TOWARDS AUTOMATIC INSTRUMENTATION BY LEARNING TO SEPARATE PARTS IN SYMBOLIC MULTITRACK MUSIC

Hao-Wen Dong[1]   **Chris Donahue**[2]   **Taylor Berg-Kirkpatrick**[1]   **Julian McAuley**[1]

[1] University of California San Diego   [2] Stanford University

hwdong@ucsd.edu   cdonahue@cs.stanford.edu   tberg@ucsd.edu   jmcauley@ucsd.edu

ABSTRACT

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. In this paper, we aim to further extend this idea and examine the feasibility of automatic instrumentation—dynamically assigning instruments to notes in solo music during performance. In addition to the online, real-time-capable setting for performative use cases, automatic instrumentation can also find applications in assistive composing tools in an offline setting. Due to the lack of paired data of original solo music and their full arrangements, we approach automatic instrumentation by learning to separate parts (e.g., voices, instruments and tracks) from their mixture in symbolic multitrack music, assuming that the mixture is to be played on a keyboard. We frame the task of part separation as a sequential multi-class classification problem and adopt machine learning to map sequences of notes into sequences of part labels. To examine the effectiveness of our proposed models, we conduct a comprehensive empirical evaluation over four diverse datasets of different genres and ensembles—Bach chorales, string quartets, game music and pop music. Our experiments show that the proposed models outperform various baselines. We also demonstrate the potential for our proposed models to produce alternative convincing instrumentations for an existing arrangement by separating its mixture into parts. All source code and audio samples can be found at `https://salu133445.github.io/arranger/`.

1. INTRODUCTION

Music is an art of time and sound. It often contains complex textures and possibly *parts* for multiple voices, instruments and tracks. While jointly following the global style and flow of the song, each part possesses its own characteristics and can develop different musical ideas independently. For example, in pop music, guitar and piano tend to play chords and might span across a large pitch
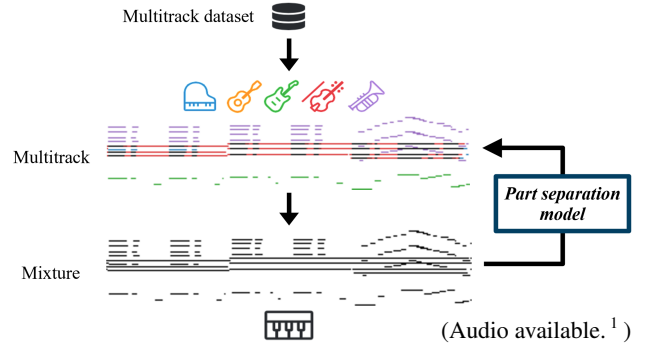
Figure 1. Proposed pipeline. By downmixing a symbolic multitrack into a single-track mixture, we acquire paired data of solo music and its instrumentation. We then use these paired data to train a *part separation* model that aims to infer the part label (e.g., one out of the five instruments in this example) for each single note in a mixture. Automatic instrumentation can subsequently be accomplished by treating input from a keyboard player as a downmixed mixture (bottom) and separating out the relevant parts (top). The music is visualized in the piano roll representation, where the x- and y-axes represent time and pitch, respectively. Colors indicate the instruments.

range, while bass is usually monophonic and stays in a lower range. While playing multiple instruments usually requires multiple performers, keyboardists potentially have the ability to control many instruments at once. Modern keyboards often offer the functionality of *zoning*, which allows a player to divide the pitch range into zones and assign each zone to a certain instrument. However, zoning is not ideal given its low flexibility that requires careful configuration and sometimes rearrangement of the music, and incapability for handling certain genres of music that have close and possibly overlapping harmony.

In this paper, we aim for the more ambitious goal of automatic instrumentation—a process that we define as dynamically assigning instruments to notes in solo music. A real-time, online automatic instrumentation model could allow a musician to have their keyboard performance instantaneously and seamlessly performed by a different ensemble. In addition to performative use cases, an offline automatic instrumentation model can also be useful to assist composers in suggesting proper instrumentation or providing a starting point for arranging a solo piece, especially for composers who have less experience arranging
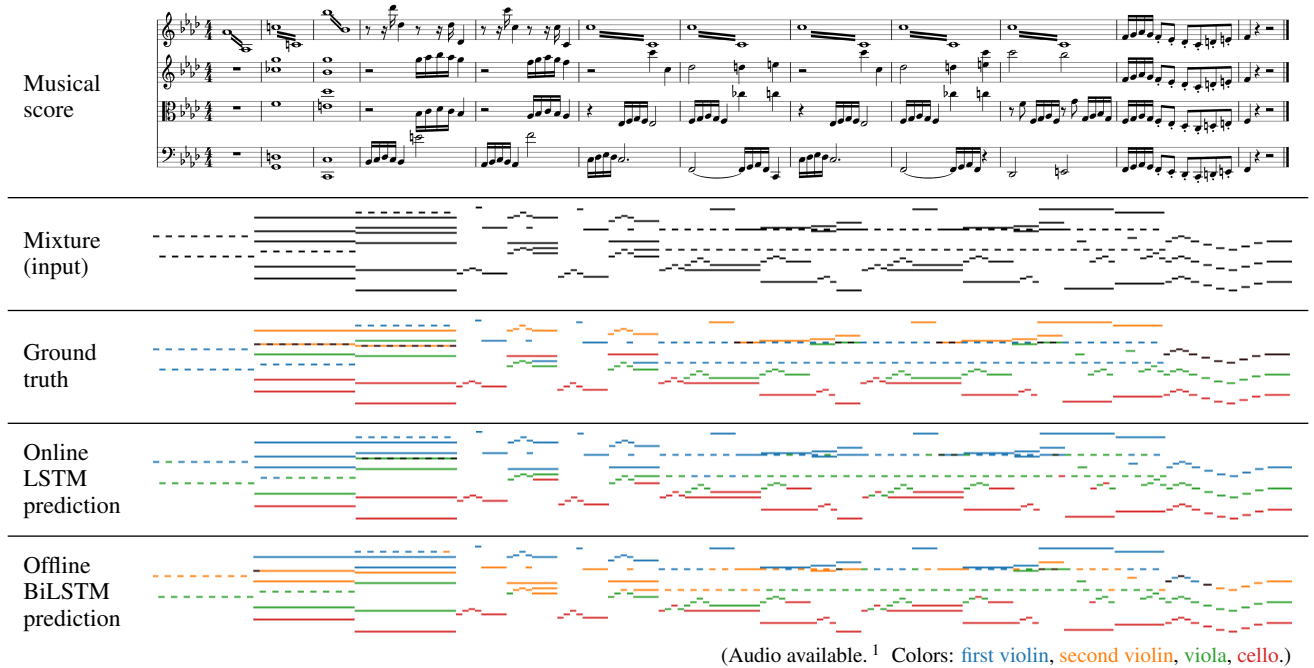
(Audio available. [1] Colors: first violin, second violin, viola, cello.)

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 for the second violin, viola and cello (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.

for a particular ensemble.

Automatic instrumentation is challenging as it requires domain knowledge of each target instrument, i.e., which pitches, rhythms, chords, and sequences thereof are playable, and it is hard to specify such knowledge by some fixed set of rules. In view of recent advances in machine learning, we propose to adopt a data-driven approach to this task. However, it can be laborious to acquire paired data of original solo music and their full arrangements. Given the abundance of multitrack music data, we approach automatic instrumentation by learning to separate parts from their mixture in multitrack music, a task we call *part separation* (see Figure 1). Assuming that the mixture is to be played on a keyboard and the multitrack is the target arrangement that we want to generate by an automatic instrumentation model, we thereby have paired data for solo music and full arrangements.

We frame the new task of part separation as a sequential multi-class classification problem that aims to map sequences of notes into sequences of part labels. We adopt long short-term memory (LSTM) [1] and Transformer [2] models for the task. We conduct an extensive empirical evaluation showing the superiority of our proposed models to baselines for the related task of voice separation as well as strategies found in commodity keyboards. To showcase the potential of our proposed models, we also demonstrate their ability to produce alternative convincing instrumentations for existing arrangements. Audio for all examples and more samples are available on the demo website. [1] All source code can be found in the project repository. [2]

---

2. PRIOR WORK

Voice separation is a related task to part separation which involves separating blended scores into individual monophonic voices. While useful, voice separation is agnostic to constraints imposed by specific instruments—a composer using a voice separation algorithm would have to manually align voices to appropriate instruments. Some prior work investigates voice separation in small, carefully-annotated pop music datasets [3, 4]. Some prior work on voice separation allows synchronous or overlapping notes in a voice [5–8]. However, their results are only reported on small test sets in certain genres. Others have adopted multilayer perceptrons [3, 9] and convolutional neural networks [10] with hand-crafted input features for voice separation. Another relevant work on hand detection in piano music used LSTMs to separate notes played by right and left hands in piano MIDI data [11]. To the best of our knowledge, no past work has examined the task of part separation in a general setting for multiple music genres.

In addition to voice separation, prior work has explored automatic music arrangement. The primary focus of prior work for automatic music arrangement has been on reduction—mapping musical scores for large ensembles to parts playable by a single specific instrument such as the piano [12–17], guitar [18–20] or bass [21]. This past work focuses on identifying the least important notes to delete so that the resultant score is playable on a single instrument, whereas our work seeks to preserve the original score in its entirety and satisfy playability for multiple instruments simultaneously. As an exception, Crestel and Esling [22] explore strategies for arranging orchestral music from pi-

ano, though their approach does not guarantee that all notes in the input piano map to parts in the output.

Music generation is another body of work that has used neural network sequential models for processing symbolic music [23]. Simon and Oore [24] proposed a convenient approach for music generation which involved training recurrent neural network language models on a language-like "event-based" representation of music. Subsequently, recent work has explored event-based representations using Transformers [2, 25–30]. In this work, we explore a more compact input representation of music that passes all of the information about a note into the model at once, rather than spreading it out across several events. We also note that Payne [26] generate music which contains parts for several instruments, but their model cannot be directly used to perform part separation of existing musical material.

3. PROBLEM FORMULATION

Mathematically, we consider a piece of music x as a sequence of notes $(x_1, \ldots, x_N)$, where N is the number of notes. Each note is represented by a tuple of time t_i and pitch p_i, i.e., $x_i = (t_i, p_i)$. Alternatively, we could also include duration d_i as an input and have $x_i = (t_i, p_i, d_i)$. Each note is associated with a label $y_i \in \{1, \ldots, K\}$ that represents the part it is in, where K is the number of parts. The goal of part separation is to learn the mapping between notes and part labels. This formulation is rather flexible and has no assumptions on whether a part is monophonic or not—it can be a voice, an instrument, a track, etc.

In terms of the context given for predicting the label of each note, we can categorize part separation models into three classes: An *independent model* predicts the label for each note independently, without any context. An *online model* predicts the label of the current note x_i given only past information, i.e., notes $(x_1, \ldots, x_{i-1})$, as context. An *offline model* predicts the label of the current note x_i given past and future information, i.e., the full sequence of $(x_1, \ldots, x_N)$, as context. While independent and online models are preferable for use cases that require real-time outputs, e.g., live performance. Moreover, the inability to look into the future makes the real-time setting more challenging than the offline setting. On the other hand, offline models can find applications in assisstive composing tools.

4. MODELS

We consider the following input features for our models—(1) *time*: onset time, in time step,[3] (2) *pitch*: pitch as a MIDI note number, (3) *duration*: note length, in time step, and (4) *frequency*: fundamental frequency of the pitch, in Hz, computed by the formula $f = 440 \cdot 2^{(p-69)/12}$. In addition, we also consider features that encode the metric time grid of music similar to the BAR and POSITION events proposed in [28]—(5) *beat*: onset time, in beat, and (6) *position*: position within a beat, in time step.

---

[3] Assuming that the music is in metrical timing, a time step is a factor of some musically-meaningful unit (e.g., a quarter note) and can be adjusted to match the desired temporal resolution.

Moreover, to help the models better disambiguate parts, we also include two simple hints—(7) *entry hints*: onset position for each instrument, encoded as a unit step function centered at its onset time and all zero if the instrument is not used, and (8) *pitch hints*: average pitch of each track. These hints allow the musician to use interactively to make the instrumentation process more controllable. For example, entry hints can be used to control the instruments available as they serve as switches for the instruments.

For the machine learning models, we consider the LSTM [1] and its bidirectional version (BiLSTM) [35]. We use a three-layer stacked LSTM with 128 hidden units in each layer (64 hidden units per layer for BiLSTM). We also consider two variants of Transformer [36]—one based on the encoder (Transformer-Enc) and one based on the decoder (Transformer-Dec). They share the same architecture that is composed of three Transformer blocks, each of which has 128 hidden units and 8 heads in self-attention computation and 256 hidden units in the internal feedforward network. However, they have different attention masks: Transformer-Enc uses only the padding mask, while Transformer-Dec uses both the padding mask and the lookahead mask, which blocks its access to future information and makes it a online model. In this paper, the LSTM and Transformer-Dec models are made online models, and the BiLSTM and Transformer-Enc models are made offline models that take durations as inputs.

5. BASELINE MODELS

In order to gain an insight into how the proposed models perform, we include two heuristic algorithms and a voice separation model from the literature in our empirical study.

5.1 Zone-based algorithm

This algorithm simulates a common feature in modern keyboards where a player can preassign a pitch range (i.e., the 'zone') for each instrument and notes will automatically be assigned to the corresponding instrument as the player performs. This algorithm finds the optimal zones for the whole training data and uses these optimal zones at test time. For the oracle case, the optimal zones for each sample are computed and used at test time. We note that the oracle case might not be easily achievable as it can be hard for a musician to set the zones optimally beforehand, especially for improvisation.

5.2 Closest-pitch algorithm

The closest-pitch algorithm keeps track of the last active pitches p_i' for each track i. For each incoming pitch p, it finds the pitch among the last active pitches that has the closest pitch to p and assigns the upcoming note with the same label as the chosen pitch. This is a casual model and it also relies on the onset hints. We can formulate this algorithm as follows. For $i = 1, \ldots, N$, we have

$$\hat{y}_i = \begin{cases} y_i, & \text{if } x_i \text{ is an onset} \\ \underset{j \in \{1, \ldots, K\}}{\arg\min} (p_i - p_j')^2 + Ma_i, & \text{otherwise} \end{cases},$$

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.

where p'_i is the last active pitch of track i before time t and a_i indicates whether track i is active, i.e., a concurrent note has not yet been released. We set M to a large positive number when we assume each part is monophonic, which we will refer to as the 'mono' version of this algorithm, otherwise set $M = 0$.

5.3 Multilayer perceptron (MLP)

We adapt the voice separation model proposed in [9] to the task of part separation. This model uses multilayer perceptron (MLP) to predict the label for the current note based on hand-crafted features that encodes its nearby context. We use entry hints rather than predicting them by the proposed voice entry estimation heuristics. We remove the 'interval' feature as there is no upper bound for the number of concurrent notes and change the proximity function to L1 distance. The oracle case of this model replaces error-prone prior predictions with ground truth history labels. In our implementation, we use three fully-connected layers with 128 hidden units each.

6. DATA

In order to examine the effectiveness of the proposed models, we consider four datasets—(1) *Bach chorales* in Music21 [31], (2) *string quartets* in MusicNet [32], (3) *game music* in Nintendo Entertainment System (NES) Music Database [33] and (4) *pop music* in Lakh MIDI Dataset [34], which are diverse in their genres, sizes and ensembles (see Table 1 for a comparison).

As these datasets are noisy in different ways, we need to further clean the data. For the game music dataset, we discard the percussive noise track in the original dataset as they do not follow the standard 128-pitch system used in other tracks. For the pop music dataset, we use a cleaned subset derived in [37], which contains only pop songs. We mapped the instruments to the five most common instrument families—piano, guitar, bass, strings and brass. We follow the General MIDI 1 specification on the mapping from an instrument to its instrument family. Instruments that fall outside of these five families are discarded. We note that the lead melody track might occasionally be discarded during the mapping process due to the high variance on instruments used for the melody track.

Moreover, we discard songs with only one active track as the task becomes trivial in this case. We note that all Bach chorales, string quartets and most pop songs are in metrical timing, where a time step corresponds to some fraction of a quarter note. Thus, we downsample them into 24 time steps per quarter note, which can cover 32nd notes
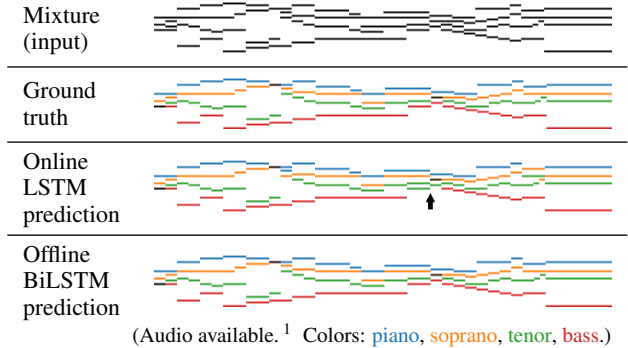

(Audio available. [1] Colors: piano, soprano, tenor, bass.)

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.
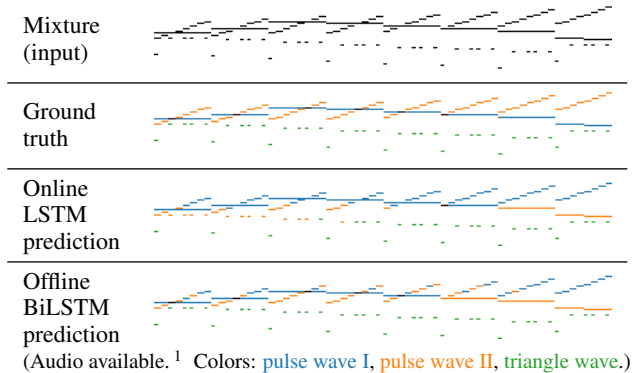

(Audio available. [1] Colors: pulse wave I, pulse wave II, triangle wave.)

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.

and triplets. As songs in the game music dataset are in absolute time, we downsample them to a temporal resolution equivalent to 24 time steps per quarter note in a tempo of 125 quarter notes per minute (qpm).

Finally, we split each dataset into train–test–validation sets with a ratio of $8 : 1 : 1$ except the game music dataset, where we use the original splits provided with the NES Music Database. We use MusPy [38] and music21 [31] for processing MIDI and MusicXML files.

7. EXPERIMENTS

7.1 Implementation details

We use a batch size of 16, a sequence length of 500 for training and a maximum sequence length of 2000 for validation and testing. We clip the time by 4096 time steps
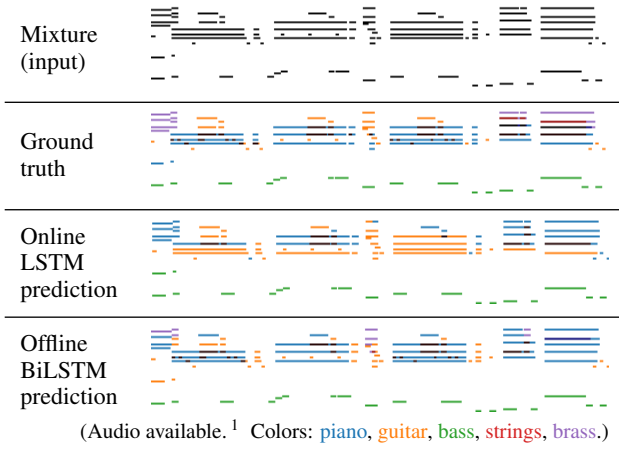
(Audio available. [1] Colors: piano, guitar, bass, strings, brass.)

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.

(i.e., roughly 170 quarter notes), the beat by 4096 beats, and durations by 192 time steps (i.e., 8 quarter notes). We randomly transpose the music by -5 to +6 semitones during training for data augmentation. We use the cross entropy loss with the Adam optimizer with $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [39]. We apply dropout [40] to prevent overfitting and layer normalization [41] to speed up the training. All models are implemented in TensorFlow [42] and experiments are run on NVIDIA GeForce RTX 2070s.

7.2 Qualitative results and error analysis

We present in Figures 2 to 5 several examples in the four datasets. Some representative cases include overlapping pitch ranges or chords for two polyphonic instruments (see Figures 2 and 5), overlapping melodies and chords (see Figure 5) and a sequence of short notes crossing a single long note (see Figure 4).

7.3 Quantitative results

We conduct an extensive empirical evaluation over different dataset and models in different settings. We present the results in Table 2.[4] First, we notice the improved performance for the oracle cases on the MLP baseline. The large gap of performance is possibly because it predicts each note independently and the errors can propagate over time. This emphasizes the need to incorporate sequential models for this task. Moreover, the BiLSTM model outperforms its LSTM counterpart for most cases. This is reasonable as the BiLSTM model has access to the future information, which could, for example, help identify the direction of an arpeggio. Further, the LSTM and BiLSTM models outperform their Transformer counterparts—Transformer-Dec and Transformer-Enc, respectively—across all settings. However, the Transformer models benefits from faster inference speed at test

---

[4] Due to high computation cost, we report the oracle cases for the zone-based algorithm and MLP model on a subset of 100 test samples, and omit the oracle case of the zone-based algorithm for the pop music dataset.

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 (%).

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.

time as compared to the LSTM models. Finally, we notice that the proposed models perform relatively poorly on the string quartets and game music datasets, possibly because the two violins in the string quartets dataset and the two pulse waves in the game music dataset are sometimes used interchangeably. We examine the use of pitch hints to help the models in the following section.

7.4 Effect of input features

In order to compare the effectiveness of different input features, we also report in Table 3 the performance for the LSTM model with different input features. First of all, pitch, note and beat embedding leads to improvements on all datasets, especially significant on the string quartet (30% gain) and pop music (15% gain) datasets. Second, entry hints improve the performance by 10% for the game music dataset, which is possibly because it helps disambiguate the two interchangeable pulse wave tracks. Interestingly, they have negative impacts on the Bach chorales and pop music datasets. Third, duration inputs are always helpful and help achieve the highest accuracy on the Bach chorales and pop music datasets. For example, durations
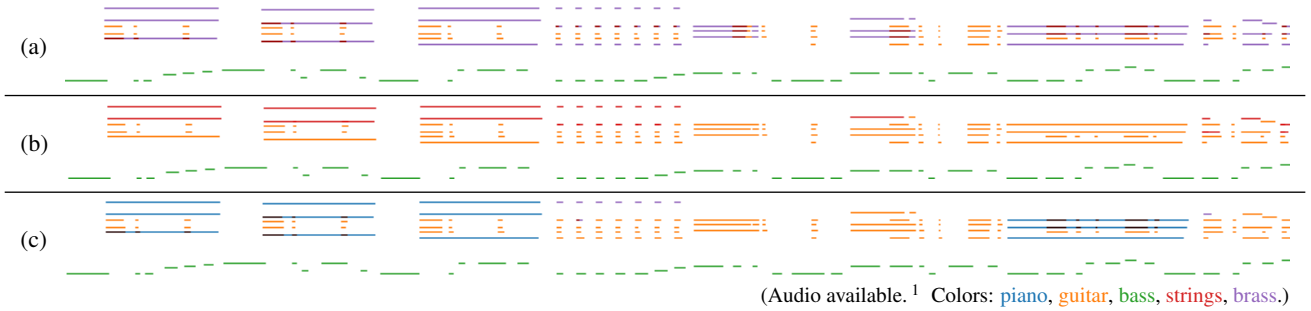
(Audio available.[1] Colors: piano, guitar, bass, strings, brass.)

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.

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 (%).

would be critical in distinguishing the overlapping guitar melody and piano chords in the example shown in Figure 5. Last, pitch hints improve the performance for all datasets but Bach chorales, possibly because the vocal ranges for SATB are strict in chorales. Pitch hints help achieve the highest accuracies for the string quartets and game music datasets as they help disambiguate interchangeable tracks.

7.5 Effects of time encoding

In this experiment, we examine the effects of time encoding. In particular, we consider four variants—(1) raw time as a number, (2) raw beat and position as two numbers, (3) time embedding and (4) beat and position embedding (see Section 4 for the definition of beat and position). We report in Table 4 the results and we can see that using raw time gives the worst performance. Interestingly, the other three encoding strategies achieve comparable performance.

7.6 Effects of data augmentation

In this experiment, we compare the following three strategies of data augmentation—(1) no augmentation, (2) *light augmentation*, where each song is randomly transposed by -1 to +1 semitone during training and (3) *strong augmentation*, where each song is randomly transposed by -5 to +6 semitones during training. We report in Table 4 the results. We can see that data augmentation is generally harmful for the Bach chorales and string quartets datasets, possibly be-

cause classical music has strict rules on the pitch ranges of voices and instruments. However, for game and pop music datasets, where rules on keys and pitch ranges in classical music are loosened, the models yield better performance with proper data augmentation.

8. DISCUSSION

In Figure 6, we depict the original instrumentation of the song *Quando Quando Quando* alongside the instrumentations generated by our best performing models for both the online and offline settings. While neither model produces an instrumentation identical to that of the original, both produce instrumentations that "cluster" notes similarly to the original and are reasonable rearrangements of the song. This indicates a fundamental ambiguity of the task, though we note that such ambiguity is more present in some genres than others—our models are able to achieve high accuracy on the Bach chorales dataset despite its small size, suggesting that this domain is less ambiguous.

In addition to its immediate musical applications, we believe that our proposed part separation task may be useful for large-scale pre-training of symbolic music models. Pre-training music generation models on large, heterogeneous music corpora has already been observed to improve performance for music generation of smaller, homogenous corpora [27, 43]. Given that our proposed task represents a substantial additional source of musical knowledge supervision, we speculate that additionally pre-training on this task could improve performance for many downstream symbolic music tasks.

9. CONCLUSION

In this paper, we have proposed a new task of part separation in multitrack music and examined its feasibility under both the online and offline settings. Through a comprehensive empirical evaluation over four diverse datasets, we showed the effectiveness of our proposed models against various baselines. We also presented promising results for applying part separation models to automatic instrumentation. Moreover, we discussed the fundamental ambiguity of the task and future research directions.

10. REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[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," *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[3] P. Gray and R. Bunescu, "A neural greedy model for voice separation in symbolic music," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[4] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, "Comparing voice and stream segmentation algorithms," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[5] J. Kilian and H. H. Hoos, "Voice separation — a local optimisation approach," in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, 2002.

[6] E. Cambouropoulos, "'Voice' separation: theoretical, perceptual and computational perspectives," in *Proceedings of the 9th International Conference on Music Perception & Cognition (ICMPC)*, 2006.

[7] I. Karydis, A. Nanopoulos, A. Papadopoulos, and E. Cambouropoulos, "VISA: The voice integration/segregation algorithm," in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[8] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, 2008.

[9] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[10] 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.

[11] A. Hadjakos, S. Waloschek, and A. Leemhuis, "Detecting hands from piano MIDI data," in *Mensch und Computer 2019 - Workshopband*, 2019.

[12] S.-C. Chiu, M.-K. Shan, and J.-L. Huang, "Automatic system for the arrangement of piano reductions," in *Proceedings of the 2009 11th IEEE International Symposium on Multimedia (ISM)*, 2009.

[13] S. Onuma and M. Hamanaka, "Piano arrangement system based on composers' arrangement processes," in *Proceedings of the 2010 International Computer Music Conference (ICMC)*, 2010.

[14] J.-L. Huang, S.-C. Chiu, and M.-K. Shan, "Towards an automatic music arrangement framework using score reduction," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 1, pp. 1–23, 2012.

[15] E. Nakamura and S. Sagayama, "Automatic piano reduction from ensemble scores based on merged-output hidden Markov model," in *Proceedings of the 2005 International Computer Music Conference (ICMC)*, 2015.

[16] H. Takamori, H. Sato, T. Nakatsuka, and S. Morishima, "Automatic arranging musical score for piano using important musical elements," in *Proceedings of the 14th Sound and Music Computing Conference, Aalto, Finland*, 2017, pp. 35–41.

[17] E. Nakamura and K. Yoshii, "Statistical piano reduction controlling performance difficulty," *APSIPA Transactions on Signal and Information Processing*, vol. 7, 2018.

[18] D. R. Tuohy and W. D. Potter, "A genetic algorithm for the automatic generation of playable guitar tablature," in *Proceedings of the 2005 International Computer Music Conference (ICMC)*, 2005.

[19] G. Hori, Y. Yoshinaga, S. Fukayama, H. Kameoka, and S. Sagayama, "Automatic arrangement for guitars using hidden Markov model," in *Proceedings of the 9th Sound and Music Computing Conference (SMC)*, 2012, pp. 450–456.

[20] G. Hori, H. Kameoka, and S. Sagayama, "Input-output HMM applied to automatic arrangement for guitars," *Information and Media Technologies*, vol. 8, no. 2, pp. 477–484, 2013.

[21] Y. Abe, Y. Murakami, and M. Miura, "Automatic arrangement for the bass guitar in popular music using principle component analysis," *Acoustical Science and Technology*, vol. 33, no. 4, pp. 229–238, 2012.

[22] L. Crestel and P. Esling, "Live orchestral piano, a system for real-time orchestral music generation," *arXiv preprint arXiv:1609.01203*, 2016.

[23] J.-P. Briot, G. Hadjeres, and F. Pachet, "Deep learning techniques for music generation: A survey," *arXiv preprint arXiv:1709.01620*, 2017.

[24] 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

[25] 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," in *Proceedings of the 7th International Conference for Learning Representations (ICLR)*, 2019.

[26] C. Payne, "MuseNet," OpenAI, 2019. [Online]. Available: https://openai.com/blog/musenet/

[27] 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," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[28] Y.-S. Huang and Y.-H. Yang, "Pop music Transformer: Generating music with rhythm and harmony," in *Proceedings of the 28th ACM International Conference on Multimedia (ACMMM)*, 2020.

[29] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the Transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[30] 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," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[31] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[32] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[33] C. Donahue, H. H. Mao, and J. McAuley, "The NES music database: A multi-instrumental dataset with expressive performance attributes," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (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.

[35] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

[37] 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," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[38] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "MusPy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 56, pp. 1929–1958, 2014.

[41] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *NeurIPS 2016 Deep Learning Symposium*, 2016.

[42] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, 2016.

[43] H.-T. Hung, C.-Y. Wang, Y.-H. Yang, and H.-M. Wang, "Improving automatic jazz melody generation by transfer learning techniques," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019.