LEARNING NOTE-TO-NOTE AFFINITY FOR VOICE SEGREGATION AND MELODY LINE IDENTIFICATION OF SYMBOLIC MUSIC DATA

Yo-Wei Hsiao **Li Su**
Institute of Information Science, Academia Sinica, Taiwan
{willyhsiao, lisu}@iis.sinica.edu.tw

ABSTRACT

Voice segregation, melody line identification and other tasks of identifying the horizontal elements of music have been developed independently, although their purposes are similar. In this paper, we propose a unified framework to solve the voice segregation and melody line identification tasks of symbolic music data. To achieve this, a neural network model is trained to learn note-to-note affinity values directly from their contextual notes, in order to represent a music piece as a weighted undirected graph, with the affinity values being the edge weights. Individual voices or streams are then obtained with spectral clustering over the learned graph. Conditioned on minimal prior knowledge, the framework can achieve state-of-the-art performance on both tasks, and further demonstrates strong advantages on simulated real-world symbolic music data with missing notes and asynchronous chord notes.

1. INTRODUCTION

Identifying the horizontal elements of music (e.g., melody, accompaniment, voice, stream, and counterpoint) is crucial for understanding musical data. As a mandatory step in music transcription [1] and generation [2, 3], this problem has been widely discussed; related tasks include melody extraction [4] and multi-pitch streaming [5, 6] for audio music data and voice segregation [7] and melody line identification [8] for symbolic music data. In this paper, we will focus on the case of symbolic music data.

It should be noted that the afore-mentioned tasks are rarely considered under a unified framework, but are solved individually according to the music texture of the input music. For example, voice segregation is only for *polyphony*, the texture with multiple independent melody lines; while melody line identification is only for *homophony*, the texture with one predominant melody line plus an accompaniment. The input music piece with hybrid or unknown texture cannot be discussed under these frameworks. Besides, most of these frameworks still heavily rely on further information of the input (e.g., number of voices, whether each voice is monophonic, whether chord notes are perfectly synchronized) and strictly follow some predefined rules (e.g., avoiding voice crossing), and turn out to be inflexible to real-world performance data with missing notes or asynchronous chord notes.

In this paper, we propose a unified horizontal element extraction framework which works with minimal constraints on musical textures and pre-defined rules. The major idea is to let the model learn the configuration of voice directly from the training data, and learn the *note-to-note affinity* for arbitrary pairs of notes within a musical segment from their shared contextual notes: the model outputs 1 if a pair of notes are in the same horizontal elements, while 0 if they are not. Then, based on the learned affinity values, a clustering algorithm is used to estimate the number of horizontal elements, and to partition the notes which are linked with high affinity values into one element. Finally, these elements extracted from different musical segment can be merged without any perceptual or musical assumptions by applying the *minimal overlapping* principle proposed in this paper.

To verify our ideas, the same framework is applied on multiple tasks with various conditions, including 1) the polyphony voice segregation task with unknown number of voices, missing notes and asynchronous chord notes for simulating real-world performance, and 2) the melody line identification task of homophonic music. The framework achieves state-of-the-art performance on both tasks and shows strong advantages on simulated real-world cases. Furthermore, we demonstrate the potential of using the graph constructed with the learned note-to-note affinity as a tool in computational analysis of general music data.

2. RELATED WORK

2.1 Voices, streams, and their perceptual rules

A *voice* or a *stream* is a horizontal music structure which is *perceived* as single sonority by humans. A voice is a sequence of monophonic and non-overlapped musical tones in polyphony texture, such as the S, A, T, and B in a 4-part chorale. On the other hand, a stream can be either a monophonic voice or a multi-tone sonority fused by several musical lines, such as the predominant *melody line* and *accompaniment* in homophonic texture. A monophonic note sequence may also contain multiple voices. The perception of voice or stream in music is highly subjective. The voice analysis for the very same music piece might end up with diverse results [9]. With abuse of terminology, the terms of voice, stream, and horizontal element are used interchangeably in the paper.

A number of perceptual rules have been proposed for voice segregation and melody identification tasks [10]. The *pitch proximity* and *temporal continuity* rules suggest that the temporal and pitch distance between two neighboring notes in a voice should be minimized, and large leaps or rests should be avoided [11]. The *new stream* rule suggests that the number of streams in a music piece should be minimized. The *voice collision* rule states that common tones shared between different voices should be avoided. The *voice crossing* suggests that two voices do not cross each other even when their pitch ranges overlap significantly.

2.2 Prior art

Voice segregation and melody line identification methods can be categorized into three classes. First, the *rule-based* methods, such as local optimization [12], contig mapping [13–16], graph-based method [17], complexity-based method [18], and the voice integration and segregation algorithm (VISA) [19] utilize heuristics or perceptual principles as constraints for tracing voices. These methods do not strictly designate the role of each stream (e.g., one stream should be melody and the other should be accompaniment), and therefore can be applied to the data having arbitrary configurations of streams without labels [20]. The major limitation of these methods is that they are less flexible dealing with real-world performance data.

Second, the *data-driven* methods introduce either classifiers to predict the voice or stream labels of each note from annotated data [7, 21–23], or regression models to predict the ratings over all the mappings from notes to voices for each chord [24, 25]. Representative examples include the convolutional neural network (CNN) model which predict the position of melody notes on a piano roll [26], or a feedforward neural network to classify voice indices from note-level features [27]. Different from the rule-based methods, data-driven methods are based on supervised learning. Therefore, the output dimension of the model is usually restricted by the label classes in the training data. This issue can be solved with neural greedy search such as [28], which implicitly indicated the importance of learning note-to-note affinity.

Besides the rule-based and data-driven approaches, most of the methods are *hybrid* ones which incorporate both perceptual rules and supervised learning in voice segregation and melody line identification. For example, in the hidden Markov model (HMM)-based voice segregation method, the probability of note transition is defined according to the perceptual principles, while the pitch score and gap score in probability function can be tuned to fit the training data [29]. In the classification-based methods, hand-crafted input features which consider the perceptual principles have been proposed in various ways [21,24,27]. Some of these features can be used only when the number of voice, the metric positions and other note attributes of the input music are known.

3. PROPOSED METHOD

We represent a symbolic music piece as a undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each vertex $v_i \in \mathcal{V}$ represents a note in MIDI, and each weighted edge $w_{ij} \in \mathcal{E}$ corresponds to the affinity between two notes v_i and v_j. We assume $w_{ij} = 1$ if v_i and v_j are in the same voice or melodic line, while $w_{ij} = 0$ if they are situated in different parts. The task of voice segregation is then equivalent to the task of learning w_{ij} from the training data having the binary-valued w_{ij} as the ground truth label.

Denote the affinity matrix of $\mathcal{G}$ as W, and $|\mathcal{V}|$ the vertex count of $\mathcal{G}$. Then, we have $W \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, and w_{ij} is the (i,j)th element of W. To learn W, we adopt a CNN model which takes the information carried by a pair of notes (v_i, v_j) as inputs, and outputs an affinity value $\hat{w}_{ij} \in [0,1]$ which minimizes the binary cross-entropy (BCE) between $\hat{w}_{ij}$ and w_{ij}. Once the predicted affinity matrix $\hat{W}$ is obtained, the task of voice segregation is simplified into a clustering problem. With the help of spectral clustering algorithm, $\mathcal{G}$ is partitioned into multiple subgraphs, each of which represents a voice.

3.1 Data representation

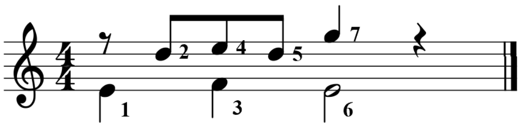
For simplicity, the vertex v_i directly represents the content of the ith note: each note event $v_i := [p_i, o_i, d_i]^T$ is a 3-dimensional vector composed of its pitch (in MIDI number), onset time, and duration (both in second); see Figures 1a and 1b. The order index i of each note is obtained by sorting all the note events with the following rules: 1) notes are sorted by its onset time in ascending order; 2) if multiple notes have the same onset time, they are then sorted by their pitch in ascending order; and 3) if multiple notes happen to have identical onset time and pitch value, they are sorted by their duration in descending order. The adopted data representation of each v_i, denoted as x_i, is simply constructed by v_i and its neighbouring notes. More specifically, x_i is defined as

$$x_i := [\bar{v}_{i-M,i}, \bar{v}_{i-M+1,i}, ..., \bar{v}_{i+M-1,i}, \bar{v}_{i+M,i}]^T \ , \quad (1)$$

and we have $x_i \in \mathbb{R}^{(2M+1)\times 3}$. It should be noted that $\bar{v}_{j,i}$ is the content of the jth note event with its onset time expressed relative to the ith note, i.e. $\bar{v}_{j,i} := [p_j, o_j - o_i, d_j]$. This operation makes the onset time information in each x_i be centered at the same temporal position. For $\bar{v}_{j,i}$ with $j \leq 0$ or $j > |\mathcal{V}|$, the note sequence is zero-padded (i.e., add virtual notes with its MIDI pitch, onset time and duration being all zeros) such that the $(M+1)$th row of x_i is $\bar{v}_{i,i}$, as shown in Figure 1c.

The idea behind the above data representation is that it simulates human behavior. When given tasks like voice segregation, humans do not plainly judge the affinity of a pair of notes merely by their own pitch and position, but by the local musical context lying in the structure. The hyperparameter M determines the context window, and we set $M = 60$ notes in this paper.

In the setup of affinity learning, the training data is then the pairs of the data representation given a binary label



(a) The score

$$v_1 = (64, 0, 0.5)$$
$$v_2 = (74, 0.25, 0.25)$$
$$v_3 = (65, 0.5, 0.5)$$
$$v_4 = (76, 0.5, 0.25)$$
$$\dots$$

(b) Note events

$$x_2 = \begin{bmatrix} 0 & 0 & 0 \\ 64 & -0.25 & 0.5 \\ 74 & 0 & 0.25 \\ 65 & 0.25 & 0.5 \\ 76 & 0.25 & 0.25 \end{bmatrix}$$

(c) The matrix

Figure 1: An example of data representation for $M = 2$.

$w_{ij} \in \{0, 1\}$. A training sample (x_i, x_j) is labeled as $w_{ij} = 1$ if v_i and v_j are in the same voice, whereas (x_i, x_j) is labeled as $w_{ij} = 0$ if they are not.

3.2 Model training

We employ a multi-layer deep 1-D CNN $f(X)$ to classify whether x_i and x_j are in the same musical stream. More specifically, given $X_{ij} \in \mathbb{R}^{(2M+1)\times 6}$ the concatenation of two matrices x_i and x_j, we have $\hat{w}_{ij} = f(X_{ij})$ so as to minimize $\text{BCE}(w_{ij}, \hat{w}_{ij})$. The architecture of the CNN contains six 1D convolution layers, with the kernel size of each layer being [32, 16, 16, 8, 8, 4], and the number of each kernel being [32, 32, 64, 64, 128, 128]. The output of the final convolution layer is flattened and mapped to an output logit using a fully connected layer.

For a music piece with $|\mathcal{V}|$ notes, there are totally $|\mathcal{V}|(|\mathcal{V}| - 1)/2$ pairs of notes that can be used for training. However, taking all the pairs for training is computationally intensive, and unnecessary from the perspective of music perception. When people listen to music, it is impractical to have them distinguish whether a pair of notes several bars apart belong to the same voice. Instead, listeners tend to focus on a restricted time interval and identify a voice from others according to the relationship among the notes in the interval. Therefore, we choose all of the pairs (x_i, x_j) with $|i - j| \leq N$ for training, where the hyperparameter N represents the maximum distance of two notes. In this paper, we set $N = 30$ notes according to our study.

3.3 Voice extraction with graph clustering

We employ spectral clustering [30, 31], one of the most widely used graph-based clustering techniques, to separate the voices or streams according to the learned affinity matrix $\hat{W} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ for the music data graph $\mathcal{G}$. However, it should be noted that the model $f(X_{ij})$ outputs $\hat{w}_{ij}$ (i.e. the estimated value between v_i and v_j) only for $|i - j| \leq N$; other $\hat{w}_{ij}$ are still unknown. Therefore, the following assignment processes are adopted to adjust the affinity matrix:

$$\begin{aligned} \hat{w}_{ij} &:= 1 \quad \text{if} \quad \hat{w}_{ij} > 0.5 \\ \hat{w}_{ij} &:= \epsilon_1 \quad \text{if} \quad \hat{w}_{ij} \leq 0.5 \\ \hat{w}_{ij} &:= \epsilon_2 \quad \text{if} \quad \hat{w}_{ij} \text{ is unknown} \end{aligned} \tag{2}$$

where $:=$ is the assignment operator. We assume $\epsilon_1 < \epsilon_2 < 1$ in order to represent the uncertainty of the affinity for distant note pairs (i.e. with $|i - j| > N$). Therefore, in this paper we set $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-3}$. Our study showed that these discretized values give stable eigendecomposition in the spectral clustering process and perform better than directly using predicted values.

It should be noted that performing spectral clustering over the whole music piece is unfeasible, as the eigendecomposition of a large affinity matrix tends to be highly unstable, an unfavorable effect in spectral clustering. To address this issue, we divide a musical piece into multiple overlapping segments and perform spectral clustering for each segment. Each of the segments contains S consecutive note events and each pair of consecutive segments are overlapped by O note events, where $0 \leq O < S$.

For $\hat{W}_i$, which denotes the affinity matrix of the ith segment, the normalized graph Laplacian L_i for spectral clustering is represented as

$$L_i = I - D^{-\frac{1}{2}} \hat{W}_i D^{\frac{1}{2}} , \tag{3}$$

where I is the identity matrix, and the degree matrix D is a diagonal matrix, whose ith diagonal element $d_{ii} := \sum_j \hat{w}_{ij}$ is the sum of the affinity measure between v_j and v_i for all v_j which are connected with v_i. Given the Laplacian L, a matrix $Z = [z_1, z_2, ..., z_k] \in \mathbb{R}^{n\times k}$ is formed by stacking the top-k largest eigenvectors z_i of L, and the k subsets is obtained by applying the k-means algorithm to the rows of Z [31].

3.4 Estimating the number of clusters

The number of clusters, or the number of estimated voices or streams, k, is a pre-determined parameter in the spectral clustering process. For the melody line identification task, k is simply 2. For the voice segregation task, we assume that k is unknown and needs to be estimated. An intuitive estimation is to set k as the maximal number of synchronous note events occurring in a piece [13]. However, this method may not be suitable for voices or streams having overlapped notes, such as human-performed MIDI or homophonic music data.

To address the issue, the *eigengap* of Laplacian L is employed to predict k of each segment. Let $\{\lambda_i\}_{i=1}^{N}$ be the eigenvalues of L sorted in descending order. According to graph theories, the number of clusters k of the graph can be estimated by the kth eigenvalue having the most significant eigengap of L [32]. To implement this, we calculate the top-10 largest eigenvalues of each L_i, and find the index k_i satisfying that the difference between λ_{k_i} and λ_{k_i+1} is the largest one. To simplify our discussion, we assume that the number of voices of the whole music piece is a constant, meaning that each voice in a piece contributes at least one note in any segment of the piece. The number of voices, k, is therefore obtained by the mode of k_i over all segments.

3.5 Segment merging

After we apply spectral clustering to $\hat{W}_i$, note events in the ith segment are partitioned into k segregated voices (or streams). Let $\mathcal{C}_i = \{\mathcal{S}_{ij}\}_{j=1}^{k}$ denote the ith segment with voice labeling, where $\mathcal{S}_{ij}$ denotes the jth voice obtained from the spectral clustering result of the ith segment. We consider two methods to merge the segmented voices into complete ones. The first method, called the *pitch proximity* method, is performed straightforwardly by sorting the segmented voices $\mathcal{S}_{ij}$ according to their average pitch for each i, and connecting the segmented voices with the same sorting index. Being stable and perceptually plausible, the pitch proximity method however assumes the prior knowledge of part-crossing rule (i.e. voices tend not to cross with respect to pitch) [11], which is no longer valid in the situations such as voice crossing nearby the end of a segment.

The second method, coined as the *minimal overlapping* method, is a newly proposed method which does not rely on any perceptual rules. In a nutshell, this method attempts to find a one-to-one mapping for voices from two adjacent segments $(\mathcal{C}_i, \mathcal{C}_{i+1})$ to attain a newly merged segment $\mathcal{C}_* = \{\mathcal{S}_{*j}\}_{j=1}^{k}$ such that

- a voice in the new segment $\mathcal{S}_{*j}$ is merged from two voices, $\mathcal{S}_{im}$ and $\mathcal{S}_{(i+1)n}$, $1 \leq m, n \leq k$, if and only if they are overlapped (connectivity condition);
- each note in either $\mathcal{C}_i$ or $\mathcal{C}_{i+1}$ should be in $\mathcal{S}_{*j}$ for some j and therefore in $\mathcal{C}_*$ (consistency condition);
- the total number of notes which belong to multiple merged voices of $\{\mathcal{S}_{*j}\}_{j=1}^{k}$ should be minimized (minimal redundancy condition).

These conditions are implemented with the following procedures. First, we list all the possible merged voices satisfying the connectivity condition, denoted as $\Sigma_1 := \{\mathcal{S}_{ij} \cup \mathcal{S}_{(i+1)l} \mid \mathcal{S}_{ij} \cap \mathcal{S}_{(i+1)l} \neq \varnothing\}$. A merged segment $\mathcal{C}_*$ with these merged voices is constructed by choosing k elements from Σ_1. Denote all candidates of such k chosen elements as $\binom{\Sigma_1}{k}$. Given a fixed-valued k and the consistency condition, we narrow down the set of candidates to $\Sigma_2 := \{\binom{\Sigma_1}{k} \mid \{\bigcup_{j=1}^{k} \mathcal{S}_{*j}\} = \mathcal{C}_i \cup \mathcal{C}_{i+1}\}$. To apply the minimal redundancy condition, we begin with calculating the number of notes existing in multiple voices (#NMV) for an arbitrary merged segment $\mathcal{C}'_*$, which is defined as

$$\#\text{NMV} := \sum_{j=i+1}^{k} \sum_{i=1}^{k-1} |\mathcal{S}'_{*i} \cap \mathcal{S}'_{*j}|, \quad (4)$$

where $\mathcal{S}'_{*i}, \mathcal{S}'_{*j} \in \mathcal{C}'_*$. Then, we pick the segment $\mathcal{C}_*$ with the smallest #NMV among all possible merged segments in Σ_2 to be our final choice. Finally, because we require that a note should be classified to one voice strictly in our study, we remove notes existing in multiple voices from a random voice to ensure $\mathcal{C}_*$ behaves like an ideal segment.

Once a larger segment is obtained, it works like a crystal nucleus. It will continue growing its size by merging with another adjacent segment when we feed them into the algorithm. In the end, there would be only one merged section left, which is then the final result. We set the size of the segment to $S = 40$ notes. The overlap size is set to $O = 0$ for the pitch proximity method, and $O = 30$ notes for the minimal overlapping method.

4. DATASETS

Three major datasets are used to evaluate our method. For voice segregation, we use the Bach Chorales Dataset (BCD) collected from IMSLP.org, which originally contains 364 four-voice chorales composed by Johann Sebastian Bach. To test the robustness of the models, the dataset is augmented with voice dropping, note dropping, and onset/offset shifting. As a result, the dataset includes the original four-voice pieces, 2,184 two-voice pieces 1,456 three-voice pieces, and 364 four-voice pieces with onset/offset shifting, all of which are augmented with three different note dropping rates (see the data augmentation process described as below). For melody extraction, we use two datasets, Mozart Piano Sonatas (MPS) [26] and Americans Folks (AF) [8]. MPS consists of 38 movements from Mozart's piano sonatas, and their melody lines were annotated by a professional pianist. AF is the subset of "Big Dataset," [1] and contains 1,262 folk songs in MIDI format. Every folk song contains a Soprano track, which is picked as the melody line. For AF, we crop the piece so that the melody line consistently exists among note events, and merge all the other note events that do not belong to the melody into a single accompaniment voice.

To further enhance the generalization ability of the model, the following data augmentation techniques are applied for the training data:

1. Tempo scaling: the tempo of each piece is by 2^i times faster (or slower), where i is uniformly sampled from the interval $[-1, 1]$.
2. Key shifting: each piece is transposed by n semitones, for n being uniformly sampled from $\{-6, -5, ...5, 6\}$.
3. Note dropping: a piece has an equal chance for dropping 0%, 5%, or 10% of its notes.
4. Voice dropping: the voices in a Bach 4-part chorales are randomly dropped with rates $p(c)$, $c \in \{0, 1, 2\}$, where $p(c)$ donates the probability of dropping c voices. In our setting, we set $p(0) = 0.6$, $p(1) = 0.3$, and $p(2) = 0.1$. Note that for the melody line identification task, voice dropping is not used because there are only two voices (i.e. melody and accompaniment) in the training data and no more voices can be further dropped.
5. Onset and offset shifting: the onset and offset time of a note event are stretched or shrunk by $(1 + r)$ times of its duration, respectively; r is a sample from the truncated normal distribution [33], whose PDF is set to $f(r; \mu = 0, \sigma = 0.15, a = -0.15, b = 0.15)$; see [33] for detailed implementation.

[1] https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/

Voices	Original (4-voice)			2-voice			3-voice			Onset-offset		
Drop rate	0%	5%	10%	0%	5%	10%	0%	5%	10%	0%	5%	10%
Skyline	95.76	88.63	82.23	98.87	96.32	93.71	97.45	92.25	87.62	26.26	26.82	27.49
VoSA	97.03	95.42	93.77	99.06	98.67	98.29	98.08	97.22	96.27	-	-	-
HMM	**97.79**	96.10	93.50	**99.28**	**98.99**	98.44	**98.59**	**97.75**	96.29	67.43	68.95	67.75
Ours (Min)	97.40	**96.33**	**95.04**	99.03	98.74	98.44	98.28	97.64	97.02	95.56	**94.44**	92.70
Ours (Pitch)	97.43	96.26	94.99	99.03	98.75	**98.46**	98.30	97.66	**97.03**	**96.62**	94.43	**92.84**

Table 1: Frame-level accuracy for voice segregation (in %) on BCD.

Data	Model	P	R	F1
MPS	Skyline	88.49	93.91	91.09
	CNN	93.00	89.69	91.22
	Ours (Min)	91.30	92.04	91.60
	Ours (Pitch)	**95.80**	**95.53**	**95.64**
AF	Skyline	73.33	74.40	73.74
	CNN	69.00	**89.61**	77.27
	Ours (Min)	78.19	82.10	79.09
	Ours (Pitch)	**85.10**	85.04	**84.77**

Table 2: Results (in %) for melody identification

Finally, to facilitate the experiment process, we assume that one note belongs to only one voice. In the voice segregation task, if there exist identical note events in different voices, we assign it to one voice randomly and remove others. As for the the same situation in the melody line identification task, the note is assigned to the melody part.

5. EVALUATION AND DISCUSSION

The proposed neural networks are implemented with Tensorflow 2.0. We adopt Adam optimization [34] and the learning rate is set to 10^{-3}. All experiments are run with one NVIDIA GTX1060 GPU. The training time for every $1M$ pairs of notes is approximately one minute. Source codes are available at our website [2].

Four baseline methods are considered. For voice segregation, we consider the skyline algorithm [26, 28], the VoSA algorithm [13] (which is re-implemented by ourselves), and the HMM-based voice segregation methods [29]. For melody line identification, the skyline algorithm and the state-of-the-art CNN-based melody extractor [26] are considered. These methods are compared with our proposed methods with two segment merging modes, which are denoted by Pitch (the pitch proximity method) and Min (the minimal overlapping method).

Several metrics are reported in this section. For the performance of neural networks, the first evaluation metric we consider is simply the *pairwise accuracy*, the accuracy of the binary prediction (i.e. voice connection of note pairs) of our 1D-CNN model. For voice segregation, we use frame-level accuracy, the ratio of correctly predicted frames to the total frames, to present the results. For melody line identification, the frame-level precision (P), recall (R), and F1-score (F1) are used [26].

[2] `https://github.com/Wiilly07/musical-stream-segregation`

5.1 Results

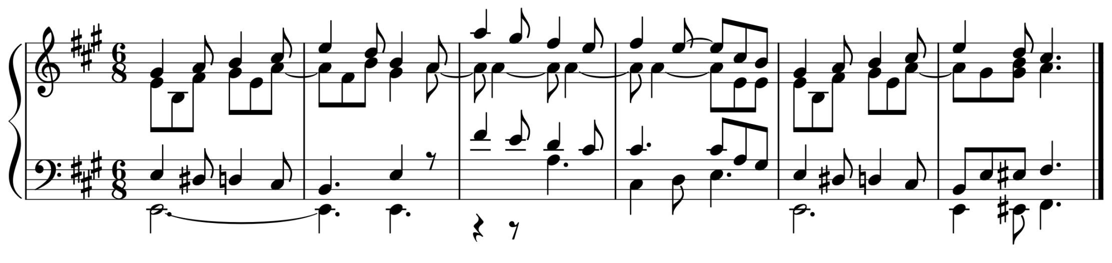
To evaluate the performance of our system, we performed 9-fold cross-validation on BCD under all the conditions mentioned in Section 4. We split each fold on MIDI files to ensure that all the note pairs from the same music piece are in the same fold. All the augmentation methods were also applied to the training data. The training and validation pairwise-accuracy of the 1D-CNN are 95.41% and 96.81%, respectively. Among all the validation data (including augmented data), our system can correctly predict the cluster number k over 99.8% (12865/12888) by eigengap. The result of voice segregation is shown in Table 1. First, for zero note drop and zero onset/offset shifting, HMM remains as the most superior method. However, the proposed methods prevails as the note dropping rate increases; for example, Ours (Min) outperforms HMM by 1.54 percentage points in 4-voice and 10% note dropping rate. In addition, in the case of onset-offset shifting, the proposed systems outperform all the others by at least 25 percentage points. These findings highlight the advantage of the proposed model on high tolerance to noisy data. Furthermore, the differences of performance between Ours (Min) and Ours (Pitch) are very small (within 0.1% for most of the cases), suggesting that the proposed method can work without imposing perceptual rules.

Similarly, in the task of melody line identification, we also performed 9-fold cross-validation on MPS and AF dataset, respectively. Only the first three data augmentation methods in Section 4 were applied in training. The training and validation pairwise accuracy values are 98.28% and 93.96% for MPS, and 91.87% and 88.43% for AF, respectively. From Table 2, we observe that our method with minimal overlapping is on par with the CNN baseline [26] on both MPS and AF. [3] When pitch proximity is applied, the proposed method outperforms others by at least 4 percentage points in F1-score.

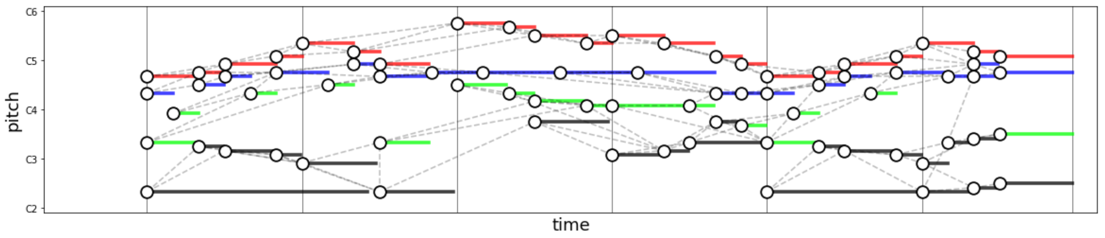
5.2 Discussion

To investigate the behaviors of our proposed framework, we conduct voice segregation and melody line identification for *unseen* types of data using the models we trained. Figure 2 takes the first six bars of the first movement of Beethoven's Sonata No. 28 as an example. Figure 2b and 2c show the graphs constructed with the note-to-note affinity inferred by the BCD and AF models, respectively. Both

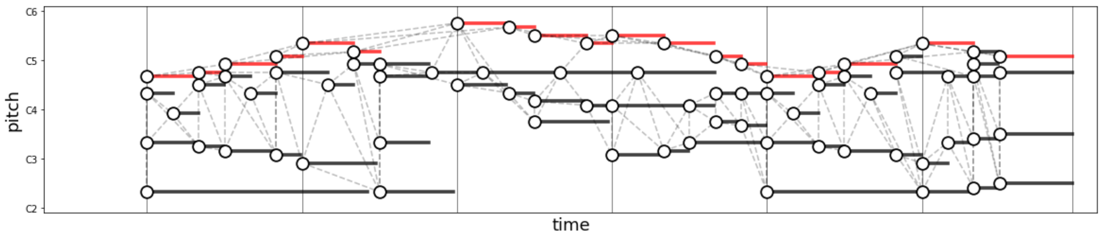
[3] The CNN-based model for evaluating MPS is directly provided from [26]. We retrained a model using the source code to evaluate AF.



(a) The sheet music



(b) The result of voice segregation by the model trained in BCD



(c) The result of melody extraction predicted by the model trained in AF

Figure 2: Results on the first 6 bars of Beethoven Sonata No. 28 Mov. 1. Dashed lines in the piano rolls represent the edges with affinities $\hat{w}_{ij} > 0.5$ of the learned graph. The links between distant notes are omitted for better visualization.

graphs exhibit the "interpretation" from the two models. For the BCD model, notes tend to be linked in the horizontal direction and long-distance links are enforced to construct the voices. By contrast, the AF model prefers to establish vertical links in lower pitches (e.g., accompaniment), while enforcing horizontal links in higher pitches (e.g., melody). It is intriguing to see in Figure 2c that although the note A5 in the third bar is linked with several accompaniment notes, A5 is still classified as melody because of the even more abundant links to melody notes.

According to the directions of note stems, four voices can be identified from Figure 2a. The eigengap of the graph generated by the BCD model does give an estimation of $k = 4$, which is consistent with such a common interpretation. By comparing Figures 2a and 2b, most interpretation of the BCD model are consistent with the score sheet except some interesting exceptions. For example, in the first bar, the BCD model assigns B3 and E4$'$ of the second voice (denoted as E4–B3–F#4–G#4–E4$'$–A4) to the third voice, while D#3, D♮3, and C#3 in the third voice (E3–D#3–D♮3–C#3) are assigned to the fourth voice. That means, the model treats the second voice as *pseudo-polyphony* and manages to derive two valid voices from it. An explanation of this phenomenon is that the BCD model learns the principles of counterpoint and tends to have more large leaps for lower voices. It can be found that the notes in the fourth voice from the BCD model are highly overlapped, although perceiving them as a voice is indeed possible for human, if the duration information is ignored. The characteristics of the training data may also provide another explanation of this phenomenon; the density of notes in the fourth voice of this excerpt is sparser than the density of bass notes in BCD. Therefore, the model tends to build more links between the lowest note E2 and its contextual notes, and merges all of them into the same stream.

6. CONCLUSION

In this paper, we have presented a new, generalizable, and straightforward framework to learn note-to-note affinity for symbolic music data. The framework, taking only note attributes as training features, can outperform several state-of-the-art horizontal element extraction methods in various musical textures and in real-world scenarios. The graph representation induced from the framework can also serve as a tool for in-depth music analysis on the relationship among the notes if the label configuration describing the training data of interest is given. The major limitation of this work is that it has not considered the case of varying number of voices (e.g., voices shrinking or expansion) in the music data, which can however be well solved with a modified segment merging process and more accurate prediction on the eigengaps within segments. Adopting more advanced neural networks and statistical methods to achieve this goal is left as our future work.

7. REFERENCES

[1] Y.-T. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 2796–2809, 2020.

[2] 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 *Proc. of the AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018, pp. 34–41.

[3] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, "Pypianoroll: Open source python package for handling multitrack pianoroll," *Late-breaking/Demo paper in International Society Music Information Retrieval Conference (ISMIR)*, 2018.

[4] D. Basaran, S. Essid, and G. Peeters, "Main melody extraction with source-filter NMF and CRNN," in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 82–89.

[5] Z. Duan, J. Han, and B. Pardo, "Multi-pitch Streaming of Harmonic Sound Mixtures," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 22, no. 1, pp. 138–150, 2014.

[6] C.-Y. Kuan, L. Su, Y.-H. Chin, and J.-C. Wang, "Multipitch streaming of interwoven streams," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 311–315.

[7] A. Jordanous, "Voice separation in polyphonic music: A data-driven approach," in *Proc. of the 2008 International Computer Music Conference (ICMC)*, Belfast, Ireland, 2008.

[8] W.-T. Lu and L. Su, "Deep learning models for melody perception: An investigation on symbolic music data," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Honolulu, USA: IEEE, 2018, pp. 1620–1625.

[9] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 2008.

[10] D. Deutsch, "Grouping mechanisms in music," in *The Psychology of Music*. Elsevier, 2013, pp. 183–248.

[11] D. Huron, "Tone and voice: A derivation of the rules of voice-leading from perceptual principles," *Music Perception*, vol. 19, no. 1, pp. 1–64, 2001.

[12] J. Kilian and H. H. Hoos, "Voice separation-a local optimization approach." in *Proc. 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.

[13] E. Chew and X. Wu, "Separating voices in polyphonic music: A contig mapping approach," in *International Symposium on Computer Music Modeling and Retrieval (CMMR)*, U. K. Wiil, Ed. Esbjerg, Denmark: pringer Berlin Heidelberg, 2004, pp. 1–20.

[14] A. Ishigaki, M. Matsubara, and H. Saito, "Prioritized contig combining to segregate voices in polyphonic music," in *Proc. Sound and Music Computing Conference (SMC)*, vol. 119, Padova, Italy, 2011, p. 58.

[15] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, "Comparing voice and stream segmentation algorithms," in *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 493–499.

[16] ——, "Improving voice separation by better connecting contigs," in *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 164–170.

[17] B. Duane and B. Pardo, "Streaming from MIDI using constraint satisfaction optimization and sequence alignment," in *Proc. of the 2009 International Computer Music Conference (ICMC)*, Montreal, Canada, 2009.

[18] S. T. Madsen and G. Widmer, "A complexity-based approach to melody track identification in midi files," in *Proc. of the International Workshop on Artificial Intelligence and Music*, Hyderabad, India, 2007.

[19] I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos, "Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data," in *Proc. of Sound and Music Computing Conference (SMC)*, Lefkada, Greece, 2007, pp. 299–306.

[20] D. Makris, I. Karydis, and E. Cambouropoulos, "Visa3: Refining the voice integration/segregation algorithm," in *Proc. Sound and Music Computing Conference (SMC)*, Hamburg, Germany, 2016, pp. 266–273.

[21] P. B. Kirlin and P. E. Utgoff, "Voise: Learning to segregate voices in explicit and implicit polyphony." in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005, pp. 552–557.

[22] D. Rizo, P. J. P. De León, C. Pérez-Sancho, A. Pertusa, and J. M. I. Quereda, "A pattern recognition approach for melody track selection in midi files." in *Proc. of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, Victoria, Canada, 2006, pp. 61–66.

[23] R. de Valk, T. Weyde, E. Benetos *et al.*, "A machine learning approach to voice separation in lute tablature," in *Proc. of the 14th International Society for Music

Information Retrieval Conference (ISMIR), Curitiba, Brazil, 2013, pp. 555–560.

[24] T. Weyde and R. de Valk, "Chord- and note-based approaches to voice separation," in *Computational Music Analysis*. Springer, 2016, pp. 137–154.

[25] 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.

[26] F. Simonetta, C. E. C. Chacón, S. Ntalampiras, and G. Widmer, "A convolutional approach to melody line sdentification in symbolic scores," in *Proc. of the 20th Int. Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 924–931.

[27] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 281–288.

[28] P. Gray and R. C. Bunescu, "A neural greedy model for voice separation in symbolic music." in *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 782–788.

[29] A. McLeod and M. Steedman, "Hmm-based voice separation of midi performance," *Journal of New Music Research (JNMR)*, vol. 45, no. 1, pp. 17–26, 2016.

[30] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 22, no. 8, pp. 888–905, 2000.

[31] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," Vancouver, Canada, 2001, pp. 849–856.

[32] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[33] J. Burkardt, "The truncated normal distribution," *Department of Scientific Computing Website, Florida State University*, pp. 1–35, 2014.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.