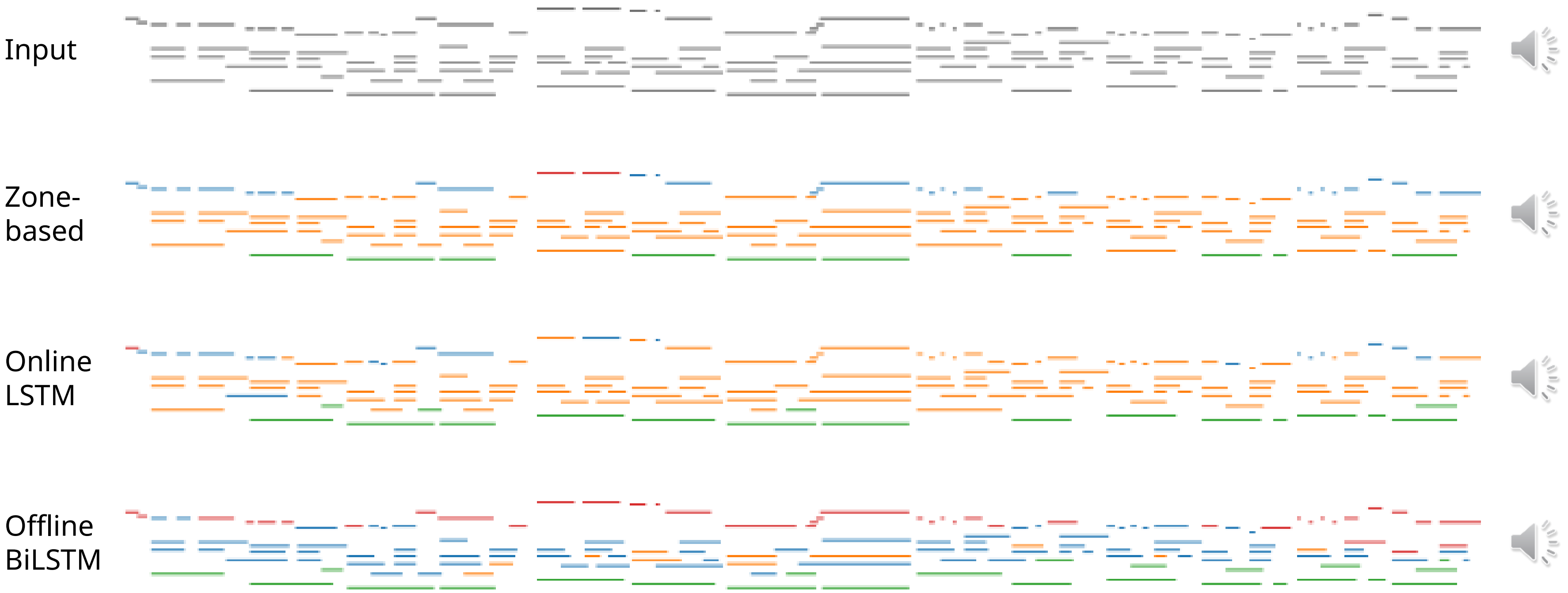
- No augmentation
- Light augmentation—randomly transposed by -1 to +1 semitone (during training only)
- Strong augmentation—randomly transposed by -5 to +6 semitones (during training only)

Strategy	Bach	String	Game	Pop
Time encoding				
Raw time	91.97	37.26	44.10	37.92
Raw beat and position	**93.13**	66.72	48.60	68.42
Time embedding	92.21	**68.31**	49.32	70.64
Beat and position emb.	93.02	67.43	**50.22**	**74.14**
Data augmentation				
No augmentation	**93.03**	**69.36**	49.03	70.73
Light augmentation	92.85	68.66	46.38	71.10
Strong augmentation	93.02	67.43	**50.22**	**74.14**

Table 4. Comparisons of time encoding and data augmentation strategies for the online LSTM model. Performance is measured in accuracy (%).

Out-of-distribution testing

- Test on POP909 [1] (trained on LMD)



[1] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A Pop-song Dataset for Music Arrangement Generation," *ISMIR*, 2020.

Out-of-distribution testing

- Test on POP909 [1] (trained on LMD)

[1] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A Pop-song Dataset for Music Arrangement Generation," *ISMIR*, 2020.

Discussion & Conclusion

Ambiguity of the task

- Automatic instrumentation is essentially a **one-to-many mapping**.

Original

LSTM without entry hints

BiLSTM with entry hints

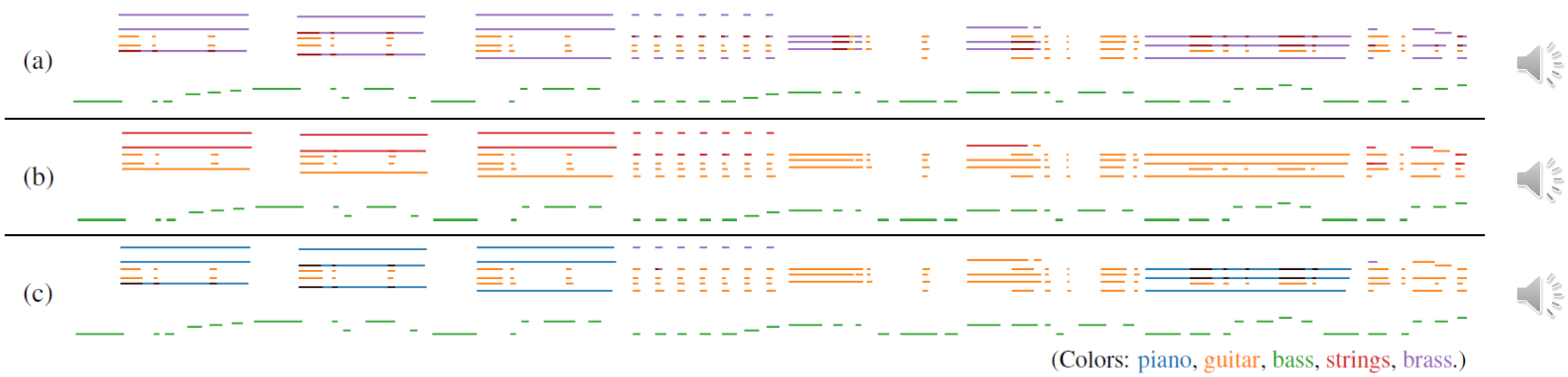
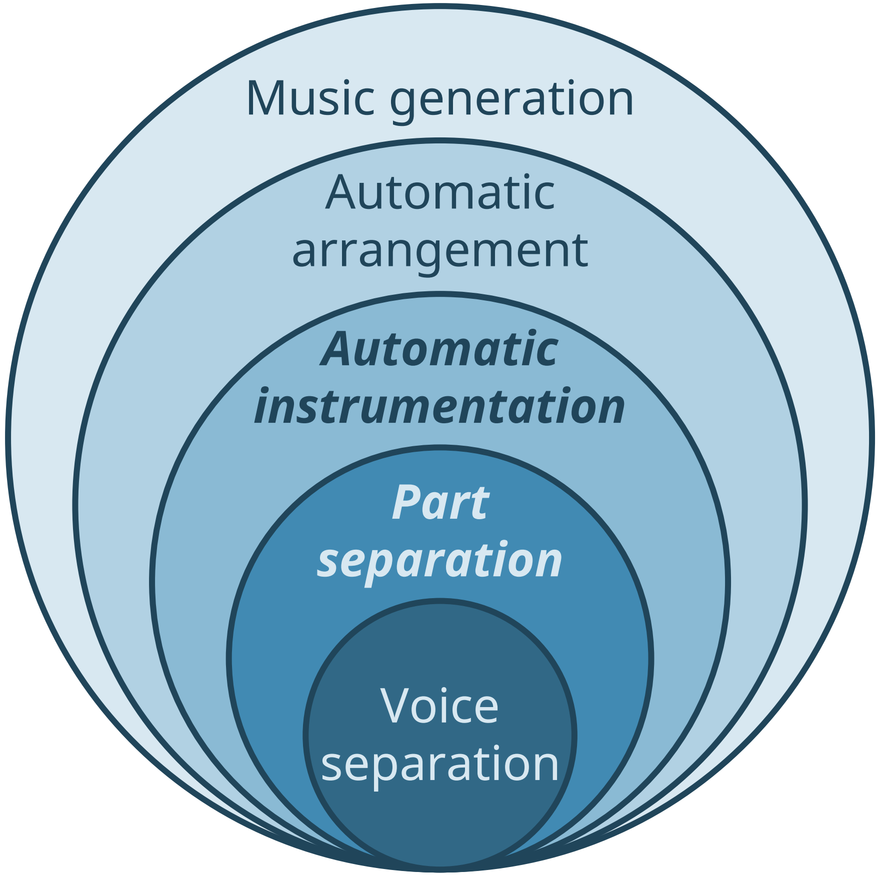


Figure 6. *Quando Quando Quando* by Tony Renis—(a) original instrumentation and the versions produced by (b) the online LSTM model without entry hints and (b) the offline BiLSTM model with entry hints. The LSTM model assigns the chords to the guitar, the most common instrument in the pop music dataset except the high pitches, which are assigned to the strings. The BiLSTM model is able to separate the long chords from the short ones and assigns the former to the piano. Audio can be found in the Supplementary Material.

Limitations

- Part separation is not a perfect surrogate task for automatic instrumentation.
  - Input mixture might not be playable
- Automatic instrumentation needs to balance between accuracy and diversity.
  - Could benefit from generative modeling



Future directions

- Generative modeling of automatic instrumentation
  - Learn the one-to-many mapping
  - Balance between accuracy and diversity
- Unpaired automatic instrumentation
  - Easy to find solo music and easy to find multitrack music
  - Can we unsupervisedly learn an automatic instrumentation model?
- Large-scale pretraining for symbolic music models
  - Part separation could be an additional source of music knowledge supervision
  - Could improve performance for other downstream symbolic music tasks

Conclusion

- Proposed a new task of part separation and framed it as a sequential multi-class classification problem
- Examined the feasibility of part separation under both the online and offline settings
- Showed that our proposed models outperform various baselines through a comprehensive empirical evaluation over four diverse datasets
- Presented promising results for applying part separation models to automatic instrumentation
- Discussed the ambiguity of the task and suggested future directions

Thank you!

[Source code] github.com/salu133445/arranger

[Demo website] salu133445.github.io/arranger/