A methodology for automatic composition of polyphonic music

Angélica Abadia Paulista Ribeiro[1], João Luís Garcia Rosa[1]

Universidade de São Paulo, Brazil
angelribeiro@usp.br
joaoluis@icmc.usp.br

Abstract.

Automatic music composition presents challenges that exceed those of picture or video synthesis, requiring precise temporal modeling and the synchronization of several instrumental voices with unique dynamics. This study presents three Generative Adversarial Network (GAN) architectures—the Improvisation, Composition, and Hybrid models—for the automated generation of polyphonic notation for multiple instruments. Utilizing a corpus of over 100,000 symphonic music pieces in Wave, MIDI, and MusicXML formats (including a dataset curated by the primary author), the models include completely extracted MIDI features derived from the conversion of all Wave files to MIDI. Assessment via quantitative intra- and inter-instrument metrics, supplemented by a musician-centric study, demonstrates that the models can create music from inception, produce coherent four-measure excerpts, and engage in human-machine co-composition to transform single-instrument melodies into comprehensive arrangements.

CCS Concepts: • **Computing methodologies** → **Deep Learning**.

Keywords: computer music, neural Networks, machine learning

1. INTRODUCTION

Creating aesthetically pleasing, realistic music is one of the most challenging AI projects. Researchers have dramatically advanced automatic video, image, and text synthesis in recent years, mainly by employing generative adversarial networks [Goodfellow et al. 2019; Radford et al. 2015; Vondrick et al. 2016; Saito et al. 2017; Yu et al. 2017]. Naturally, they have also attempted similar approaches for music composition, yet the task remains challenging for the following reasons.

Foremost, music evolves as art. Composers organise music hierarchically, building phrases from simpler patterns like measures and notes. When listening to music, musicians focus on structural patterns such as coherence, harmony, rhythm, tension, and emotional flow [Herremans and Chew 2017]. Therefore, a mechanism that describes temporal structure becomes essential.

Due to the polyphonic nature of music, composers frequently incorporate multiple instruments or voices in a composition. A symphonic orchestra consists of strings (violins, violas, cellos, double basses, harps), woodwinds (flutes, piccolos, oboes, English horns, clarinets, bass clarinet, bassoons, contra-bassoons), percussion (timpani, triangle, snare drums, bass drum, cymbals, tubular bells, etc.), and keyboard instruments. These instruments interact and evolve together. Music theorists extensively discuss composing techniques such as counterpoint and harmony.

Composers typically structure music through melodies (horizontal), chords (vertical), and arpeggios (diagonal). While effective for monophonic music, horizontal note arrangements fail to capture the vertical dependencies essential to polyphony. Consequently, many studies simplify composition by reducing polyphonic structures to monophonic representations, often through linear sequencing or the combination of single-voice lines.

This article avoids these simplifications as much as possible. Our objective is to create polyphonic music with (1) harmonic and rhythmic structure, (2) multi-instrument interdependence, and (3) temporal structure.

A temporal model can compose music from scratch or learn to follow the temporal structure of an underlying tune provided by the musician's instrument. We propose three composition styles for instrument interactions: *improvisation*, where a dedicated generator independently generates each instrument; *composer*, where a single generator jointly composes all instruments; and *hybrid*, which combines both approaches by assigning each instrument its generator while sharing additional inputs among them. We employ transposed convolutional neural networks (CNNs) to compose music using measures instead of individual notes, thereby capturing local translation-invariant patterns.

We define objective intra- and inter-instrument measures to monitor learning and quantitatively evaluate the models' results. Although we focus on automatic music creation, researchers can apply our network design to construct polyphonic sequences in other domains. Our key contributions are as follows: we propose a GAN-based approach to generate polyphonic musical notation for various instruments; our model enables conditional melody generation to support cooperative human–machine composition and musical accompaniment; we introduce the Solfejo Dataset (SD)[1], which contains 2,436 technical pieces in WAVE, MIDI, and MusicXML formats; and we evaluate the artificially generated musical notation using the defined intra- and inter-instrument objective measures.

2. GENERATIVE ADVERSARIAL NETWORKS

To achieve adversarial learning, GANs employ two networks: the *generator* and the *discriminator* [Goodfellow et al. 2019]. The generator creates plausible data to mislead the discriminator, while the discriminator attempts to distinguish between real and generated data. The generator maps noise z, sampled from a prior distribution, to the data space. During training, the discriminator learns to identify real data, and the generator learns to deceive the discriminator. This training process forms a two-player minimax game between the generator G and the discriminator D:

$$\min_G \max_D \mathbf{E}_{\mathbf{x}\sim p_d}[\log(D(\mathbf{x}))] + \mathbf{E}_{\mathbf{z}\sim p_z}[1 - \log(D(G(\mathbf{z})))] \tag{1}$$

where p_d and p_z denote the real data distribution and the prior distribution of z, respectively.

arjovsky2017wasserstein showed that using the Wasserstein distance instead of the Jensen-Shannon divergence can stabilize training and mitigate *mode collapse*. To satisfy the K-Lipschitz constraint, the original WGAN enforces *weight clipping*; however, this strategy often causes optimisation difficulties. To address this issue, [Gulrajani et al. 2017] introduced a gradient-penalty term into the discriminator's objective, resulting in the following loss for D:

$$\mathbf{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] - \mathbf{E}_{\mathbf{z}\sim p_z}[D(G(\mathbf{z}))] + \mathbf{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}\left[(\nabla_{\hat{\mathbf{x}}}\|\hat{\mathbf{x}}\| - 1)^2\right] \tag{2}$$

In the model, $p_{\hat{\mathbf{x}}}$ represents samples obtained by linearly interpolating between points from p_d and p_g. The WGAN-GP improves optimisation stability, accelerates convergence, and reduces the need for hyperparameter adjustments; for these reasons, we adopt it as the generator in this study.

3. RELATED WORK

Researchers have extensively explored GANs for music generation. C-RNN-GAN [Mogren 2016] employs RNNs in both the generator and discriminator to compose polyphonic music with a fixed order

[1]The dataset is publicly available at `https://github.com/AngelicaRibeiro/Solfege-Dataset` and the source code could be requested by the email.

of 10 notes. SeqGAN [Yu et al. 2017] combines reinforcement learning with GANs to construct discrete token sequences for monophonic music. MidiNet [Yang et al. 2017] applies convolutional GANs to generate melodies from scratch, follow a chord progression, or build on the melodies of previous measures.

Several authors have proposed new models for music composition, as discussed by [Richard et al. 2025]. Researchers have used neural networks to generate various types of music, including monophonic melodies [Chowdhury et al. 2025] and four-part chorales [Dhar and Victor 2024]. Notably, RNN-RBM [Chuan and Herremans 2018], a generalisation of RT-RBM, generates polyphonic notation for a single instrument. In [Cheston et al. 2025], the authors used hierarchical RNNs to create lead-sheet accompaniments (melody and chord symbols) and monophonic percussion tracks.

4. PROPOSED MODEL

yang2017midinet states that measures serve as the fundamental unit of music composition because musicians often use harmonic alterations, such as chord transitions at the beginning or conclusion of a measure, as building blocks. Hyperparameters were automatically optimized using a Python library to identify the best-performing configurations.

4.1 Data Representation

To model polyphonic music, we represent many instruments using the MIDI attribute format. The MIDI attributes form binary matrices that indicate the presence of notes at different times. We describe a piece as a collection of these characteristics. We represent a set of M instruments in a measure as a tensor $\mathbf{x} \in \{0,1\}^{R \times S \times M}$, where R and S correspond to the measure's time instants and candidate notes, respectively.

We denote the representation of a set of M instruments over T measures as $\vec{\mathbf{x}} = \{\vec{\mathbf{x}}(t)\}_{t=1}^{T}$, where $\vec{\mathbf{x}}(t) \in \{0,1\}^{R \times S \times M}$ represents the M instruments at measure t. For both real and generated data, we represent the instrument set for each measure with a fixed-size matrix, which enables the use of CNNs.

4.2 Modeling the Interdependence among Multiple Instruments

There are two musical composition methods. The first is *improvisation*, where players play different instruments and produce music spontaneously without a preset arrangement. The second way involves a composer who organizes the composition and arranges instruments according to harmonic structure and instrumentation knowledge. The musicians then follow the score and execute the work. We propose two compositional models and a third hybrid model based on this logic.

The Improvisation Model involves numerous generators individually making music for their respective instruments using a random vector $\mathbf{z}_i$, $i = 1, 2, \ldots, M$, where M denotes the number of generators (or instruments). These generators receive feedback from separate discriminators.

Composer Model: A single generator outputs separate channels, each corresponding to a distinct instrument, reflecting the composer's intent. This model uses one shared random vector z and one discriminator to detect if the input music is authentic or fraudulent by examining all instruments M. Only one generator and discriminator are needed, regardless of M.

We suggest a hybrid model that combines the ideas of the two models above. Each of the M generators requires an *inter-instrument* random vector z and an *intra-instrument* random vector z_i as input. The inter-instrument vector, G_i, coordinates the development of works, similar to a composer. We evaluate all M instruments with one discriminator. This setup requires M generators but just one discriminator. Unlike the composer model, the hybrid model allows flexible network architectures—changing the number of layers, filter sizes, and input configurations—for each of the M generators. This flexibility lets instrument generation vary without losing dependency.

4.3 Temporal Structure Model

Multi-instrument music generated by the methods above may lack cohesion across measures. As music is time-dependent, a temporal model is needed to generate more extended sequences like musical phrases. We develop two strategies to address this.

To generate fixed-length musical phrases, the first method uses measure progression as a dimension to upscale the generator. The generator has two subnetworks: the temporal-structure generator. G_{temp} and measure generator G_{bar}. G_{temp} maps a noise vector z to a sequence of latent vectors $\vec{\mathbf{z}} = \{\vec{\mathbf{z}}^{(t)}\}_{t=1}^{T}$. The output $\vec{\mathbf{z}}$, Assuming temporal information, G_{bar} generates consecutive sets of instruments, measure by measure:

$$G(\mathbf{z}) = \left\{G_{\text{bar}}\left(G_{\text{temp}}(\mathbf{z})^{(t)}\right)\right\}_{t=1}^{T} \tag{3}$$

saito2017temporal used a similar concept for video creation.

Conditional generation: The second method assumes that the sequence of measures $\vec{\mathbf{y}}$ musicians supply for one instrument, and it learns its temporal structure while creating the other instruments to finish the piece. The conditional instrument generator G° produced sequentially by the conditional measure generator G°_{bar}. The sets of multiple remaining instruments for measure t are generated by G°_{bar}, which requires two inputs: the condition $\vec{\mathbf{y}}^{(t)}$ and a time-dependent random noise vector $\vec{\mathbf{z}}^{(t)}$.

To achieve conditional generation with high-dimensional conditions, we train an additional encoder E to map $\vec{\mathbf{y}}^{(t)}$ to the latent space of $\vec{\mathbf{z}}^{(t)}$. yang2017midinet has adopted similar approaches. We can formulate the entire procedure as follows.

$$G^{\circ}(\vec{\mathbf{z}}, \vec{\mathbf{y}}) = \left\{G^{\circ}_{\text{bar}}\left(\vec{\mathbf{z}}^{(t)}, E\left(\vec{\mathbf{y}}^{(t)}\right)\right)\right\}_{t=1}^{T} \tag{4}$$

The encoder should extract *inter-instrument* features from the supplied instrument, as *intra-instrument* features are not relevant for producing remaining instruments. This temporal model incorporation method is effective for cooperative human–machine composition and musical accompaniment.

4.4 GAN

We now present *GAN*, an integration and extension of the proposed multi-instrument and temporal models. The input to GAN, denoted by $\bar{\mathbf{z}}$, consists of four components: time-independent, inter-instrument random vectors z, time-independent intra-instrument random vectors z_i, time-dependent, inter-instrument, random vectors z_t, and time-dependent intra-instrument random vectors $z_{i,t}$.

For each instrument $i = 1, \ldots, M$, the shared temporal generator G_{temp} and the instrument-specific generator $G_{temp,i}$ map the time-varying noise vectors z_t and $z_{i,t}$ to latent sequences that encode inter- and intra-instrument timing, respectively. We concatenate these sequences with the time-invariant noises z and z_i and pass them to the measure generator, which then sequentially outputs the instrument set. We can formulate the generation procedure as:

$$G(\bar{\mathbf{z}}) = \left\{G_{\text{bar},i}\left(\mathbf{z}, G_{\text{temp}}\left(\mathbf{z}_t\right)^{(t)}, \mathbf{z}_i, G_{\text{temp},i}\left(\mathbf{z}_{i,t}\right)^{(t)}\right)\right\}_{i,t=1}^{M,T} \tag{5}$$

5. IMPLEMENTATION

This section introduces a new technical music collection and describes all datasets used in this work.

5.1 Solfejo Dataset

The author built the dataset by performing pieces from technical music books, including Alexis Garaude's *Solfejo* [Garaudé 1811] and Mario Mascarenhas' volumes [Mascarenhas 1961]. They recorded music in WAVE format using piano, alto saxophone, and recorder as timbres.

The current release contains 2,436 pieces. The technical content significantly increased network training complexity. Besides this proprietary set, researchers used public datasets. They converted WAVE recordings to MIDI using the `audio-to-midi` Python module.

We applied custom algorithms to extract score-level information from MusicXML files—such as note names, key signatures, clefs, accidentals, and more—to build gold-standard descriptions. Our package includes musical descriptions covering temporal structure, meter, and melody, together with manually generated text, MXL files, and audio signals.

We archived the gold standards, audio files (.wav, .midi), and scores (.xml) in a `.xlsx` spreadsheet. The resultant compilation is designated as the Solfejo Dataset (SD).

This dataset facilitates transcription, automated music composition, and computational musicology tasks, as it is derived from technical studies and music education exercises.

5.2 Additional Datasets

To calibrate the model's rhythmic component, we used the Groove MIDI Dataset (GMD) [gro 2019], which contains 13.6 hours of time-aligned expressive drum MIDI and synthesised audio (1,150 MIDI files and over 22,000 drum measures).

The MAESTRO dataset (MIDI and Audio Edited) [mae 2019] provides over 200 hours of virtuoso piano performances aligned to within 3 ms, thereby strengthening the model by covering all 88 notes.

We also utilised an aggregated MIDI collection [Raffel 2019] (77,153 pieces) and the Lakh MIDI Dataset (LMD) [Raffel 2016], which includes 176,581 unique files.

We converted all MIDI files into instrument matrices. To capture triplets and semiquavers, we set the pitch dimension to 128 and the temporal resolution to 96 per measure. We used the Python library `pypianoroll` [Dong et al. 2018] to evaluate and process MIDI files. All datasets, metadata, and conversion tools are cross-compatible.

5.3 Data Pre-processing

The dataset contains considerable noise because most MIDI files originate from the Web. To address this, we added three cleaning steps. First, we observed that specific instruments play only a few notes in each piece, which increases data sparsity and hampers learning. To mitigate class imbalance, we merged related instruments by summing their MIDI properties. For each composition, we merged MIDI-attribute matrices based on the number of instruments. Although this introduced some noise, we empirically found it preferable to using empty measures. After completing this step, we obtained 60,987 multi-instrument sets.

We categorised instruments into melody, rhythm, and percussion, following prior research [Raffel 2016], since distinguishing between a melody and its accompaniment remains challenging [Chu et al. 2016; Yang et al. 2017]. Second, we filtered the dataset using metadata confidence: we retained only entries with reliable time signatures, such as 4/4, 2/2, and 3/4-and removed compound meters like 3/8 and 6/8, so we could use only simple meters. This process yielded a clean corpus.

Third, we applied the state-of-the-art *Structural Features* algorithm [McCallum 2019] to segment each instrument part and extract metrically compliant phrases for training the temporal model. We defined a phrase as four measures and reduced longer segments to this fixed length, which provided 55,623 training phrases. While the proposed models generate fixed-length segments, the conditional model supports variable-length music generation based on the input. To eliminate rarely used pitches, we removed notes below C1 and above C8. As a result, the target output tensor has dimensions 4 (measures) $\times$ 96 (time instants) $\times$ 84 (pitches) $\times$ N (instruments).

5.4 Model settings

Both G and D are deep CNNs. The G expands first along the time axis and then the pitch axis, whereas D contracts. To comply with [Gulrajani et al. 2017], we update G every five updates of D and perform batch normalisation exclusively to G. Each generator has a 128-length random input vector (s). On an NVIDIA GeForce GTX 950M GPU (4,055 MB), each model takes 1.5 weeks to train. During testing, we binarize G output using `tanh` activation functions in the final layer with zero bias.

6. OBJECTIVE EVALUATION METRICS

We assess each model with six metrics: four *intra-instrument*—**CV** (percentage of empty measures), **NCN** (note-class count per measure), **NQ** (share of notes at least a *fusa* long), and **PP** (percentage of notes that fall on the metrical grid)—plus two *inter-instrument* metrics: **DT**, the *tonal distance* of [Harte et al. 2006], gauging harmonic proximity, and **RI**, the fraction of time intervals in which parts overlap, indicating textural density.

6.1 Training-Data Analysis

Applying the six metrics to the training set (see the first rows of Tables I–II) yielded several insights. **CV** corroborates the decision to group instruments into families. **NCN** distinguishes melody carriers—saxophone and recorder, with NCN7.0 - from chordal instruments such as violin, piano, and strings, with NCN8.0. High **NQ** shows the MIDI-attribute matrices are not overly fragmented, while **PP** indicates that 93% of percussion notes align with composite-meter pulses (eighth or sixteenth notes). **DT** clusters around 5.50 when comparing a melodic and a harmonic part, and around 6.00 for two harmonic parts; shuffling instrument pairs raises **DT** slightly, confirming its sensitivity to harmonic relationships. Finally, **RI** emerges as one of the most informative statistics of the dataset.

		bars with pauses (CV; %)					Used Notes (NCN)				Notes qualitie (NQ; %)				Intervals Ratios RI(%)
		S	Pe	V	Pi	C	S	V	Pi	C	S	V	Pi	C	Pe
Training data		13.06	13.06	24.4	29.8	15.1	6.71	8.08	8.28	8.38	95	86.9	93.4	94.6	93.6
Scratch Composition	Simple	97.4	100	17.5	5.68	5.00	6	7.88	7.32	9.72	5	27.8	36.1	31.2	5
	Improvising	11.59	7.33	23.3	27.6	11.1	6.53	8.69	9.13	9.09	76.5	61.6	67.2	68.1	98.2
	Composer	5.01	33.9	6.34	5.02	5.01	7.51	9.2	9.89	10.19	54.5	52.4	54.9	57.5	80.3
	Hibrid	7.14	34.7	16.7	22.8	11.04	7.35	9.76	10.45	10.24	49.6	48.2	50.5	57	76.3
Conditional composition	Improvising	9.6	8.47	18.3	15.7	8.44	7.05	8.79	7.12	9.23	78.9	63.8	64.44	67.3	96.6
	Composer	5.65	25.7	6.97	25.1	6.49	7.51	9.57	6.49	10.1	58.5	53.4	70.62	64	89.5
	Hibrid	7.09	9.53	15.3	28.5	9.05	7.86	9.43	8.33	9.32	48.3	60.6	72.27	72.1	76.8

Table I. Intra-instrument evaluation (S: saxophone, Pe: percussion, V: violin, Pi: piano, C: strings; values closer to the first row are better).

		Tonal Distance (DT)					
		S-V	S-C	S-Pi	V-C	V-Pi	C-Pi
Treining		1.22	1.23	1.16	0.75	0.67	0.69
Random Training		1.24	1.24	1.21	0.79	0.77	0.78
Scratch composing	Improvising	1.21	1.25	1.19	0.7	0.64	0.7
	Composer	1.02	1.01	0.95	0.6	0.63	0.56
	Hibrid	0.99	1	0.97	0.5	0.5	0.48
Composing Condicional	Improvising	1.16	1.18	1.15	0.69	0.6	0.65
	Composer	1.06	1.01	1.05	0.61	0.66	0.6
	Hibrid	1.04	1.01	1.03	0.61	0.59	0.6

Table II. Inter-instrument evaluation (lower values are better).

7. EXPERIMENTS AND RESULTS

7.1 Sample Results

We compare the initial input excerpts with the outputs generated by the composer and hybrid models. Across the samples, the instruments largely remain within a standard musical scale, and several passages reveal chord intervals. The saxophone typically plays the lowest pitches and stays predominantly monophonic. At the same time, the percussion tracks follow rhythmic patterns that align strictly with the meter grid, usually in eighth- or sixteenth-note pulses. Meanwhile, the violin, piano, and string sections often articulate chords with overlapping pitch ranges, consistently revealing a strong harmonic relationship among these parts.

7.2 Objective Evaluation

We generated 25,000 measures with each model and assessed them using the proposed metrics (Tables I–II). In the conditional-generation setting, we provided a piano melody and synthesized the remaining four parts. We used a stripped-down composer variant without batch normalisation as a baseline, since it learns virtually nothing.

The improvisation model scores highest across the *intra-instrument* metrics, which aligns with its architecture of dedicating one generator per instrument. All models (except the baseline) exhibit strong pulse alignment (**PP**). However, they use more note classes (**NCN**) and produce far fewer qualified notes (**NQ**) than the training data, suggesting that binarising the continuous outputs of G introduces fragmentation. Despite the relatively high **CV** for percussion in the composer and hybrid outputs, the models still capture rhythmic patterns effectively.

For the *inter-instrument* metric **DT**, the composer and hybrid models produce lower values than the improvisation model, which implies tighter harmonic coherence. Their performance remains consistent across instrument pairs, confirming that the hybrid architecture preserves the composer's harmonic quality while introducing additional flexibility.

7.3 Training Process

We analysed the composer model trained *from scratch*—the other architectures behave similarly—to track learning dynamics. The discriminator loss $\mathcal{L}_D$ drops sharply, then plateaus, and finally drifts upward, signalling the moment at which the generator G begins to capture meaningful features. The progression: G rapidly infers each instrument's pitch range, places notes (initially fragmented but pitch-constrained), forms event clusters in the saxophone's lower register, and enables the violin, piano, and strings to adopt longer, sustained durations—all evidence of steady improvement. The benefit of the technical-music dataset shows accelerated learning. It confirms that G eventually matches the correct number of note classes. In contrast, it reveals a divergence in **QN** from the training data, indicating that the curated dataset further enhances the generator's performance.

8. CONCLUSION

This study presented a GAN-based architecture employing deep convolutional networks to generate multi-instrument polyphonic music. Trained on a custom dataset and evaluated through objective metrics and a user case study, the model demonstrates the ability to capture meaningful musical structure. Despite limitations in aesthetic quality, the system offers a promising foundation for EEG-driven composition in Brain–Computer Interface applications, with potential to control high-level musical parameters such as tempo, tonality, and motif selection and more.

We conducted ablation studies to assess the role of the temporal module, observing apparent declines in musical coherence when it was removed. Parametric variations further confirmed the importance of embedding size and dropout in maintaining model stability. Comparisons with state-of-the-art models showed our method offers competitive performance, particularly in harmonic structure and symbolic interpretability. The model comprises 4.2M parameters with $\mathcal{O}(n^2)$ complexity due to self-attention, and achieves sub-second inference on mid-range GPUs.

ACKNOWLEDGMENT

The authors thank the financial support from Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) Process Codes 88887.822266/2023-00, 88887.503298/2020-00, 88887.473768/2020-00, 88887.695319/2022-00; and FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brazil, process 2020/14115-8.

REFERENCES

Groove dataset. `https://magenta.tensorflow.org/datasets/groove`, 2019. Acessado: 29-09-2019.

Maestro dataset. `https://https://magenta.tensorflow.org/datasets/maestro`, 2019. Acessado: 29-09-2019.

Cheston, H., Bance, R., and Harrison, P. Deconstructing jazz piano style using machine learning. *arXiv preprint arXiv:2504.05009*, 2025.

Chowdhury, S. R., Biswas, S., Nandy, S., Maity, S. K., and Chatterjee, D. Music generation using deep learning. *Power Devices and Internet of Things for Intelligent System Design*, 2025.

Chu, H., Urtasun, R., and Fidler, S. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*, 2016.

Chuan, C.-H. and Herremans, D. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Dhar, A. and Victor, A. Neural harmony: Advancing polyphonic music generation and genre classification through lstm-based networks. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, pp. 1–6, 2024.

Dong, H.-W., Hsiao, W.-Y., and Yang, Y.-H. Pypianoroll: Open source python package for handling multitrack pianoroll, 2018.

Garaudé, A. d. *Méthode complète de chant: oeuv. 40.* A la Classe de Chant de l'Auteur, 1811.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2019.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems*. pp. 5767–5777, 2017.

Harte, C., Sandler, M., and Gasser, M. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. ACM, pp. 21–26, 2006.

Herremans, D. and Chew, E. Morpheus: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 2017.

Mascarenhas, M. *120 Músicas Favoritas Para Piano.* Irmãos Vitale, 1961.

McCallum, M. C. Unsupervised learning of deep features for music segmentation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 346–350, 2019.

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

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Raffel, C. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching.* Ph.D. thesis, Columbia University, 2016.

Raffel, C. Midi dataset. `https://composing.ai/dataset`, 2019. Acessado: 29-09-2019.

Richard, G., Lostanlen, V., Yang, Y.-H., and Müller, M. Model-based deep learning for music information research: Leveraging diverse knowledge sources to enhance explainability, controllability, and resource efficiency [special issue on model-based and data-driven audio signal processing]. *IEEE Signal Processing Magazine* 41 (6): 51–59, 2025.

Saito, M., Matsumoto, E., and Saito, S. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE International Conference on Computer Vision*. pp. 2830–2839, 2017.

Vondrick, C., Pirsiavash, H., and Torralba, A. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*. pp. 613–621, 2016.

Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.