An Interpretable, Flexible, and Interactive Probabilistic Framework for Melody Generation

Stephen Hahn
stephen.hahn@duke.edu
Duke University
Durham, NC, USA

Rico Zhu
rico.zhu@duke.edu
Duke University
Durham, NC, USA

Simon Mak
sm769@duke.edu
Duke University
Durham, NC, USA

Cynthia Rudin
cynthia@cs.duke.edu
Duke University
Durham, NC, USA

Yue Jiang
yue.jiang@duke.edu
Duke University
Durham, NC, USA

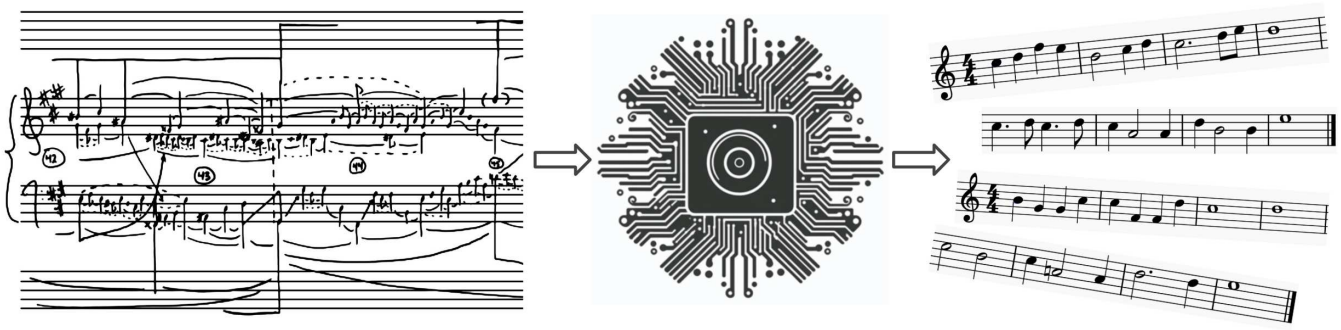

Figure 1: Schenkerian analysis [15] processed by a computer to generate music.

ABSTRACT

The fast-growing demand for algorithmic music generation is found throughout entertainment, art, education, etc. Unfortunately, most recent models are practically impossible to interpret or musically fine-tune, as they use deep neural networks with thousands of parameters. We introduce an interpretable, flexible, and interactive model, *SchenkComposer*, for melody generation that empowers users to be creative in all aspects of the music generation pipeline and allows them to learn from the process. We divide the task of melody generation into steps based on the process that a human composer using music-theoretical domain knowledge might use. First, the model determines phrase structure based on form analysis and identifies an appropriate number of measures. Using concepts from Schenkerian analysis, the model then finds a fitting harmonic rhythm, middleground harmonic progression, foreground rhythm, and melody in a hierarchical, scaffolded approach using a probabilistic context-free grammar based on musical contours. By incorporating theories of musical form and harmonic structure, our model produces music with long-term structural coherence. In extensive human experiments, we find that music generated with our approach successfully passes a Turing test in human experiments while current state-of-the-art approaches fail, and we further demonstrate superior performance and preference for our melodies compared to existing melody generation methods. Additionally, we developed and deployed a public website for SchenkComposer, and conducted preliminary user surveys. Through analysis, we show the strong viability and enjoyability of SchenkComposer.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Computing methodologies** → **Knowledge representation and reasoning**; **Machine learning approaches**; • **Information systems** → *Web applications*.

KEYWORDS

Algorithmic Music Generation; Interpretable Machine Learning; Schenkerian Analysis; Musical Form; Probabilistic Context-Free Grammars

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599772

ACM Reference Format:
Stephen Hahn, Rico Zhu, Simon Mak, Cynthia Rudin, and Yue Jiang. 2023. An Interpretable, Flexible, and Interactive Probabilistic Framework for Melody Generation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599772

1 INTRODUCTION

Algorithmic music generation (AMG) is becoming increasingly important in art and entertainment. For instance, many video games take advantage of AMG to further engross the player in the game's environment [10]. AMG has also been used to enhance visual art [29] and narratives [27]. Experienced composers use AMG as a tool for inspiration, collaboration [31] and education [19]. Perhaps most meaningfully, AMG can empower those who have little or no musical background with the ability to engage in the creative process.

Unfortunately, it is practically impossible to interpret and manipulate many recent AMG models because the vast majority of them use deep neural networks with thousands of parameters, even for potentially simple tasks. There have been attempts to "disentangle" deep neural networks using an interpretable latent space [50]. However, it is still practically impossible to understand how the models move from their latent spaces to their generated products. Indeed, it has become clear over the last several years of the value of interpretability in human-centered machine learning design for scientific and engineering advancements [6, 25, 39].

As we will show, it is possible to create machine learning models that incorporate general musical domain knowledge that lead to similar – if not better – results as the deep learning approaches, with the added benefits of *interpretability* and *flexibility*. An additional benefit of our approach is that it requires *much less data* than deep learning, which relies on vast amounts of training data that is generally not readily available without intensive data cleaning and processing. Our model is *far less complex* than the deep learning models, allowing users to interpret and tinker with the model's inner mechanisms to fit their needs and desires. In fact, with enough domain knowledge to set user-defined parameters, our model can produce reasonably convincing music with little training data.

One particularly difficult problem in AMG is modeling long-term structure. Two major branches of music theory coexist to describe this long-term structure in Western classical music: (1) form theory [3, 5, 21, 42], which describes music's structure in terms of section repetition and variation, and (2) Schenkerian analysis [4, 12, 40, 41], which aims to understand music's hierarchical harmonic-melodic structure. By involving and adapting these music-theoretical concepts of form and harmonic-melodic structure, we show that our model can produce convincing melodies with structural coherence.

Our model, which we call *SchenkComposer*, uses a novel grammatical approach that incorporates music-theoretical domain knowledge, yet it is simple enough for a user with little to no musical background to adjust its components and generate unique, personal results. In particular, our model defines a probabilistic context-free grammar (PCFG) for contours between notes at varying levels of structure. It also incorporates Markovian structures for deeper levels of structure and harmony. A high-level overview of our framework may be seen in Figure 2. The proposed model is novel in its use of Schenkerian analysis in the three *middleground* stages as well as the *foreground melody*.

Section 2 discusses related works. Section 3 provides background for music theory and mathematical notation. The framework in Figure 2 is described in detail in Section 4. We present human experiments in Section 5. Finally, in Section 6, we describe our online Web application.

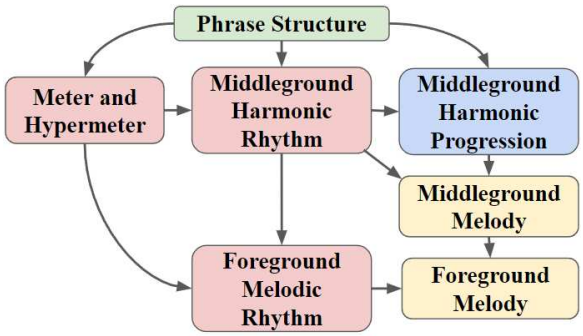

Figure 2: High-level overview of model components and their dependencies. Foreground melody is the final goal.

2 RELATED WORK

One of the greatest challenges in music generation for both humans and computers is to construct pieces with cohesive long-term structure. Most recent approaches use deep learning architectures such as the transformer neural network [46]; Huang et al. [17] and Huang and Yang [18] use transformers to produce expressive piano performances note by note. Before these transformer-based models, recurrent neural networks (RNNs) such as long-short-term-memory (LSTM) networks were most widely used for sequential tasks. Hadjeres et al. [14] use LSTMs for Bach-ian counterpoint, and Medeot et al. [28] and Wu et al. [49] generate melodies using LSTMs. Dai et al. [8] use a combination of both transformers and LSTMs for melody generation, with a hierarchical music representation used to generate pieces with coherent structures. Alternatively, Roberts et al. [37] generates melodies using a deep hierarchical variational autoencoder (VAE). Recently, the popular transformer-based language model, ChatGPT [32], has shown to be capable of writing music with limited success [11]. However, ChatGPT is still not capable of generating music anywhere near the level of other models, and therefore will not be used for comparison.

While nonparametric black box methods (specifically deep learning approaches) have had successes, they also have key limitations, both in their ability to capture long-term musical structure and to incorporate user feedback. Here, we ask the question of whether simpler, easier-to-use modeling techniques can achieve similar (or even better) performance. Many semantic models such as Flow Machines [33] and [47] make extensive use of Markovian models, which are simple and transparent. In particular, we consider *probabilistic context-free grammars* (PCFGs) which are commonly used for music analysis [9, 16, 24, 36, 44, 48]. PCFGs are also used for music synthesis: Rader [35] uses PCFGs to generate simple musical rounds, Bel and Kippen [2] for the generation of North Indian tabla drumming patterns, Assayag and Dubnov [1] for musical improvisation, and Tsushima et al. [45] for part of its model for automatic harmonization. Most related to our work, Keller and Morrison [19], Rodríguez et al. [38], and Nakamura et al. [30] use PCFGs for jazz improvisation, salsa improvisation, and generalized melody synthesis, respectively. None of these works incorporate Schenkerian

analysis to look at deeper levels of musical structure to provide deeper cohesion.

By incorporating Schenkerian analysis, we directly handle the problem of incorporating long-term structure. Schenkerian analysis looks at the hierarchical relationships between tones and harmonies, showing various layers of musical structure. Although many genres of Western music can be characterized using Schenkerian analysis, few computational systems for music analysis and generation explicitly use it. Nakamura et al. [30] experiment with music synthesis using a theory of tonal music described by Lerdahl and Jackendoff [23], which was influenced by natural language processing techniques and Schenkerian analysis. Gilbert and Conklin [13] reduces music to a pseudo-Schenkerian background structure using a PCFG of intervals. Using a "quasi-grammatical" binary-tree structure, Marsden [26] provides a proof of concept for computational Schenkerian analysis, and Kirlin [20] provides a dynamic programming algorithm for Schenkerian analysis using a combination of PCFGs and a maximal outerplanar graph representation. One significant issue with Marsden [26] and Kirlin [20] is an unrealistic assumption regarding prolongations used to manage the search space of potential analyses. When generating music, however, such limitations are not necessary. Note that all previous works incorporating Schenkerian analysis focus on analysis, not generation, as we do here.

3 RELEVANT BACKGROUND

3.1 Form Analysis

Form analysis is a field of study that breaks a musical piece into constituent *sections* based on similarities in rhythm, melody, and harmony. Generally, sections are broken into *phrases*, which may be broken further into *subphrases* (or *motifs*). Sections, phrases, and subphrases are most commonly labelled using lowercase and uppercase letters, where relatively larger structures use uppercase letters. In Figure 3, we show the subphrase structure of a single phrase in a Beethoven piano sonata in red, based mainly on related rhythms and melodic contours. Subphrase a is repeated with variation a' (2 measures each), followed by b and b' (1 measure each), and c and c' (half a measure each). Subphrase d (1 measure) ends the phrase.

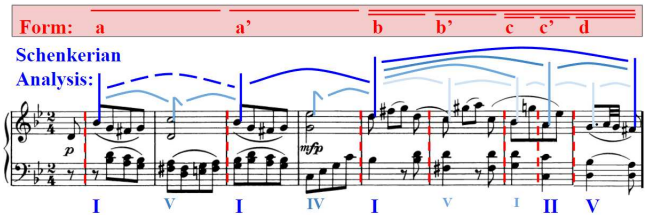

Figure 3: Form and Schenkerian analysis of the first phrase from Beethoven's Piano Sonata Op. 49/1. Form analysis is in red and Schenkerian analysis is in blue.

Laitz [21] describes two main categories of common phrase structures in Western classical music, the *sentence* and the *period*, both of which we use in our experiments. The simplest implementation of a sentence takes the form of $a - a' - b$, where the proportional length of each subphrase is $1:1:2$ and b ends with a cadence, is often divided into a sentence structure itself. In fact, Figure 3 is an example of a sentence within a sentence within a sentence. A period consists of two phrases, an antecedent (A) and a consequent (B or A'), each ending with a cadence, where the first cadence is "weaker" than the second. Cadences ending on an "open" harmony such as V are considered weakest, while "closed" cadences such as those ending on I are stronger. Phrases that make up a period may be sentences themselves or made of less organized subphrases.

3.2 Schenkerian Analysis

Named after its inventor, Heinrich Schenker (1868-1935), Schenkerian Analysis aims to understand the hierarchical harmonic-melodic structure of a piece of music. From a Schenkerian perspective, a piece of music as it is written in the score is the musical *foreground*. Non-harmonic tones such as passing, neighboring, and anticipating tones may be pruned from the foreground to find the next level of structure. This process may be repeated until the background structure (the *Ursatz* or "fundamental structure") is revealed [34, 41]. The upper voice's background structure is known as the *Urlinie* or "fundamental line." Any and all levels of structure between the foreground and background are said to be part of the middleground structure. In Schenkerian terms, the background is said to be "deeper" than the foreground, with levels of the middleground distinguished by their relative depths.

An important concept in Schenkerian analysis is that of *prolongation*. A note is "prolonged" when the note "governs" a section of music at a certain level of depth, even if it is not actually present at all times. An example is shown in the final four measures of Figure 3 in blue. In the G minor melody, D-C-B♭-A-G-F♯, the notes C (m. 6), B♭ (m. 7), A (m. 7), and G (m. 8) may be understood as structurally subsidiary, prolonging the motion from D (m. 5) to F♯ (m. 8); D is considered structurally "in control" until the occurrence of F♯. At a shallower depth (towards the foreground), C and B♭ may be understood as prolonging the motion from D to A, and G prolonging the motion from A to F♯. In Figure 3, longer stems and darker colors represent deeper levels of structure. The bass line is vital to the structure; however, only the melody and harmony are relevant to this paper, so the bass annotations are omitted to simplify the figure.

3.3 Probabilistic Context Free Grammars

A context-free grammar (CFG) $\mathcal{G}$ is defined by four components: (1) $\mathcal{V}$, a finite set of variables known as *nonterminals*, where each nonterminal defines a sub-language of the language defined by $\mathcal{G}$, (2) Σ, a finite set of *terminals*, which exist at the "foreground" of a language, (3) $\mathcal{R}$, the finite set of *production rules*, where a nonterminal may "produce" any number of nonterminals and terminals, and (4) $\mathcal{S}$, the start variable (in $\mathcal{V}$), which represents an entire single realization of the grammar. *Probabilistic* CFGs (PCFGs) simply extend CFGs with the addition of $\mathcal{P}$, a set of probabilities associated with production rules of $\mathcal{R}$.

4 METHODOLOGY

Here we provide an overview of our model along with detailed descriptions of individual components. A high-level diagram of our model is provided in Figure 2. It is designed to imitate the

process that a human composer might take when composing music in a "top-down" approach, thus allowing for *interpretable* music generation. At the top, *phrase structure* is generated independently. By phrase structure, we refer to a series of alphabetic characters that describe a series of subphrases and their relationships (Section 4.1). Meter and hypermeter determine the lengths of measures and subphrases respectively, (Section 4.2.1).

Together with the phrase structure and metrical layout of the piece, the model can determine the *middleground harmonic rhythm* (Section 4.2.2) as shown in Figure 2. Harmonic rhythm describes where changes from one harmony to another occur. The particular *middleground harmonic progression* is then sampled from a Markov chain or PCFG to satisfy the harmonic rhythm. The *middleground melody* is generated via our novel Schenker-inspired PCFG based on the harmonic progression and rhythm (Section 4.4.3). We then determine the *foreground melodic rhythm* (Section 4.2.3). Finally, using the Schenker-inspired PCFG once more, we generate the *foreground melody*, completing the generation process (Section 4.4.4).

4.1 Phrase Structure Generation

Our model samples phrase structure from common forms found in the literature such as the variations of periods and sentences (e.g., ab, $aa'baa''c$, $abac$). We use common phrase structures of Western classical music here, but such structure can naturally be adapted to fit *any* musical style. The sampled phrase structures inform the structure of the generated melody. For our model, we identified 12 phrase structures used in a prior dataset of 41 Schenkerian analyses collected from textbooks and music theory faculty [20]; our model chooses any of these as possibilities, weighted by how often they occur in this repertoire.

4.2 Rhythm Generation

All aspects of rhythm (meter/hypermeter, middleground, and foreground) are sampled from occurrences in the chosen repertoire, again weighted by how often they occur. In other words, meter/hypermeter is chosen from a small set of possibilities, and portions of middleground and foreground rhythms are sampled and combined to generate a new rhythmic framework.

4.2.1 Meter and Hypermeter Generation. Meter is generated independently as any combination of measure subdivision (duple, triple, quadruple) and beat subdivision (simple, compound) to form time signatures ($\frac{4}{4}$, $\frac{3}{4}$, $\frac{6}{8}$, etc.). For instance, Figure 3 is simple duple ($\frac{2}{4}$).

Hypermeter refers to the number of measures for each subphrase or phrase. In Western classical music, sentences ($a - a' - b$) are commonly composed with subphrases of 2, 2, and 4 measures respectively, such as in the main sentence structure of Figure 3. Periods ($A-B$) are often composed as two 8-measure phrases. Our model fits the sampled phrase structure with a proper proportion of measures for each subphrase (1:1:2 for sentences or 1:1 for periods).

4.2.2 Middleground Harmonic Rhythm Generation. Middleground harmonic rhythm is determined based on where middleground harmonic shifts occur. In Figure 3, the harmonic rhythm changes at the beginning of every measure except the penultimate one, which also changes halfway through. Our model examines each phrase and uniformly samples a common harmonic rhythm based on its length from all possibilities. For example, Figure 3 shows middleground harmonic shifts in the darkest blue; harmonic rhythm follows the pattern 4-4-5-1-2 beats in $\frac{2}{4}$.

4.2.3 Foreground Melodic Rhythm Generation. For the foreground melodic rhythm, sections determined by the middleground harmonic rhythm are subdivided into sample rhythms from the repertoire. For instance, many samples from the Beethoven example fill the space of a half note using combinations of notes with shorter durations or a half note itself.

4.3 Harmonic Progression Generation

Once we have the middleground harmonic rhythm, the next step is to fill in the particular harmonic progression. We experiment with two methods to determine this progression: a simple Markov chain and a PCFG of harmonic entities.

Harmonic entities can consist of any notes specified by the user or gathered from a dataset's harmonies. For instance, Western classical or pop style may sample from pitch class sets represented by Roman numerals:

$$\mathcal{H} = \Big\{ I : \{0, 4, 7\},\ ii : \{2, 5, 9\}, \ldots,\ vii^{o} : \{2, 5, 11\} \Big\}.$$

Here, $\mathcal{H}$ is the set of possible harmonies, which are represented as sets of chromatic pitch classes ($0 = C$, $1 = C\sharp/D\flat, \ldots, 11 = B$). Our implementation makes use of the *RomanNumeral* class in the Python package *Music21* [7], which can be straightforwardly manipulated to represent any pitch class set.

4.3.1 Harmonic Markov Chain. A single-order Markov chain makes a strict assumption that one element depends only on the element that comes immediately "before" it. That is,

$$P(h_t \mid h_{t-1}, h_{t-2}, \ldots, h_0) = P(h_t \mid h_{t-1}),$$

where h is a particular harmony in $\mathcal{H}$ and at discrete time step t. Our Markov chain generates harmonies backwards from a goal harmony. This backwards generation produces goal-oriented progressions within the number of allotted harmonic changes. In most Western styles of music, a phrase may begin practically anywhere, but lead towards a limited number of goal harmonies (e.g., I or V).

4.3.2 Harmonic Probabilistic Context-Free Grammar. For our harmonic PCFG we define a CFG with nonterminal variables $\mathcal{V}$ and terminals Σ as follows,

$$\mathcal{V} = \Big\{ f_i : i \in \mathbb{N}_{\geq 2} \Big\}, \quad \Sigma = \mathcal{H}, \tag{1}$$

where $f \in \mathcal{F}$ is a functional category expansion of length i. For instance, in Western classical music, the set $\mathcal{F}$ might include tonic (T), predominant (PD), and dominant (D) functional categories. The start variable $\mathcal{S}$ might lead to a string of variables based on the given harmonic rhythm, such as $T_4 - PD_2 - D_2 - I$, which may then break into the sequence,

$$[I - IV - V - I^6]_T - [IV - ii^6]_{PD} - [V^8 - V^7]_D - I.$$

We see that the PCFG for harmony imposes long-term harmonic structure on the generated music (unlike the Markov chain) and allows for the flexibility to use numerous styles.

4.4 Melody Generation

4.4.1 Contour Probabilistic Context Free Grammar. We define a simple PCFG of contour sequences. Let the set of variables be

$$\mathcal{V} = \left\{ \rightarrow_i^c,\ \nearrow_i^c,\ \searrow_i^c \ :\ i \in \mathbb{N}_{\geq 2} \text{ and } c \in \{nht, \varnothing\} \right\},$$

where each arrow describes the contour (general direction) from one note to another, superscript c indicates whether the contour leads to a *non-harmonic tone* (NHT), and subscript i indicates the number of contours that are put together to make the larger compound contour. That is, a note's pitch can be the same, higher, or lower in relation to another's, and a particular sequence of contours may lead to a broader contour. The set of terminals is defined as

$$\Sigma = \left\{ \rightarrow_1,\ \nearrow_1,\ \searrow_1,\ \rightarrow_1^{nht},\ \nearrow_1^{nht},\ \searrow_1^{nht} \right\}.$$

That is, contours with a subscript of 1 cannot be broken down further into more contours.

To demonstrate, Figure 4 shows three levels of contours from the first two measures of Figure 3. The larger green $\nearrow$ contour is comprised of three smaller orange contours that follow the sequence, $\searrow\rightarrow\nearrow$. The middle orange contour, $\rightarrow$, is further broken down into two smaller red contours that follow the sequence, $\searrow\nearrow$.

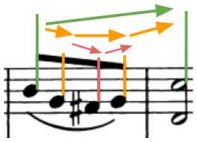

Figure 4: Toy example of the background, middleground, and foreground contours in the first two measures of Beethoven Op. 49/1.

The set of production rules for the excerpt may be

$$\mathcal{R} = \left\{ \ \nearrow_4 \Rightarrow \searrow_1 \rightarrow_2 \nearrow_1,\ \ \rightarrow_2 \Rightarrow \searrow_1^{nht} \nearrow_1 \ \right\}$$

where terminals are arrows with subscript 1. The start variable in this case is $\mathcal{S} = \nearrow_4$. Each production rule is given a probability by the user or based on its frequency in a given dataset. In the trivial example in Figure 4, each production rule in $\mathcal{R}$ has a probability of 1.

4.4.2 Schenkerian Analysis Dataset. We gather most contour data from the Schenker41 dataset [20], which consists of 41 Schenkerian analyses from textbooks and a music theory professor. Other contour data comes from the authors' own analyses. For each prolongation, we extract contour information and keep track of production frequency to be used in our PCFGs. Our data are centered around Western classical music of the common practice era, as that is the style Schenkerian analysis was designed for. However, concepts of Schenkerian analysis can be broadly applied to other genres of music [22, 43].

4.4.3 Middleground Melody Generation. Given the harmonic rhythm, the model generates a middleground note at each harmonic change using a PCFG similar to the one defined in Section 4.4.1. The only difference is that no terminal may be an NHT. The start variable produces a contour from the first note to the last, representing a pseudo-Schenkerian *Urlinie*. Once all contours are determined by the PCFG, the notes are filled in to satisfy the harmonies and *smoothest voice-leading*. Smoothest voice-leading is defined as the smallest intervallic distance between each consecutive note. In other words, the smoothest melody uses the fewest semitones necessary to travel from the first note to the last.

4.4.4 Foreground Melody Generation. Given all other components for the melody (harmony, middleground melody, foreground rhythm, and phrase structure), we produce the foreground melody, which gives the piece its unique character.

Because the foreground rhythm is known, contours derived from our PCFG can be mapped to notes in our melody. Using these contours and a measure for *smoothness* ($\mathbb{S}$), we place melody notes within the harmonic/rhythmic framework. All harmonic tones are handled first. If the next note in the melody is native to the harmony and we set $\mathbb{S} = 0$, the next note will be set as the nearest harmonic tone in the direction of the contour. Greater values of $\mathbb{S}$ offset the next note as shown in Figure 5; higher $\mathbb{S}$ increases the intervallic distance between consecutive notes.

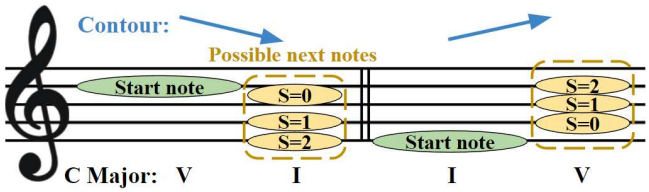

Figure 5: Melodic notes are determined by the contour, harmony, and smoothness measure. The left half shows a V-I progression moving from D5 with $\searrow$ contour. Yellow shows the next possible notes depending on smoothness $\mathbb{S}$. The right half shows a similar action with a rising contour.

On the other hand, if the next note is an NHT, then it is chosen in relation to its two surrounding notes and the prescribed contour from a set of common NHT types,

$$\{N,\ P,\ IN,\ SUS,\ ANT,\ RET\},$$

which stand for neighboring, passing, incomplete neighbor, suspension, anticipation, and retardation respectively.

Neighbor tones may occur as the result of a $\rightarrow_2$ contour. Passing tones occur when notes are a third apart and the contours are $\nearrow_1^{nht}\nearrow_1$ or $\nearrow_1^{nht}\nearrow_1^{nht}$ or their inverses. Incomplete neighbors may occur with any interval and either $\nearrow_1^{nht}\searrow_1$ or $\searrow_1^{nht}\nearrow_1$ contours, suspensions with $\rightarrow_1^{nht}\searrow_1$ contours, anticipations with $\nearrow_1^{nht}\rightarrow$ or $\searrow_1^{nht}\rightarrow$ contours, and retardations with $\rightarrow_1^{nht}\nearrow$ contours.

It should be noted that using contours rather than precise intervals allows the model to be more easily generalized to other styles. However, using contours rather than precise intervals loses some of the structural benefits of Schenkerian analysis, which prescribes that we consider specific notes, and thus, their intervals. If using precise intervals, large amounts of data from a specific genre are required to generate the foreground melody. On the other hand, the foreground of any genre can be generated from very small

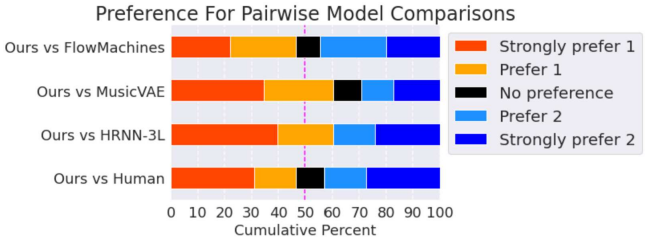

Figure 6: Pairwise preferences in model comparisons. SchenkComposer's melodies (1) are in orange on the left. Other models (2) are on the right in blue.

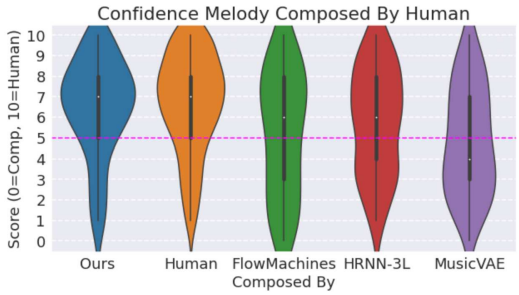

Figure 7: Turing test results (human included as control).

amounts of data by using contours. In the future, we will incorporate a method that searches for a possible foreground realization using precise intervals, but falls back to using contours if precise intervals are not available.

The model generates subphrase variations using a randomly sampled variation technique. One such technique involves copying the original subphrase rhythms and contours, but altering the harmonies. The melody notes are then regenerated to fit the new harmonies. Another technique follows the same process, but with altered contours instead of harmonies. We may also vary aspects such as the final subphrase harmony and contour, the smoothness factor, and foreground rhythm.

5 EXPERIMENTS

To evaluate the performance of our model's generated melodies, we conducted a survey experiment that compared the output of our model against other state-of-the-art melody generation models. We also performed a Turing test to see whether our model's output was convincingly "human." Surveys were conducted anonymously using the Amazon Mechanical Turk platform. An additional ablation study may be found in the Supplemental Materials[1] to determine "effects" and relative importance of each component of SchenkComposer in successfully generating music.

5.1 Survey Design and Evaluation

We compared SchenkComposer with human-composed melodies and melodies from three leading melody generation models, HRNN-3L [49], MusicVAE [37], and Flow Machines [33]. Other methods [8, 28] with no reproducible code or sample recordings were not evaluated. We randomly generated over 20 melodies from each model and uniformly sampled from them to evaluate all pairwise comparisons between SchenkComposer and comparitors. Each excerpt was either 8 or 16 measures and lasted between 16-34 seconds, giving reviewers enough information to make an informed judgement on the quality of the melody. Excerpts were presented in a randomized order as piano MIDI recordings.

For each pair of melodies we asked the following: **1)** On a scale of 0 (not enjoyable) to 10 (very enjoyable), how would you rate melody X? **2)** On a scale of 0 (certain it's by a computer) to 10 (certain it's by a human), what is your degree of belief that a human composed melody X? **3)** Which melody do you prefer? (a) strongly prefer 1, (b) prefer 1, (c) no clear preference, (d) prefer 2, (e) strongly prefer 2. **4)** Were there any parts of melody X that stood out as sounding weird or bad to you? (yes=1, no=0). The full survey instrument, reproducible code, and excerpts are available in the Supplemental Materials.

We compared mean enjoyability for each competing excerpt vs. SchenkComposer using a paired t-test. For the Turing test, we evaluated mean confidence that each excerpt was composed by a human compared to the actual human-composed excerpt using a paired t-test. We calculated exact (Clopper-Pearson) binomial confidence intervals for the proportion of participants that strictly preferred SchenkComposer compared to the competitors; this was doubly conservative both in the type of confidence interval used as well as in excluding "no preference" participants from being counted in favor of SchenkComposer. Finally, we evaluated whether there was a difference in the proportion of respondents that identified a "weird or bad" sounding excerpt for each competing excerpt vs. SchenkComposer using a chi-square test.

Eighty people participated in our study; two were removed for claiming to have studied music but providing nonsensical answers for screening questions (see supplement), resulting in a final analysis dataset of $n = 78$.

5.2 Survey Results

Table 1: Enjoyability (higher is better).

Method	Mean	95% CI	p-value
SchenkComposer	7.11	(6.85, 7.38)	ref.
Human	7.29	(7.04, 7.55)	0.334
HRNN-3L	6.23	(5.63, 6.82)	0.004
MusicVAE	5.42	(4.77, 6.06)	<0.001
FlowMachines	6.22	(5.63, 6.82)	0.008

Table 1 demonstrates *similar mean enjoyability for SchenkComposer compared to human-composed excerpts.* We additionally find sufficient statistical evidence suggesting *greater enjoyability scores for our excerpts compared to the current state-of-the-art automated melody generators.* Of particular note in Figure 7 and Table 2 is that *SchenkComposer successfully passed the Turing test* whereas HRNN-3L, MusicVAE, and Flow Machines did not. This can be seen from the p-values rejecting the hypothesis that the other methods' melodies were composed by a human. Figure 6 and Table 3 suggest

[1] github.com/stephenHahn88/SchenkComposer_Supplement

Table 2: Confidence of being composed by human.

Method	Mean	95% CI	p-value
Human	6.68	(6.37, 7.00)	ref.
SchenkComposer	6.45	(6.14, 6.76)	0.300
HRNN-3L	5.80	(5.20, 6.40)	0.007
MusicVAE	4.84	(4.17, 5.51)	<0.001
FlowMachines	5.22	(4.32, 6.13)	<0.001

Table 3: Proportion of respondents strictly preferring SchenkComposer (higher is better).

Method	Proportion	95% CI
vs. Human	0.47	(0.34, 0.59)
vs. HRNN-3L	0.60	(0.47, 0.72)
vs. MusicVAE	0.60	(0.47, 0.72)
vs. FlowMachines	0.47	(0.33, 0.61)

Table 4: Proportion of excerpts identified as containing "weird or bad" segments (lower is better).

Method	Proportion	95% CI	p-value
SchenkComposer	0.25	(0.18, 0.31)	ref.
Human	0.33	(0.27, 0.40)	0.019
HRNN-3L	0.61	(0.49, 0.74)	<0.001
MusicVAE	0.52	(0.39, 0.64)	<0.001
FlowMachines	0.27	(0.13, 0.40)	0.637

a *general preference for our method compared to the current state-of-the-art*, and demonstrate *non-inferiority to human-composed excerpts and Flow Machines*. Finally, Table 5 suggests that SchenkComposer generally has *significantly lower incidence of "weird or bad" segments compared to the current state-of-the-art.*

6 DEPLOYMENT

We implemented a Web application[2] allowing the public to interact with SchenkComposer and the melody generation process. The application allows the user to follow the flow of the SchenkComposer model, generating a melody in a top-down style (see Figure 2). The website allows the user limited manual access to adjust phrase structure, meter, hypermeter, harmonic rhythm, harmonic transition matrix, and harmonic progression. This section will discuss the technology and design of the website (Section 6.1), and preliminary user feedback and an analysis of user data (Section 6.2).

6.1 Technology and Design

The "deep" structure and flow of the website is depicted in Figure 8. The website follows a typical Client-Server model, where the client interacts with a responsive user interface, and the server safely processes client requests by communicating with the database and machine learning model.

The MongoDB database stores melodies and user information, including survey results. Users may create and login to their accounts

[2]https://melody.cs.duke.edu:8000

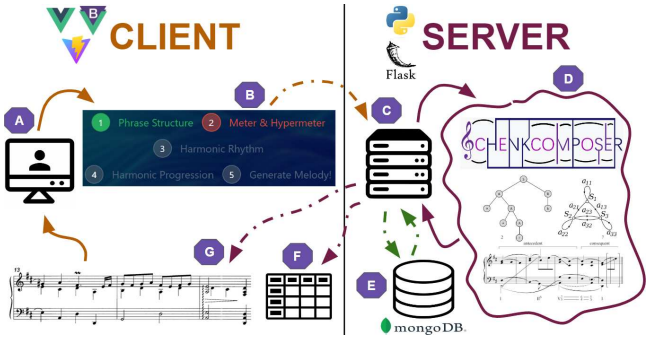

Figure 8: Architecture of the SchenkComposer website. "A" represents a human visiting the public website. "B" is the user interface, which reactively guides the user through the melody generation process. Throughout this process, http requests are sent to "C," representing the Flask server. "C" processes requests by working with the SchenkComposer model and MongoDB, represented by "D" and "E" respectively. The server ("C") responds to the client, providing generated melodies and records of saved data ("G" and "F" respectively). The user ("A") can then process the result of "G" and "F" and update model parameters, repeating the process.

in order to save and return to melodies and model parameters they produced. Melodies and their parameters are viewable in a table in the user's melodies screen (see Figure 13).

The website's front-end server is implemented using the Vite, Vue.js, and BootstrapVue frameworks along with the Vexflow, Graphly D3, and Tone.js libraries. Vexflow is used throughout the application to generate professional-grade music notation in the browser. Graphly D3 makes it simple to generate the interactive Markov chain found in the Harmonic Progression page (see Figure 12). Lastly, Tone.js is used to perform the music from the playback panel (see Figure 10).

The back-end server, written using the Python Flask framework, handles the log-in procedure, saving and loading information to the MongoDB database, and interaction with the SchenkComposer model.

Each step of the melody generation process is explained via an optionally displayed tutorial panel. Music and machine learning concepts are explained and visualized using musical scores and graphs. Tutorial videos and hoverable popup texts help the user find their way through the website.

6.2 Preliminary User Feedback and Analysis of User Data

We designed an informal survey to gather opinions on the current state of the website. The survey consists of four questions or responses: **1)** How would you rate the usability of SchenkComposer, from 0 (very difficult) to 10 (intuitive)? **2)** How likely would you recommend SchenkComposer to a friend or colleague on a scale of 0 (not likely) to 10 (very likely)? **3)** Are there any particular features that you would like to see added to the website? **4)** Please provide additional comments on the website, such as parts you enjoyed or

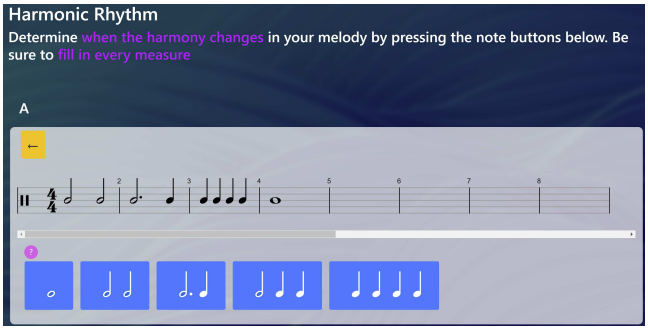

Figure 9: Screenshot of the Middleground Harmonic Rhythm page. Notes are placed in the score (built with Vexflow) measure by measure by pressing the buttons at the bottom.

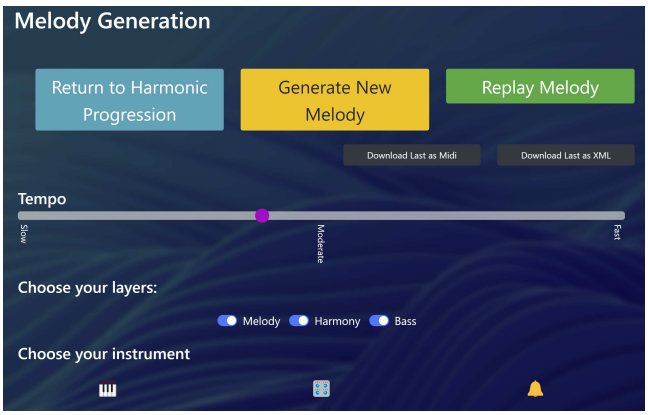

Figure 10: Screenshot of the Melody Generation and playback page. The user can determine their tempo, which layers are playing, and the instrumental ensemble they wish to use. The music is played through the browser using the Tone.js library. The user may also download their melody as a midi file or a musicXML score.

bugs you encountered, here. We received 10 survey responses from a range of users from professional musicians to music enthusiasts.

For usability, the mean score was 8.11 with a standard deviation of 1.79. For likeliness to recommend, the mean score was 9.00 with a standard deviation of 1.33. Several respondents expressed how they enjoyed the process and musicality of the outcome. They also appreciated the tutorial videos that made it very manageable to figure out how to proceed through the website. Users reported no major bugs found, except that the model had trouble generating larger phrases. We have found this occurs with unusual parameter settings and could be resolved with more contour data. Desired features included more rhythm and phrase options, improved educational features for non-musicians, simpler navigation, and use of non-diatonic harmony. We will continue to work towards including and improving these features.

We also included a survey that pops up once a melody is generated. This brief survey, shown in Figure 14, requests the user's reaction to the melody in terms of five emojis ranging from upset

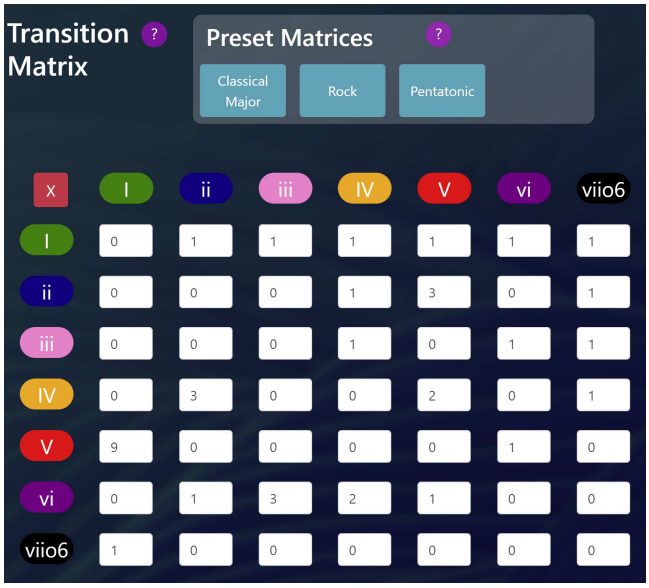

Figure 11: Screenshot of the transition matrix on the Harmonic Progression page. The matrix corresponds to the Markov chain shown in Figure 12. The user may alter the inputs to the matrix with any non-negative real number, directly controlling the way harmonies are produced by SchenkComposer.

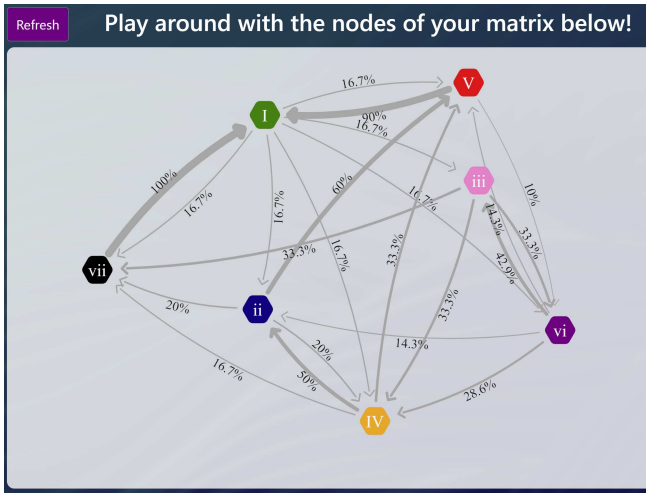

Figure 12: Screenshot of the Markov chain produced fdrom the transition matrix in Figure 11. Harmonic labels act as nodes and links between nodes are weighted with probabilities. Nodes may be dragged around, pulling and pushing other nodes using the Graphly D3 physics.

to joyous. The emoji scores were encoded as 0 (upset) to 4 (joyous). Based on 20 responses, the mean score was 3.4 and the standard deviation, 0.50.

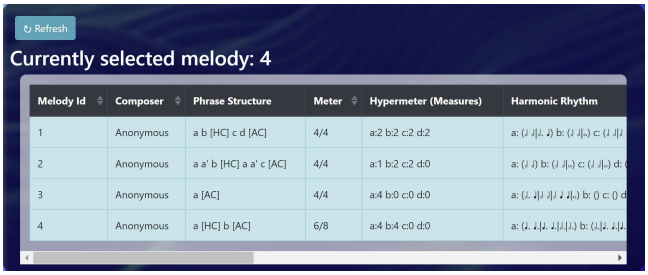

Figure 13: Screenshot of the user's melodies table. The table displays all parameters and outcomes for each particular melody. Clicking on a row will load a particular melody into the system, allowing parameters to be updated.

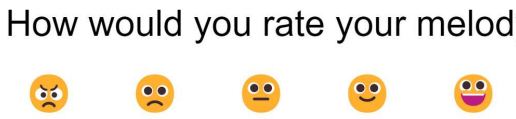

Figure 14: Screenshot of the melody reaction popup survey. Users may choose how they enjoyed their melody by choosing one of five emoji expressions.

7 CONCLUSIONS

Western music generation cannot be complete without structure, harmony, and melody, which is handled gracefully by SchenkComposer. Our experiments show that SchenkComposer produces melodies that are structurally coherent and musically preferable over current state-of-the-art deep learning models, and successfully pass as human-composed in a blinded experiment. SchenkComposer is also vastly less complex, allowing users to fully grasp and manipulate its inner mechanisms, with simple changes to the model parameters allowing for melodies of different genres.

REFERENCES

[1] Gérard Assayag and Shlomo Dubnov. 2004. Using factor oracles for machine improvisation. *Soft Computing* 8, 9 (2004), 604–610.

[2] Bernard Bel and Jim Kippen. 1992. Modelling music with grammars: formal language representation in the Bol Processor.

[3] Wallace Berry. 1966. *Form in Music.* Prentice Hall.

[4] Allen Clayton Cadwallader and David Gagné. 2007. *Analysis of Tonal Music: a Schenkerian Approach.*

[5] William E Caplin. 2001. *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven.* Oxford University Press.

[6] Jialei Chen, Simon Mak, V Roshan Joseph, and Chuck Zhang. 2020. Function-on-function kriging, with applications to three-dimensional printing of aortic tissues. *Technometrics* (2020), 1–12.

[7] Michael Scott Cuthbert and Christopher Ariza. 2010. music21: A toolkit for computer-aided musicology and symbolic music data. *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)* (2010).

[8] Shuqi Dai, Zeyu Jin, Celso Gomes, and Roger B Dannenberg. 2021. Controllable deep melody generation via hierarchical music structure representation. *arXiv preprint arXiv:2109.00663* (2021).

[9] Diana Deutsch and John Feroe. 1981. The internal representation of pitch sequences in tonal music. *Psychological Review* 88 (11 1981), 503–522. https://doi.org/10.1037/0033-295X.88.6.503

[10] Alvaro E Lopez Duarte. 2020. Algorithmic interactive music generation in videogames: A modular design for adaptive automatic music scoring. *SoundEffects-An Interdisciplinary Journal of Sound and Sound Experience* 9, 1 (2020), 38–59.

[11] Marc Evanstein. 2022. Can ChatGPT write a good melody? https://www.youtube.com/watch?app=desktop&v=ogfYRBgzZPU&ab_channel=MarcEvanstein%2Fmusic%E2%80%A4py

[12] Allen Forte and Steven E Gilbert. 1982. *Introduction to Schenkerian analysis.* Norton.

[13] Édouard Gilbert and Darrell Conklin. 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.* Citeseer, 83–94.

[14] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. 2017. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning.* PMLR, 1362–1371.

[15] Stephen Hahn. 2019. Continuous Harmonic Structure in J.S Bach's Triple Fugues in The Well-Tempered Clavier and Art of Fugue. https://digital.library.unt.edu/ark:/67531/metadc1538652/

[16] Daniel Harasim, Martin Rohrmeier, and Timothy J O'Donnell. 2018. A Generalized Parsing Framework for Generative Models of Harmonic Syntax.. In *ISMIR.* 152–159.

[17] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer. https://doi.org/10.48550/ARXIV.1809.04281

[18] Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions. https://doi.org/10.48550/ARXIV.2002.00212

[19] Robert M Keller and David R Morrison. 2007. A grammatical approach to automatic improvisation. In *Proceedings of the Sound and Music Computing Conference.* Citeseer, 330–337.

[20] Phillip B Kirlin. 2014. *A Probabilistic Model of Hierarchical Music Analysis.* University of Massachusetts Amherst.

[21] Steven Geoffrey Laitz. 2012. *The Complete Musician: An Integrated Approach to Tonal Theory, Analysis, and Listening.* Oxford University Press New York.

[22] Steve Larson. 1998. Schenkerian analysis of modern jazz: questions about method. *Music Theory Spectrum* 20, 2 (1998), 209–241.

[23] Fred Lerdahl and Ray S Jackendoff. 1996. *A Generative Theory of Tonal Music.* MIT press.

[24] Björn Lindblom and Johan Sundberg. 1970. *Towards a Generative Theory of Melody.* Department of Phonetics, Institute of Linguistics, University of Stockholm.

[25] Simon Mak, Chih-Li Sung, Xingjian Wang, Shiang-Ting Yeh, Yu-Hung Chang, V Roshan Joseph, Vigor Yang, and C.-F. Jeff Wu. 2018. An efficient surrogate model for emulation and physics extraction of large eddy simulations. *J. Amer. Statist. Assoc.* 113, 524 (2018), 1443–1456.

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

[27] Ryan Martin. 2018. Generative Music with the Living Machine: Using Rule-Based Improvisation to Generate Narrative and Soundtrack. *Critical Studies in Improvisation/Études Critiques en Improvisation* 12, 2 (2018).

[28] Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, and Kevin Webster. 2018. StructureNet: Inducing Structure in Generated Melodies.. In *ISMIR.* 725–731.

[29] Maximilian Muller-Eberstein and Nanne van Noord. 2019. Translating visual art into music. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.* 0–0.

[30] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. 2016. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 276–280.

[31] Steven Nicholls, Stuart Cunningham, and Richard Picking. 2018. Collaborative artificial intelligence in music production. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion.* 1–4.

[32] OpenAI. 2022. ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/. Accessed: 2022-12-20.

[33] François Pachet, Pierre Roy, and Benoit Carré. 2020. Assisted music creation with Flow Machines: towards new categories of new. https://arxiv.org/abs/2006.09232

[34] Thomas Pankhurst. 2008. *SchenkerGUIDE: A Brief Handbook and Website for Schenkerian Analysis.* Routledge.

[35] Gary M Rader. 1974. A method for composing simple traditional music by computer. *Commun. ACM* 17, 11 (1974), 631–638.

[36] Curtis Roads and Paul Wieneke. 1979. Grammars as representations for music. *Computer Music Journal* (1979), 48–55.

[37] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *International Conference on Machine Learning (ICML).* http://proceedings.mlr.press/v80/roberts18a.html

[38] Brayan Rodríguez, Raúl Gutiérrez de Piñérez, and Gerardo M Sarria M. 2017. Using Probabilistic Parsers to Support Salsa Music Composition. In *International Conference on Mathematics and Computation in Music.* Springer, 361–372.

[39] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* 16, none (2022), 1 – 85. https://doi.org/10.1214/21-SS133
[40] Carl Schachter. 1999. *Unfoldings: Essays in Schenkerian theory and analysis.* Oxford University Press on Demand.
[41] Heinrich Schenker. 2001. *Free Composition.* Pendragon Press, Hillsdale, NY. Translated and edited by Ernst Oster.
[42] Arnold Schoenberg, Leonard Stein, and Gerald Strang. 1967. *Fundamentals of Musical Composition.* Faber & Faber London.
[43] Jonathan Stock. 1993. The application of Schenkerian Analysis to Ethnomusicology: problems and possibilities. *Music Analysis* 12, 2 (1993), 215–240.
[44] David Temperley. 2009. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research* 38, 1 (2009), 3–18.
[45] Hiroaki Tsushima, Eita Nakamura, Katsutoshi Itoyama, and Kazuyoshi Yoshii. 2017. Function-and Rhythm-Aware Melody Harmonization Based on Tree-Structured Parsing and Split-Merge Sampling of Chord Sequences.. In *ISMIR.* 502–508.
[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).
[47] Gilbert Wassermann and Mark Glickman. 2020. Automated harmonization of bass lines from Bach chorales: a hybrid approach. *Computer Music Journal* 43, 2-3 (2020), 142–157.
[48] Terry Winograd. 1968. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory* 12, 1 (1968), 2–49.
[49] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. 2019. A hierarchical recurrent neural network for symbolic melody generation. *IEEE Transactions on Cybernetics* 50, 6 (2019), 2749–2757.
[50] Ruihan Yang, Dingsu Wang, Ziyu Wang, Tianyao Chen, Junyan Jiang, and Gus Xia. 2019. Deep music analogy via latent representation disentanglement. *arXiv preprint arXiv:1906.03626* (2019).

A SUPPLEMENTARY MATERIALS OVERVIEW

Complete supplementary materials may be found in the following repository:
https://github.com/stephenHahn88/SchenkComposer_Supplement

In our supplementary materials, we provide our full survey instrument including all musical excerpts, a discussion of our ablation experiments with musical excerpts, and a README file describing the use of SchenkComposer. Here, we provide additional figures and statistics for our survey experiments in Appendix B. Appendix C discusses our experiments producing various musical genres.

B ADDITIONAL SURVEY ANALYSIS

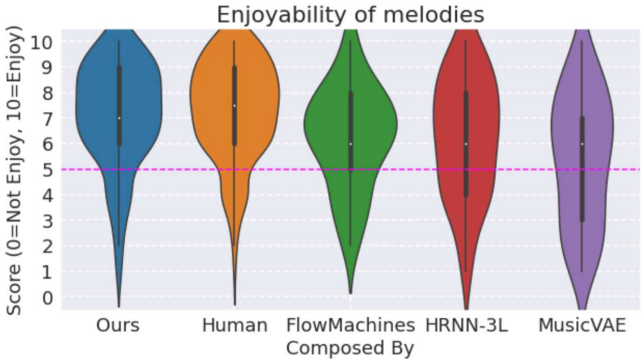

Figure 15: "Enjoyability" of melodies by various models on a scale of 0-10.

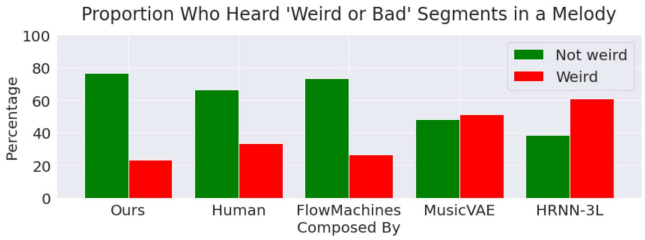

Figure 16: Percentage of participants who heard "weird or bad" segments in melodies by various models.

Question	Low	Medium	High
Years Studied	n=40	27	10
Hours Listened	9	43	25

Question	Nonsense	Wrong	Correct
Meter	13	21	43
Interval	20	38	19

Table 5: Number of participants who answered background and knowledge question. For years studied: low="I have not studied music with a teacher or in school," medium="I have studied music for 5 years or less," high="I have studied music for more than 5 years." For hours listened: low="I listen to music less than 1 hour per week," medium="I listen to music between 1 and 15 hours per week," high="I listen to music more than 15 hours per week."

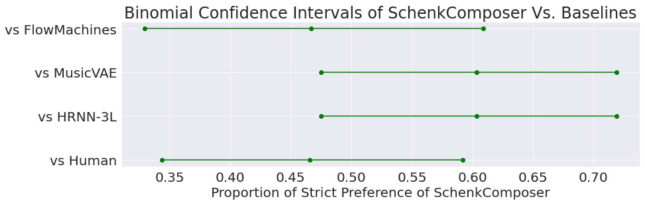

Figure 17: Binomial confidence intervals for strict preference of SchenkComposer over baseline models (higher is better).

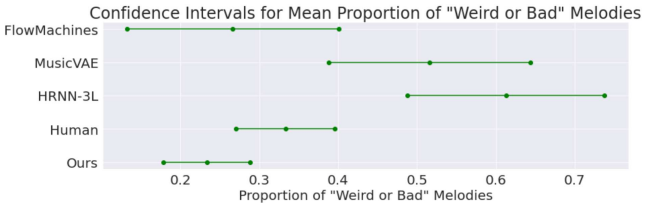

Figure 18: Confidence intervals for "weird or bad" melodies (lower is better).

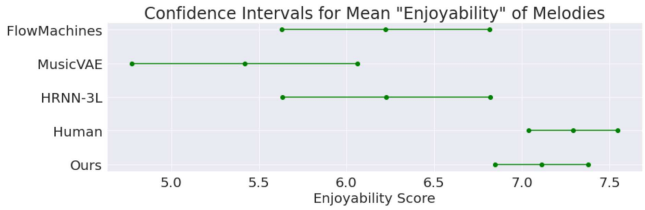

Figure 19: Confidence intervals for enjoyability scores (higher is better).

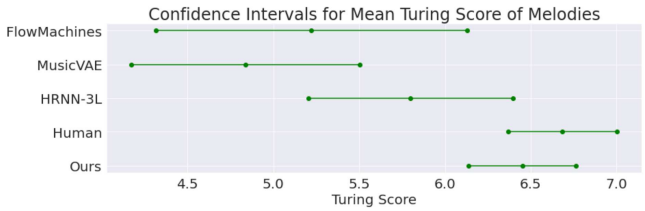

Figure 20: Confidence intervals for Turing scores (higher is better).

C GENRE EXPERIMENTS

For our genre experiments, we attempt to adjust the parameters of SchenkComposer to generate melodies of specific genres. We attempt to generate style-specific melodies without changes to the contour probabilistic context-free grammar. We are also forced to use a Markov chain for harmony generation due to our lack of hierarchical data in other genres. Despite this lack of hierarchical analyses in other genres, we are able to achieve moderate success. Recordings can be found in the Genre experiments folder.

C.1 Rock/Pop

For the rock/pop genre we simply change the harmonic transition matrix, which may be visualized in Figure 21. Additionally, the foreground rhythms are changed to include more syncopated patterns.

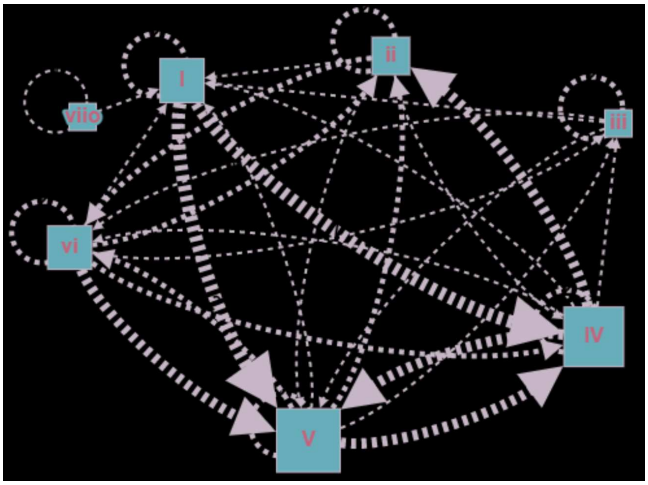

Figure 21: Rock transition graph. Arrows point to harmonies that precede the source harmony. Arrow width indicates the relative probability from one harmony to another.

C.2 Pentatonic

Using the major pentatonic scale, we attempted to generate melodies of the traditional Chinese style. We replace the transition matrix with a simple four-harmony system that only includes notes of the pentatonic scale (see Figure 22).

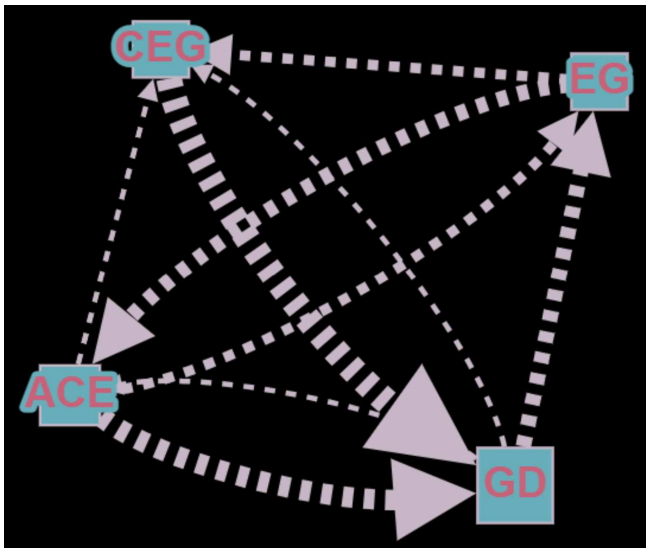

Figure 22: Pentatonic transition graph. Arrows point to harmonies that precede the source harmony. Arrow width indicates the relative probability from one harmony to another.

C.3 Gagaku

Out of sheer curiosity, we attempted to generate melodies in the style of Japanese Gagaku music, although it is far from the authors' expertise. The foreground rhythms were slowed down to last up to multiple measures. We used a four-harmony transition matrix including *Bo*, *Otsu*, *Gyo*, and *Ichi*. Because we are unaware of the transition probabilities, the weights between all harmonies were equal. The melody was based on a subset of the Dorian scale. We based these decisions on the information found in the following website: https://gagaku.stanford.edu/en/.