AutoSchA: Automatic Hierarchical Music Representations via Multi-Relational Node Isolation

Stephen Ni-Hahn*[1], Rico Zhu*[1], Jerry Yin[2], Yue Jiang[1], Cynthia Rudin[1], Simon Mak[1]

[1]Duke University
[2]Stanford University
{stephen.hahn, rico.zhu}@duke.edu

Abstract

Hierarchical representations provide powerful and principled approaches for analyzing many musical genres. Such representations have been broadly studied in music theory, for instance via Schenkerian analysis (SchA). Hierarchical music analyses, however, are highly cost-intensive; the analysis of a single piece of music requires a great deal of time and effort from trained experts. The representation of hierarchical analyses in a computer-readable format is a further challenge. Given recent developments in hierarchical deep learning and increasing quantities of computer-readable data, there is great promise in extending such work for an automatic hierarchical representation framework. This paper thus introduces a novel approach, AutoSchA, which extends recent developments in graph neural networks (GNNs) for hierarchical music analysis. AutoSchA features three key contributions: 1) a new graph learning framework for hierarchical music representation, 2) a new graph pooling mechanism based on node isolation that directly optimizes learned pooling assignments, and 3) a state-of-the-art architecture that integrates such developments for automatic hierarchical music analysis. We show, in a suite of experiments, that AutoSchA performs comparably to human experts when analyzing Baroque fugue subjects.

Supplemental Material —
github.com/stephenHahn88/AutoSchA_Supplement.git

1 Introduction

Music theory is an art that aims to understand how music of various styles is designed, constructed, and composed. Whether it is understood implicitly or explicitly, one's own theory of music is vital for any musician to generate original works of art with influence from what they have heard in the past. Hierarchical structures provide an elegant framework for music representation and are broadly used in the music community. In particular, Schenkerian analysis (SchA) is a widely taught representation which fuses elements of melody, harmony, form, and counterpoint. Many composers use SchA to inform their compositions by enhancing their understanding of other works, incorporating structural ideas and techniques into their own (Schenker 2000; Jackson 2001). While SchA was originally designed for Western common practice tonal music, aspects have been used in a broad range of musical genres from rock and jazz to Chinese opera and African folk music (Nobile 2014; Stock 1993; Larson 2009).

In this work, we use machine learning to identify the hierarchical structure of a musical piece – a task we call *automatic hierarchical music representation*. Such automatic representations provide users a better understanding of the direction a piece can take, in turn leading to more convincing generation at larger scales. While previous work on machine models for music theory and generation have shown promise for small-scale structures and surface-level counterpoint (Ferreira, Limongi, and Fávero 2023; Huang et al. 2018, 2019; Wu et al. 2019), there is scant literature on our target problem of automatic hierarchical music representation. This may be due to two inherent difficulties. First, to learn complex hierarchical structure, the employed model requires careful analyses from trained experts. Such training data are *cost-intensive* to generate: an analysis of a single musical phrase can take longer than 30 minutes depending on complexity. Second, compactly representing hierarchical analyses in a computer-readable data format for model training remains a challenge, with many published analyses being incomplete and not showing all details needed to present the entire analysis in a computer-readable format.

There has been recent promising work on hierarchical deep learning aimed at addressing these difficulties. In particular, graph neural networks (GNNs) have made strides in hierarchical data processing with pooling methods such as Diffpool (Ying et al. 2018) and top-k pooling (Cangea et al. 2018). Given recent developments in hierarchical deep learning and the increasing amount of computer-readable data, there is promise in extending such work for automatic hierarchical representation of music.

We thus introduce the first deep learning model for SchA, called *AutoSchA*, which extends recent developments in GNNs for effective automatic music representation. Our novel contributions include 1) a new graph learning framework for hierarchical music representation, 2) a new graph pooling mechanism based on node isolation that directly optimizes learned pooling assignments, and 3) a state-of-the-art architecture integrating such developments for automatic hierarchical music analysis.

*Equal Contribution

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

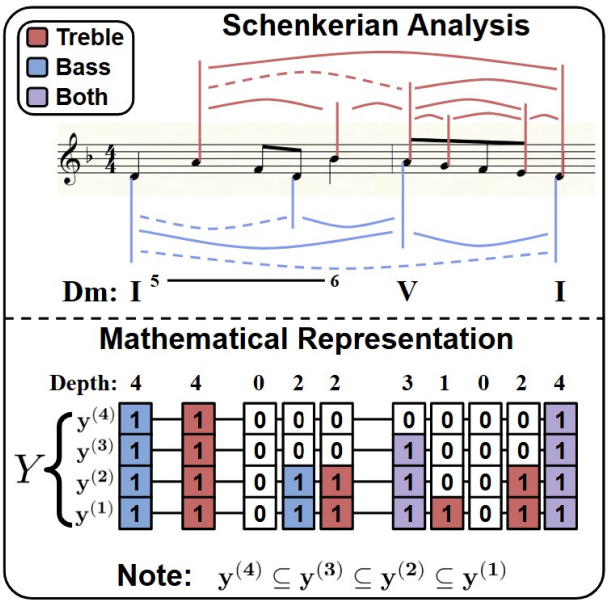

Figure 1: Top: SchA, Pachelbel's *Primi Toni* No. 1, fugue subject. Bottom: Mathematical representation of SchA as several bit arrays denoting whether notes belong in a certain depth. Note that higher depths are included in lower depths (the first note is included in all depths from 0-4).

Section 2 briefly describes SchA, previous computational models, and existing models for graph pooling. Section 3 presents our methodology, including the representation of SchA as a graph pooling problem, and the components of our novel modeling architecture. Section 4 describes our experiments, which show that we perform comparably to humans in analyzing Baroque fugue subjects. This includes ablation experiments that investigate the most important features for computational SchA and compare with other baseline deep learning models. Our results show great promise in the future of AI-assisted SchA.

2 Related Work

2.1 Hierarchical Music Analysis

Hierarchical music representations are broadly used in many fields of music theory. For instance, Hepokoski and Darcy (2006); Caplin (2013) focus on form or thematic structure, Rothstein (1989); Lerdahl and Jackendoff (1996); Foscarin, Harasim, and Widmer (2023) focus on rhythmic, phrase, or grammar-based structure, and Schenker (1935); Cadwallader, Gagné, and Samarotto (1998) focus on the linear/harmonic structure. In what follows, we focus on SchA, a hierarchical linear-harmonic framework first codified by Heinrich Schenker aiming to capture how the music analyst hears the progression of musical harmonies on various levels of structure as they are "unfolded" by linear melodic motions.

Figure 1 displays an example SchA for a Pachelbel fugue subject, outlining the hierarchy of tones and the harmonies these tones produce over time. Longer note stems indicate deeper levels of structure, slurs indicate prolongations between notes at a given level, and dotted slurs indicate prolongations between notes of the same pitch. Roman numerals describe harmonies unfolded by melodic tones. The analysis shows that the first four notes outline the root position tonic ("I") triad, the fifth note destabilizes the tonic function with a 5–6 exchange as it moves towards the dominant "V," which is expressed through the subsequent descending four notes. The final note shows a return to "I." The downward and upward note stems indicate the bass and treble voices respectively, with the bass voice tending to outline harmonies with larger leaps and the treble and inner voices tending to act with smoother linear motions. In our example the bass follows the roots of the I-V-I harmonies (D-A-D) and the treble moves entirely stepwise (A-B♭-A-G-F-E-D).

SchA depends on the various structural levels ("depths") of the music. The music as written in the score is the "foreground," which hosts all details that make a piece unique. As layers of decoration and relatively less structural tones are removed, deeper levels of "middleground" structure are revealed until finally we reach the deepest level, the "background" or *Ursatz* structure, often represented with open note heads. For our model, notes on the foreground are considered depth 0, notes that have a stem are considered part of depth 1, notes with relatively longer stems are included in depth 2, etc., until the deepest level is reached. In Figure 1, the maximal depths of each note are notated at the bottom. This depth notation is not typically explicit in SchA. The pre-penultimate note (F4) belongs to depth 0 in the treble voice, but it does not belong in depth 1. The fourth-to-last note (G4) belongs to depths 0 and 1, but not depth 2, since its analytical stem is at the shortest level. Corresponding notes in the bass and treble such as the first two notes share the same maximal depth.

2.2 Computational Schenkerian Analysis

While structural analyses such as SchA provide vital insights into symbolic music, there is a paucity of research on computational SchA. The vast majority of past approaches rely on heuristics or rule-based systems due to the absence of computer-readable data (Kassler 1975; Frankel, Rosenschein, and Smoliar 1976; Smoliar 1979; Mavromatis and Brown 2004; Gilbert and Conklin 2007). The first steps toward a data-driven approach were taken by Marsden (2010), who introduced a "goodness metric" based on a corpus of six Mozart analyses. This metric was used to determine high scoring analyses out of a prohibitively large search space of possible analyses.

Most notably, Kirlin introduced a probabilistic approach based on a new corpus of 41 SchAs (Kirlin and Jensen 2011; Kirlin 2014; Kirlin and Jensen 2015). Such models employ random forests to predict how likely certain notes *prolong* others, where a prolongation is defined as two notes note_1 and note_α that are more structural than a group of intervening notes $(\text{note}_2, ..., \text{note}_{\alpha-1})$. These analyses were represented as restricted versions of Yust's maximal outerplanar graphs (MOPs; Yust 2015).

While their work is based in Schenkerian theory, the model of Kirlin and Jensen (2015) does not perform a proper Schenkerian analysis, nor do the authors claim that it does so. Their model reads a melodic line as a single theoretical

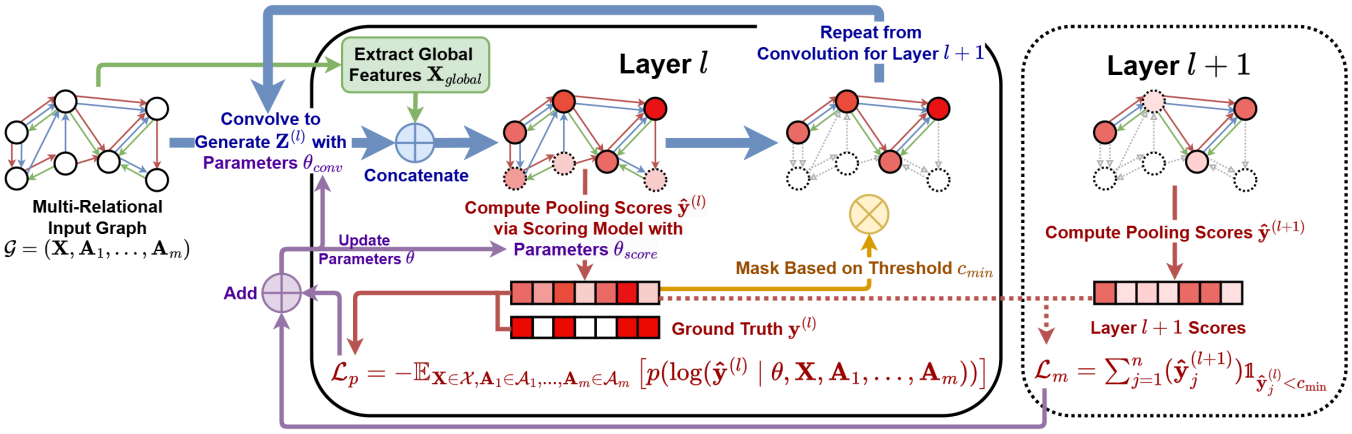

Figure 2: Overview of model architecture: diagram for the pooling GNN. Blue arrows indicate the major flow of GNN convolution, green arrows represents global feature extraction, yellow indicates threshold masking, red describes information regarding the scoring model and loss functions, and purple arrows show the flow of backpropogation. Given input graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}_1, ..., \mathbf{A}_m)$, we first perform a directed multi-relational convolution (parameterized by θ_{conv}) over input features $\mathbf{X}$ to generate node embeddings $\mathbf{Z}^{(l)}$, and global feature matrix $\mathbf{X}_{\text{global}}$ (see Figure 4 for details). $\mathbf{Z}^{(l)}$ and $\mathbf{X}_{\text{global}}$ are concatenated and passed to the scoring model (parameterized by θ_{score}), generating pooling scores $\hat{\mathbf{y}}^{(l)}$. We aim to minimize the cross-entropy $\mathcal{L}_p$ between the pooling scores and ground truth assignments. Based on $\hat{\mathbf{y}}^{(l)}$ and threshold c_{min}, the nodes are masked or "isolated" for the next GNN layer $l+1$, to which $\mathbf{Z}^{(l)}$ is passed in place of $\mathbf{X}$. To ensure monotonicity in pooling scores, we add a regularizer term $\mathcal{L}_m$ computed from scores at layer l and layer $l+1$.

voice, often misinterpreting the compound polyphonic nature of the music. In fact, it is incapable of reading multiple musical lines simultaneously, which is vital to SchA even in monophonic settings where music is notated as a single voice. Without an understanding of compound melody and counterpoint between voices, it is impossible to describe musical structure as we hear it.

2.3 Graph Neural Networks for Graph Pooling

We introduce a model for computational SchA as a graph pooling problem framed using GNNs. GNNs learn mappings from a node parameter space to some latent space, where the geometric relations of the embedded nodes reflect the structure of the original graph (Hamilton, Ying, and Leskovec 2017). In graph classification, it is necessary to condense node-level representations into graph-level representations, an operation termed *graph pooling*. Early approaches mainly focused on adapting existing graph clustering algorithms (Bruna et al. 2013; Defferrard, Bresson, and Vandergheynst 2017), e.g., the Graclus algorithm (Dhillon, Guan, and Kulis 2007). The current state-of-the-art in graph pooling operators permit dynamic learning of optimal pooling assignments to adapt to downstream tasks (Grattarola et al. 2021). Of this new family of learnable pooling operators, two main approaches have emerged: clustering approaches, which learn grouping matrices from the node embedding space (Ying et al. 2018; Bianchi, Grattarola, and Alippi 2020; Bacciu and Di Sotto 2019; Yuan and Ji 2020; Noutahi et al. 2019; Tsitsulin et al. 2023), and top-k approaches, which fit a scoring function over the node set and prune the lowest scoring nodes, the number of which to prune at each layer being determined by a predefined percentage (Gao and Ji 2019; Cangea et al. 2018; Lee, Lee, and Kang 2019; Ranjan, Sanyal, and Talukdar 2020; Zhang et al. 2019) or score threshold (Knyazev, Taylor, and Amer 2019).

Real-world networks often involve modeling the heterogeneous interactions between a collection of entities, motivating graph learning algorithms that operate on a multi-relational domain. Schlichtkrull et al. (2018) and Khan and Blumenstock (2019) propose methods to compute node embeddings that aggregate information across all relations present in a heterogeneous graph. *Pooling* multi-relational graphs, however, is relatively underexplored, and existing methods cannot be applied directly for SchA. For instance, Wu et al. (2021) describes pooling for a heterogeneous vertex set, not edge sets as in SchA. Khan and Blumenstock (2019) first compute a consensus graph over all edge types and then perform a manifold ranking (Zhou et al. 2003) to compute node saliency, but this approach is structural and not learnable in that it does not incorporate node embedding information to compute the pooling filtration. Finally, Lin et al. (2021) proposes an algorithm that is able to leverage node information, but is non-adaptive in that the number of clusters the model learns is fixed as a hyperparameter.

3 Methodology

3.1 Notation and Problem Formulation

We consider multi-relational graphs of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the set of n vertices $\mathcal{V} = \{v_1, ..., v_n\}$ indexed by j, $\mathcal{E} = \{E_1, ..., E_m\}$ is the union of edge sets over all m edge types (indexed by i), and $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the input feature matrix where the jth row describes features of the corresponding jth node. The number of edge types is constant throughout the entire dataset; thus, any given graph

$\mathcal{G}$ can be fully characterized by $\mathbf{X} \in \mathbb{R}^{n \times p}$, as well as adjacency matrices $\mathbf{A}_1, ..., \mathbf{A}_m \in \mathbb{R}^{n \times n}$.

We express a corresponding Schenkerian analysis of depth d as a sequence of bit arrays $Y = (\mathbf{y}^{(1)}, ..., \mathbf{y}^{(d)})$, each indicating whether particular vertices belong within a particular depth (see Figure 1). Higher depths are more sparse, containing more structural notes. Since Schenkerian analysis can be understood as a recursive process where all notes that are removed at a given level cannot appear in deeper levels, the sequence is nested: $\mathbf{y}^{(l)} \subseteq \mathbf{y}^{(k)}$ for all $l \geq k$.

3.2 Graph Representation for Symbolic Music

In order to process music through our GNN, we first convert musical scores to multi-relational graph data structures (see Figure 3). Following Jeong et al. (2019), we represent nodes as musical notes and edges as relationships between notes; going forward, "note" and "node" are used interchangeably.

In our representation, nodes are described by various continuous and categorical features. We include three pitch features: 1) pitch class: the discrete note pitch name (e.g., A♯ vs. D♭, etc.); 2) normalized midi pitch, which encodes a proxy for a note's location on the piano; and 3) scale degree: the note's relationship to the home key. We also include three rhythmic features: 1) normalized duration: the relative length of a note to other notes in the excerpt; 2) normalized offset: the relative location in time of a note; and 3) metric strength: the relative strength of the onset of a note (e.g., downbeats vs. offbeats).

For edge types, we first adopt the edges described by Jeong et al. (2019). These include 1) onset edges, which relate notes that begin at the same time; 2) forward edges, which relate notes with notes that immediately follow in time; 3) voice edges, which are similar to forward edges, but for notes within the same contrapuntal line; 4) rest edges, which are similar to forward edges, but connecting notes separated by rests; 5) sustain edges, which relate all notes that occur while one is held; and 6) slur edges, which relate notes that share a slur. Because music only flows forwards in time, the forward edge graph is acyclic.

To address deeper structural connections, we add novel *intervalic* edges. For each note, we create an edge to the next note up or down a certain diatonic interval if it exists. For example, in Figure 3, different edge types describe the relationships between the notes of the score. The first note (D4) is connected with the penultimate note (E4) by a "next 2nd up" edge. Similarly, the third note (F4) is connected to the 7th note (G4) with the same edge and to the penultimate note (E4) with a "next 2nd down" edge.

3.3 Pooling Structure

One way to interpret SchA is to "remove" structurally less important notes at various depths, only keeping those describing linear and harmonic function of the music at each depth. Top-k pooling approaches provide a natural formulation for this task as they follow a related procedure of node removal. However, such methods have not been used for heterogeneous graphs and often require k (the percentage of nodes to drop at each depth) to be set in advance as a hyperparameter. For SchA, this k can vary from piece to piece

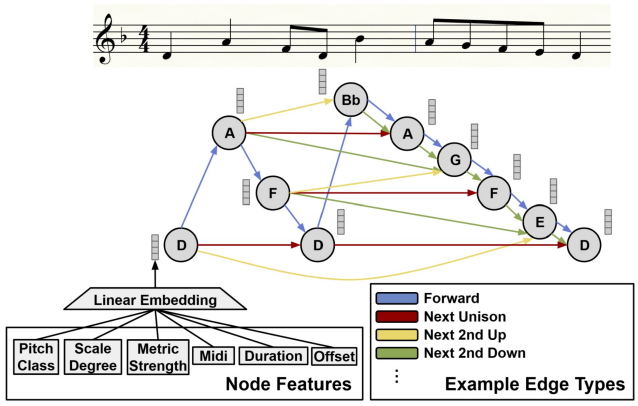

Figure 3: *Primi Toni* No. 1 as a multi-relational graph.

and from depth to depth, necessitating an adaptive method for removing nodes.

While pooling is typically a byproduct in classification tasks, it is vital to SchA. To our knowledge, our model is the first to directly optimize a sequence of pooling assignments as its main objective. This section introduces our novel multi-relational pooling layer based on node isolation, a general framework that can explicitly learn pooled node representations for classification based on any given criteria for SchA or related tasks.

Multi-Relational Graph Convolution The first step in processing node embeddings for SchA is graph convolution (recall Figure 2). The standard graph convolution operator introduced in Kipf and Welling (2016) is given by

$$\mathbf{Z}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(l)}\mathbf{W}^{(l+1)}),$$

where $\mathbf{Z}^{(l)}$ is the node embedding matrix after l GNN layers ($\mathbf{Z}^{(0)} = \mathbf{X}$), $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the adjacency matrix with self-loops, $\tilde{\mathbf{D}}$ is the associated degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{p \times q}$ is a matrix of learnable parameters for layer l where p and q are hyperparameters, and σ is a non-linear activation function (e.g., ReLU). The expression $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(l)}$ spreads node features to their neighbors while scaling them based on node degree. This allows each node to learn from its neighbors without letting high-degree nodes overwhelm the information. The result is then linearly transformed by $\mathbf{W}^{(l+1)}$ to update the node embeddings.

A natural extension of this operator to the multi-relational setting is given in Schlichtkrull et al. (2018). The matrix form of the node embedding equation is

$$\mathbf{Z}^{(l+1)} = \sigma\left(\sum_{i=1}^{m} \tilde{\mathbf{D}}_i^{-\frac{1}{2}}\tilde{\mathbf{A}}_i\tilde{\mathbf{D}}_i^{-\frac{1}{2}}\mathbf{Z}^{(l)}\mathbf{W}_i^{(l+1)}\right).$$

More generally, the sum over edge types can be replaced with any aggregation method (e.g., concatenation). We extend the multi-relational convolution operator to the directed case, letting $\tilde{\mathbf{M}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ denote the normalized forward edge adjacency matrix as in Rossi et al. (2024):

$$\mathbf{Z}^{(l+1)} = \sigma\left(\sum_{i=1}^{m}\left(\alpha\tilde{\mathbf{M}}_i\mathbf{Z}^{(l)}\mathbf{W}_{i,\rightarrow}^{(l+1)} + \bar{\alpha}\tilde{\mathbf{M}}_i^{\intercal}\mathbf{Z}^{(l)}\mathbf{W}_{i,\leftarrow}^{(l+1)}\right)\right).$$

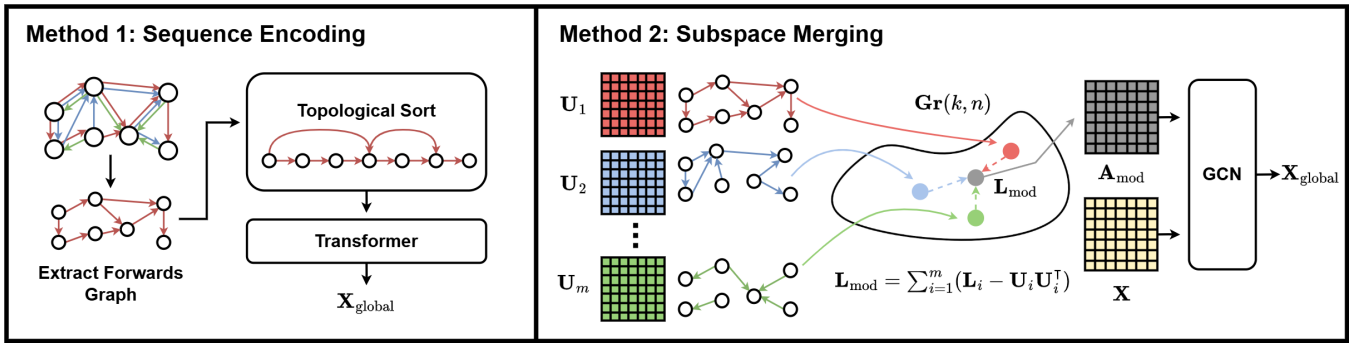

Figure 4: Two approaches to compute the global embedding, $\mathbf{X}_{\text{global}}$. (1) A sequential approach, leveraging a transformer to encode long-range dependencies. We obtain a canonical sequence representation of our graph via the topological order of the forwards edges. (2) A subspace merging approach, based on computing a unified global topology fusing all edge types. We compute a fused global graph $\mathbf{A}_{\text{mod}}$ given fixed embeddings $\mathbf{U}_1, ..., \mathbf{U}_m$ from the Laplacians $\mathbf{L}_1, ..., \mathbf{L}_m$, and then convolve the original features $\mathbf{X}$ over $\mathbf{A}_{\text{mod}}$. Further details are in the technical appendix.

Here $\mathbf{W}_{i,\rightarrow}, \mathbf{W}_{i,\leftarrow}$ denote the learnable weights for the forward and backward direction edges for edge type i, and $\alpha \in [0, 1]$ is a hyperparameter weighing the forwards and backwards edges in tandem with $\bar{\alpha} = 1 - \alpha$.

Node Isolation for Multi-Relational Pooling One of the main challenges in SchA is that the number of nodes we wish to drop at each level varies with respect to the input graph. For example, a passage with a high density of neighbor or repeated tones would result in a large percentage of notes being dropped, as neighbor tones are non-structural with regard to the foreground of the piece. On the other hand, a passage with a variety of harmonic shifts along with few non-harmonic tones (NHTs) and repetitions would drop a low percentage of notes at each level of Schenkerian depth.

We now outline a novel node downsampling mechanism, "node isolation." We first compute a score $\hat{\mathbf{y}}_j$ for each node in the input graph (similar to Lee, Lee, and Kang 2019) using the multi-relational convolution of Rossi et al. (2024) described earlier:

$$\hat{\mathbf{y}} = \sigma\left(\sum_{i=1}^{m}\left(\alpha\tilde{\mathbf{M}}_i\mathbf{Z}\mathbf{W}_{i,\rightarrow} + \bar{\alpha}\tilde{\mathbf{M}}_i^\intercal\mathbf{Z}\mathbf{W}_{i,\leftarrow}\right)\right),$$

where $\mathbf{W}_{i,\rightarrow}, \mathbf{W}_{i,\leftarrow} \in \mathbb{R}^{p\times 1}$ and $\sigma : \mathbb{R}^n \rightarrow [0, 1]^n$ represents the sigmoid function. Next, we prune the adjacency matrices $\mathbf{A}_1, ..., \mathbf{A}_m$ in accordance with the indicator vector $\mathbf{idx} = \mathbb{1}_{\hat{\mathbf{y}}_j \geq c_{\min}}$ for some minimum threshold hyperparameter $c_{\min} \in (0, 1)$:

$$\begin{aligned}\mathbf{A}_i' &= (\mathbf{A}_i \odot \mathbf{idx})^\intercal \odot \mathbf{idx},\\ \mathbf{Z}' &= \mathbf{Z} \odot \hat{\mathbf{y}},\end{aligned} \tag{1}$$

where $\odot$ denotes a row-wise product. The indicator vector, $\mathbf{idx}$, masks the rows and columns of the adjacency matrix to ensure all edges to and from isolated nodes are pruned. The node embeddings are then weighted by their scores $\hat{\mathbf{y}}$ before being passed to the next convolutional layer.

While the nodes below the scoring threshold remain in the graph, by pruning edge connections, we create isolated nodes that cannot affect other nodes; no message passing occurs in or out of the isolated nodes.

Through this approach, our method easily extends existing classification models with access to ground truth node values to learn pooling assignments. We train learnable weights $\mathbf{W}$ of our pooling assignment model to minimize the binary cross-entropy between the generated score vector $\hat{\mathbf{y}}$ and the underlying ground truth $\mathbf{y}$ at each depth (see Figure 2). The objective is then summed over all d depths to obtain the pooling loss term:

$$\begin{aligned}\mathcal{L}_p(\theta) =& -\mathbb{E}_{\mathbf{X}\in\mathcal{X},\mathbf{A}_{1,\dots,m}\in\mathcal{A}_{1,\dots,m}}[\log p(\hat{\mathbf{y}}|\theta, \mathbf{X}, \mathbf{A}_1, ..., \mathbf{A}_m)]\\ =& -\sum_{l=1}^{d}\mathbf{y}^{(l)}\log(\hat{\mathbf{y}}^{(l)}) + (1-\mathbf{y}^{(l)})\log(1-\hat{\mathbf{y}}^{(l)}),\end{aligned}$$

where θ represents learnable parameters $\mathbf{W}_{i,o}^{(l)}$ for all $1 \leq i \leq m$, $1 \leq l \leq d$, and $o \in \{\leftarrow, \rightarrow\}$. Note that the variables of $\hat{\mathbf{y}}$ we want to optimize are contained in θ.

To ensure $\hat{\mathbf{y}}^{(1)}, \dots, \hat{\mathbf{y}}^{(l)}, \dots, \hat{\mathbf{y}}^{(d)}$ is a valid filtration, we introduce a second novel loss term that acts as a monotonicity regularizer. This new loss term encourages nodes that have a score below the cutoff threshold at level l to remain below the threshold at level $l + 1$ (and thus at all remaining levels), which leads to the loss term:

$$\mathcal{L}_m(\theta) = \sum_{l=2}^{d-1}\sum_{j=1}^{n}\hat{\mathbf{y}}_j^{(l+1)} \cdot \mathbb{1}_{\hat{\mathbf{y}}_j^{(l)} < c_{\min}}.$$

Thus, the complete objective is $\mathcal{L}(\theta) = \mathcal{L}_p(\theta) + \mathcal{L}_m(\theta)$.

3.4 Global Feature Aggregation

In addition to local information such as interval relations between nearby notes, global information (e.g., key, large-scale tonicization) is also vital to SchA. Thus, the model requires understanding of global feature input in order to create a suitable node scoring function. We investigate two approaches below, based on sequential and topological views of the music graph. Figure 4 visualizes the two proposed methods side by side.

Sequential Approach Previous approaches for algorithmic symbolic music composition and harmonization treat

music as a sequence of discretized tokens. Transformer models in particular demonstrate strong ability to leverage long term harmonic information encoded within a passage globally in performing composition tasks (Huang et al. 2018; von Rütte et al. 2022).

We use a transformer encoder on the input features to a given layer l (see Method 1 of Figure 4). To compute an appropriate sequence representation of our graph, we leverage the fact that the forward edge graph should be a directed acyclic graph by first performing a topological sort of our nodes, breaking ties in vertical order (treble to bass):

$$\mathbf{x}_{\text{global}}^{(l)} = \text{Transformer}\big(\text{Topological-Sort}(\mathbf{Z}^{(l)})\big),$$

$$\mathbf{Z}'^{(l)} = \begin{bmatrix} \mathbf{Z}_1^{(l)} \\ \vdots \\ \mathbf{Z}_n^{(l)} \end{bmatrix} \Big\| \begin{bmatrix} \mathbf{x}_{\text{global}}^{(l)} \cdot \hat{\mathbf{y}}_1^{(l-1)} \\ \vdots \\ \mathbf{x}_{\text{global}}^{(l)} \cdot \hat{\mathbf{y}}_n^{(l-1)} \end{bmatrix},$$

where $\|$ represents column concatenation.

Here, Topological-Sort$(\cdot)$ performs a topological sort of the rows of $\mathbf{Z}^{(l)}$. $\mathbf{Z}'^{(l)}$ is then passed into the scoring GNN to compute pooling scores. We then reweigh the global embedding with the node attention scores from the previous layer. This prevents the global embeddings from disrupting monotonicity by reducing the influence of nodes that have already been dropped out or are close to dropping out. All together, this approach helps the model understand each note's place within the whole of the piece.

Subspace Merging Approach Alternatively, the subspace merging approach aims to find a unified global topology that can capture the node connections over all edge types (see Method 2 of Figure 4 and the technical appendix for full technical details). For simplicity, we focus on the undirected representation of our input graphs, $\hat{\mathbf{A}}_i$ (computed as the logical *or* between $\mathbf{A}_i$ and $\mathbf{A}_i^\intercal$), which we use to generate a fused global graph $\mathbf{A}_{\text{mod}}$. Using this global topological information, we then convolve the initial embeddings over the fused graph, generating node-level features which integrate information over all layers of the graph. We find that this approach improves predictive accuracy over the baseline model with a comparable number of parameters.

3.5 Voice Assignment and Inference

SchA makes the distinction between different theoretical voices. Western tonal music particularly relies on distinct structural roles for treble and bass voices. Inner voices may also host important motivic and structural support, but generally not at the same level as the indicated outer voices. We thus pose our task as a multi-label prediction task over all graph nodes, with class options in $\{$*treble*, *bass*, *inner*$\}$. Since voice assignment is determined exclusively using information from the first GNN convolution layer, we need only to generate the voice assignment once and propagate the assignment to each depth. We trained a Directed R-GCN as described in Section 3.3 with sigmoid activation to perform this task.

We first predict the voice assignment for each node in the input graph and then compute scores for each musical depth represented in our model. For both voice and depth inference, a human expert may choose their preferred threshold $c_{\min}$ to produce their ideal results. If a higher number of depths are required or ambiguities arise in the highest depth given by the model, the relative confidence of scores can be used to inform expert decisions.

4 Experiments and Results

4.1 Experimental Design

We present a suite of experiments to assess performance of our methodology. We use the Schenkerian analysis dataset introduced by Ni-Hahn et al. (2024). The dataset consists of >140 analyses in computer-readable format. Particularly, we focus on 88 analyses of fugue subjects by Bach and Pachelbel as they are the most abundant and relatively homogeneous subset of analyses in the dataset. Because we are estimating structure for each note individually, this totals 3807 inferences per estimated depth (there are 3807 total notes, each note could be in each depth). To augment the data, we transpose the analyses to 12 musical keys, leading to a total 45,684 inferences per estimated depth.

Relative Performance To measure the relative performance of our model, we compare with other deep sequence and graph-based approaches. We separately train each baseline for each level of SchA to minimize the binary cross-entropy of node scores for specific depths. We compare with two non-graph-based models, a multi-layer perceptron (MLP) and transformer. We encode the nodes with positional information given by the topological order of the forwards graph as in Section 3.4. We also compare with three graph-based models: GCN (Kipf and Welling 2016), Graph Attention Networks (GATs) (Veličković et al. 2018), and R-GCN (Schlichtkrull et al. 2018). For the GCN and GAT, we use the undirected representation of our graphs with adjacency matrix equal to the union of all edge types. Accuracy is measured by the proportion of node scores on the correct side of the threshold as compared to the ground truth of expert-annotated SchA, and monotonicity loss was defined previously in Section 3.3). We report the means and standard deviations of the maximum accuracy and lowest monotonicity loss attained across test samples in Table 1.

Human Experiments To assess the subjective performance of AutoSchA's analyses, we surveyed five experts in music theory and Schenkerian analysis. The survey presented 9 scores and midi recordings, each with three corresponding Schenkerian analyses. We compare the human analysis from the dataset, AutoSchA's analysis, and an analysis purposefully designed to be unmusical, which we denote as "Flawed" analyses. We do not compare against Kirlin and Jensen (2015) because their model does not perform proper Schenkerian analysis (see Section 2.2). Analyses from AutoSchA were created by choosing an optimal threshold at each depth, chosen by an expert in Schenkerian analysis. It is important to note that this evaluation represents the largest number of Schenkerian experts ever polled for experiments in computational SchA (experiments by Kirlin and Jensen 2015 relied on 3 experts). Experts with

graduate-level training in the relatively small field of SchA are extremely difficult to find, and in this case, they each provided hours of effort analyzing SchA attempts.

The three analyses for each score were presented in random order to avoid positional bias. For each analysis, we assessed overall quality as a letter grade, asking "What letter grade would you assign this analysis" as if grading a student (A+, A, A-, B+, etc.). To judge perceived musicality of an analysis we asked, "How would you score the musicality of this analysis on a scale of 0 (not musical) to 10 (very musical)?". We performed a Turing test, asking "How certain are you that the analysis was written by a human on a scale of 0 (certain it's by a computer) to 10 (certain it's by a human)?". Lastly, we asked participants for a yes/no response to the question, "Are there any clearly awkward or weird portions of the analysis?" asking if they would optionally explain their response if any were found. Results are in Table 2.

Ablation Experiments and Case Study For ablation experiments (see Appendix A), we removed certain node and edge features, then trained 30 model copies for 3 epochs, evaluating maximum validation accuracy for each ablated model over the combined 90 epochs the model copies were trained. We additionally explored how model performance changes with different values of threshold $c_{\min}$ and edge direction weighting α. Our case study (see Appendix B) shows a comparison of AutoSchA's analysis of Pachelbel's *Primi Toni 1* with that of the human analyst.

4.2 Results

Relative Performance Table 1 presents performance of AutoSchA vs. various existing models, suggesting higher best-case accuracy and low best-case monotonicity loss for our proposed methodology. Notably, our model with sequential encoding for global features attains the highest accuracy.

Method	Accuracy	Monotonicity
MLP	0.714 (0.002)	0.014 (0.001)
Transformer	0.707 (0.011)	0.011 (0.004)
GCN	0.663 (0.006)	0.007 (0.002)
GAT	0.675 (0.006)	0.012 (0.018)
R-GCN	0.729 (0.012)	0.013 (0.002)
AutoSchA (Base)	0.731 (0.015)	**0.003 (0.002)**
AutoSchA (Grassmann)	*0.738 (0.021)*	0.016 (0.020)
AutoSchA (Sequence)	**0.749 (0.015)**	*0.004 (0.003)*

Table 1: Mean (SD) best-case validation set accuracy (higher is better) and monotonicity loss (lower is better).

Human Experiments Table 2 suggests that AutoSchA is significantly better than the flawed analysis and relatively close to the performance of the human analyst. The average grade for AutoSchA is around a B+, whereas the human received around an A-, and the flawed analysis a C. Similarly, when judging musicality, AutoSchA again nears the level of the human analyst, easily outperforming the flawed model. Although participants tended to believe most analyses were human-generated, *AutoSchA was rated similarly to human analyses* while the flawed analyses were statistically dissimilar. Finally, regarding the presence of "weird analyses," AutoSchA statistically outperforms the flawed analysis, but still has a significantly larger number of perceived awkward components compared with the human analyses, suggesting room for future work.

Method	Mean	95% CI	p-value
Grade Point Average (higher is better)			
AutoSchA	3.216	(3.032, 3.399)	ref.
Human	3.551	(3.363, 3.740)	0.012
Flawed	2.158	(1.871, 2.445)	<0.001
Musicality (higher is better)			
AutoSchA	7.196	(6.682, 7.709)	ref.
Human	8.038	(7.469, 8.607)	0.029
Flawed	4.653	(4.022, 5.284)	<0.001
Confidence in Human Annotation (higher is better)			
Human	6.467	(5.824, 7.109)	ref.
AutoSchA	6.227	(5.654, 6.800)	0.672
Flawed	5.236	(4.597, 5.874)	0.010
"Weird" Analysis Proportion (lower is better)			
AutoSchA	0.667	(0.523, 0.810)	ref.
Human	0.333	(0.190, 0.477)	0.001
Flawed	0.956	(0.893, 1.)	<0.001

Table 2: Evaluation of AutoSchA performance metrics.

Ablation Experiments Our ablation experiments suggest that rhythmic features are most vital to model performance, and including too many pitch features seems to confuse the model. Maximum accuracy was achieved by removing two of the three pitch features: pitch class and midi. We hypothesize that larger, more complex musical scores, and scores of different styles, may show very different variable importance metrics. For instance, a style that includes many overlapping suspensions would not rely on metric strength as much, for important structural tones would often be offset from strong metrical beats. Regarding threshold and edge direction weighting, the ideal threshold is 0.5 and the optimal alpha is 0.75, indicating that music analysis is mostly a forwards-looking phenomenon. The backwards component is also essential based on the steep drop from alpha=0.75 to alpha=1.0.

5 Conclusion

In this study, we describe a novel deep learning framework capable of generating convincing Schenkerian analyses near the level of a human analyst. These can be improved by human analysts more easily than creating them from scratch, and they can be used for downstream tasks like music generation and a broad set of applications to music theory analysis. We build on recent developments in GNNs and hierarchical deep learning to represent hierarchical musical analysis as a graph pooling problem, along with a novel graph pooling mechanism to directly model and learn the pooling process. By demonstrating the benefits of graph data structures for automatic hierarchical music representation, we open exciting new research avenues that advance music theory and methods development via GNNs.

References

Bacciu, D.; and Di Sotto, L. 2019. A non-negative factorization approach to node pooling in graph convolutional neural networks. In *AI*IA 2019–Advances in Artificial Intelligence: XVIIIth International Conference of the Italian Association for Artificial Intelligence*, 294–306. Springer.

Bianchi, F. M.; Grattarola, D.; and Alippi, C. 2020. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning (ICML)*, 874–883. PMLR.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral Networks and Deep Locally Connected Networks on Graphs. *arXiv preprint arXiv:1312.6203*.

Cadwallader, A. C.; Gagné, D.; and Samarotto, F. 1998. Analysis of tonal music: a Schenkerian approach. *(No Title)*.

Cangea, C.; Veličković, P.; Jovanović, N.; Kipf, T.; and Liò, P. 2018. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*.

Caplin, W. E. 2013. *Analyzing classical form: An approach for the classroom*. Oxford University Press, USA.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2017. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*.

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2007. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11): 1944–1957.

Ferreira, P.; Limongi, R.; and Fávero, L. P. 2023. Generating music with data: application of deep learning models for symbolic music composition. *Applied Sciences*, 13(7): 4543.

Foscarin, F.; Harasim, D.; and Widmer, G. 2023. Predicting Music Hierarchies with a Graph-Based Neural Decoder. arXiv:2306.16955.

Frankel, R. E.; Rosenschein, S. J.; and Smoliar, S. W. 1976. A LISP-based system for the study of Schenkerian analysis. *Computers and the Humanities*, 21–32.

Gao, H.; and Ji, S. 2019. Graph u-nets. In *International Conference on Machine Learning (ICML)*, 2083–2092. PMLR.

Gilbert, É.; and Conklin, D. 2007. A probabilistic context-free grammar for melodic reduction. In *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence*, 83–94.

Grattarola, D.; Zambon, D.; Bianchi, F. M.; and Alippi, C. 2021. Understanding Pooling in Graph Neural Networks. *arXiv preprint arxiv:2110.05292*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Hepokoski, J.; and Darcy, W. 2006. *Elements of sonata theory: Norms, types, and deformations in the late-eighteenth-century sonata*. Oxford University Press.

Huang, C.-Z. A.; Cooijmans, T.; Roberts, A.; Courville, A.; and Eck, D. 2019. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

Jackson, T. L. 2001. Heinrich Schenker as Composition Teacher: The Schenker-Oppel Exchange. *Music Analysis*, 20(1): 1–115.

Jeong, D.; Kwon, T.; Kim, Y.; and Nam, J. 2019. Graph neural network for music score data and modeling expressive piano performance. In *International Conference on Machine Learning (ICML)*, 3060–3070. PMLR.

Kassler, M. 1975. *Proving musical theorems I: The middleground of Heinrich Schenker's theory of tonality*. 103. Basser Department of Computer Science, School of Physics, University of Sydney.

Khan, M. R.; and Blumenstock, J. E. 2019. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 606–613.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kirlin, P.; and Jensen, D. 2015. Learning to Uncover Deep Musical Structure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Kirlin, P. B. 2014. A Data Set for Computational Studies of Schenkerian Analysis. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*, 213–218.

Kirlin, P. B.; and Jensen, D. D. 2011. Probabilistic Modeling of Hierarchical Music Analysis. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*, 393–398.

Knyazev, B.; Taylor, G. W.; and Amer, M. 2019. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32.

Larson, S. 2009. *Analyzing Jazz: A Schenkerian Approach*. ACLS Humanities E-Book. Pendragon.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International Conference on Machine Learning (ICML)*, 3734–3743.

Lerdahl, F.; and Jackendoff, R. S. 1996. *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press.

Lin, Z.; Kang, Z.; Zhang, L.; and Tian, L. 2021. Multi-view attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*, 35(2): 1872–1880.

Marsden, A. 2010. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3): 269–289.

Mavromatis, P.; and Brown, M. 2004. Parsing context-free grammars for music: A computational model of Schenkerian analysis. In *Proceedings of the 8th International Conference on Music Perception & Cognition*, 414–415.

Ni-Hahn, S.; Xu, W.; Yin, J.; Zhu, R.; Mak, S.; Jiang, Y.; and Rudin, C. 2024. A New Dataset, Notation Software, and Representation for Computational Schenkerian Analysis. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*.

Nobile, D. F. 2014. *A structural approach to the analysis of rock music*. Ph.D. thesis.

Noutahi, E.; Beaini, D.; Horwood, J.; Giguère, S.; and Tossou, P. 2019. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv preprint arXiv:1905.11577*.

Ranjan, E.; Sanyal, S.; and Talukdar, P. 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5470–5477.

Rossi, E.; Charpentier, B.; Di Giovanni, F.; Frasca, F.; Günnemann, S.; and Bronstein, M. M. 2024. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference*, 25–1. PMLR.

Rothstein, W. N. 1989. *Phrase rhythm in tonal music*. Schirmer Books.

Schenker, H. 1935. *Free Composition: Volume III of new musical theories and fantasies*, volume 1. Pendragon Press.

Schenker, H. 2000. *The art of performance*. Oxford University Press.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference*, 593–607.

Smoliar, S. W. 1979. A computer aid for Schenkerian analysis. In *Association for Computing Machinery Proceedings of the 1979 Annual Conference*, 110–115.

Stock, J. 1993. The application of Schenkerian Analysis to Ethnomusicology: problems and possibilities. *Music Analysis*, 12(2): 215–240.

Tsitsulin, A.; Palowitch, J.; Perozzi, B.; and Müller, E. 2023. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127): 1–21.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

von Rütte, D.; Biggio, L.; Kilcher, Y.; and Hofmann, T. 2022. Figaro: Generating symbolic music with fine-grained artistic control. *arXiv preprint arXiv:2201.10936*.

Wu, C.; Wu, F.; Huang, Y.; and Xie, X. 2021. User-as-Graph: User Modeling with Heterogeneous Graph Pooling for News Recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1624–1630.

Wu, J.; Hu, C.; Wang, Y.; Hu, X.; and Zhu, J. 2019. A hierarchical recurrent neural network for symbolic melody generation. *IEEE Transactions on Cybernetics*, 50(6): 2749–2757.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, 31.

Yuan, H.; and Ji, S. 2020. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yust, J. 2015. Voice-leading transformation and generative theories of tonal structure. *Music Theory Online*, 21(4).

Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Yao, C.; Yu, Z.; and Wang, C. 2019. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*.

Zhou, D.; Weston, J.; Gretton, A.; Bousquet, O.; and Schölkopf, B. 2003. Ranking on Data Manifolds. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems*, volume 16. MIT Press.