Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Special Track on AI, the Arts and Creativity

Graph-based Polyphonic Multitrack Music Generation

Emanuele Cosenza , **Andrea Valenti** and **Davide Bacciu**
University of Pisa
e.cosenza3@studenti.unipi.it, andrea.valenti@phd.unipi.it, davide.bacciu@unipi.it

Abstract

Graphs can be leveraged to model polyphonic multitrack symbolic music, where notes, chords and entire sections may be linked at different levels of the musical hierarchy by tonal and rhythmic relationships. Nonetheless, there is a lack of works that consider graph representations in the context of deep learning systems for music generation. This paper bridges this gap by introducing a novel graph representation for music and a deep Variational Autoencoder that generates the structure and the content of musical graphs separately, one after the other, with a hierarchical architecture that matches the structural priors of music. By separating the structure and content of musical graphs, it is possible to condition generation by specifying which instruments are played at certain times. This opens the door to a new form of human-computer interaction in the context of music co-creation. After training the model on existing MIDI datasets, the experiments show that the model is able to generate appealing short and long musical sequences and to realistically interpolate between them, producing music that is tonally and rhythmically consistent. Finally, the visualization of the embeddings shows that the model is able to organize its latent space in accordance with known musical concepts.

1 Introduction

The automatic generation of artistic artifacts is gathering increasing interest, also thanks to the possibilities offered by modern deep generative models. The visual arts of painting and photography [Ramesh *et al.*, 2022], the written expressions of prose and poetry [Brown *et al.*, 2020], and intricate art forms such as music [Agostinelli *et al.*, 2023] are all domains where neural models can be leveraged to produce realistic, if not artistically appealing, artifacts.

Despite these achievements, a closer inspection is often enough to detect whether a piece of art is the outcome of an automatic artificial process or not. While being very good at approximating the external appearance of the artworks, artificial models still lack a way to convey an artistic message to the overall experience. This results in artworks that are convincing but soulless, lacking a general coherence and a deeper meaning. This is particularly true in the case of music, where the artist needs to be very aware of the emotions evoked by a particular sequence of notes in order to stimulate a specific mood in the listener.

A way to circumvent the above issues is to look at deep learning models as a powerful support to the human artist, instead of as a replacement. The models can thus be used as a way to automatize the low-level routine sub-tasks of the creative process, while leaving the artist free to concentrate on the overall picture. Thus, the neural network becomes an extremely versatile tool in the hand of the artist, which should be able to control and shape the output of the network in any way they see appropriate.

In this paper, we introduce a new model for the automatic generation of symbolic sequences of multitrack, polyphonic music. The generation process is carried out through the use of a novel graph-based internal representation, which allows to explicitly model the different chords in the song and the relations between them. This representation allows the human artist to perform controlled changes to the output of the neural network in order to control specific aspects of the artistic performance, while leaving the model free to generate the remaining part in a coherent way.

The main contributions of this paper are the following:

1. We propose a novel graph representation of multitrack, polyphonic music, where nodes represent the chords played by different instruments and edges model the relationships between them.

2. We introduce a deep Variational Autoencoder [Kingma and Welling, 2013] that generates musical graphs by separating their rhythmic structure and tonal content. To the best of our knowledge, this is the first time in literature that Deep Graph Networks [Bacciu *et al.*, 2020] are used to generate multitrack, polyphonic music.

3. We show a new generative scenario enabled by our approach in which the user can intuitively condition generation by specifying which instruments have to be played at specific timesteps.

2 Related Works

In recent years, there have been many attempts at generating symbolic music with deep learning. Various works have focused on sequential models such as LSTMs [Chu *et al.*, 2016; Brunner *et al.*, 2017; Roberts *et al.*, 2018] and, more recently, Transformers [Huang *et al.*, 2018; Valenti *et al.*, 2021]. When considering specific representations (e.g. pianorolls), music can also be processed by convolutional networks, with associated applications to music generation [Chuan and Herremans, 2018; Huang *et al.*, 2019].

Variational Autoencoders (VAE) [Kingma and Welling, 2013] and Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014] have emerged as plausible candidates for symbolic music generation. MidiNet [Yang *et al.*, 2017], C-rnn-gan [Mogren, 2016] and MuseGAN [Dong *et al.*, 2018] are all models in which a convolutional or recurrent generator produces music from a random sample, and a discriminator is trained to distinguish generated samples from real ones. For what concerns VAEs, an early approach to music generation is Midi-VAE [Brunner *et al.*, 2018], where separate GRU encoder/decoder pairs are used for pitch, instrument and velocity, while sharing the same latent space. In [Roberts *et al.*, 2018], instead, a high-level conductor LSTM takes the latent code generated by an encoder and produces latent variables corresponding to different segments of music. These are then processed by a lower-level decoder LSTM, which focuses on the generation of smaller subsections one note at a time. The same hierarchical approach is followed by PianoTree VAE [Wang *et al.*, 2020], which uses multiple GRUs to compute bar decodings from a latent code representing the entire piece, and chord decodings from bar decodings. The authors also exploit note-chord hierarchy priors, computing chord embeddings from note embeddings. An interesting middle ground between VAEs and GANs is represented by Adversarial Autoencoders (AAE) [Makhzani *et al.*, 2015], which have been used in the context of music generation to impose arbitrary priors to latent variables [Valenti *et al.*, 2020; Valenti *et al.*, 2021].

A challenge in devising symbolic generators is choosing an appropriate representation for music data. Researchers have therefore started to experiment with graph-based representations, where musical entities and their relationships are modeled, respectively, by nodes and edges. Musical graphs have been built at the note level [Liu *et al.*, 2010; Ferretti, 2018; Ferretti, 2017], associating nodes to notes and edges to temporal or tonal relationships, as well as at a higher level of the hierarchy, using melodic segments [Simonetta *et al.*, 2018] and bars [Wu *et al.*, 2020; Zou *et al.*, 2021] as building blocks.

In the literature, there is a substantial lack of studies that consider graph representations in the context of deep learning for symbolic music. The VAE-based performance renderer in [Wu *et al.*, 2020] and the cadence detector in [Karystinaios and Widmer, 2022] are, to the best of our knowledge, the only systems that use Deep Graph Networks to process musical graphs. In both works, graphs are constructed at the note level and edges represent both tonal and temporal relationships between musical entities. For what concerns generation, the only attempts at using graphs with deep learning are represented by PopMNet [Wu *et al.*, 2020] and MELONS [Zou *et al.*, 2021]. Both works use GANs and recurrent networks, enforcing the typical structure of human music through a bar-level graph representation. These graphs are used to condition the generation of monophonic music, which is carried out by the recurrent networks. In contrast to these works, our approach uses graphs at a lower level, leveraging Deep Graph Networks to automatically learn meaningful tonal and rhythmic concepts in the context of polyphonic multitrack music generation.

3 Graph-based Music Generation

The proposed model processes polyphonic, multitrack music. Input songs are assumed to be available as an $N \times I \times T \times P$ multitrack pianoroll binary tensor, where N is the number of bars, I the number of tracks, T the number of timesteps in a bar and P the number of possible pitches. An example of a multitrack pianoroll is shown in Figure 1a. The number of timesteps in a bar, T, is fixed to 32, which allows to represent notes with rhythmic value $1/32$. A note is defined by its pitch and duration values. Songs are assumed to contain a set of tracks played by non-percussive instruments together with a drum/percussion track (possibly silenced). This can be easily enforced during the preprocessing phase.

3.1 Graph-based Music Representation

We propose to represent polyphonic multitrack music by a *chord-level graph* $g = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of (multi-type) edges, $\mathcal{A}$ the set of edge features and $\mathcal{X}$ the set of node features. An example of a chord-level graph is shown in Figure 1c.

The *structure* $\mathcal{S}$ of g is represented by the sets $\mathcal{V}$, $\mathcal{A}$ and $\mathcal{E}$. Each node $v \in \mathcal{V}$ corresponds to the activation of a chord in a specific track and timestep. We identify three types of edges $(u, v) \in \mathcal{E}$: *track* edges, *onset* edges and *next* edges. Track edges connect nodes that represent consecutive activations of a single track. Onset edges connect nodes that represent simultaneous activations of different tracks. Finally, next edges connect nodes that represent consecutive activations of different tracks in different timesteps. In order to model different tracks, a separate track edge type is instantiated for each track. Track edges model *intra-track* relationships since they only connect nodes belonging to a single track. On the other hand, onset and next edges model *inter-track* relationships since they connect nodes related to different tracks. Each edge feature $a_{uv} \in \mathcal{A}$ contains the type of the edge (u, v) as well as the distance in timesteps between the two nodes.

The *content* $\mathcal{C}$ of g is represented by the set of node features $\mathcal{X}$. Node features $x_v \in \mathcal{X}$ contain the list of notes played in correspondence of node v. The number of maximum notes in a chord, Σ, is fixed a priori. Each note is represented as a feature vector of dimension D. The vector contains information about pitch and duration stored as a one-hot token pair. The pitch token can assume 131 different values, which correspond to 128 MIDI pitches with the addition of $\texttt{SOS}_P$, $\texttt{EOS}_P$ and $\texttt{PAD}_P$ tokens. Similarly, the duration token can assume 99 different values, which correspond to 96 different

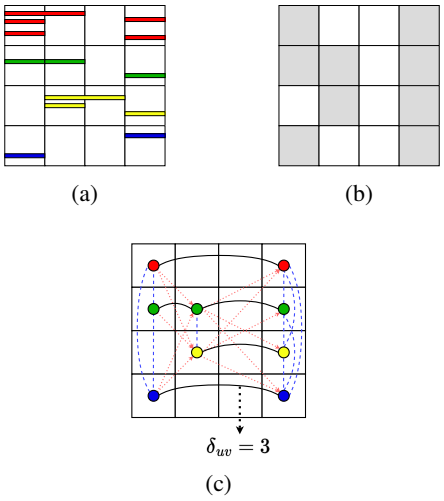


Figure 1: (a) Illustration of a single bar of a multitrack pianoroll with four tracks (rows) and four timesteps (columns). Colored rectangles in the grid represent the notes being played in the sequence, while their position inside each cell indicates their pitch. (b) A structure tensor computed from the pianoroll. (c) The resulting chord level graph. In the image, black, solid connections indicate track edges. Red, dotted connections indicate next edges. Blue, dashed connections indicate onset edges. Edge features δ_{uv} indicate the distance in timesteps between two nodes. The content of the graph is omitted for simplicity.

durations (yielding a maximum duration of 3 bars) with the addition of $\texttt{SOS}_D$, $\texttt{EOS}_D$ and $\texttt{PAD}_D$ tokens.

The structure of g is encoded by the tensor $\mathbf{S} \in \{0,1\}^{N\times I\times T}$, where $\mathbf{S}_{n,i,t} = 1$ if and only if there is an activation of at least one note in the track i at timestep t of the n-th bar. Intuitively, $\mathbf{S}_{n,i,t}$ indicates whether track i is active (not counting the sustain of notes) at timestep t in the n-th bar. An example of a structure tensor is shown in Figure 1b. The content of a chord-level graph, on the other hand, can be encoded through a tensor $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times\Sigma\times D}$ after fixing an ordering of $\mathcal{V}$.

3.2 Deep Graph Network for Music

Our graph-based representation of music is processed by a deep VAE [Kingma and Welling, 2013] that reconstructs the structure $\mathcal{S}$ and the content $\mathcal{C}$ of a chord-level graph $g = (\mathcal{S}, \mathcal{C})$. Its encoder models the encoding distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{S},\mathcal{C})$, where $\boldsymbol{z} \in \mathbb{R}^d$. The decoder network, on the other hand, models $p_{\boldsymbol{\theta}}(\mathcal{S},\mathcal{C}|\boldsymbol{z})$. After introducing the latent variables $\boldsymbol{z}_{\mathcal{S}} \in \mathbb{R}^d$ and $\boldsymbol{z}_{\mathcal{C}} \in \mathbb{R}^d$, the generative process can be formalized as follows:

$$p_{\boldsymbol{\theta}}(\mathcal{S},\mathcal{C},\boldsymbol{z}_{\mathcal{S}},\boldsymbol{z}_{\mathcal{C}}|\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\mathcal{S}}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\mathcal{C}}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\mathcal{S}|\boldsymbol{z}_{\mathcal{S}})p_{\boldsymbol{\theta}}(\mathcal{C}|\boldsymbol{z}_{\mathcal{C}},\mathcal{S}) \quad (1)$$

A high-level representation of the model is shown in Figure 2. The encoder consists of two separate submodules, namely a *content encoder* and a *structure encoder* which output, respectively, the codes $\boldsymbol{z}_{\mathcal{S}}$ and $\boldsymbol{z}_{\mathcal{C}}$. The two codes are finally combined into a graph code $\boldsymbol{z}_g$ with a linear layer. The decoder, on the other hand, generates the structure $\mathcal{S}$ and the content $\mathcal{C}$ of g one after the other. First, symmetrically to the encoder, it decomposes $\boldsymbol{z}$ into two separate latent vectors $\boldsymbol{z}_{\mathcal{S}}$ and $\boldsymbol{z}_{\mathcal{C}}$ through a linear layer. Then, it generates $\mathcal{S}$ from $\boldsymbol{z}_{\mathcal{S}}$ through a *structure decoder* and the content $\mathcal{C}$ from $\mathcal{S}$ and $\boldsymbol{z}_{\mathcal{C}}$ through a deep graph *content decoder*. The content and the structure decoder model, respectively, the distributions $p_{\boldsymbol{\theta}}(\mathcal{S}|\boldsymbol{z})$ and $p_{\boldsymbol{\theta}}(\mathcal{C}|\mathcal{S},\boldsymbol{z})$.

Content Encoder. The content encoder (Figure 3a) develops progressively higher-level representations for notes, chords, bars and the whole piece. This module first embeds each note in a d-dimensional space with a note encoder, which uses separate embedding matrices for pitches and durations. Next, a chord encoder processes the list of notes associated to each node, producing d-dimensional chord representations. In our instantiation of the model, the chord encoder is implemented as a linear layer that takes a concatenation of the Σ note representations and yields a final chord embedding. These chord representations are the initial node states $\boldsymbol{h}_v^0$ of an encoder Graph Convolutional Network (GCN) [Bacciu *et al.*, 2020] with L layers. Combining the techniques employed in [Schlichtkrull *et al.*, 2018; Simonovsky and Komodakis, 2017; Gilmer *et al.*, 2017], the GCN constructs new node states by taking into account the discrete information regarding both the edge types and the distances between nodes. Residual connections are used between consecutive layers in the GCN. This has proven to be beneficial in mitigating oversmoothing problems with large values of L [Li *et al.*, 2018; Li *et al.*, 2019]. Batch Normalization [Ioffe and Szegedy, 2015] is used after each graph convolutional layer to speed up convergence and improve the generalization capability of the model. We refer the reader to the supplementary material[1] for details about the implementation of the GCN. After L graph convolutional layers, a readout layer aggregates the information contained in each subgraph g^n of g related to the n-th bar of the musical sequence. This layer, which resembles the ones in [Jeong *et al.*, 2019] and [Li *et al.*, 2015], produces bar embeddings $\boldsymbol{z}_{\mathcal{C}}^1, \ldots, \boldsymbol{z}_{\mathcal{C}}^N$ using a soft attention layer, which is in charge of learning the importance of single track activations. The N bar embeddings $\boldsymbol{z}_{\mathcal{C}}^1, \ldots, \boldsymbol{z}_{\mathcal{C}}^N$ are concatenated and passed through a bar compressor, which is implemented as a linear layer, to obtain the final content representation $\boldsymbol{z}_{\mathcal{C}}$.

Structure Encoder. The structure encoder (Figure 3b) takes as input the structure tensor $\mathbf{S} \in \mathbb{R}^{N\times I\times T}$ and computes the code $\boldsymbol{z}_{\mathcal{S}}$. This module first encodes each bar $\mathbf{S}_n \in \mathbb{R}^{I\times T}$ into a latent representation $\boldsymbol{z}_{\mathcal{S}}^n \in \mathbb{R}^d$ through a CNN [Goodfellow *et al.*, 2016] made of two convolutional

[1] https://emanuelecosenza.github.io/polyphemus/assets/suppmaterials.pdf

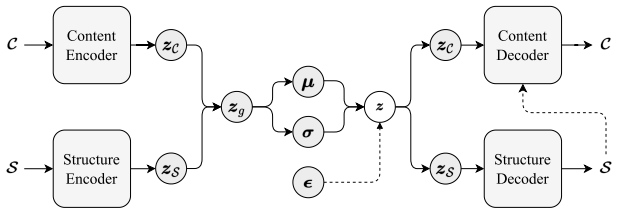


Figure 2: High-level visualization of the model.

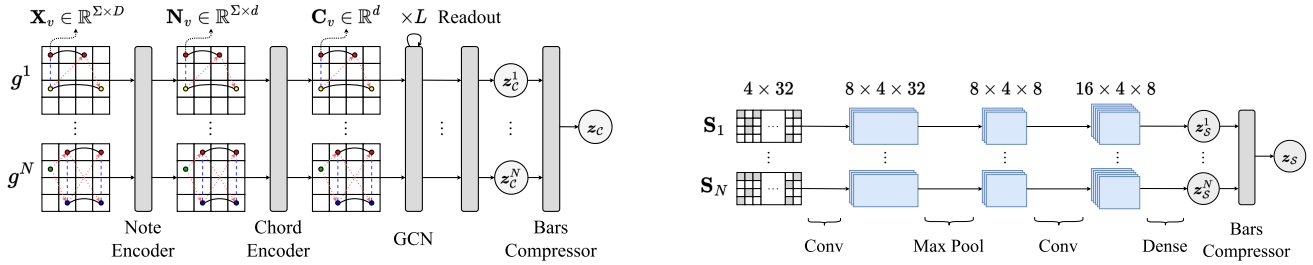


Figure 3: Visualization of the content encoder (a) and the structure encoder (b).

layers with ReLU activations and Batch Normalization, interleaved by max pooling. The bar representations $\boldsymbol{z}_{\mathcal{S}}^1, \ldots, \boldsymbol{z}_{\mathcal{S}}^N$ are then computed by passing the signal through two dense layers. These representations are finally concatenated and passed through a linear layer to obtain $\boldsymbol{z}_{\mathcal{S}}$.

Structure Decoder. The structure decoder (see Figure 4b) is specular to the structure encoder. It first decompresses $\boldsymbol{z}_{\mathcal{S}}$ into N structure bar representations $\boldsymbol{z}_{\mathcal{S}}^1, \ldots, \boldsymbol{z}_{\mathcal{S}}^N$ and decodes each of them into a structure tensor $\mathbf{S}_n \in \mathbb{R}^{I \times T}$ with a bar decoder. The bar decoder mirrors the bar encoder, with the difference that upsample layers are interleaved with convolutional layers to obtain the original resolution of the pianoroll. Finally, a sigmoid layer produces probability values which are stacked to form the probabilistic structure tensor $\tilde{\mathbf{S}}$.

Content Decoder. It reconstructs the content of g from $\boldsymbol{z}_{\mathcal{C}}$ and $\mathcal{S}$. The decoder first decompresses $\boldsymbol{z}_{\mathcal{C}}$ into $\boldsymbol{z}_{\mathcal{C}}^1, \ldots, \boldsymbol{z}_{\mathcal{C}}^N \in \mathbb{R}^d$. Each $\boldsymbol{z}_{\mathcal{C}}^n$ is used to initialize the states of the nodes in the subgraph g^n, which represents the connected component related to the n-th bar of the structure $\mathcal{S}$. From there, a GCN identical to the one employed in the encoder computes the final states $\mathbf{h}_v^L \in \mathbb{R}^d$ for each node v. At this point, a (linear) chord decoder transforms each final node state $\mathbf{h}_v^L$ into the corresponding Σ note representations of dimension d. Such note representation vectors are split into two halves: each half is transformed by a pitch and a duration decoder, respectively, into pitch and duration information. As in the encoder, two separate pitch embedding matrices are used for drum and non-drum pitches. Finally, a softmax layer outputs two separate probability distributions over pitches and durations, yielding the probabilistic tensors $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{D}}$, which contain, respectively, pitch and duration probabilities.

3.3 Training

The model is trained to minimize the following loss

$$\mathcal{L}(g) = \mathbb{E}[-\log p_{\boldsymbol{\theta}}(g|\boldsymbol{z})] + \beta D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|g) \,||\, \mathcal{N}(\mathbf{0}, \boldsymbol{I})), \quad (2)$$

where $D_{KL}(\cdot||\cdot)$ is the KL divergence and the expectation is taken with respect to $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Following the β-VAE framework [Higgins *et al.*, 2016], the hyperparameter β controls the trade-off between reconstruction accuracy and latent space regularization.

Since the generative process is divided in two parts, the log-likelihood term in Equation 2 can be decomposed as follows:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(g|\boldsymbol{z}) &= \log \left(p_{\boldsymbol{\theta}}(\mathcal{S}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\mathcal{C}|\boldsymbol{z}, \mathcal{S})\right) \\ &= \log p_{\boldsymbol{\theta}}(\mathcal{S}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\mathcal{C}|\boldsymbol{z}, \mathcal{S}). \end{aligned} \quad (3)$$

The first term in Equation 3 can be derived in the following way:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathcal{S}|\boldsymbol{z}) = \sum_{n,i,t} \mathbf{S}_{n,i,t} \log \tilde{\mathbf{S}}_{n,i,t} + \\ + (1 - \mathbf{S}_{n,i,t}) \log(1 - \tilde{\mathbf{S}}_{n,i,t}), \end{aligned} \quad (4)$$

where independence is assumed between variables.

Computing the content log-likelihood in Equation 3 is trickier, since the structure generated by the structure decoder may be different from the real one. We circumvent this problem by using a form of teacher forcing, where the content is obtained by filling the real structure in place of the one generated by the structure decoder. In this way, the following likelihood can always be computed:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathcal{C}|\boldsymbol{z}, \mathcal{S}) = \sum_i \sum_\sigma \log(\tilde{\mathbf{P}}_{i,\sigma})^T \mathbf{P}_{i,\sigma} + \\ + \log(\tilde{\mathbf{D}}_{i,\sigma})^T \mathbf{D}_{i,\sigma}, \end{aligned} \quad (5)$$

where $\mathbf{P}$ and $\mathbf{D}$ are tensors containing, respectively, real one-hot pitch and duration tokens, while $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{D}}$ represent their probabilistic reconstructions. Indepedence is assumed between all pitch and duration variables.

4 Experiments

Following [Roberts *et al.*, 2018; Valenti *et al.*, 2020; Valenti *et al.*, 2021], we experiment on short and long sequences of MIDI music. The experiments probe the generative capabilities of the model comparing, whenever possible, to state of the art approaches. We further examine a novel scenario enabled by our methodology where generation is conditioned on user-specified structures. Finally, pitch, duration and chord embeddings are visualized to show that the model is able to learn known tonal and rhythmic concepts. We refer the reader to the source code[2] and the additional material[3], which contains the audio samples produced in the experimental phase.

[2]https://github.com/EmanueleCosenza/polyphemus
[3]https://emanuelecosenza.github.io/polyphemus/

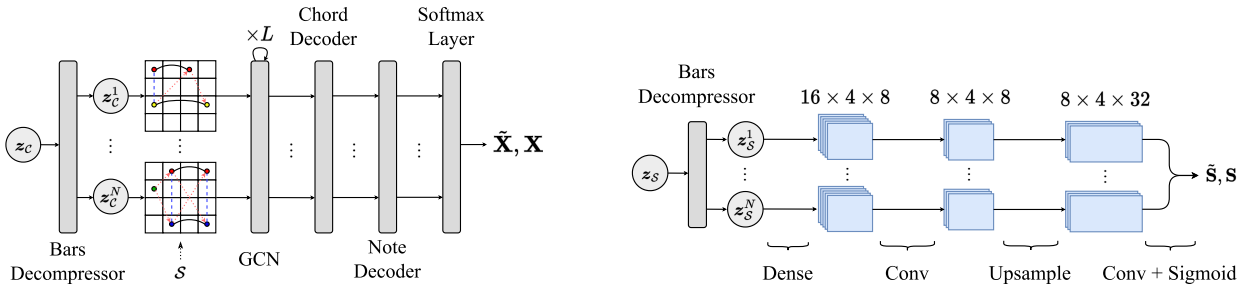


Figure 4: Visualization of the content decoder (a) and the structure decoder (b).

Datasets	2 Bars	16 Bars
LMD-matched	6,813,946	2,842,739
MetaMIDI Dataset	11,076,635	27,251,322

Table 1: Size of the datasets obtained in the preprocessing phase.

4.1 Data and Experimental Setup

We consider the 'LMD-matched' version of the Lakh MIDI Dataset [Raffel, 2016], which contains a total of 45,129 MIDI songs scraped from the internet. Additionally, we train our models on the more challenging MetaMIDI Dataset (MMD) [Ens and Pasquier, 2021], a recent and unexplored large scale MIDI collection totalling 436,631 songs. For each dataset, we obtain two new datasets containing, respectively, 2-bar and 16-bar sequences represented as chord-level graphs. The preprocessing pipeline is similar to that in [Roberts *et al.*, 2018; Valenti *et al.*, 2020; Valenti *et al.*, 2021]. The details about preprocessing can be found in the supplementary material. At the end of this phase, each sequence is composed of 4 tracks: a drum track, a bass track, a guitar/piano track and a strings track. The sizes of the resulting datasets are shown in Table 1.

The experiments focus on two versions of the model, one for 2-bar sequences and one for 16-bar sequences. We use for both a 70/10/20 split. The number of layers L of the encoder and decoder GCN is fixed to 8. The value d is set to 512. Adam [Kingma and Ba, 2014] is used as the optimizer for both models. The initial learning rates are set to 1e-4 and 5e-5, respectively, for the 2-bar and the 16-bar model. In both cases, the learning rate is decayed exponentially after 8000 gradient updates with a decay factor of $1 - 5\text{e-}6$. The hyperparameter β is annealed from 0 to 0.01. In the first 40,000 gradient updates, β is always 0, allowing the model to focus on the reconstruction task to find good initial representations. After this phase, the hyperparameter is annealed every $u = 40,000$ gradient updates by adding 0.001 to its current value. The batch size b is set to 256 and 32 respectively for the 2-bar and the 16-bar model.

4.2 Generation

The first set of experiments concerns the analysis of sequences generated from random codes $\boldsymbol{z}$. A qualitative visual inspection of the samples suggests that the models can consistently generate realistic music. Figure 5 shows an example of a 2-bar generated sequence. As can be seen from the pianoroll, the generated structures are well organized rhythmically, with drum and bass events played at the same timesteps. In the listening analysis[4], the 2-bar models appear to be particularly consistent, producing reasonable chord progressions, melodic segments and drum patterns. The 16-bar models are also coherent, both rhythmically and tonally. However, 16-bar sequences generally lack variability as the models tend to repeat musical structures across bars with slight differences. The rhythmic consistency of the model is enforced by the fact that the content decoder can focus on the generation of reasonable rhythmic patterns. The tonal consistency, instead, is ensured by the expressiveness of the GCN decoder, which is able to fill chord-level graphs realistically. To provide a more quantitative assessment, following previous works [Dong *et al.*, 2018; Valenti *et al.*, 2021], we measure the generative ability of the trained models by computing the following metrics on 20,000 generated sequences:

- EB (Empty Bars): ratio of empty bars.
- UPC (Used Pitch Classes): number of used pitch classes (12) per bar.
- DP (Drum Patterns): ratio of notes in 16-beat patterns, which are common in popular music (in %).

[4]Audio samples of generated 2-bar and 16-bar samples can be found here: https://emanuelecosenza.github.io/polyphemus/generation.html

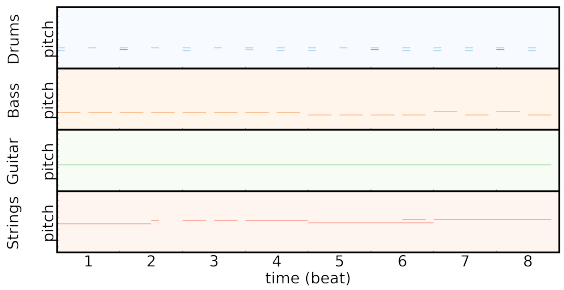


Figure 5: A pianoroll of a generated 2-bar sequence.

		EB				UPC ↓			DP ↑
		D	B	G/P	S	B	G	S	D
LMD-matched	jamming	6.59	2.33	20.45	6.10	**1.53**	3.91	4.09	93.2
	composer	0.01	28.9	1.35	0.01	2.51	4.55	5.19	75.3
	hybrid	2.14	29.7	14.75	6.04	2.35	5.11	5.24	71.3
	Calliope	0.0	0.0	0.0	0.0	2.08	3.87	2.52	94.84
	Ours (2-bars)	4.58	4.39	20.46	17.74	2.27	2.53	2.72	**96.97**
	Ours (16-bars)	1.96	3.37	11.38	12.02	1.79	**2.38**	**2.07**	96.59
MetaMIDI Dataset	**Ours (2-bars)**	5.38	8.31	23.49	21.54	1.85	2.03	2.24	96.28
	Ours (16-bars)	4.20	7.20	18.39	17.56	1.34	1.66	1.39	95.92

Table 2: Generation metrics of the proposed model, Calliope and the jamming, composer and hybrid versions of MuseGAN (EB: empty bars (%), UPC: number of used pitch classes, DP: drum patterns (%), D: drums, B: bass, G/P: guitar/piano, S: strings).

Table 2 shows the results obtained by our models, comparing our approach to different versions of MuseGAN [Dong *et al.*, 2018] and Calliope [Valenti *et al.*, 2021]. We also include metrics for the models trained on MetaMIDI Dataset with the goal of stimulating research on larger MIDI collections. The EB values are never equal to zero, which indicates that there are no issues with holes in the latent space and that the models do not ignore the latent codes during decoding. The UPC values are consistently low, indicating that the models have learned to stick to specific tonalities in the context of single bars. Additionally, the DP values for the proposed model are the highest, confirming its consistency on the rhythmic level. These results further validate the proposed methodology and confirm the rhythmic and tonal coherence of the model.

To inspect the structure of the latent space learned by the 2-bar Lahk MIDI Dataset model, we interpolate random latent codes linearly and we examine the music obtained by concatenating the resulting 2-bar sequences[5]. In the majority of cases, the interpolations created with the model are smooth and remain coherent, both tonally and rhythmically, throughout their entirety. Moreover, when the starting samples differ substantially, the model manages to create appealing transitions between distant styles. This suggests that the model has learned to organize its latent space in accordance with known musical semantics.

4.3 Structure-conditioned Generation

The separation of structure and content in our approach allows for the replacement of the generated structure tensor $\mathbf{S}$ with a new tensor $\hat{\mathbf{S}}$ during the decoding process. This new tensor can be modified in a similar fashion to pianoroll editing in Digital Audio Workstations (DAW). For instance, the user can specify that a certain instrument should only be played at a specific time in the sequence by filling the desired positions in the binary activation grid. To show this, we operate as follows, focusing on the 2-bar model trained on the Lahk MIDI Dataset. We start by sampling a random latent code $\boldsymbol{z}$, from which we obtain the two representations $\boldsymbol{z}_{\mathcal{S}}$ and $\boldsymbol{z}_{\mathcal{C}}$. We then let the structure decoder produce the corresponding structure tensor $\mathbf{S}$ from $\boldsymbol{z}_{\mathcal{S}}$. At this point, we modify $\mathbf{S}$ to our liking, obtaining a new structure tensor $\hat{\mathbf{S}}$. This corresponds to adding or removing nodes from the chord-level graph being generated. Finally, we let the content decoder compute two separate content tensors $\mathbf{X}$ and $\hat{\mathbf{X}}$, corresponding to two final music sequences. For our purposes, the content decoder should be robust to changes in the structure, replicating the same musical content represented by $\boldsymbol{z}_{\mathcal{C}}$. When listening to the audio samples generated in this way, the model appears to be able to preserve the rhythmic and tonal features of the original sequence, rearranging the musical content while abiding by the imposed structure. As an example, Figure 6a shows a generated structure tensor $\mathbf{S}$. The resulting sequence contains a recognizable I-IV progression in the key of B, supported by 8-beat bass and drum patterns[6]. We edit the tensor by making the drums sparser, keeping only the nodes at the start of each beat, and by making the strings more active, adding new nodes at the start of beats. This yields a new structure tensor $\hat{\mathbf{S}}$, which is shown in Figure 6b. The resulting music pro-

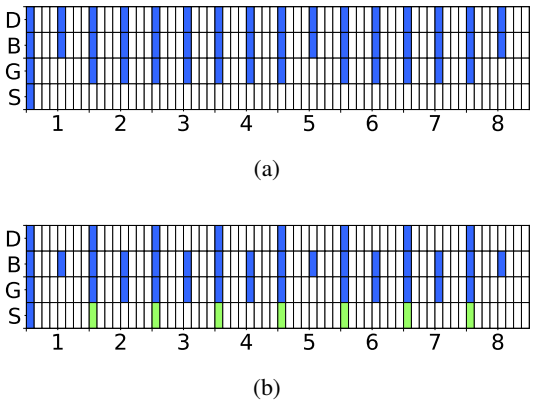
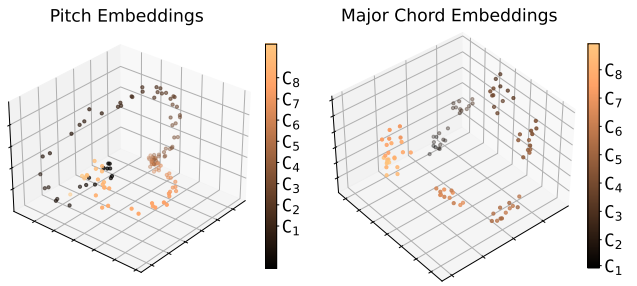


Figure 6: A visualization of the structure editing process. (a) A structure tensor $\mathbf{S}$ generated from a random latent code $\boldsymbol{z}$ (D: drums, B: bass, G: guitar, S: strings). Blue entries indicate the activation of single instruments at specific timesteps. Beats are numbered on the horizontal axis. (b) The edited structure tensor $\hat{\mathbf{S}}$. Green entries indicate the addition of new activations.

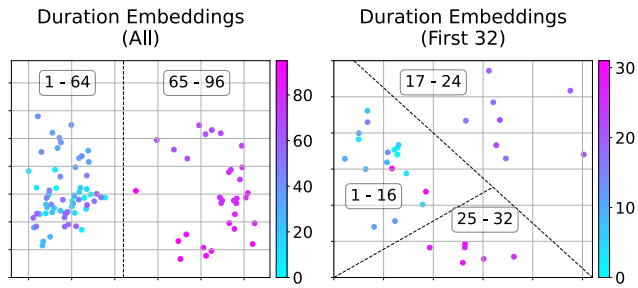
[5]Audio samples and pianorolls of interpolations can be found at the following link: https://emanuelecosenza.github.io/polyphemus/interpolation.html

[6]This and other examples related to conditioned generation can be found here: https://emanuelecosenza.github.io/polyphemus/conditioned-generation.html



(a) (b)

Figure 7: (a) PCA projection of pitch embeddings. (b) PCA projection of major chord embeddings. The major chords are obtained by picking as roots each note between C_1 and B_8.



(a) (b)

Figure 8: 2D PCA projections of duration embeddings, considering all 96 durations (a) and the first 32 (b).

duced by the content decoder maintains the same harmonic progression of the original sequence. The bass and guitar tracks remain unaltered with very slight variations. Finally, the strings play a new melodic line in the right key, while the drums play a steady 4-beat hi-hat pattern. Overall, this shows that the content decoder can adapt to new structures specified by the user, opening the door to a new form of human-computer music co-creation.

4.4 Embedding Visualization

Similarly to [Wang *et al.*, 2020] we explore the pitch, duration and chord embeddings by visualizing their principal components, focusing on the encoder network of the 2-bar model trained on the Lahk MIDI Dataset. Figure 7a shows the PCA projection in 3D space of all the 128 pitch embeddings. Pitch projections follow a circular path along the clockwise direction, suggesting that the model has learned the tonal relationships between different pitches. Figure 7b shows a 3D PCA projection of chord embeddings considering every major chord obtained by picking as roots the notes between C_1 and B_8. Durations are fixed to 1 beat. Similarly to what happens for pitches, chord embeddings follow a circular path in the space and form clusters related to specific octaves.

Figure 8 shows the PCA projections in 2D space of duration embeddings considering, respectively, all the possible 96 durations (i.e. up to three bars) and the first 32 durations (i.e. up to a bar). In the first case (Figure 8a), two distinct clusters contain, respectively, durations above 64 (i.e. above 2 bars) and durations below 64 (i.e. below 2 bars). In the second plot (Figure 8b), three clusters can be identified with, respectively, durations below 16 (i.e. below 2 beats, left of the plot), durations between 16 and 24 (i.e. between 2 and 3 beats, upper-right of the plot) and durations above 24 (i.e. between 3 beats and a bar). The plots suggest that the model has learned to organize its duration space in accordance to the rhythmic concepts of beats and bars.

5 Conclusions

In this work, we introduced a new graph representation for polyphonic multitrack music and a model that generates musical graphs by separating their structure and content. We then tested our methodology on both known and unexplored MIDI datasets, considering short and long sequences. As seen in the qualitative analysis and the comparison with the state of the art, our approach has revealed to be beneficial with regards to the rhythmic and tonal consistency of the generated music. Through manual experiments, we showed that the models are able to replicate the same musical content when varying the structure of the graphs. This allows for a new generative scenario where users can specify the activity of particular instruments in a music sequence. To conclude, we further validated our methodology by visualizing the pitch, chord and duration embeddings learned by the model. In each case, the embedding spaces are organized in accordance with known tonal and rhythmic concepts.

This work represents a first attempt at generating music with graph-based deep methodologies and should be considered as a starting point for further research on the topic. In the future, we aim to extend our work by taking into account MIDI velocity values, by training our model on other datasets and by studying new feasible graph representations and model configurations. Recurrent networks may be tested in the bar compressors and decompressors to check for improvements in the variability across bars. Again, a new *sustain* edge type could model the sustain of notes in different track activations. To conclude, we believe that the model has the potential to support human-computer co-creation, and it will be interesting to find possible applications of our methodology in modern software audio tools.

References

[Agostinelli *et al.*, 2023] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[Bacciu *et al.*, 2020] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot

learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Brunner *et al.*, 2017] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. Jambot: Music theory aware chord based generation of polyphonic music with lstms. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 519–526. IEEE, 2017.

[Brunner *et al.*, 2018] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. *arXiv preprint arXiv:1809.07600*, 2018.

[Chu *et al.*, 2016] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*, 2016.

[Chuan and Herremans, 2018] Ching-Hua Chuan and Dorien Herremans. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Dong *et al.*, 2018] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Ens and Pasquier, 2021] Jeffrey Ens and Philippe Pasquier. Building the metamidi dataset: Linking symbolic and audio musical data. In *ISMIR*, pages 182–188, 2021.

[Ferretti, 2017] Stefano Ferretti. On the modeling of musical solos as complex networks. *Information Sciences*, 375:271–295, 2017.

[Ferretti, 2018] Stefano Ferretti. On the complex network structure of musical pieces: analysis of some use cases from different music genres. *Multimedia Tools and Applications*, 77(13):16003–16029, 2018.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Higgins *et al.*, 2016] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[Huang *et al.*, 2018] 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. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[Huang *et al.*, 2019] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*, 2019.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[Jeong *et al.*, 2019] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In *International Conference on Machine Learning*, pages 3060–3070. PMLR, 2019.

[Karystinaios and Widmer, 2022] Emmanouil Karystinaios and Gerhard Widmer. Cadence detection in symbolic classical music using graph neural networks, 2022.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Li *et al.*, 2015] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[Li *et al.*, 2019] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019.

[Liu *et al.*, 2010] Xiao Fan Liu, K Tse Chi, and Michael Small. Complex network structure of musical compositions: Algorithmic generation of appealing music. *Physica A: Statistical Mechanics and its Applications*, 389(1):126–132, 2010.

[Makhzani *et al.*, 2015] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[Mogren, 2016] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[Raffel, 2016] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[Ramesh *et al.*, 2022] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[Roberts *et al.*, 2018] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning*, pages 4364–4373. PMLR, 2018.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[Simonetta *et al.*, 2018] Federico Simonetta, Filippo Carnovalini, Nicola Orio, and Antonio Rodà. Symbolic music similarity through a graph-based representation. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, pages 1–7. 2018.

[Simonovsky and Komodakis, 2017] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[Valenti *et al.*, 2020] Andrea Valenti, Antonio Carta, and Davide Bacciu. Learning style-aware symbolic music representations by adversarial autoencoders. *arXiv preprint arXiv:2001.05494*, 2020.

[Valenti *et al.*, 2021] Andrea Valenti, Stefano Berti, and Davide Bacciu. Calliope–a polyphonic music transformer. *arXiv preprint arXiv:2107.05546*, 2021.

[Wang *et al.*, 2020] Ziyu Wang, Yiyi Zhang, Yixiao Zhang, Junyan Jiang, Ruihan Yang, Junbo Zhao, and Gus Xia. Pianotree vae: Structured representation learning for polyphonic music. *arXiv preprint arXiv:2008.07118*, 2020.

[Wu *et al.*, 2020] Jian Wu, Xiaoguang Liu, Xiaolin Hu, and Jun Zhu. Popmnet: Generating structured pop music melodies using neural networks. *Artificial Intelligence*, 286:103303, 2020.

[Yang *et al.*, 2017] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[Zou *et al.*, 2021] Yi Zou, Pei Zou, Yi Zhao, Kaixiang Zhang, Ran Zhang, and Xiaorui Wang. Melons: generating melody with long-term structure using transformers and structure graph. *arXiv preprint arXiv:2110.05020*, 2021.