From Bytes to Beats: Music Generation Using WGAN-GP and Self-Attention Mechanism

Hemangani Nagarajan
University of Massachusetts, Amherst
hemanganinag@umass.edu

Sudharshan Govindan
University of Massachusetts, Amherst
sgovindan@umass.edu

December 16, 2023

Abstract

This project explores the cutting-edge domain of automated music generation, a field at the intersection of artificial intelligence (AI) and artistic creativity. We focus on the challenge of developing algorithms capable of understanding the complexities of music and generating compositions that are both technically sound and aesthetically appealing. We utilize Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP), enhanced with a self-attention mechanism, to address the challenge of creating algorithms that can understand and replicate music's complexities. Our approach is tested using the Lakh MIDI dataset comprising 17 diverse instrumental recordings across various genres. This rich dataset facilitates an in-depth exploration of generating intricate and coherent musical compositions.

1. Introduction

Automated music generation represents a fascinating and innovative area of AI research, where the technical prowess of machine learning meets the creative essence of musical composition. This field presents a unique challenge: to devise algorithms that not only grasp the subtleties of music but also produce technically proficient and aesthetically pleasing compositions.

In our work, we have adapted the WGAN-GP framework, renowned for its enhanced training stability, to better suit the intricate demands of music creation. A key innovation in our approach is the integration of a self-attention mechanism. This addition allows our model to capture long-range dependencies within musical sequences, a critical aspect often overlooked in traditional models. By doing so, we aim to generate melodies that are aesthetically pleasing and exhibit consistent and complex musical structures over extended sequences. This addresses a notable challenge in AI-generated music, where maintaining coherence over longer durations can be particularly challenging.

We use a combination of subjective and objective metrics to evaluate our model's music generation capabilities. Human listener feedback assesses the compositions' appeal. At the same time, technical measures like Polyphonic Rate, Self-Similarity Matrix provide insights into the complexity, structure, and pitch variety of AI-generated music. Collectively, these metrics offer a holistic view of the model's performance.

The potential applications of our model are significant, offering valuable tools for composers and musicologists. The use of WGAN-GP in music generation, enhanced with a self-attention mechanism, represents a significant stride in the quest to enable AI systems to emulate human artistic creativity.

Our research leverages the Lakh MIDI dataset, an extensive collection of audio recordings featuring various instruments and genres. This dataset, comprising 17 distinct instrumental tracks, provides a comprehensive platform for our exploration into AI-based music production. Each track in the dataset offers unique harmonic and rhythmic challenges, enhancing our understanding of the intricacies involved in generating AI-based music. The diversity of instruments, including guitar, drums, piano, and more, allows us to investigate the nuances of various musical elements in AI-generated compositions. Through this project, we aim to explore the limits of AI in creating individual melodies and orchestrating these elements into complex, coherent musical pieces, thereby advancing the field of automated music generation.

2. Problem Statement

Recent breakthroughs have greatly influenced the field of automated music generation in artificial intelligence. Conventional techniques frequently face difficulties in producing original and harmonically cohesive music, particularly when working with several instruments.

The Lakh MIDI dataset offers unparalleled possibili-

ties for AI-based music production but also brings distinct obstacles. An obstacle that arises is ensuring that the music produced by AI retains a coherent and structurally sound composition, particularly when handling various instruments and genres. Ensuring the preservation of temporal coherence in generated music poses a significant difficulty. The model must acquire the ability to generate both harmonically and rhythmically pleasant music at a specific point in time and sustain these qualities consistently throughout the composition. Training Generative Adversarial Networks (GANs), particularly on intricate tasks such as music production, frequently encounters instability and can result in problems such as mode collapse.

Although there has been extensive research on employing neural networks to generate music, the particular use of WGANs with GP, along with self-attention to generate multi-instrumental music from the Lakh MIDI dataset, has not been fully explored. This gap provides an opportunity to examine how the Wasserstein loss with gradient penalty can enhance the quality of generated music, specifically in a multi-instrumental setting.

The primary object of our study revolves around creating and refining a WGAN-GP model that can effectively handle the complexity and diversity of the Lakh MIDI dataset and ensure stability while training. Our methodology entails instructing the model to identify and reproduce the unique attributes of each instrument and the interaction among them, therefore facilitating the creation of melodious and dynamic compositions.

3. Literature Survey

In our project on music generation using GANs with a MIDI dataset, we drew inspiration from two key papers in the field: "MIDINET: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation"[10] and "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment."[2] While these papers provided a solid foundation in the use of GANs for music generation, our approach diverged by incorporating Deep Convolutional GANs (DCGAN) and Wasserstein GAN with Gradient Penalty (WGAN-GP).

In the initial stages, we experimented with a hybrid model combining Long Short-Term Memory (LSTM) networks and 1D convolutional layers, aiming to leverage LSTM's temporal data handling and the convolutional layer's sequential data processing capabilities. However, this approach led to significant mode collapse, prompting us to seek more robust solutions.[7]

We chose DCGAN for its stability in training and its proficiency in handling the high-dimensional data characteristic of MIDI files. This architecture, known for generating coherent and structurally sound outputs, was ideal for capturing the complex patterns in music sequences. The convolutional layers in DCGAN were particularly effective in learning and replicating the intricate structures found in musical compositions.

To further enhance our model, we integrated WGAN-GP, addressing the common challenges of mode collapse and training instability in traditional GANs. This was a crucial step in ensuring the diversity and quality of our generated music. WGAN-GP's gradient penalty term was pivotal in stabilizing the training process, allowing our model to explore a wider variety of musical styles without sacrificing realism.

4. Technical Approach

In this section, we present our technical strategy for the use of Generative Adversarial Networks (GANs) in the realm of automated music composition. We detail the architecture of our model, the training process, and the dataset that informs our network, setting the stage for a deeper dive into the mechanics of our approach and the future directions we intend to explore.

4.1. Discriminator/Critic Network

In a standard GAN[3], the Discriminator generates a probability value (using a sigmoid activation function) representing the likelihood of a particular input being either real or fake. In the case of WGAN, the Discriminator in our model (also known as critic) generates a score (without sigmoid) that indicates the degree of genuineness or falsity of the input. This modification is essential for the effective functioning of the Wasserstein loss.

This discriminator architecture utilizes convolutional layers, incorporating Parametric Rectified Linear Units (PReLU) as an activation function. PReLU offers greater flexibility than traditional ReLUs, enhancing the model's ability to handle non-linearities and stabilizing the learning process.

To normalize the feature maps and accelerate convergence, batch normalization[5] (BatchNorm2d) is applied to the convolutional neural network layers before the activation function (PReLU). Additionally, Self Attention[9] is applied after a certain number of convolutional layers. It helps the Discriminator to differentiate between real and generated music more effectively by focusing on key features in the input pianorolls. Following this, the output is channeled through a linear layer, which produces the final discrimination score.

4.2. Generator Network

The Generator's architecture incorporates Transpose Convolutional 2D (Transpose CNN 2D) layers, specifically designed to upscale the input and efficiently generate sequential data. The network incorporates activation functions,

such as PReLU and sigmoid, to introduce non-linearity and accurately map the outputs to the intended range [0,1].

Like its counterpart in the Discriminator, batch normalization and Self Attention is also key component within the Generator's layers. Batch normalization ensures consistent scales of activations throughout the network and effectively reduces internal covariate shift, leading to smoother and faster convergence. Batch normalization, combined with the Transpose CNN 2D layers, facilitates the network's ability to expand the noise input into a more structured and complex output, in this case, a piano roll. Self-attention assists in synthesizing more realistic and complex music. It allows the Generator to focus on specific aspects of the music sequence, such as melody or rhythm, thereby enhancing the authenticity of the generated tracks.

4.3. Wasserstein Loss

WGAN-GP network introduces a critical modification to the original GAN loss function called Wasserstein Loss, which measures the Earth Mover's distance between the distribution of generated data and real data. This loss provides smoother gradients and a more meaningful measure of distance between distributions compared to traditional GAN loss functions.[1][4]

The loss functions for the discriminator (critic) D and the generator G in a WGAN are defined as follows:

Discriminator Loss:

$$L_D = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] \tag{1}$$

Generator Loss:

$$L_G = -\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] \tag{2}$$

Where:

E represents the expectation.

$\mathbb{P}_r$ is the real data distribution.

$\mathbb{P}_g$ is the generator's data distribution.

$\tilde{x}$ are samples generated by the generator.

x are real data samples.

4.4. Gradient Penalty

Although traditional WGANs are effective, they may encounter challenges during training. Incorporating the Gradient Penalty (GP) into the WGAN architecture enhances the stability of the training process.

The gradient penalty (GP) is a technique used to ensure smooth gradient transitions in the Discriminator. It achieves this by incorporating a penalty term into the Discriminator's loss function. This penalty term is directly proportional to the square of the difference between the gradient norm and its target value, typically set to 1. The inclusion of this regularization term effectively mitigates the occurrence of significant updates to the Discriminator, resulting in a more consistent and reliable training process. The penalty term is formulated as follows:

$$GP = \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1)^2] \tag{3}$$

Where:

λ is the penalty coefficient

E denotes the expectation over the interpolated samples $\hat{x}$.

$\nabla_{\hat{x}}D(\hat{x})$ represents the gradient of the D's output w.r.t $\hat{x}$.

$\|\cdot\|_2$ is the L2 norm.

4.5. Dataset

The LPD-17 dataset is an extensive collection of musical compositions formatted as seventeen-track pianorolls. It is derived from the Lakh MIDI Dataset (LMD)[8], specifically designed for tasks in automatic music generation and analysis. The dataset offers a unique representation of MIDI files, making it highly suitable for training generative models.

In the LPD-17 dataset, each track belongs to one of seventeen categories based on the instrument family. This categorization aligns with the program numbers in the MIDI files and the specifications of General MIDI Level 1. The seventeen tracks represented are:

1. Drums
2. Piano
3. Chromatic Percussion
4. Organ
5. Guitar
6. Bass
7. Strings
8. Ensemble
9. Brass
10. Reed
11. Pipe
12. Synth Lead
13. Synth Pad
14. Synth Effects
15. Ethnic
16. Percussive
17. Sound Effects

We choose 1000 such piano rolls from the 'rock' genre, offering a substantial variety of instruments for training purposes. Post-training, the model can predict each of the 17 channels independently. These predicted channels are then synthesized to form a single, cohesive music track. The model underwent training for 100 epochs, using the structure of the LPD-17-Cleansed pianorolls to learn the generation of diverse musical tracks. Each pianoroll's multi-instrumental nature allowed the model to understand and

Figure 1. Architecture of Discriminator network

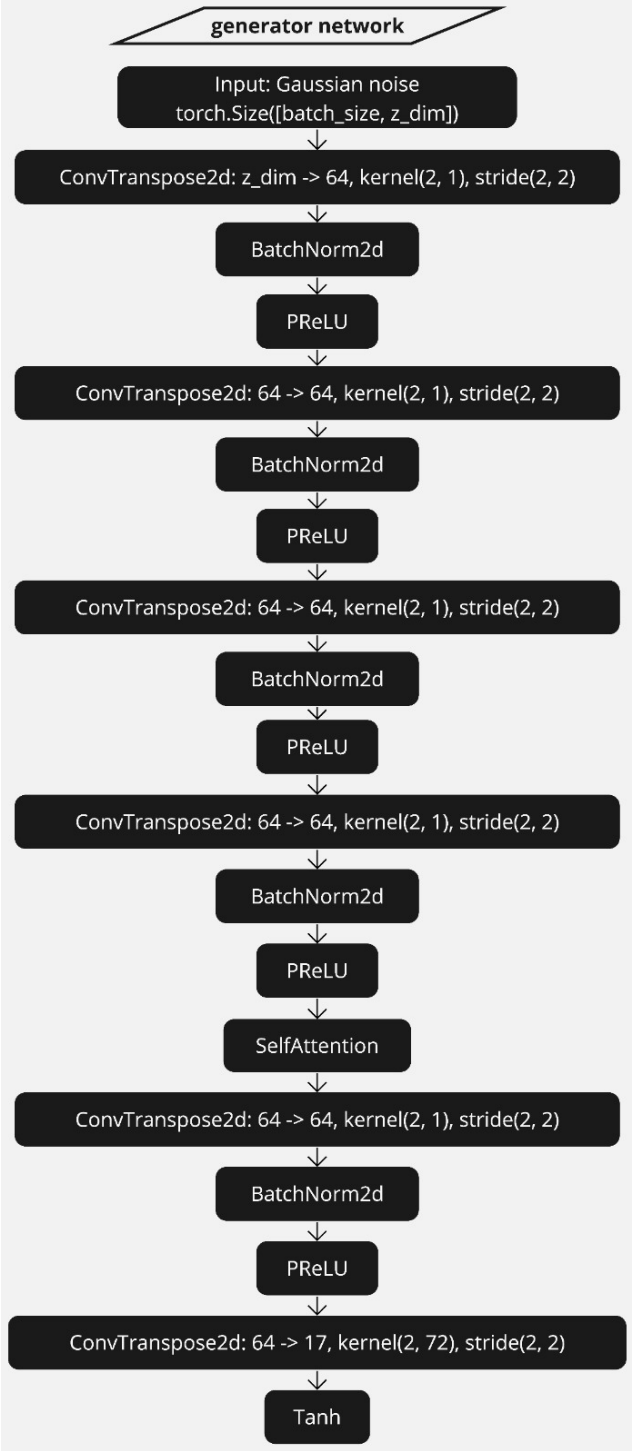

Figure 2. Architecture of Generator network

predict intricate musical patterns across various instrument families.

4.6. Training

In our project, we employed the model utilizing the advanced computational capabilities of an Nvidia T4 GPU with 15GB of GPU RAM, complemented by 50 GB of CPU RAM. The generator and discriminator are initialized and transferred to GPU. This setup was crucial for handling the large-scale data efficiently.

Adam optimizers[6] are used for both the generator and discriminator, with distinct learning rates (0.0003 for the generator, 0.0006 for the discriminator) and betas parameters (0.5 and 0.9). These settings were carefully chosen to balance training stability and convergence speed.

Step decay learning rate schedulers are applied to both optimizers. The learning rates are reduced by a factor of 0.75 every 25 epochs, aiding in fine-tuning the models as training progresses.

The discriminator is trained to differentiate between real and generated images, providing a measure of the generated images' authenticity relative to real data, using a Wasserstein loss with gradient penalty. This loss is computed and backpropagated to update the discriminator.

The generator is updated after every ncritic iterations of discriminator training. The variable, ncritic, is pivotal as it determines the frequency at which the generator is trained in relation to the discriminator, ensuring a balanced learning process. The generator aims to produce images that are classified as real by the discriminator, with its loss being the negation of the discriminator's output.

For each batch, the generator and discriminator losses are recorded, offering insight into the training dynamics. Every 10 epochs, produced images are saved alongside real ones for qualitative analysis. Every 20 epochs, model states and loss data are stored. This not only makes progress monitoring easier, but it also allowed us to resume training from these checkpoints if necessary.

5. Results

Figures above shows the loss curves for the generator and discriminator of a normal GAN during training. These curves provide valuable insights into the training dynamics and stability of our model, illustrating how the generator and discriminator evolved over time to produce the final results. Throughout the epoch, the generator loss fluctuates, which is typical in GAN training due to the adversarial nature of the generator and discriminator learning process and the loss of the discriminator reduces as we train more. We can see smoother loss graph when WGAN-GP is used.

Figure 6. shows a continuous range of generated values from the generator represented in reversed grayscale, where pure black represents a value of 1 (note on) and pure white represents a value of 0 (note off), with various shades of gray indicating values in between. Figure 7. shows an image that is a binary thresholded version, where all activations below 0.9 have been set to 0. Only the values at or above 0.9 are left as black, representing a note being played.

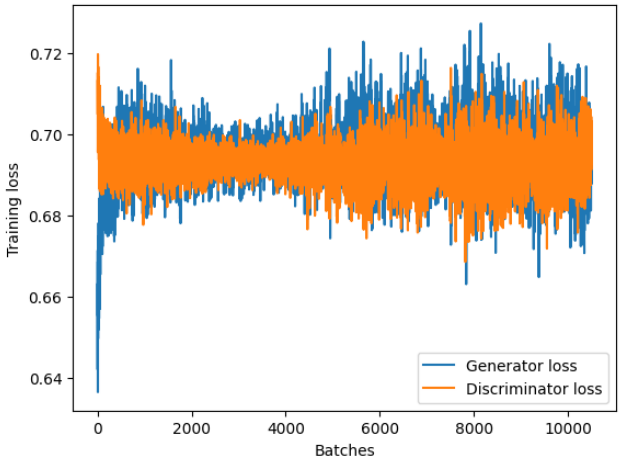

Figure 3. Generator and Discriminator Loss training with GAN

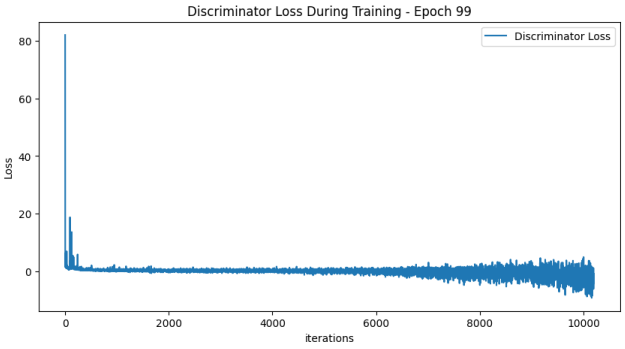

Figure 4. Discriminator loss of WGAN-GP Network

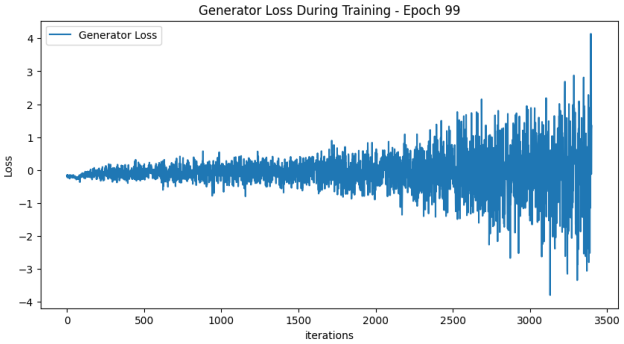

Figure 5. Generator loss of WGAN-GP Network

We present a detailed analysis of the performance of our AI model in music generation, based on the rigorous evaluation metrics we employed. We have implemented a multi-faceted evaluation approach. This includes gathering feed-

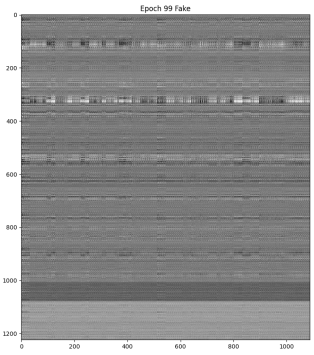

Figure 6. Generated Piano Rolls without threshold

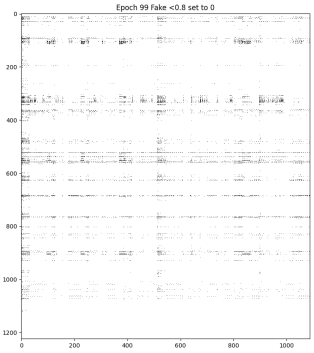

Figure 7. Generated Piano Rolls with threshold

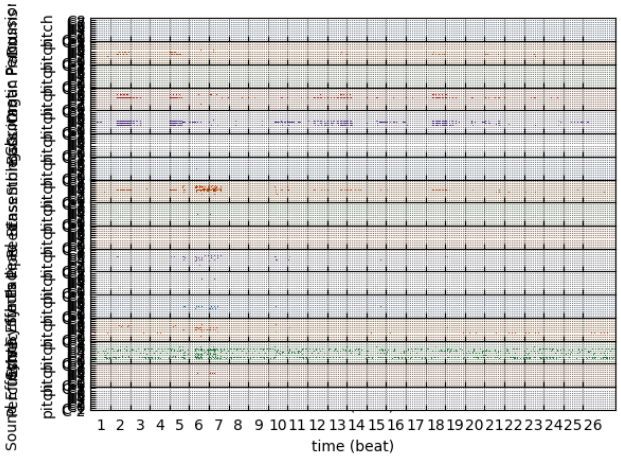

Figure 8. Generated Multitrack

back from human listeners, who provide subjective insights into the quality and appeal of the AI-generated compositions. Additionally, we measure the Polyphonic Rate to understand the complexity and richness of the harmonies produced. The Self-Similarity Matrix is employed to analyze the structural patterns and repetitions within the music, offering a quantitative view of the compositions' internal coherence. These diverse metrics collectively offer a comprehensive evaluation of our model, capturing both the technical proficiency and the artistic quality of the AI-generated compositions.

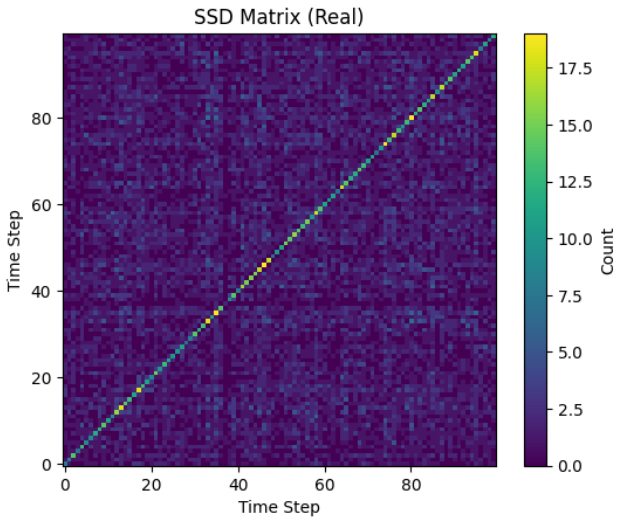

Figure 9. Real Self-Similarity Matrix

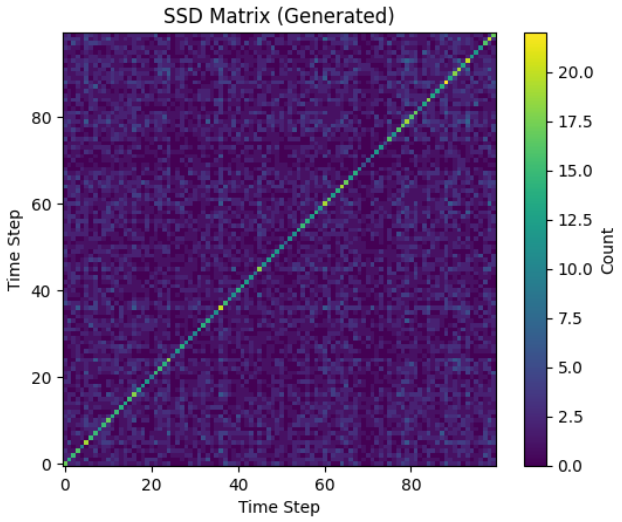

Figure 10. Generated Self-Similarity Matrix

A key part of our evaluation involved human listeners. We conducted a study with 20 individuals, asking them to differentiate between AI-generated and real musical compositions. Remarkably, only 20% of the participants were able to correctly identify the AI-generated pieces, suggesting that our model's outputs are quite convincing and closely resemble human-composed music. Furthermore, 90% of these listeners reported finding the AI-generated music pleasing, a strong indicator of the aesthetic quality of our model's outputs.

In the evaluation of generative models for music, Self-

Similarity Matrices provide a quantitative visual tool to assess the structural learning and replication capabilities of the model. Firstly, the presence of a clear main diagonal indicates that the model has effectively learned a degree of temporal coherence; the generated music has a self-consistent structure at individual time steps, which is fundamental for any musical piece. Furthermore, the diffused nature of off-diagonal patterns, while less structured than in the real data Self-Similarity Matrix, suggests that the generated music does possess variation and avoids repetitive monotony, which could be seen as a creative exploration beyond simple mimicry of the training data. This indicates that the model is not overfitting and is capable of generating novel content that, while it may currently lack some structural complexity of real-world music, provides a foundation for further refinement towards more complex composition generation.

The polyphonic rate measures the average number of simultaneous notes per time step, providing insight into the complexity and richness of a musical piece. The real data's polyphonic rate of 1.237 suggests that, on average, there are about 1.24 notes played together at any given time. In comparison, the generated data has a slightly higher polyphonic rate of 1.296, indicating that the generative model tends to produce compositions with a marginally higher degree of note overlap. This could imply that the model has learned to create more complex, layered music, which is a positive sign of its ability to capture and replicate the polyphonic nature of the training dataset.

To give a more tangible sense of the music generated by our model, we have also included Generated Piano Rolls images and music samples in our results section. These visual and auditory representations allow readers to directly experience the quality and complexity of the music produced by our AI system. The inclusion of these samples not only demonstrates the capabilities of our model but also provides a more engaging and comprehensive understanding of our research outcomes.

Link to video - ***LINK***

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. 3

[2] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017. 2

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. 3

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 2

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[7] Sanidhya Mangal, Rahul Modak, and Poorva Joshi. Lstm based music generation system. *IARJSET*, 6(5):47–54, 2019. 2

[8] Colin Raffel. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. 2016. 3

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[10] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation, 2017. 2