Note-level Nonchord-tone Identification with Graph Neural Networks

Yui Uehara* and Satoshi Tojo†
* Kanagawa University, Japan
E-mail: yuiuehara@kanagawa-u.ac.jp Tel: +81-(0)45-481-5661
† Asia University, Japan
E-mail: tojo_satoshi@asia-u.ac.jp Tel: +81-422-36-3241

***Abstract*—This paper presents a method for identifying nonchord tones at the note level using graph neural networks. In this approach, musical scores are represented as graphs, where each node corresponds to a musical note. While traditional methods often segment or tokenize musical scores, representing them as graphs allows us to grasp the relationships between notes. This representation is particularly effective for tasks that require note-level recognition. We conduct experiments on three collections by J.S. Bach: *Harmonized Chorales*, *Little Organ Book*, and *The Well-Tempered Clavier*. Although written by the same composer, each collection varied in musical complexity. The proposed method achieved an accurate recognition of approximately 87% F1-score on *Harmonized Chorales*. Although the smallest collection of *The Well-Tempered Clavier* resulted in the lowest F1-score among the three, *Little Organ Book* attained an F1-score exceeding 80% despite containing complex non-harmonic decoration.**

I. Introduction

Non-chord tones (NCTs) are, as the name suggests, tones that are not included in chords. They are complementary in definition, but their musical roles are different. Chord tones (CTs) form chords and determine the structure of a piece, such as chord progressions and keys, while NCTs decorate the surroundings of CTs and give the piece individuality and expressive richness[1].

Although both are essential to music, research on NCTs is less extensive than on harmony and tonality. A key challenge here is how musical scores are represented. In general music information processing, one common method of this is representing a musical score as a time series of segments; previous research on NCT identification has also adopted this method[2]. However, in this approach, a single note could be split across multiple segments because segments are generated at the point where a new note onset occurs, and other notes may still be sustained. Considering that NCT is a note-level concept, this treatment seems unnatural and prevents recognition at the note level. Another method converts musical notes into text as a series of events for a Transformer model[3]. However, this representation strays from natural musical notation, resulting in a large model size.

In this paper, we represent musical scores as graphs and build note-level NCT identification models using graph neural networks (GNNs). This approach is inspired by researches that use graph representation and GNNs in harmonic analysis[4] or voice separation[5]. Considering each musical note as a node, it makes sense to represent a musical score as a graph. This transformation can be done naturally, without resorting to overly artificial rules. Because NCT identification is a task that deals with each note individually, graph-based methods are considered suitable for this purpose.

For the GNN, we conducted experiments using two models: GraphSAGE[6] and Graph Attention Network v2 (GATv2)[7]. We also tested multiple configurations for the number of layers and the number of heads in GATv2.

We apply the proposed method to J.S. Bach's three different collections of pieces: *Harmonized Chorales (Chorales)*, *Little Organ Book (Organ)*, and *The Well-Tempered Clavier (WTC)*. *Chorales* and the pieces in *Organ* are based on the same chorale (hymn) melodies; however, as organ instrumental pieces, the latter feature complex counter-melodies that are rich in NCTs. *WTC* is characterized by instrumental expressions such as arpeggio.

Our proposed method achieves a maximum F1-score of approximately 87% and an accuracy of over 95% on *Chorales*. This high recognition accuracy demonstrates that graph representation of musical scores and feature extraction using GNNs effectively recognize non-chord tones. Regarding the *WTC*, we achieve an F1-score of 66% despite there being only 24 pieces in the entire corpus. Furthermore, F1-score of over 80% is obtained for the *Organ*. The experimental results demonstrate that our model can effectively work on more complex pieces than the simple *Chorales*, even with only 44 pieces.

II. Related Studies

In simpler pieces, NCTs can be identified by rule-based models designed with musical knowledge[8], [9]. However, these rules cannot be adapted to diverse compositions, as the contrapuntal elaboration of music varies widely[1], [9], [10].

Therefore, machine learning can be a promising method for handling diverse compositions. Hu and Arthur developed a logistic regression model for NCT identification[11]. Ju et al.[2] introduced deep learning models, and tested them on J. S. Bach's *Chorales*; however, they did not provide note-level predictions due to their reliance on time segments. McLeod and Rohrmeier[12] developed a note-level method and tested it with more complex compositions than *Chorales*, aiming to improve chord quality recognition. However, their model was

979-8-3315-7206-8/25/$31.00

designed for post-processing and included chord information in the input, unlike our model.

This study employs GNNs for this task. A musical score consists of multiple series of pitches (i.e., melodies) rather than a single time series. The graph format is versatile and can represent various types of data. Several researchers have noted that musical scores can be effectively represented as graphs by treating each note as a node[4], [5], [13]. In addition, GNNs have been reported to be effective in tasks such as audio generation[13], musical part (voice) detection[5], and harmonic analysis[4].

III. Methodology

A. Problem formulation of NCT identification

We simply assume that NCTs are notes that are not part of a chord and focus on the binary classification of chord-tone or not. The model estimates the probability y that a specific note is an NCT, based on the feature vector $\hat{\boldsymbol{x}}$ of the original note.

$$y = f\left(\hat{\boldsymbol{x}}; \mathcal{C}\right) \tag{1}$$

f represents a function; it is used to predict the probability y of the note being an NCT from the input feature vector $\hat{\boldsymbol{x}}$. This function is designed as a neural network. It utilizes additional context, denoted as $\mathcal{C}$, which consists of information about neighboring notes, along with the input $\hat{\boldsymbol{x}}$, to predict y.

We use a graph neural network (GNN) to effectively incorporate contextual information by aggregating features from neighboring notes with the focused note's features. While a fully connected graph conceptually resembles a Transformer model[14] in that it gathers and weights information from all instances, NCTs are more influenced by local conditions like preceding and following chords. Therefore, we chose a GNN focusing on note connections rather than applying attention to all notes or using artificial positional encoding. Although the model is not given chord information, we hope the GNN will learn to identify consonant sounds from the neighboring notes.

B. Graph representation of a musical score

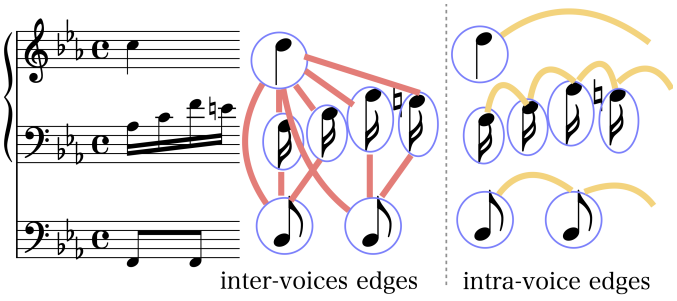

Fig. 1. Graph representation of a musical score.

We transform a musical score into a graph, as shown in Fig. 1. Each node represents a single note or *rest*. The input node feature consists of the note's pitch class, MIDI note number, and type of beat, which are each one-hot vectorized independently and then combined. The pitch class ranges from 0 to 11, and the MIDI note number ranges from 0 to 127. The beats are classified into three types: the first beat of each measure, strong beats except the first beat, and upbeats. Therefore, the node feature vector has 143 dimensions. The node feature of a *rest* is a concatenation of a zero vector for pitch class and MIDI, and a beat feature vector.

Three types of edges are attached. The musical scores used in this paper are in MusicXML format, providing information about the voices.

- **Inter-voice edges** are edges between notes that belong to different voices, but that overlap in time.
- **Intra-voice edges** are edges between adjacent notes or rests in the same voice.
- A **self-loop** is an edge that connects to the node itself. Self-loops ensure that a node's own feature is also considered during the feature update process in GNN. (For ease of viewing, the self-loop is omitted in Fig. 1.)

Although the node feature does not explicitly have the information on note length, the graph's topology gives it since a note with a longer duration tends to overlap with a greater numbers of other notes, and thus receives more information from them as neighboring notes.

The edge feature for inter-voice edges is determined by the time overlap between nodes. For example, in Fig. 1, the edge features between the quarter note and the sixteenth notes in the top two voices are 0.25 (assuming the quarter note is 1.0). In contrast, intra-voice edges use the inter-onset interval as their feature. For instance, the edge feature between the two eighth notes in the bottom part of Fig. 1 is 0.5. Although we have not shown directions on the edges, they are intrinsically bidirectional. Most edges have the same feature value in both directions; however, intra-voice edges are an exception and have signed features based on their direction. A self-loop edge possesses the feature, corresponding to its duration. Lastly, a three-dimensional one-hot vector for edge types is added to the edge features.

The conversion of MusicXML sheet music into a graph is carried out automatically according to the design described above. The complete source code of this work, which includes the MusicXML-to-graph converter and the NCT identification models, is available[1].

C. NCT identification using graph neural networks

The input to the proposed NCT identification model is the graph representation of the musical score: the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the feature vectors of all notes and edges. We employ two types of *message-passing*-style graph neural networks for feature encoding: GraphSAGE[6] and Graph Attention Network v2 (GATv2)[7]. In *message-passing*, the feature of node u is updated using the feature values of neighboring nodes connected to it (context $\mathcal{C}$). As shown in Fig. 2 the feature of node u (filled note) is updated using only the features of the nodes directly connected to u rather than the features from the entire graph.

[1]https://github.com/yui-u/gnn-ncti

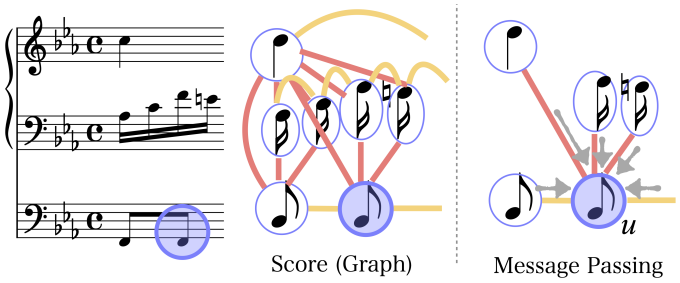

Fig. 2. Message Passing.

This message-passing is performed multiple times. The k-th update refers to updating the node features by performing message passing from the features obtained up to the $k-1$-st update with the parameters of the k-th layer in the GNN network. While each update can only draw information from directly connected nodes, the information from neighboring nodes is incorporated with each layer's output. As a result, information from more distant nodes (notes) can be accessed through multiple layers.

1) GraphSAGE: In GraphSAGE[6], node features are updated as (2).

$$\boldsymbol{x}_u^{'(k)} = \mathrm{ReLU}\left(W_1^{(k)}\boldsymbol{x}_u^{(k-1)} + W_2^{(k)} \cdot \mathrm{MEAN}_{v\in\mathcal{N}_u}\boldsymbol{x}_v^{(k-1)}\right) \tag{2}$$

This is a simple method that adds the linear transformation of the focused node's feature value to the linear transformation of the mean of the neighboring nodes' feature values. $W_{(\cdot)}^{(k)}$ is the linear weight parameter to be learned. The subscript u indicates a focused node, and v is a neighboring node connected with u, which is an element of the set of neighboring nodes $\mathcal{N}_u$. The bias term is omitted to avoid complications. The formula of (2) differs slightly from the original paper[6] but is essentially equivalent, reducing the number of parameters. In addition, since each node's degree in the graph is small, no sampling is conducted.

Layer Normalization[15] is applied after (2) to produce the k-th updated node feature of $\boldsymbol{x}_u^{(k)}$.

$$\boldsymbol{x}_u^{(k)} = \frac{\boldsymbol{x}_u^{'(k)} - \mu^{(k)}}{\sigma^{(k)}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{3}$$

$\mu^{(k)}$ is the mean of all $\boldsymbol{x}_u^{'(k)}$ and $\sigma^{(k)}$ is the variance of them. $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learnable affine parameters.

2) Graph Attention Network v2: In GATv2[7], node features are updated as (4–5), where LR indicates Leaky ReLU activation function.

$$\boldsymbol{x}_u^{'(k)} = \mathrm{ReLU}\left(\sum_{v\in\mathcal{N}_u} \alpha_{u,v} W_t^{(k)} \boldsymbol{x}_v^{(k-1)}\right) \tag{4}$$

$$\alpha_{u,v} = \frac{\exp\left(\boldsymbol{a}^\top \mathrm{LR}(W_1^{(k)}\boldsymbol{x}_u^{(k-1)} + W_2^{(k)}\boldsymbol{x}_v^{(k-1)} + W_3^{(k)}\boldsymbol{e}_{(u,v)})\right)}{\sum_{v'\in\mathcal{N}_u}\exp\left(\boldsymbol{a}^\top \mathrm{LR}(W_1^{(k)}\boldsymbol{x}_u^{(k-1)} + W_2^{(k)}\boldsymbol{x}_{v'}^{(k-1)} + W_3^{(k)}\boldsymbol{e}_{(u,v')})\right)} \tag{5}$$

Unlike GraphSAGE, GATv2 incorporates edge features to calculate the weights of neighbors, as shown in (5). In addition, GATv2 can have multiple heads, and (4) is the calculation for a single head. After the calculation of (4) for each head, the node features of all heads are concatenated and applied the Layer Normalization to become the new node feature of layer k, $\boldsymbol{x}_u^{(k)}$.

3) Prediction of non-chord tones: To predict NCTs, we first transform the node features of the input graph as described in equation (6).

$$\boldsymbol{x}_u^{(0)} = \mathrm{ReLU}\left(W_0\hat{\boldsymbol{x}}_u + \boldsymbol{b}_0\right) \tag{6}$$

Then, we perform message passing K times using GraphSAGE or GATv2 to obtain the final feature vector for each node, denoted as $\boldsymbol{x}_u^{(K)}$. The obtained final node feature is converted into a 2D vector by a multilayer perceptron (MLP) with one hidden layer. Since this is a binary classification problem, we could convert it into a 1D scalar and use the sigmoid function. However, we converted it into a 2D vector and applied the softmax function instead. This approach optimizes the threshold for binary classification by using the relative values of two parameters instead of focusing on a single value, potentially leading to improved performance compared to the 1D method. Then, the softmax function is applied to obtain the prediction probability $\boldsymbol{y}_u$.

$$\boldsymbol{y}_u = \mathrm{softmax}\left(\mathrm{MLP}_2(\boldsymbol{x}_u^{(K)})\right) \tag{7}$$

4) Training: The training data (labels of CT/NCT distinction) is automatically generated from the annotated data of harmonic analysis, and used in the supervised learning. The semi-automatic annotation process is detailed later in IV-B. The loss function is the average cross-entropy loss for the note nodes. Although *rest* nodes are used as part of the graph when calculating the node features, they are not included in the loss calculation.

IV. Experiments

A. Dataset

1) Harmonized Chorales (Chorales): This collection consists of simple harmonizations based on the hymn melodies (chorales), and many of them are for four voices. Although the collection contains 371 pieces, duplicates of nearly identical ones are removed. Additionally, pieces whose harmonic analysis annotations used for NCT labeling do not match the key signatures or measure(bar) numbers in the scores are also excluded, resulting in 306 pieces.

In the previous study that conducted NCT identification on *Chorales*, they reported that 140 pieces were used[2]. Since the same dataset is unavailable, we obtain annotations for the *Chorales* from When-in-Rome[16], a meta-corpus for harmonic analysis. However, we observed discrepancies of measure numbers between the harmonic analysis annotations and sheet music for some pieces in the meta-corpus, likely due to differences in how repeats were handled. As a result, we sourced the target sheet music from Music21[17].

2) Little Organ Book (Organ): This is a collection of 46 short pieces for organ solo. Although the pieces are based on the same chorale melodies found in *Chorales*, they are more complex and rich in NCTs (e.g., Fig. 3). In addition, each piece is more diverse than *Chorales*.

The scores in MusicXML format, along with manually annotated harmonic analyses, are made publicly available[2]. The annotation of harmonic analysis are made by organ performance experts. Unlike the other two collections, harmonic analysis does not include annotations of inversions, but only annotations of key and Roman numerals (degrees). Except for one minor revision and one piece lacking harmonic analysis of the opening section, a total of 44 pieces are used for experiments.

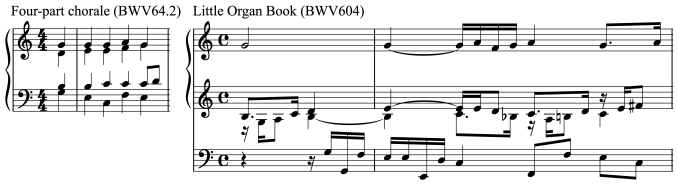

Fig. 3. Examples of the beginning of the *Chorale* (BWV64.2) and the piece in the *Organ Book* (BWV604). Both are based on the same chorale melody.

3) The Well-Tempered Clavier (WTC): The *Well-Tempered Clavier (WTC)* is a collection of 24 keyboard pieces, each originally consisting of a prelude and a fugue. However, since the harmonic analysis for the fugue is not available, we only use the 24 preludes.

Similar to *Organ*, each piece is rich in diversity (e.g., Fig. 4). Some contain many NCTs, while others feature arpeggios of chord tones as their motif. Some pieces have as few as two voices (e.g., Fig. 4), and it is expected to pose a challenge in NCT identification, since all the notes of the harmony are not present at the same time.

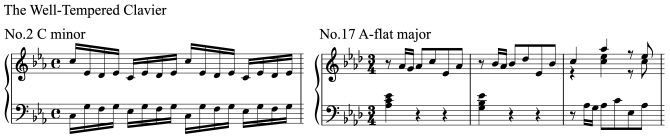

Fig. 4. Examples of pieces in *WTC*.

B. Semi-automatic generation of training labels

This study focuses on binary classification of whether a note is a chord tone (CT) or a non-chord tone (NCT). Since chord tones are identified through harmonic analysis, we automatically generate training labels for CT/NCT using manually annotated harmonic analysis data. While the harmonic analysis label may change during the note's duration, the harmonic analysis label used for the automatic training label generation is determined based on the analysis at the note's onset.

As mentioned in IV-A2, the annotation of the *Organ* only includes information on root degrees, and the constituent notes of seventh chords or more are unknown. Therefore, when the root degree is V, the seventh note of the dominant seventh chord is added to the three constituent notes. Hereafter, we denote this treatment as *triad+dominant*, while the CT/NCT labeling of training data based on full Roman numerals (if available) is denoted as *full*. As described above, the difference between *triad+dominant* and *full* only appears in seventh chords where the root is not V. For example, in Fig. 5, only the notes marked with circles (the 7th of the viio7) differ: the note is labeled as NCT in *triad+dominant*, and CT in *full*.

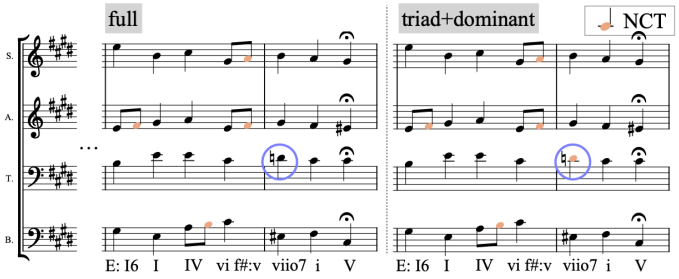

Fig. 5. Automatic generation of NCT labels for training.

The statistics for the automatically generated labels for CT and NCTs are shown in Table I. As can be seen from the table, *Organ* and *WTC* have a higher proportion of NCTs than *Chorales*, suggesting that they feature more complex melodic decoration, typical of instrumental pieces.

TABLE I
CT AND NCT STATISTICS.

dataset	chord type		CT	NCT	Total
Chorales	full	# notes	63473	8558	72031
		prop.(%)	88.12	11.88	-
Chorales	triad+dom.	# notes	62362	9669	72031
		prop.(%)	86.58	13.42	-
Organ	triad+dom.	# notes	16684	6261	22945
		prop.(%)	72.71	27.29	-
WTC	full	# notes	13785	3585	17370
		prop.(%)	79.36	20.64	-
WTC	triad+dom.	# notes	12857	4513	17370
		prop.(%)	74.02	25.98	-

C. Experimental settings

We conduct 5-fold cross-validation, with three folds for training, one for validation, and one for testing. Each of the 5 folds are experimented with 3 random seeds, resulting in averages from 15 data combinations. The mini-batch size is 4, and the Adam optimizer is used with a learning rate of 1e-3. Gradient clipping is set to a maximum of 1.0, and node feature dimensions in the GNN are 32. During training, a dropout rate of 25% is applied to each layer's output. The training lasts 1024 epochs, and the model with the highest F1-score on the validation data is selected. Since changing the pitch of the entire score does not alter the CT and NCT types for each note, the input to the model was normalized so that it does not have a key signature.

[2]https://github.com/yui-u/little-organ-analyses

V. Results

A. Experiments for different GNN settings on Organ dataset

We conduct experiments for different types and sizes of GNNs on *Organ* dataset, which has the highest proportion of NCTs (Table I). The number of layers in the GNN was tested from 1 to 5.

TABLE II
RESULTS WITH DIFFERENT MODEL SETTINGS ON *Organ*.

model	head	layer	acc.	prec.	rec.	f1
GraphSAGE	-	1	85.87	77.45	68.16	72.48
GraphSAGE	-	2	88.15	79.02	77.22	78.08
GraphSAGE	-	3	88.30	79.91	76.45	78.11
GraphSAGE	-	4	88.30	80.33	75.83	77.97
GraphSAGE	-	5	88.14	79.20	76.99	78.01
GATv2	1	1	83.09	70.55	65.67	67.90
GATv2	1	2	84.17	72.30	70.13	70.91
GATv2	1	3	82.89	67.85	71.17	69.45
GATv2	1	4	83.45	71.54	66.80	68.93
GATv2	1	5	79.66	62.34	65.56	63.54
GATv2	2	1	86.68	77.55	72.42	74.79
GATv2	2	2	89.14	81.26	78.56	79.87
GATv2	2	3	89.41	82.27	78.37	80.23
GATv2	2	4	89.36	81.92	78.52	80.16
GATv2	2	5	88.92	81.39	77.38	79.31
GATv2	3	1	87.10	77.38	74.85	76.06
GATv2	3	2	89.08	80.81	78.99	79.87
GATv2	3	3	89.63	82.01	79.75	80.83
GATv2	3	4	**89.83**	**82.30**	**80.19**	**81.21**
GATv2	3	5	89.24	81.01	79.62	80.27

As shown in Table II, scores for models with single layer were slightly lower, and models with 2 or more layers were stable. In GATv2, the single-head model led to lower performance compared to models with 2 or 3 heads.

The model can achieve an accuracy of around 73% if all notes are identified as chord tones, since 72.71% of gold labels are chord tones, as shown in Table I. In the experimental results shown in Table II, models with 2 or more heads achieved an accuracy of over 85%, confirming the effectiveness of the training. The best accuracy, precision, recall, and F1-score were achieved by GATv2 with 4 layers and 3 heads.

Regarding the performance differences between GNN types, GraphSAGE and GATv2 with two heads tended to have similar F1-scores. The number of parameters in GraphSAGE is less than half that of GATv2 with two heads, and GraphSAGE achieved equivalent performance even though it does not use edge features. GATv2 with a single head has nearly the same number of parameters as GraphSAGE, but its performance is inferior to that of GraphSAGE. Although the tasks are different, models that use text-based music token sequences and Transformers for music generation have been reported to have a model size of approximately 41M[3], which is significantly larger than the GNN used in this study, the largest model size of which is only about 0.1M (GATv2, head=3, layer=5).

B. Ablation studies

The results in Table III indicated that the edge features had a positive effect. When the edge features were not used, GraphSAGE (that did not use them by default) performed better than GATv2. The ablation study also confirmed that pitch class and beat information were useful. In contrast, there was little change in performance even when the MIDI note number information was removed; this result suggested that pitch range information did not contribute much to the recognition of NCTs. The score has actually increased slightly by removing MIDI information. The number of node feature dimensions was reduced by 128 by removing it, which simplified the node features and possibly made learning easier.

TABLE III
ABLATION STUDY ON *Organ* DATASET.

model	head	layer	acc.	prec.	rec.	f1
GraphSAGE	-	3	88.30	79.91	76.45	78.11
– pitch class	-	3	88.10	80.01	75.35	77.58
– MIDI	-	3	88.97	81.80	76.56	79.07
– beat	-	3	87.50	79.11	73.98	76.43
GATv2	3	4	89.83	82.30	80.19	81.21
– edge feature	3	4	87.50	77.99	75.79	76.84
– pitch class	3	4	88.39	79.70	77.51	78.55
– MIDI	3	4	90.25	83.46	80.23	81.79
– beat	3	4	88.80	80.08	78.82	79.42

C. Results on different J. S. Bach's collections

TABLE IV
RESULTS ON DIFFERENT DATASETS WITH GRAPHSAGE(LAYER=3) AND GATV2 (HEAD=3, LAYER=4)

dataset	chord type	model	acc.	prec.	rec.	f1
Chorales	full	GraphSAGE	96.50	87.86	82.03	84.80
Chorales	full	GATv2	96.96	88.23	85.92	87.04
Chorales	triad+dom.	GraphSAGE	96.03	88.16	81.52	84.66
Chorales	triad+dom.	GATv2	97.00	89.82	87.67	88.70
Organ	triad+dom.	GraphSAGE	88.30	79.91	76.45	78.11
Organ	triad+dom.	GATv2	89.83	82.30	80.19	81.21
WTC	full	GraphSAGE	87.17	71.63	58.88	64.47
WTC	full	GATv2	87.36	71.11	62.04	66.13
WTC	triad+dom.	GraphSAGE	84.61	73.88	60.67	66.48
WTC	triad+dom.	GATv2	85.61	74.25	66.60	70.08

The proposed model achieved F1-score of 87% with GATv2 (head=3, layer=4) on *Chorales*. In a previous study using the same collection, the F1-score was found to be 72%[2]. However, that study used only about half (140 pieces) of the collection, and the content is currently unavailable. Additionally, it did not perform note-level recognition like our work; instead, they used a segment-based approach, which makes direct comparisons impossible. Despite these differences, our accurate results demonstrate the effectiveness of the proposed model.

The F1-score for *WTC(full)* with GATv2 was approximately 66%, indicating that there is room for improvement. Although the limited amount of data may have impacted this score, pretraining with *Chorales* did not lead to much improvement. On the other hand, the proposed model achieved an F1-score of over 80% on *Organ*, which retained maximum proportion of NCTs. It is also worth noting that *Organ* achieved reasonable performance with only 44 pieces. Although issues remained in *WTC*, the proposed method has proven effective for complex instrumental pieces that contain many NCTs.

VI. Conclusions

In this paper, we verified the adequacy and the efficiency to employ GNNs to identify NCTs. By representing musical notes as nodes in a graph and expressing the vertical and horizontal connections between notes as edges, we have realized a note-level NCT identification without relying on the artificial conversion of musical score information.

We tested two types of GNN (GraphSAGE and GATv2) and multiple numbers of heads and layers settings. As a result, we achieved a maximum F1-score of 87% on *Harmonized Chorales (Chorales)*, 81% on *Little Organ Book (Organ)*, and 66% on *The Well-Tempered Clavier (WTC)*. We achieved high performance on *Organ*, which was much more complex than the *Chorales* and rich in melodic decorations with NCTs.

The ablation study results indicated that edge features, pitch class, and beat information were effective, while pitch range information contributed minimally to the performance. In addition, although GATv2 achieved the highest score, GraphSAGE proved to be superior to GATv2 under conditions of small model parameter sizes.

A limitation of this work was that it automatically generated training data for NCTs based on the assumption of complementary relationship between CTs and NCTs, using annotation data from harmonic analysis. The assumption is natural and has been adopted in the previous study[2]. However, in harmonic analysis, temporary fluctuations in chords are sometimes not annotated. This lack of annotation may not provide enough information to determine NCTs. Therefore, an immediate challenge for us is to validate the quality of automatically generated labels against NCTs analyzed by human experts. Furthermore, considering the complementarity of CT and NCT, the simultaneous recognition of NCT and harmonic analysis is also an important area for future work.

More detailed analysis of non-chord tones (NCTs), such as neighboring tones and passing tones, is another important future work. To analyze these detailed labels, it is essential to understand the relationships between preceding and following notes. The achievement of NCT identification at the note level in this study represents an important step toward more detailed and musical automatic classification of non-chord tones.

Acknowledgment

This work was in part supported by grants from JSPS KAKENHI Grant Numbers 23K20011, 25H01169, and JST ACT-X Grant Number JPMJAX24C6. We used ABCI 3.0 provided by AIST and AIST Solutions with support from "ABCI 3.0 Development Acceleration Use".

References

[1] W. Piston, *Counterpoint*. W. W. Norton & Company, Inc., 1947.

[2] Y. Ju, N. Condit-Schultz, C. Arthur, and I. Fujinaga, "Non-chord tone identification using deep neural networks," in *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, Association for Computing Machinery, 2017, pp. 13–16.

[3] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," ser. MM '20, Association for Computing Machinery, 2020.

[4] E. Karystinaios and G. Widmer, "Roman numeral analysis with graph neural networks: Onset-wise predictions from note-wise features," in *Proceedings of the 24th conference of the International Society for Music Information Retrieval Conference*, 2023, pp. 597–604.

[5] E. Karystinaios, F. Foscarin, and G. Widmer, "Musical voice separation as link prediction: Modeling a musical perception task as a multi-trajectory tracking problem," in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 2023, pp. 3866–3874.

[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" In *International Conference on Learning Representations*, 2022.

[8] M. Rohrmeier and I. Cross, "Statistical properties of tonal harmony in bach's chorales.," in *10th International Conference on Music Perception and Cognition*, 2008, pp. 619–627.

[9] C. Arthur, "A corpus approach to the classification of nonchord tones across genres," in *14th International Conference on Music Perception and Cognition*, 2016.

[10] D. Temperley, "The melodic-harmonic 'divorce' in rock," *Popular Music*, vol. 26, pp. 323–342, 2007.

[11] T. Hu and C. Arthur, "A statistical model for melody reduction," in *Proceedings of the Future Directions of Music Cognition International Conference*, 2021.

[12] A. McLeod and M. Rohrmeier, "Detecting chord tone alterations and suspensions," *Journal of New Music Research*, pp. 1–11, 2024.

[13] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 3060–3070.

[14] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450.

[16] M. Gotham, G. Micchi, N. N. López, and M. Sailor, "When in Rome: A Meta-corpus of Functional Harmony," *Transactions of the International Society for Music Information Retrieval*, vol. 6, no. 1, pp. 150–166, Nov. 2023.

[17] 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*, 2010, pp. 637–642.