ChromaFlow: Modeling And Generating Harmonic Progressions With a Transformer And Voicing Encoding

David Dalmazzo, Ken Déguernel, Bob L. T. Sturm

ChromaFlow: Modeling And Generating Harmonic Progressions With a Transformer And Voicing Encoding

David Dalmazzo[1][0000−0002−3262−4091], Ken Déguernel[2][0000−0001−7919−3463], and Bob L. T. Sturm[1][0000−0003−2549−6367]

[1]KTH-Royal Institute of Technology, Stockholm, Sweden
[2]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
{dalmazzo,bobs}@kth.se*, ken.deguernel@cnrs.fr

Abstract. Modeling harmonic progressions in symbolic music is a complex task that requires generating musically coherent and varied chord sequences. In this study, we employ a transformer-based architecture trained on a comprehensive dataset of 48,072 songs, which includes an augmented set of 4,300 original pieces from the *iReal Pro* application transposed across all chromatic keys. We introduce a novel tokenization and voicing encoding strategy designed to enhance the musicality of the generated chord progressions. Our approach not only generates chord progression suggestions but also provides corresponding voicings tailored for instruments such as piano and guitar. To evaluate the effectiveness of our model, we conducted a listening test comparing the harmonic progressions produced by our approach against those from a baseline model. The results indicate that our model generates progressions with more fluid voicings, coherent harmonic motion, and plausible chord suggestions, effectively utilizing repetition and variation to enhance musicality.

Keywords: Music Generation · Chord Progressions · The Transformer Network.

1 Introduction

In the field of music generation using neural network architectures, various approaches have been proposed to address different dimensions of music creation, such as melody generation [17], harmony [4], accompaniment [11], rhythmic section or beat generation [15,9], orchestration [13,1], chord identification [14], and even music recording [5]. However, commercial generative music platforms typically employ machine learning to produce complete compositions that mimic human-created Western pop music. These platforms leverage vast databases from unknown sources, enabling them to surpass human production capabilities in terms of quantity and time efficiency. Consequently, such platforms externalize the agency of music creation [18] by eliminating the need for musical expertise on the user's part.

In contrast, this project aims to develop tools tailored for musicians, thereby preserving their creative agency [7]. Specifically, we focus on modeling harmony by creating an AI assistant designed to outline the structure of potential compositions. This assistant provides suggestions for completing the next consecutive chord or sequences of chord progressions, proposes voicings for piano or guitar arrangements, generates MIDI files based on the suggested chord progressions, and propose a tokenization strategy that includes all harmonic vocabulary available. The dataset and code reference can be found in the GitHub repository `https://github.com/Dazzid/chromaflow`.

2 Related Work

Neural network methodologies for modeling music information are found in the literature; perhaps a pioneer work is *ChordRipple* [10]. It introduced the Chord2Vec encoder, an adaptation of the Word2Vec model, using datasets derived from Bach's chorales and the corpus of *The Rolling Stone's Top 200* songs. Choi et al. [3] used Long Short-Term Memory (LSTM) networks trained on textual representations of chords, such as '`C:7 E:min`'. They trained both a character-level model and a word-level model (meaning `"E:min"` is represented by five tokens in the first model and only by one in the second). Both models were trained on 2,486 musical sequences from a collection scraped from Realbooks[1] and Fakebooks. In 2020, Wu and Yang [19] introduced the *Jazz Transformer*, a generative composition incorporating chords, melodies, and form. This model was trained on the Weimar Jazz Database [16], which includes 456 jazz standards spanning various styles such as Swing, Bebop, Cool, Hard Bop, and Fusion, in MIDI format.

The transformer architecture does not inherently capture the nuances of chord progressions in symbolic music notation. To address this limitation, research published in 2021 has proposed modifications to the transformer's positional embedding layer to better encode musical information. Chen et al. [2] present an encoder-based transformer model specifically designed to classify chords and elucidate harmonic progressions within classical music repertoires. Similarly, Zeng et al. [20] introduce *MusicBERT*, a model leveraging the BERT architecture for symbolic music understanding. *MusicBERT* offers capabilities such as melody completion, accompaniment suggestion, and style classification using the Meta MIDI Dataset (MMD) [8]. Li and Sung [12] developed *MrBert*, an approach for automated music generation that focuses on melody and rhythm prior to chord progressions. *MrBert* utilizes the OpenEWLD dataset and employs dual transformer encoders for melodies and rhythms. It implements a sequence-to-sequence (Seq2Seq) process for chord generation, limiting its vocabulary to triads, which constrains harmonic complexity. Additionally, chord progression modeling using the GPT-2 decoder trained on the *iRealPro* dataset is explored in [6].

[1] `https://en.wikipedia.org/wiki/Real_Book`

3 The Dataset

A dataset of original 4,300 songs from *iReal Pro* across various styles was processed using a custom parser, extracting chords and structural data from musicXML files and metadata (style, time signature, tonality, composer, title). Song information comprises bars (measures), chords (root notes, qualities, extensions, slash chords, duration), and form (repeat bars, sectional jumps, codas). A data augmentation algorithm transposes songs to all chromatic notes, seeking optimal annotation. To address *music21* library's enharmonic errors, a custom transposition method using lines of fifths (LOF) as interval reference was implemented. This method checks each chord relation backward in the LOF, ensuring the tonal center is correctly aligned with its corresponding functional relatives, correcting enharmonic dualities or misrepresentations and fifth-tonic relations. For instance, correcting errors like *'G#7'* resolving to *'Dbmaj7'* by confirming LOF distance and swapping to *'G#7'* to *'C#maj7'* when necessary. As a proof of concept of the process, Figure 1 shows a "*Giant Steps*" fragment.

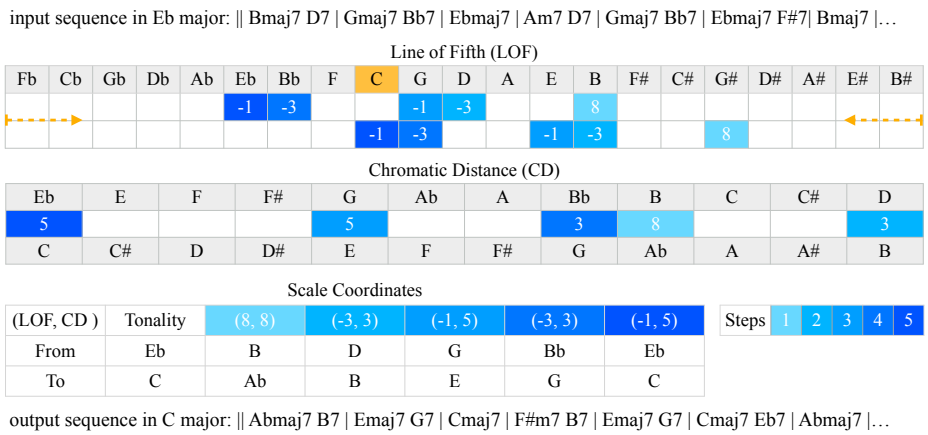

Fig. 1. Visualizing the transposition process. In this case, we transpose Eb major to C major. The example is a fragment of "*Giant Steps*". The time steps of each chord transposed are shown on a blue scale, from light blue to dark blue. A tuple distance coordinate is used to transpose; the first one is the "line of fifth" (LOF) distance, and the second is the distance in the chromatic scale (CD). The LOF is useful when checking tonality relations, e.g., if the distance of a dominant chord is 13 in the LOF, the chord is corrected to its enharmonic distance -1.

The dataset comprises 48,072 songs with a sequence of 1024 elements. A `<pad>` token was added for songs that do not complete the required length sequence. The train and test dataset is subdivided into 90% and 10%, respectively. We first extracted the percentage and then applied data augmentation separately.

4 Models

In this paper, we compare two approaches: Model 1, a previous model [6] that produced static chord blocks when suggesting chord progressions (CP), and Model 2, which includes the new voicing method, tokenization strategy, MIDI exporter, and data preparation.

We propose a strategy to optimize the training process by organizing the chord vocabulary range into its textual representation with well-defined tokens.

The Tokenization: The token sequence comprises a context token (`<style>`), tonality token (`<tonality>`), and chord progression enclosed by `<start>` and `<end>` tokens. Chord units are decomposed into discrete subsections without reducing vocabulary diversity. This is an important aspect of the proposed model, as we do not reduce chord vocabulary. Each chord element starts with a dot ('.'), followed by a **duration** token, **root**, **quality**, and **extensions**. For example, `Cmaj7 add 9` is tokenized as `'.', '4.0', 'C', 'maj7', 'add 9'`. Slash chords like `Cmaj7 / E` become `'.', '4.0', 'C', 'maj7', '/', 'E'`, with the secondary root as the principal reference. This approach results in 125 chord tokens, 22 form tokens, and 51 style tokens, totaling 198 tokens in the vocabulary.

For instance, the quality tokens are defined as follows:
'maj', 'maj6', 'maj7', 'm', 'm6', 'm7', 'm_maj7', 'dom7', 'sus', 'sus2', 'sus7', 'sus4', 'o7', 'o', 'ø7', 'power', 'aug', 'o_maj7'.

Model Architecture: Both Models are based on the architecture of GPT-2.[2] Audio samples of both models can be found in the GitHub repository `https://dazzid.github.io/chromaflow/`

The first difference is in the size of the token embedding, 192 for Model 1 and 256 for Model 2. The token embedding is combined with a learned positional embedding and a MIDI embedding.

The sequence length for Model 1 is 512 and for Model 2 is 1024. Model 2 was trained with 120 Epochs, embedding: 512, heads: 4, layers: 4, batch_size: 128, learning_rate: 3e-5, workers: 4, midi_vocabulary: 128.

Model 2 implements the voicing method, which aims to overcome the chord rigidity observed in the previous version. The voicing is intended as a twofold strategy: infer better musical representations into the positional embedding layer of the transformer network and, from the suggested CP, export MIDI files with the proposed voicing using a plausible configuration commonly used in guitar or piano.

Our method for suggesting chord voicing leverages a structured compilation of dictionaries, each outlining potential voicing dispositions per all chord qualities. For instance, as illustrated in Table 1, the *maj7* quality is represented by semitone intervals distances from the root note. The MIDI mapping configuration starts with root tokens, specified as follows: $C = 48$, $C\# = 49$, $D\flat = 49$,

[2] Our codebase follows Karpathy's implementation: `https://github.com/karpathy/minGPT`

$D = 50$, $D\# = 51$, $E\flat = 51$, $E = 52$, $F\flat = 52$, and continues in this sequence for the rest of the list. Subsequently, each element of the voicing is appended to its corresponding root note.

The Voicing Embedding (VE): The voicing algorithm adapts the distribution of the chord notes, ensuring that each note contributes meaningfully to the harmonic motion. It designates the 3rd and 7th notes of the chord as guide tones while also recognizing the 9th as a pivot extension to smooth transitions.

VE is confined within a two-octave range during both training and generation. In tandem with token inputs, VE provides information pertinent only to the current time step of the training sequence. The MIDI array is defined by a first token clarifying the root; it updates when the chord's quality appears in the sequence with associated notes defined by the voicing method. Subsequently, we extend it with extensions, sequentially enriching the VE with data until the final time-step of the chord tuple, effectively preventing information leakage. The MIDI array has eight values. The longer chord is formed out of seven notes; thus, the eighth value, 127, is reserved for the slash chord as an identifier. The MIDI vocabulary size is, by default, 128.

Voicing	Interval Structure				
v_0	0 (r),	4 (3rd),	7 (5th),	11 (7th)	
v_1	0 (r),	7 (5th),	11 (7th),	16 (3rd)	
v_2	0 (r),	11 (7th),	16 (3rd),	19 (5th)	
v_3	0 (r),	11 (7th),	14 (9th),	16 (3rd)	
v_4	0 (r),	11 (7th),	14 (9th),	16 (3rd),	19 (5th)

Table 1. Major 7th chord voicing in chromatic distances and their alongside note intervals. v_0 is the closed disposition, and the rest are open. The 9th note allows for smooth transitions.

Positional Embedding: Positional embedding is added with input embedding before the transformer encoder intake. This layer empowers the model to discern and learn specific patterns by spatially distributing information in multiple dimensions. VE transposes MIDI data into an embedding tensor, then it is reshaped to match the required architectural format and it is summed with the standard positional encoding layer as shown in Fig. 2. The Embedding Tensor (voicing_emb) is expressed as follows:

$$\text{voicing\_emb} = \mathbf{W} \cdot \text{Emb}(\text{midi\_vocab}, d_m)(m) + \mathbf{b} \tag{1}$$

Emb(midi_vocab, $d_m)(m)$ The embedding function mapping the input MIDI data m to its embeddings in $\mathbb{R}^{B \times S \times d_m}$.
W The weight matrix of the linear layer with dimensions $n_{\text{embd}} \times d_m$.
b The bias vector of the linear layer with dimension n_{embd}.

- **Embedding Layer (`Emb`)**: Transforms MIDI events into dense vectors.
- **Linear Transformation ($\mathbf{W}, \mathbf{b}$)**: Projects the embeddings into a new space with dimensionality n_{embd}.

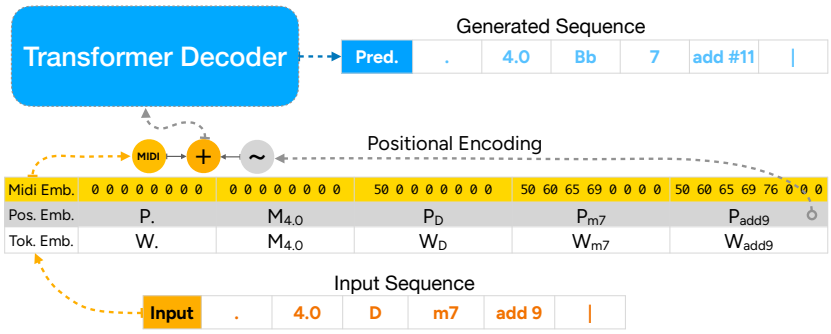

Fig. 2. Representation of the sum of Tokens, Positional, and MIDI embedding in each time-step. The token and position embedding follow the standard GPT-2 static sinusoid embedding, and we add the MIDI positional embedding following the formula explained before (Equation 1). The MIDI information is split per each time-step of the chord formation. After the dot identifier, the chord starts to accumulate information in the MIDI array sequentially until the chord is finished.

The HITS@k metric: We used the top k 1, 3, and 5, as defined in [20], to evaluate the accuracy of predictions in both models. HITS@k measures the fraction of correct answers within the top-k candidates, where k represents the rank threshold. Specifically, for each k (k = 1, 3, 5), HITS@k determines the number of correct predictions that appear within the first k positions of the ranked list of candidates. The calculation is as follows:

$$HITS@k = \frac{1}{n} \sum_{i=1}^{n} I(\text{rank}_i \leq k) \tag{2}$$

- $HITS@k$: Represents the HITS metric evaluated at the k^{th} position.
- n: The total number of instances or items under evaluation.
- $\sum_{i=1}^{n}$: Sum across all evaluated instances.
- $I(rank_i \leq k)$: An indicator function that returns 1 if the rank of the i^{th} instance is less than or equal to k, and 0 otherwise.

5 Results

We now present the findings from our evaluation using standard natural language processing metrics alongside the results from a listening questionnaire designed for human evaluation. The following Table 2 compares the HITS@K reports of Models 1 and 2.

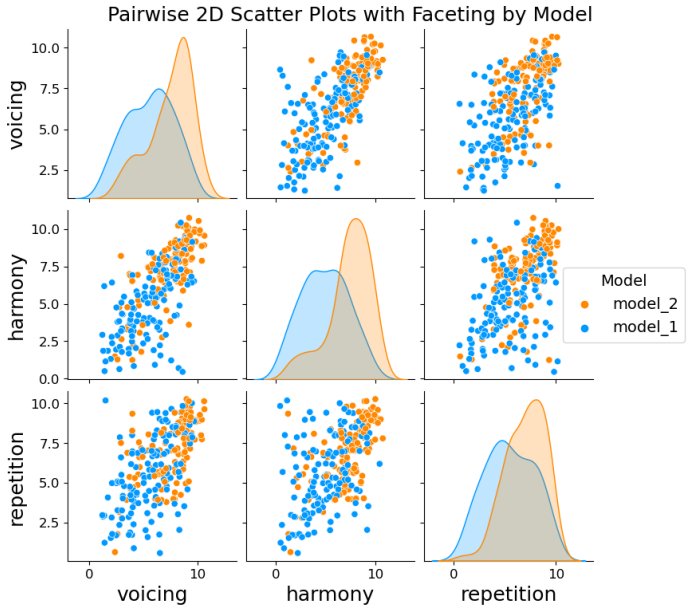

Fig. 3. Pairwise 2D scatter plot of participants' answers for 'Voicing', 'Harmony', and 'Repetition' questionnaire.

Table 2. Metrics

	Train Loss	Val. Loss	HITS@1	HITS@3	HITS@5
Model 1	0.04982	0.03645	0.9138	0.9707	0.984
Model 2	0.1612	0.2428	0.9738	0.9946	0.9967

We performed a listening test with 25 responses. The average age of the users is 40.88 ± 14.96. We asked participants to provide their ages and to rate their expertise in understanding harmony on a scale from 1 to 5 and performed a check by asking them to identify a four-chord functional progression from a randomized audio sample, with five different chord progressions: a) ii, V, I, vi, b) I, IV, V, vi, c) vi, ii, V, I, d) iii, IV, V, IV e) IV, V, VI, ii.

The listening test comprised 10 examples, with five CP generated by Model 1 and five by Model 2. Asking the participants to rate three different aspects on a scale from 1 to 10: **Voice leading**: *How cohesive and fluid do you find the voicing?* **Harmony**: *How coherent do you find the harmonic motion?* **Repetition and variation**: *To what degree do you find the progressions feature repetition and variation?* The reported evaluation is shown in Table 3:

The Pairwise Figure (Fig. 3) shows the distribution of answers comparing both models. Evaluation from a previous publication [6] approximates the same results by the scores obtained in Model 1.

Table 3. Statistical Measures

	Model 1		Model 2	
	Median	Std	Median	Std
Voicing	5.723	2.184	**8.058**	2.030
Harmony	5.063	2.336	**7.868**	2.174
Repetition	5.347	2.386	**7.300**	1.892

6 Discussion and Conclusion

Regarding differences in reported metrics in Table 2, Model 1 notifies a training and validation loss considerably smaller than those reported by Model 2. The main argument supporting this phenomenon is that Model 1 contains less information in their chord sequences. This issue was resolved in Model 2 by expanding the form. This modification has enriched the samples with a more precise structure, reducing the occurrence of padding tokens.

Model 2 generated improved CP, now incorporating form sections. When provided with a predefined chord progression sequence, Model 2 utilizes it as recursive material to generate new progressions. HITS@K metrics for Model 2 also indicated superior performance in predicting 'K' elements within the tree of ranked candidates.

As highlighted in [19] and evidenced by the outputs of Model 1, the GPT-2 decoder proves inadequate for encoding music information through symbolic notation alone. It can predict a coherent test sequence, but musical quality remains ambiguous. Given that voicing is not explicitly represented in symbols, it relies on the musician's expertise; there is still room for several modifications to explore and test better dataset representations, perhaps adding an encoder-decoder translator from symbolic to expert voicing configurations. Future modifications could explore enhanced dataset representations, incorporating an encoder-decoder translator from symbolic to expert voicing configurations alongside architectural refinements. For the next iteration, we aim to develop a web application or *Ableton Live* extension with an interface featuring form and structure visualizations, enhancing user comprehension of the proposed format.

7 Acknowledgments

This work was supported by the European Research Council under the European Union's Horizon 2020 research and innovation programme (MUSAiC project, Grant agreement No. 864189).

References

1. Cella, C.E.: Orchidea: a comprehensive framework for target-based computer-assisted dynamic orchestration. Journal of New Music Research **51**(1), 40–68 (2022)

2. Chen, T.P., Su, L.: Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models. Transactions of the Int. Society for Music Information Retrieval **4**(1) (2021)
3. Choi, K., Fazekas, G., Sandler, M.: Text-based lstm networks for automatic music composition. In: Proc. 1st Conf. on Computer Simulation of Musical Creativity. Huddersfield, UK (2016)
4. Choi, K., Fazekas, G., Sandler, M.: Text-based lstm networks for automatic music composition. arXiv preprint arXiv:1604.05358 (2016)
5. Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., Défossez, A.: Simple and controllable music generation. Advances in Neural Information Processing Systems **36** (2024)
6. Dalmazzo, D., Déguernel, K., Sturm, B.L.: The chordinator: Modeling music harmony by implementing transformer networks and token strategies. In: Evo-MUSART (2024)
7. Déguernel, K., Giraud, M., Groult, R., Gulluni, S.: Personalizing ai for co-creative music composition from melody to structure. In: Sound and Music Computing (SMC 2022). pp. 314–321 (2022)
8. Ens, J., Pasquier, P.: Metamidi dataset (Jul 2021). `https://doi.org/10.5281/zenodo.5142664`, `https://doi.org/10.5281/zenodo.5142664`
9. Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K.V., Joulin, A., Misra, I.: Imagebind: One embedding space to bind them all. In: Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition. pp. 15180–15190 (2023)
10. Huang, C.Z.A., Duvenaud, D., Gajos, K.Z.: Chordripple: Recommending chords to help novice composers go beyond the ordinary. In: Proc. of the 21st Int. Conf. on intelligent user interfaces. pp. 241–250 (2016)
11. Huang, Y.S., Yang, Y.H.: Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In: Proc. of the 28th ACM Int. Conf. on Multimedia. pp. 1180–1188 (2020)
12. Li, S., Sung, Y.: Transformer-based seq2seq model for chord progression generation. Mathematics **11**(5), 1111 (2023)
13. Lousseief, E., Sturm, B.: Mahlernet: Unbounded orchestral music with neural networks. In: the Nordic Sound and Music Computing Conference 2019 and the Interactive Sonification Workshop 2019. pp. 57–63 (2019)
14. Miller, J., Pauwels, J., Sandler, M., et al.: Polar manhattan displacement: measuring tonal distances between chords based on intervallic content (2023)
15. Nuttall, T., Haki, B., Jorda, S.: Transformer neural networks for automated rhythm generation (2021)
16. Pfleiderer, M., Frieler, K., Abeßer, J., Zaddach, W.G., Burkhart, B. (eds.): Inside the Jazzomat - New Perspectives for Jazz Research. Schott Campus (2017)
17. Sturm, B., Santos, J.F., Korshunova, I.: Folk music style modelling by recurrent neural networks with long short term memory units. In: 16th Int. society for music information retrieval Conf. (2015)
18. Wiggins, J.: Musical agency. The child as musician: A handbook of musical development pp. 102–121 (2016)
19. Wu, S.L., Yang, Y.H.: The jazz transformer on the front-line: Exploring the shortcomings of ai-composed music through quantitative measures. arXiv preprint arXiv:2008.01307 (2020)
20. Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., Liu, T.Y.: Musicbert: Symbolic music understanding with large-scale pre-training. arXiv preprint arXiv:2106.05630 (2021)