Conditional Residual GAN for Symbolic Melody Generation

Lorenzo Schiavone *2140690*
Federico Tamburlin *2123335*

***Abstract*—The generation of symbolic music using deep learning models presents a significant challenge in balancing creative novelty with musical coherence. This paper introduces a conditional Generative Adversarial Network (cGAN) for generating monophonic melodies, conditioned on both prior musical context and an explicit harmonic progression. The proposed architecture, inspired by MidiNet, employs a dual conditioning mechanism where the generator receives the preceding musical bar for temporal continuity and a chord vector for harmonic guidance. A key contribution is a robust data preprocessing pipeline that automatically extracts melody-chord pairs from a heterogeneous MIDI dataset (the Lakh dataset) using a series of heuristic filters and a scoring system. We present quantitative results based on a set of heuristic music theory metrics and qualitative analysis of generated samples, demonstrating the model's ability to produce musically plausible and harmonically appropriate melodies, without relying on discriminator capabilities. A novel contribution is the introduction of ResNet style Identity Blocks in the generator and discriminator to master the complexity and power of the networks, for better balance control.**

***Index Terms*—Generative Adversarial Networks, Deep Learning, Symbolic Music Generation, Conditional GAN, MIDI, Melody Generation.**

I. Introduction

Algorithmic music composition, particularly through deep learning, has emerged as a vibrant field of research, aiming to create models capable of generating novel and aesthetically pleasing musical pieces. Generative Adversarial Networks (GANs) [1] have shown considerable promise in this domain, leveraging their adversarial training process to learn complex data distributions. This work is situated in the context of symbolic music generation, where the goal is to produce music in a structured format like MIDI, focusing on the generation of coherent melodic lines.

While early generative models often produced musically interesting textures, they typically lacked long term structure and user control, making them unsuitable for practical composition workflows. A significant advancement was the introduction of conditioning mechanisms, allowing the generation process to be guided by external information. Models like MidiNet [2] demonstrated the efficacy of conditioning on previous musical bars and underlying chord progression to ensure temporal continuity. Most models either ignore harmony or learn it implicitly, which can lead to musically dissonant or structurally weak outputs.

[†]Department of Civil Engineering, University of Padova,
email: {name.surname}@studenti.unipd.it
This work is a project for the Neural Networks and Deep Learning course.

In this paper, we propose a conditional Generative Adversarial Network (cGAN) that addresses this limitation through a dual-conditioning mechanism. Our model generates a monophonic melody bar by bar, conditioned on two distinct inputs: 1) the preceding musical bar, providing temporal context, and 2) an explicit chord progression vector, providing harmonic context. The generator architecture incorporates residual identity blocks, inspired by ResNet [3], to facilitate the training of a deep convolutional network and easily tune the power of the network. A significant part of our contribution is a comprehensive preprocessing pipeline designed to automatically extract melody-chord pairs from complex, polyphonic MIDI files (specifically, the Lakh MIDI dataset [4]), enabling the training of our model on a diverse dataset. The applicability of this work lies in providing composers and musicians with a tool for generating harmonically aware melodic ideas that can inspire the composition of musical pieces.

The main contributions of this paper can be summarized as follows:

- **Automated Preprocessing Pipeline:** We developed and implemented a multi-stage heuristic pipeline to automatically identify and extract monophonic melody lines and their corresponding chord progressions from a large, unstructured MIDI dataset (Lakh).
- **Dual Conditioned GAN Architecture:** We present a cGAN architecture where the generator is conditioned on both temporal context (the previous bar) and harmonic context (a chord vector), enabling controllable melody generation. The generator leverages ResNet style identity blocks for improved the learning.
- **Human based Evaluation:** We defined a custom set of quantitative and objective metrics based on music theory principles (harmony, melodic range, contour, and motif structure) to evaluate the musical quality of the generated outputs against a validation set.
- **Adaptive training step balancing:** We implemented an automatic tuning of the number of training steps for the generator and the discriminator, in order to dynamically maintain a balanced training process.

This report is structured as follows. In Section II we describe the state of the art. The data model and preprocessing pipeline are presented in Section III and IV. The proposed learning framework is detailed in Section V, and its performance evaluation is carried out in Section VI. Concluding remarks are provided in Section VII.

II. Related Work

The application of deep learning to symbolic music generation has a rich history. Early approaches often relied on Recurrent Neural Networks (RNNs) and their variants, such as LSTMs, to model the sequential nature of music [5]. While capable of learning short-term dependencies, these models often struggled with maintaining long-term coherence.

Generative Adversarial Networks (GANs) [1] provided a new paradigm. MuseGAN [6] was a notable early work that used GANs to generate multi-track piano music, demonstrating the ability to learn complex polyphonic structures. However, like many early GANs, it was an unconditional model, offering little direct control over the output. The need for controllability led to the development of conditional GANs (cGANs) [7] for music. For instance, C-RNN-GAN [8] combined adversarial training with an RNN structure to generate classical music conditioned on composer style.

Our work is most directly inspired by MidiNet [2], which introduced a fully convolutional GAN architecture for symbolic music generation. A key innovation of MidiNet was its conditioner network, which allowed the generator to be conditioned on previous bars of music and chord progression, enforcing temporal continuity and harmonic coherency. The model treated the piano roll as an image, applying 2D convolutional operations for both generation and discrimination. This approach proved effective for learning local patterns and textures.

We retain the temporal conditioning on the previous bar and a second input vector representing the target chord for the current bar. This allows for explicit harmonic control, guiding the generator to produce notes that are consonant with the underlying harmony. Furthermore, our generator architecture enhances the convolutional structure with ResNet-style identity blocks [3], which have been shown to improve gradient flow and enable the training of deeper networks, a feature not present in the original MidiNet architecture. By combining these elements with a robust data extraction pipeline, adaptive training step balancing and human based evaluation, our work aims to advance the state of harmonically aware melody generation.

III. Processing Pipeline

The proposed system follows a structured pipeline from raw data ingestion to melody generation and evaluation. The pipeline is designed to be fully automated, minimizing the need for manual data annotation, especially when working with large and varied collections like the Lakh MIDI dataset. We have used the *pretty_midi*[1] library as the main tool to process Midi files. The main stages are as follows:

1) **Data Ingestion:** A dataset of polyphonic MIDI files (in this case, the Lakh 'clean_midi' dataset) serves as the raw input. These files contain multiple instrumental tracks with varying musical roles.
2) **Preprocessing and Feature Extraction:** This is the most critical stage for preparing the training data. For each MIDI file, a custom pipeline is executed:
   - **Percussive Track Filtering:** Drum and percussive tracks are identified and removed using metadata (e.g., 'is˙drum' flag) and heuristic analysis (e.g., low pitch variety).
   - **Melody Candidate Identification:** The remaining tracks are filtered to identify potential melody candidates. This involves excluding tracks based on name (e.g., 'bass', 'pad'), low pitch range, low note density, and high polyphony.
   - **Melody Scoring and Selection:** A scoring system evaluates each candidate track based on factors like instrument type (favoring common lead instruments), note density, and pitch range. The highest-scoring track is selected as the primary melody line.
   - **Melody Cleaning:** The selected melody track is processed to ensure it is monophonic (only one note per time step, keeping the highest pitch) and that very short, likely artifact notes are removed.
   - **Harmonic Analysis:** The tracks not selected as percussion are used to derive an explicit chord progression. A chromagram is computed for each bar, and the most prominent chord (major or minor) is identified via template matching.
3) **Data Representation:** The extracted features are converted into a numerical format suitable for the neural network. The monophonic melody is represented as a binary piano roll, and the chord progression is encoded into a 13-dimensional vector (12 for the root note, 1 for the quality).
4) **Data Augmentation:** To increase the diversity of the training set and improve model generalization, augmentation techniques are applied, most notably random pitch shifting, which shifts both the melody and its corresponding chord progression.
5) **Model Training:** The processed data is fed into the cGAN. The generator learns to produce a new melody bar conditioned on the previous bar and the target chord, while the discriminator learns to distinguish real melody chord pairs from generated ones.
6) **Evaluation:** The quality of the generated melodies is assessed using a custom heuristic scoring function that evaluates harmonic consonance, melodic range, contour, and motif structure.

This end to end pipeline ensures that a controllable and musically coherent generative model can be trained from a large and varied collection of raw MIDI files.

IV. Signals and Features

A. Data Source and Preprocessing Lakh dataset

The primary data source is the Lakh MIDI Dataset ('clean_midi' subset), a large collection of popular music in

[1]Colin Raffel and Daniel P. W. Ellis. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty˙midi. In Proceedings of the 15th International Conference on Music Information Retrieval Late Breaking and Demo Papers, 2014.

MIDI format. A rigorous preprocessing pipeline was developed to extract clean, paired melody and chord data.

1) **Melody Extraction:** Since most MIDI files are polyphonic, a heuristic-based approach was developed to isolate the main melody line. Tracks are first filtered out based on both metadata, name keywords, note density, vocal pitch band, and polyphony. The remaining tracks are considered melody candidates. Each candidate is then scored based on a weighted combination of factors:
   - **Instrument Type:** Higher scores are given to common lead instruments (e.g., vocals, lead synth, strings).
   - **Note Density:** Melodies typically have a moderate to high density of notes.
   - **Mean Pitch:** Melodies usually occupy a higher pitch range than accompaniment.
   - **Pitch Standard Deviation:** A wider pitch range is favored.

   The candidate tracks are made monophonic by keeping only the highest note at each time step, and short artifact notes are filtered out. Then, the track with the highest composite score is selected as the melody.
2) **Chord Extraction:** The non percussive tracks are assumed to constitute the harmony. For each bar of music, we compute an aggregated chromagram from these harmonic tracks. A cosine similarity measure is used to match this chromagram against predefined templates for all 12 major and 12 minor chords. The chord with the highest similarity is assigned to that bar. If the highest similarity is below a threshold, we assign to the bar a "N.C" to preserve the temporal alignment.

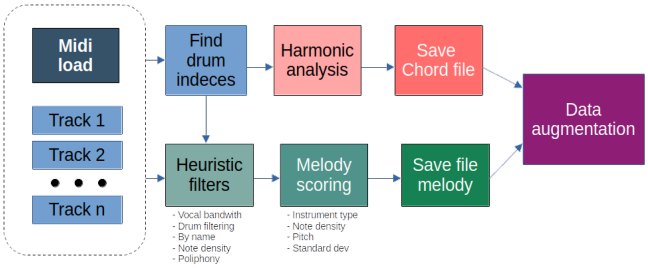


Fig. 1: Preprocessing Pipeline.

B. Feature Vector Representation

The extracted musical information is formatted into tensors for the model.

1) **Melody Representation:** The monophonic melody is converted into a binary piano roll format. Each melody is chunked into segments of `nbar` bars. A single bar is represented by a tensor of shape `(1,128,16)`, where 128 is the number of possible MIDI pitches and 16 represents the temporal resolution (16th notes). A value of 1 indicates a note presence at that pitch and time step. The full segment thus has a shape of `(1,128, nbar×16)`.
2) **Chord Representation:** The chord progression is encoded into a tensor of shape `(13, N)`, where N is the number of bars. For each bar, a 13 dimensional vector is created: the first 12 dimensions are a one-hot encoding of the chord's root note (C, C#, ..., B), and the 13th dimension represents the quality (1.0 for major, 0.0 for minor). "No Chord" (N.C.) is represented by a zero vector.

C. Preprocessing Hooktheory dataset

We also use the Hooktheory[2] dataset. This particular collection includes 550 Tabs, which are pop songs, all in 4/4 time. We chose this dataset because it gives us direct access to high-quality musical data.

A key benefit of these files is how they're set up: the first MIDI track always contains the melody, and the second track provides the chord progression in MIDI format. This arrangement means we don't need to spend time cleaning up or extracting the melody, which really simplifies our process. All the other steps we take are the same as those for our other datasets, keeping everything consistent.

D. Dataset Split and Augmentation

The final dataset consists of pairs of (`nbar`-bar melody piano roll, `nbar`-bar chord vector). This dataset is split into training (80%) and validation (20%) sets. To enhance robustness, we apply data augmentation to the training set, primarily through Random Pitch Shift. This transformation randomly transposes a sample's melody piano roll by a certain number of semitones (between -6 and +6) and applies the corresponding circular shift to the root note dimension of its chord vector, ensuring the harmonic relationship is preserved. Moreover, we also use also Random Note Shift, where single notes are shifted by two semitones with low probabilitiy, and Random Drop Notes, where some single notes are turned off.

V. Learning Framework

The core of our approach is a conditional Generative Adversarial Network (cGAN) with a ResNet-inspired architecture, designed to learn the complex relationship between melody, harmony, and temporal context.

A. Generator Architecture

The Generator (G) is responsible for creating a new melody bar. Its architecture is a deep convolutional network using transposed convolutions for upsampling and identity blocks for stable training.

- **Inputs:** G takes three inputs:
  1) A latent noise vector **z** (e.g., of size 100), providing a source of randomness.
  2) The piano roll of the previous bar, x_{prev} of shape `(B, 1, 128, 16)`, which serves as the temporal condition.

[2]https://www.hooktheory.com/theorytab

3) The target chord vector, **chord**, for the bar to be generated, of shape `(B, 13, 1)`, which serves as the harmonic condition.

- **Architecture:** The noise vector **z** is first projected by a fully connected layer into a spatial tensor. The temporal condition x_{prev} is processed by a separate conditioner network (a series of convolutional layers) to create feature maps at multiple resolutions. At each stage of the generator's upsampling path (composed of `ConvTranspose2d` layers), the feature maps from the conditioner and a tiled representation of the chord vector **chord** are concatenated to the main path's feature maps. This injects the conditioning information at multiple scales.
- **Identity Blocks:** Before each of the upsampling stages, we employ ResNet-style `IdentityBlock` modules. Every block is composed of 3 convolutional layers with Batch Normalisation and ReLU activation functions. These blocks, containing a shortcut connection that bypasses a series of convolutional layers, help to prevent vanishing gradients and allow for a deeper, more expressive network. We use these blocks in order to easily tune the power of the generator while preserving the correct dimensionalities.
- **Output:** The final layer uses a `Softmax` activation function across the pitch dimension. This forces the output to be monophonic, as it encourages only one pitch to have a high activation at each time step, which is then binarized to produce the final piano roll of shape `(B, 1, 128, 16)`.

B. Discriminator Architecture

The Discriminator (D) is a standard convolutional network responsible for distinguishing real melody-chord pairs from fake ones.

- **Inputs:** D receives two inputs:
  1) A `nbar`-bar piano roll, **X**, of shape `(B, 1, 128, nbar×16)`, which can be either a real sample from the dataset or a composition of generated sample from G.
  2) The corresponding `nbar`-bar chord progression, **chord**, of shape `(B, 13, nbar)`.
- **Architecture:** The chord progression **chord** is tiled and reshaped to match the spatial dimensions of the piano roll **X**. It is then concatenated as an additional channel to **X**. This combined tensor is processed through a series of `Conv2d` layers with `LeakyReLU` activations, spectral normalization, and dropout. The final layers flatten the feature map and pass it through a fully connected layer to produce a single logit, representing the probability of the input being real.

C. Training Strategy

The models are trained using the Adam optimizer and a binary cross-entropy (BCE) loss function. To stabilize training, we employ two common techniques:

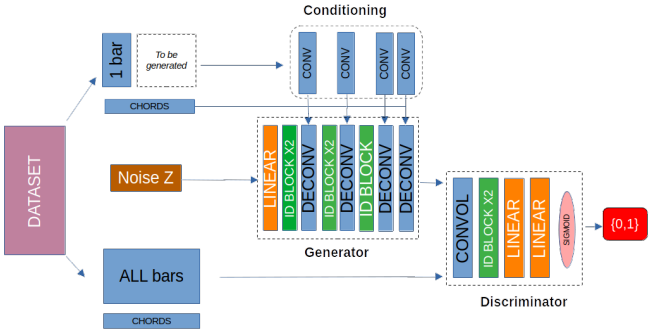


Fig. 2: Network architecture.

- **One-Sided Label Smoothing:** When training D on real samples, we use a target label of 0.9 instead of 1.0. This prevents the discriminator from becoming overly confident and helps to regularize the model.
- **Discriminator Warmup:** Before starting the adversarial training, the discriminator is pre-trained for a few epochs on real samples and simple noise. This ensures that it provides meaningful gradients to the generator from the beginning of the training process.
- **Regularization:** We implemented feature matching to regularize the generator. This technique involves incorporating additional L2 regularizers into the generator's loss function. The core idea is to encourage the generated data to closely resemble the real data by minimizing the L2 distance between their respective feature representations. The specific hyperparameters for this regularization were determined empirically as (100, 20) to be of a comparable order of magnitude as the generator's loss.

During the main training loop, the discriminator and generator are updated with the possibility of performing multiple discriminator updates for each generator update to maintain a healthy balance in the adversarial game.

VI. Results

This section presents the evaluation of our proposed model. We assessed its performance through both quantitative metrics derived from our custom scoring function and qualitative analysis of the generated musical outputs.

A. Experimental Setup

The model was trained on the HookTheory dataset or the dataset extracted from the Lakh, specifically from the Clean-MIDI subset. The hyperparameters were selected using the Optuna framework, which guided the search for an optimal configuration, where we have optimized it on maximizing "Human score". The final parameters used for training were:

- Generator Learning Rate: 2×10^{-4}
- Discriminator Learning Rate: 2×10^{-4}
- Optimizer: Adam
- Batch Size: 32
- Latent Dimension (z): 100
- Training Epochs: 50

The model was trained on a single NVIDIA T4 GPU provided by Google Colab. The discriminator was pre-trained for 4 epochs before starting the main adversarial training loop.

We performed four types of melody generation:

- unconditioned,
- conditioned only on chord (1d condition)
- conditioned only on previous bar (2d condition)
- conditioned on both chord and previous bar (1d and 2d condition)

We generate `nbars`=4 for all the models except fot the unconditioned one, where we use a value of one. The generator is able to generate one bar at the time and the function `make_bars` creates and concatenates as many bars as needed.

B. Quantitative Evaluation

To provide an objective measure of quality, we developed a heuristic scoring function, 'score_melody'. Tab. 1 shows the average scores for melodies generated by the best model after training, compared to the scores of real melodies from the validation set (derived from the datasets as a benchmark).

Model condition	Hooktheory	Lakh Clean midi
None	0.5025	0.4630
Chords	0.4823	0.4617
Previous bar	0.5398	0.5284
Both	0.5680	0.5018
Real data	0.6159	0.5569

TABLE 1: Comparison of average heuristic scores for generated melodies trained using different datasets. Higher is better.

We can see that the model with higher scores is the one with both conditioning strategies. Moreover, Table 1 shows that data on Hooktheory are cleaner than the one obtained from the Lakh dataset, and this allows a better learning process.

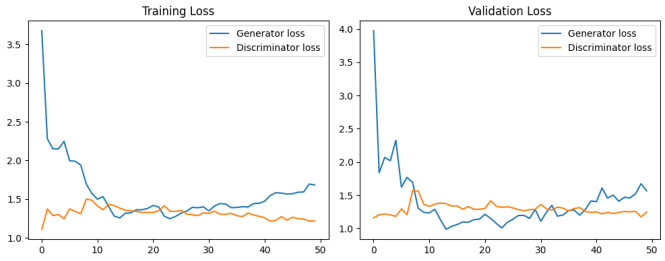


Fig. 3: Example of loss behaviour during the training of both conditioned model, using Hooktheory dataset.

From Figure 3, we can notice that the generating training loss decrease during the training and is stable at the last epochs, meaning that the generator and the discriminator are kept sufficiently balanced. Moreover, the validation losses follow the training ones, that assures no overfitting is occurring.

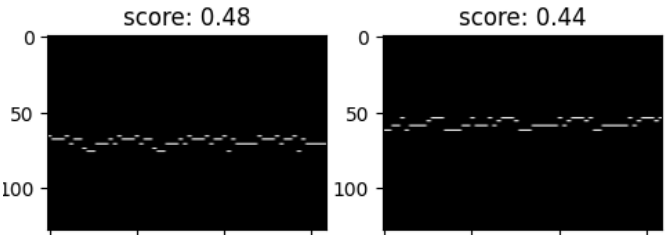
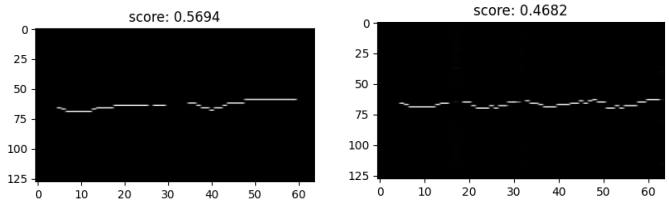


Fig. 4: Examples of midi generated by the both conditioned model. At the top the human score associated.

C. Qualitative Evaluation

Qualitative analysis was performed by visually inspecting the piano rolls of fully conditioned generation and listening to the synthesized audio. Figure 5 presents a comparison between a real four-bar melody from the validation set and a melody generated by the model given the same initial bar and chord progression. Visually, the generated piano roll exhibits similar characteristics to the real one, with a clear melodic contour and rhythmic structure. Audibly, the generated melodies are quite consistently on-key and follow the provided chord changes, confirming the success of the harmonic conditioning. The outputs are musically plausible and could serve as a creative starting point for a human composer. Audio examples can be played in the provided notebook. Comparing the four type of melody generation, it is evident that both conditioning influences the melody generation and the 2d conditioning is stronger. The constraints of the harmonic progression and previous bar shows an improving of the "human" score as expected. This comparison validates the use of this objective score, not relying on the discriminator capabilities.



(a) Example of a dataset melody (b) Example of generated melody

Fig. 5: Comparison between a genuine melody (a) and a generated melody (b).

VII. Concluding Remarks

In this report, we presented a conditional GAN with a ResNet inspired architecture for generating symbolic melodies. Our primary contribution is the improvement of a dual conditioning mechanism that guides the generative process using both immediate temporal context and an explicit harmonic progression. This approach, combined with a robust automated data preprocessing pipeline, , adaptive training step balancing and human based evaluation, allows for the training of a model capable of producing musically coherent and melodic outputs from a diverse MIDI dataset.

Our results demonstrate that the model effectively learns to adhere to the given harmonic constraints while maintaining local melodic continuity. The generated melodies are musically quite plausible and achieve high scores on our heuristic evaluation metrics. The main applicability of this work is as a tool for musicians and composers, offering a way to quickly generate melodic ideas over pre defined chord progressions, thereby augmenting the creative process.

The primary limitation of the current model is its short-term memory, as it is only conditioned on the single preceding bar. This restricts its ability to develop long-range musical structures and motifs. Future work could address this by replacing the convolutional conditioner with architectures better suited for long sequences. Additionally, the heuristic based data extraction and evaluation, while effective, could be improved by incorporating more sophisticated music information retrieval techniques or by training dedicated models for these tasks.

Furthermore, the model's capabilities could be extended by incorporating new conditioning techniques, allowing for control over generated music based on genre or artist specific styles.

From this project, we have learned the difficulties of GAN training, including the importance of stabilization techniques like label smoothing, spectral normalization, dropout regularization and architectural choices like residual connections. We also gained significant experience in MIDI data processing and the challenges of extracting meaningful features from unstructured symbolic music. The main difficulties encountered were the inherent instability of adversarial training, which required careful hyperparameter tuning, and the complexity of designing a reliable pipeline for automatically separating melody from harmony in polyphonic music files.

VIII. Contributions

This project was mostly completed by both team members through Zoom video calls. Federico mainly handled the Lakh dataset preprocessing and adaptive training, while Lorenzo focused on data augmentation, Hooktheory preprocessing, the discriminator warmup and the "human" scoring. Lorenzo contributed more to the programming side, and Federico brought a more extensive musical background to the project. Gemini 2.5 Pro was used for assistance in refining the language and improving the overall clarity of this manuscript and for coding assistance in Colab. All content was thoroughly reviewed, edited, and verified by the authors.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pp. 2672–2680, 2014.

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

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

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

[5] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, no. 4, pp. 48–56, 2002.

[6] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and 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*, vol. 32, 2018.

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

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