

Graph-based Polyphonic Multitrack Music Generation

Emanuele Cosenza*, Andrea Valenti and Davide Bacciu

University of Pisa

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.

Keywords

Symbolic music generation, Variational Autoencoders, Deep Graph Networks

1. Introduction

The automatic generation of artistic artifacts is gathering increasing interest, also thanks to the possibilities offered by modern deep generative models [1, 2, 3]. Despite these achievements, a closer inspection is often enough to detect if a piece of art is the outcome of an automatic artificial process. 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 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.


CREAI 2023 - Workshop on Artificial Intelligence and Creativity

*Corresponding author.

✉ e.cosenza3@studenti.unipi.it (E. Cosenza); andrea.valenti@phd.unipi.it (A. Valenti); davide.bacciu@unipi.it (D. Bacciu)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

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. First, 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. Second, we introduce a deep Variational Autoencoder [4] 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 [5] are used to generate multitrack, polyphonic music. Finally, 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 architectures such as LSTMs [6, 7, 8], Transformers [9, 10] and CNNs [11, 12], using Variational Autoencoders (VAE) [4], Generative Adversarial Networks (GAN) [13] and Adversarial Autoencoders (AAE) [14] as the main generative frameworks.

One of the challenges 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 [15, 16, 17, 18], associating nodes to notes and edges to temporal or tonal relationships, as well as at a higher level of the hierarchy, using melodic segments [19] and bars [20, 21] 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 [20] and the cadence detector in [22] are, to the best of our knowledge, the only systems that use Deep Graph Networks to process musical graphs. For what concerns generation, the only attempts at using graphs with deep learning are represented by PopMNet [20] and MELONS [21]. Both works use graphs to condition the generation of monophonic music, which is carried out by recurrent networks. In contrast to these works, our approach uses graphs at a lower level, leveraging Deep Graph Networks to learn tonal and rhythmic representations in the context of polyphonic multitrack music generation.

3. Graph-based Music Generation

The proposed model processes 4/4 polyphonic, multitrack fixed-length music sequences. 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. The number of timesteps in a bar, T , is fixed to 32. The division of sequences into bars is crucial since the model treats different bars separately. An example of a multitrack pianoroll is shown in Figure 1a.

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. Notice that we use the term ‘‘chord’’ loosely here to indicate any non-empty sequence of MIDI notes. 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. 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 SOS_p , EOS_p and PAD_p tokens. Similarly, the duration token can assume 99 different values, which correspond to 96 different durations (yielding a maximum duration of 3 bars) with the addition of SOS_D , EOS_D and PAD_D tokens.

The structure of g is encoded by the tensor $\mathbf{S} \in \{0, 1\}^{N \times I \times T}$. $\mathbf{S}_{n,i,t} = 1$ 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 [4] 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 $q_\phi(\mathbf{z}|\mathcal{S}, \mathcal{C})$, where $\mathbf{z} \in \mathbb{R}^d$. The decoder network, on the other hand, models $p_\theta(\mathcal{S}, \mathcal{C}|\mathbf{z})$. After introducing the latent variables $\mathbf{z}_\mathcal{S} \in \mathbb{R}^d$ and $\mathbf{z}_\mathcal{C} \in \mathbb{R}^d$, the generative process can be formalized as follows:

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

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 $\mathbf{z}_\mathcal{S}$ and $\mathbf{z}_\mathcal{C}$. The two codes are finally combined into a graph code \mathbf{z}_g with

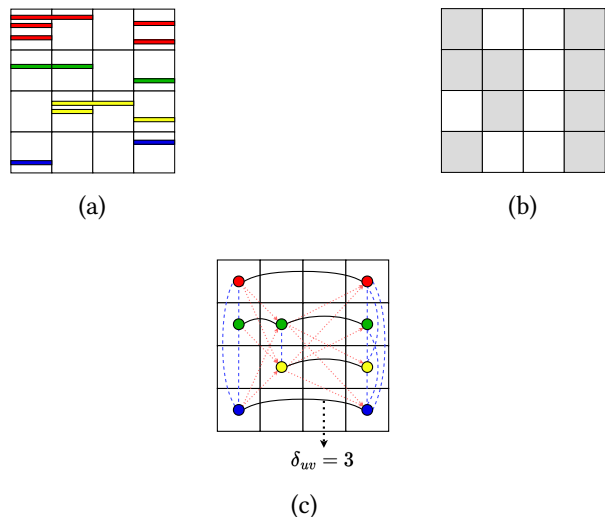


Figure 1: (a) A multitrack pianoroll with four tracks (rows) and four timesteps (columns). The positions of notes in each cell indicate their pitch. (b) A structure tensor computed from the pianoroll. (c) The resulting chord level graph. Black, solid connections indicate track edges. Red, dotted connections indicate next edges. Blue, dashed connections indicate onset edges. Edge features δ_{uv} indicate the distance in timesteps between two nodes.

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 \mathbf{z} into two separate latent vectors $\mathbf{z}_{\mathcal{S}}$ and $\mathbf{z}_{\mathcal{C}}$ through a linear layer. Then, it generates \mathcal{S} from $\mathbf{z}_{\mathcal{S}}$ through a *structure decoder* and the content \mathcal{C} from \mathcal{S} and $\mathbf{z}_{\mathcal{C}}$ through a deep graph *content decoder*. The content and the structure decoder model, respectively, the distributions $p_{\theta}(\mathcal{S}|\mathbf{z})$ and $p_{\theta}(\mathcal{C}|\mathcal{S}, \mathbf{z})$.

Content Encoder. In the content encoder (Figure 3a) each note is first embedded into a d -dimensional space with a linear note encoder. Next, a linear chord encoder processes the list of notes associated to each node, producing d -dimensional chord representations. These chord representations are processed by an encoder Graph Convolutional Network (GCN) [5] with L layers. We refer the reader to the supplementary material¹ for details about the implementation

¹<https://emanuelecosenza.github.io/polyphemus/assets/suppmaterials.pdf>

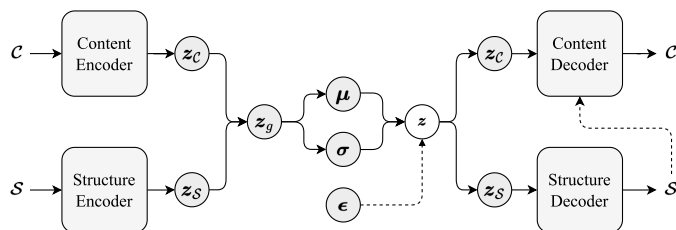


Figure 2: High-level visualization of the model.

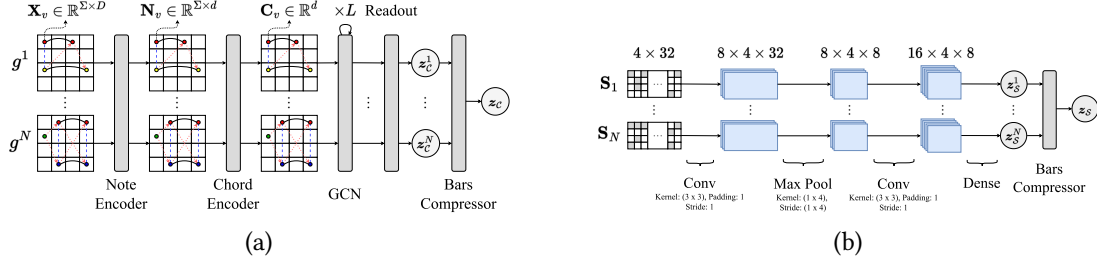


Figure 3: Visualization of the content encoder (a) and the structure encoder (b).

of the GCN. After L graph convolutional layers, a soft attention readout layer similar to that in [23] aggregates the information contained in each subgraph g^n of g related to the n -th bar of the musical sequence, producing bar embeddings z_c^1, \dots, z_c^N . The bar embeddings are finally passed through a linear bar compressor to obtain the final content representation z_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 z_s . This module first encodes each bar $\mathbf{S}_n \in \mathbb{R}^{I \times T}$ into a latent representation $z_s^n \in \mathbb{R}^d$ through a CNN [24] made of two convolutional layers with ReLU activations and Batch Normalization, interleaved by max pooling. The bar representations z_s^1, \dots, z_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 z_s .

Structure Decoder. The structure decoder (see Figure 4b) is specular to the structure encoder. It first decompresses z_s into N structure bar representations z_s^1, \dots, z_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 $\tilde{\mathbf{S}}$.

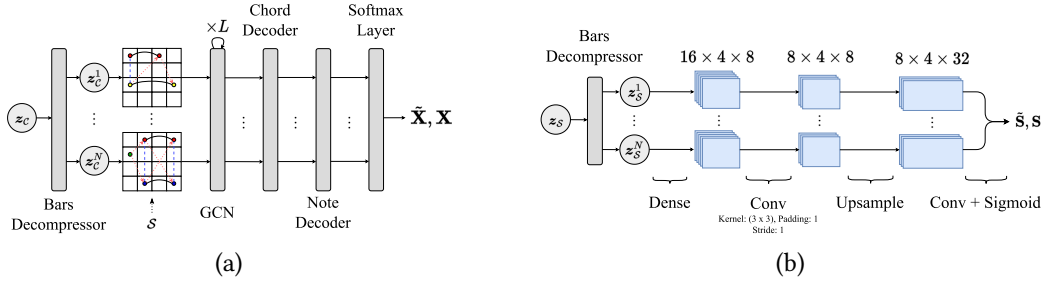


Figure 4: Visualization of the content decoder (a) and the structure decoder (b).

Content Decoder. It reconstructs the content of g from z_c and \mathcal{S} . The decoder first decompresses z_c into $z_c^1, \dots, z_c^N \in \mathbb{R}^d$. Each z_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 representations are decoded and passed through a softmax layer, which 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

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

where $D_{KL}(\cdot\|\cdot)$ is the KL divergence and the expectation is taken with respect to $\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})$. Following the β -VAE framework [25], 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:

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

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

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

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:

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

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. Independence is assumed between all pitch and duration variables.

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. Experiments

Following [8, 26, 10], 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 refer the reader to the source code² and the additional material³, which contains the audio samples produced in the experimental phase.

4.1. Data and Experimental Setup

We use the ‘LMD-matched’ version of the Lakh MIDI Dataset [27], which contains a total of 45,129 MIDI songs. We also consider the more challenging MetaMIDI Dataset (MMD) [28], an unexplored large scale MIDI collection totalling 436,631 songs. For each dataset, we obtain two new datasets containing 2-bar and 16-bar sequences represented as chord-level graphs. The preprocessing pipeline is similar to that in [8, 26, 10]. The details about preprocessing can be found in the supplementary material. Each preprocessed 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 GCNs is fixed to 8. The value d is set to 512. Adam [29] is used as the optimizer for both models, setting $1e-4$ and $5e-5$ as initial learning rates for the 2-bar and the 16-bar models. The learning rates are decayed exponentially after 8000 gradient updates with a decay factor of $1 - 5e-6$. The hyperparameter β is annealed from 0 to 0.01 during training.

4.2. Generation

The first set of experiments concerns the analysis of sequences generated from random codes \mathbf{z} . In the manual qualitative analysis⁴, both the 2-bar and the 16-bar models appear to be particularly consistent, producing reasonable chord progressions, melodic segments and drum patterns. To provide a more quantitative assessment, following previous works [30, 10], 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.

²<https://github.com/EmanueleCosenza/polyphemus>

³<https://emanuelecosenza.github.io/polyphemus/>

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

		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).

- 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 %).

Table 2 shows the results obtained by our models, comparing our approach to different versions of MuseGAN [30] and Calliope [10]. 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.

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 \mathbf{z} , from which we obtain the two representations $\mathbf{z}_{\mathcal{S}}$ and $\mathbf{z}_{\mathcal{C}}$. We then let the structure decoder produce the corresponding structure tensor \mathbf{S} from $\mathbf{z}_{\mathcal{S}}$. 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 $\mathbf{z}_{\mathcal{C}}$. When listening to the audio samples generated in this way, the model appears to be able to preserve the rhythmic

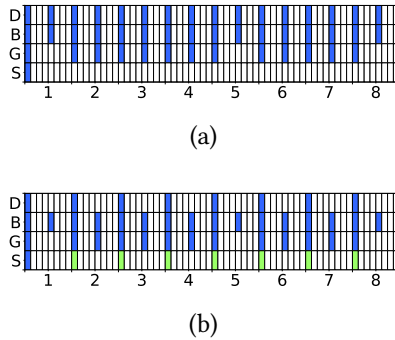


Figure 5: A visualization of the structure editing process. (a) A structure tensor \mathbf{S} generated from a random latent code \mathbf{z} (D: drums, B: bass, G: guitar, S: strings). Blue entries indicate the activation of single instruments at specific timesteps. Beats are numbered on the horizontal axis. (b) The edited structure tensor $\hat{\mathbf{S}}$. Green entries indicate the addition of new activations.

and tonal features of the original sequence, rearranging the musical content while abiding by the imposed structure. As an example, Figure 5a 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 patterns⁵. 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 5b. The resulting music produced 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 [31] 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 Lakh MIDI Dataset. Figure 6a 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 6b 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 7 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 7a), 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 7b),

⁵This and other examples related to conditioned generation can be found here: <https://emanuelecosenza.github.io/polyphemus/conditioned-generation.html>

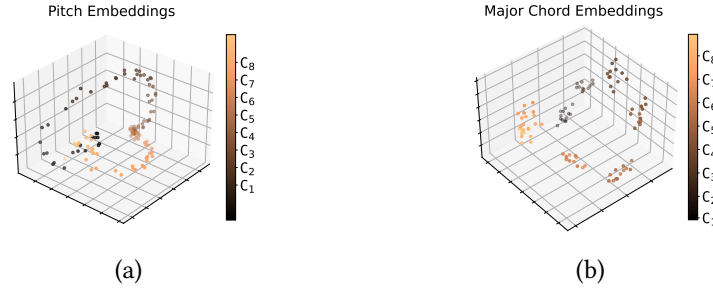


Figure 6: (a) PCA projection of pitch embeddings. (b) PCA projection of major chord embeddings. The major chords are obtained by picking as roots each note between C₁ and B₈.

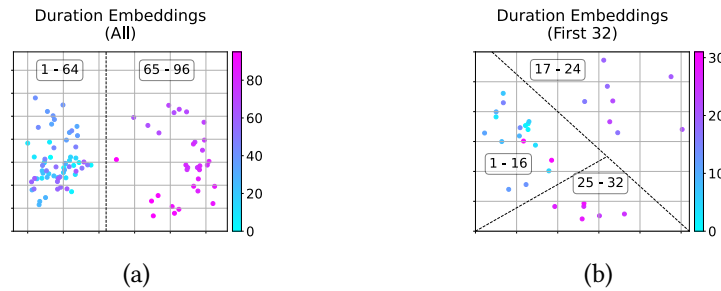


Figure 7: 2D PCA projections of duration embeddings, considering all 96 durations (a) and the first 32 (b).

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. 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 our methodology enables a generative scenario where users can specify the activity of particular instruments in a music sequence. Finally, we further validated our work 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. 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

- [1] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, Hierarchical text-conditional image generation with clip latents, arXiv preprint arXiv:2204.06125 (2022).
- [2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, *Advances in neural information processing systems* 33 (2020) 1877–1901.
- [3] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, et al., Musiclm: Generating music from text, arXiv preprint arXiv:2301.11325 (2023).
- [4] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).
- [5] D. Bacciu, F. Errica, A. Micheli, M. Podda, A gentle introduction to deep learning for graphs, *Neural Networks* 129 (2020) 203–221.
- [6] H. Chu, R. Urtasun, S. Fidler, Song from pi: A musically plausible network for pop music generation, arXiv preprint arXiv:1611.03477 (2016).
- [7] G. Brunner, Y. Wang, R. Wattenhofer, J. 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)*, IEEE, 2017, pp. 519–526.
- [8] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, D. Eck, A hierarchical latent vector model for learning long-term structure in music, in: *International conference on machine learning*, PMLR, 2018, pp. 4364–4373.
- [9] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, D. Eck, Music transformer, arXiv preprint arXiv:1809.04281 (2018).
- [10] A. Valenti, S. Berti, D. Bacciu, Calliope—a polyphonic music transformer, arXiv preprint arXiv:2107.05546 (2021).
- [11] C.-H. Chuan, D. 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.
- [12] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, D. Eck, Counterpoint by convolution, arXiv preprint arXiv:1903.07227 (2019).
- [13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, *Advances in neural information processing systems* 27 (2014).
- [14] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, arXiv preprint arXiv:1511.05644 (2015).
- [15] X. F. Liu, K. T. Chi, M. Small, Complex network structure of musical compositions: Algorithmic generation of appealing music, *Physica A: Statistical Mechanics and its Applications* 389 (2010) 126–132.
- [16] S. Ferretti, On the complex network structure of musical pieces: analysis of some use cases from different music genres, *Multimedia Tools and Applications* 77 (2018) 16003–16029.
- [17] S. Ferretti, On the modeling of musical solos as complex networks, *Information Sciences* 375 (2017) 271–295.

- [18] R. Mellon, D. Spaeth, E. Theis, Genre classification using graph representations of music (2014).
- [19] F. Simonetta, F. Carnovalini, N. Orio, A. Rodà, Symbolic music similarity through a graph-based representation, in: *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, 2018, pp. 1–7.
- [20] J. Wu, X. Liu, X. Hu, J. Zhu, Popmnet: Generating structured pop music melodies using neural networks, *Artificial Intelligence* 286 (2020) 103303.
- [21] Y. Zou, P. Zou, Y. Zhao, K. Zhang, R. Zhang, X. Wang, Melons: generating melody with long-term structure using transformers and structure graph, *arXiv preprint arXiv:2110.05020* (2021).
- [22] E. Karystinaios, G. Widmer, Cadence detection in symbolic classical music using graph neural networks, 2022. *arXiv:2208.14819*.
- [23] D. Jeong, T. Kwon, Y. Kim, J. Nam, Graph neural network for music score data and modeling expressive piano performance, in: *International Conference on Machine Learning*, PMLR, 2019, pp. 3060–3070.
- [24] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016. <http://www.deeplearningbook.org>.
- [25] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework (2016).
- [26] A. Valenti, A. Carta, D. Bacciu, Learning style-aware symbolic music representations by adversarial autoencoders, *arXiv preprint arXiv:2001.05494* (2020).
- [27] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*, Columbia University, 2016.
- [28] J. Ens, P. Pasquier, Building the metamidi dataset: Linking symbolic and audio musical data., in: *ISMIR*, 2021, pp. 182–188.
- [29] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).
- [30] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, Y.-H. 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.
- [31] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, G. Xia, Pianotree vae: Structured representation learning for polyphonic music, *arXiv preprint arXiv:2008.07118* (2020).