Queen Mary, University of London
School of Electronic Engineering and Computer Science

Music Generation using Symbolic Representation with Generative Adversarial Networks

ECS7022P – Computational Creativity

Mughees Asif | 180288337

Due: 15 April 2022

Abstract

Domain-style transfer for text, images, and music, has recently gained traction to enable creative and original digital content generation. The current trends in research for musical generation align with using deep neural networks due to the vast computational power and feasibility for producing varying styles of content. Following this light, the author has investigated and implemented the usage of Generative Adversarial Networks (GANs) with the Wasserstein Loss function to generate original musical snippets using symbolic notation. The dataset was sourced in a piano roll format, a widely used and accepted form of representing various musical tones and pitches. The GAN system was trained and tested in various configurations that show promising results. The generation of the output was aided by adding accompaniments that augment the original input samples with different tones and pitches to produce qualitatively pleasing music content.

Keywords—Content generation, Symbolic music production, Piano roll, Generative Adversarial Networks, Wasserstein Loss

Word Count—2087

Contents

List of Figures

1 Introduction

Deep learning has achieved prominent status not only in prototypical statistical tasks such as classification and prediction, but also has been highly applicable in domains including generation of creative content. Using the methodology of artificial neural networks, various deep learning models have been developed to recognise patterns or discern regularities from different data types for instance text, images, music, and videos. Recently, researchers have focused on music generation using Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to enable music production via symbolic representation, and musical melodies based on emotions [1, 2]. The use of neural networks is warranted when the learning process is reliant on high-order dependencies and is non-investigable using standard analytical formulations, such as grammar-based or rule-based generative systems [3]. However, due to architectural constraints that arise from using these methods such as division of the input into equal parts for sequential iteration or learning difficulties with distributions, researchers have recently focused on connectionist models such as Generative Adversarial Networks (GANs) and Transformers [4].

Generating music requires time sequencing and temporal modelling that can unfold over time separately [4]. The aim of this research was to develop a content generation system with a focus on custom music production. The proposed system draws inspiration from [5], where a GAN was trained on symbolic representations of music snippets, in the format of piano rolls, and was designed to add accompaniments onto input samples to generate creative and original content. For similar work, see [6, 7] where GANs have been exploited to generate melodies and deliver content that is realistic and satisfying to the audience.

2 Methodology

The following section is divided into five main sections that are involved in symbolic music generation using deep learning [3]. The outlined typology will help introduce the pertinent concepts and perspectives needed to design content generation systems based on recognising different musical melodies.

2.1 Objective

The objective of musical content generation is the determination of the nature of the output i.e., monophonic, polyphonic or genre-transfer. This dimension also determines what the output will be used for and how it will be used. For the purposes of this research, the objective can be highlighted as understanding the concept of music generation from a human-machine interface perspective. The research is centred on:

(a) Parsing a given musical sample in the piano roll format.

(b) Breaking down the sample into multi-track piano rolls with different tones.

(c) Inserting accompaniments to each new piano roll.

(d) Generating creative and original musical output.

2.2 Representation

The representation of the input data for training and testing purposes is established by determining the nature and format of the musical samples. For example, researchers have used musical waveforms with CNNs to generate music with several musical instrument digital interface (MIDI) channels, and employed Transformers with learned adversarial losses to minimise the difference between real and synthetic samples [8, 9].

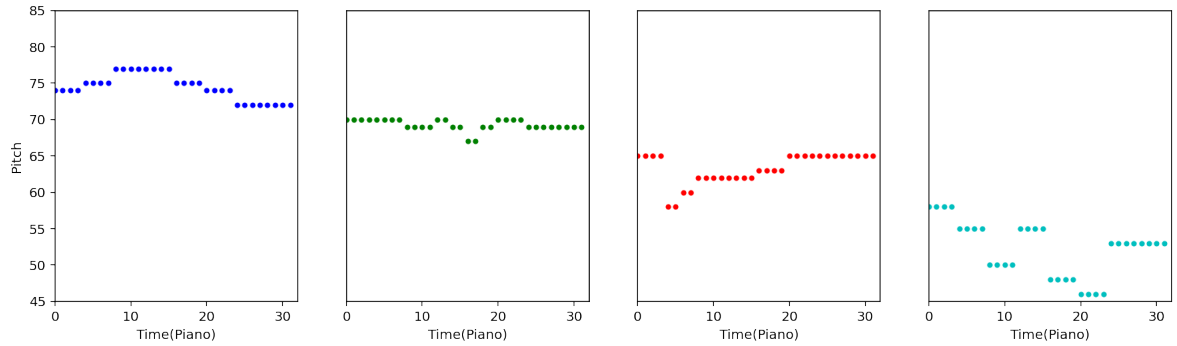


Figure 1: Piano roll representation of the input samples

Fig. 1 shows the piano roll representation of the research dataset (for more information on the specific dataset used, see Section 2.5.1), where the different dotted lines represent various tones and pitches with respect to time, and the length of each line is emblematic of the duration of each note [3].

2.3 Architecture

2.3.1 Generative Adversarial Networks (GANs)[1]

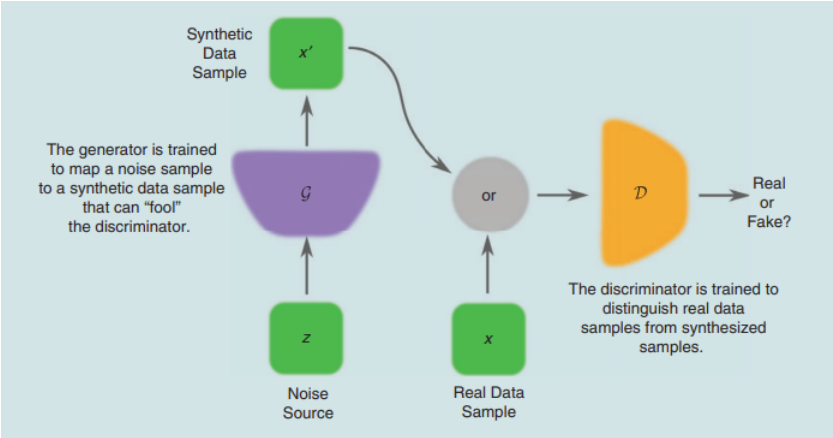


Figure 2: Generic GAN architecture

GANs are unsupervised neural network models that use two competing supervised sub-models to generate deep representations of the input data without the explicit need for

[1] All the images in this section have been sourced from [10], unless otherwise stated.

annotations [10]. The two sub-models work in concert to develop high-dimensional distributions of the input data that generalises well on unseen samples. Fig. 2 shows the adversarial architecture behind a GAN implementation, where the two main components are the generator and the discriminator:

- The generator $\mathcal{G}$ "...transforms a random noise vector into a synthetic sample, which resembles real samples drawn from a distribution of the real content" [4].
- The discriminator $\mathcal{D}$ "...estimates the probability that a sample came from the real data rather than from the generator $\mathcal{G}$" [4].

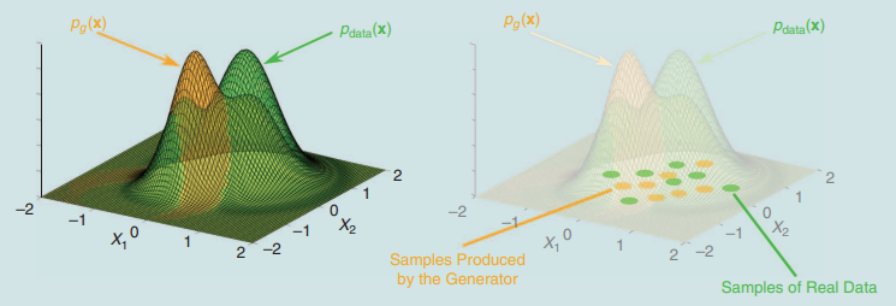


Figure 3: Probabilistic distribution for $\mathcal{G}$ and $\mathcal{D}$

The models are implemented as multi-layer convolutional neural networks where the input is represented as a scalar matrix with an associating filter/kernel to develop a feature map. The generator $\mathcal{G}$ is a differentiable function represented as a multi-layer perceptron $\mathcal{G}(z;\theta_g)$ that develops a distribution p_g over data x by defining a prior on input variables $p_z(z)$ [10]. Conversely, the discriminator $\mathcal{D}$ is also represented as a multi-layer perceptron $\mathcal{D}(z;\theta_d)$ that outputs a single scalar value denoting the probability of assigning the correct labels to the training samples and the sample from $\mathcal{G}$ (Fig. 3) [10].

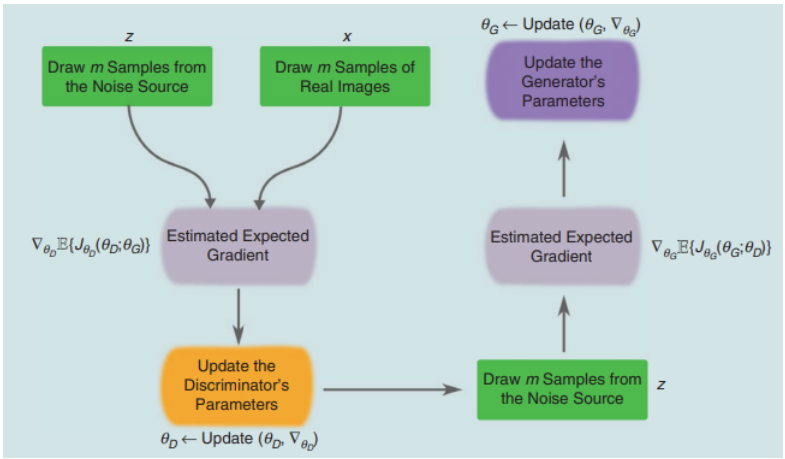


Figure 4: GAN iterative learning process

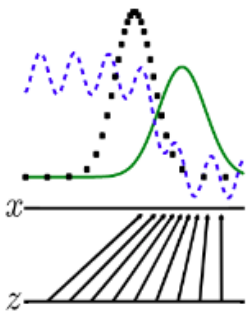
Fig. 4 displays the iterative process of developing a trained model via the minimax process where a unique solution always exists. The relationship is defined as:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G},\mathcal{D}) = \mathbb{E}_{x\backsim p_x(x)}[\log \mathcal{D}(x)] + \mathbb{E}_{z\backsim p_z(z)}[1 - \log \mathcal{D}(\mathcal{G}(z))] \tag{1}$$

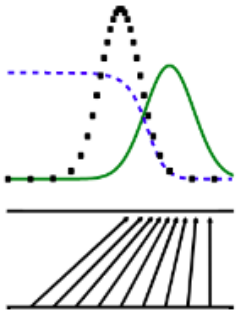
where:

$\mathcal{D}(x)$	= represents the probability that x is from the data and not p_g
$\mathbb{E}_{x \backsim p_x(x)}[\log \mathcal{D}(x)]$	= denotes $\mathcal{D}$'s objective to correctly estimate the real data
$\mathcal{D}(\mathcal{G}(z))$	= signifies the probability that a sample is from x
$1 - \log \mathcal{D}(\mathcal{G}(z))$	= represents the syntheticism of $\mathcal{D}(\mathcal{G}(z))$
$\mathbb{E}_{z \backsim p_z(z)}[1 - \log \mathcal{D}(\mathcal{G}(z))]$	= denotes $\mathcal{D}$'s objective to correctly estimate the synthetic data

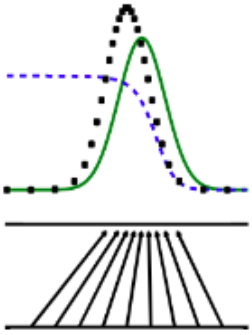
Therefore, it is $\mathcal{D}$'s objective to estimate the authenticity of *both* the real and synthetic samples, whilst $\mathcal{G}$ minimises $V(\mathcal{G}, \mathcal{D})$ [3].



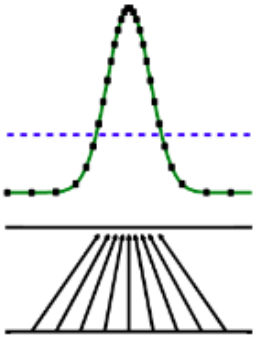
(a) $\mathcal{G}$ & $\mathcal{D}$ near convergence



(b) Update of the $\mathcal{D}$



(c) Movement of $\mathcal{G}(z)$ towards the x points



(d) $\mathcal{G}$ & $\mathcal{D}$ converge

Figure 5: Pictorial representation of the GAN learning process [11]
key:
$\mathcal{D}$, dashed line;
$\mathcal{G}$, solid line;
x-generated distribution, **dotted points**;
lower horizontal line is the domain of z;
higher horizontal line is the domain of x

The iterative process starts with drawing m samples from the real and noise sources. Fig. 5a shows $\mathcal{G}$ and $\mathcal{D}$ near convergence, where $p_z \approx p_x$. Fig. 5b represents the update of the $\mathcal{D}$ to discriminate samples from x, and Fig. 5c highlights the movement of $\mathcal{G}(z)$ towards the concentrated regions inhabited by the x points. Lastly, Fig. 5d shows the training step where the models can not improve further as $p_z = p_x$, i.e., the discriminator can not tell the difference between the real and synthetic samples.

2.3.2 Wasserstein Loss with Gradient Penalty

GANs represent a novel innovation in generative modelling with the advantage of wide generalisation capabilities. However, vanilla GANs with minimax loss as described in the original paper in [11], are hard to train due to factors including oscillating loss where no correlation between the loss of $\mathcal{G}$ and $\mathcal{D}$ can be established resulting in learning instability [5]. Therefore, Arjovsky et al. proposed the usage of a Wasserstein loss function with gradient penalty to improve the interpretability of the loss function and produce samples of higher quality [5]. The Wasserstein metric $\mathbb{W}$ is a measure of the distance between two probability distributions and defined as:

$$\mathbb{W}(p_z, p_x) = \max_{\|f\| \leq 1} \mathbb{E}_{x \backsim p_z}[f(x)] - \mathbb{E}_{x \backsim p_x}[f(x)] \tag{2}$$

The gradient of the discriminator is penalised to ensure optimal differentiation between the real and synthetic samples [5]. This is achieved by implicitly defining $p_{\hat{x}}$ which represents uniform sampling along straight lines between a pair of points drawn from p_z and p_x [5]. The resulting loss function $\mathbb{L}$ can be parametrised to allow neural network training with two learnable functions for the discriminator $\mathbb{D}_w$ and generator $\mathbb{G}_\theta$:

$$\mathbb{L}(p_x, p_\theta, p_{\hat{x}}) = \max_{w \in \mathbb{W}} \mathbb{E}_{x \sim |_x}(\mathbb{D}_w(x)) - \mathbb{E}_{z \sim p(z)}(\mathbb{D}_w(\mathbb{G}_\theta(z)) + \lambda \mathbb{E}_{\hat{x} \sim |_{\hat{x}[(\|\nabla_{\hat{x}} \mathbb{D}_w(\hat{x})\|_2 - 1)^2]}} \tag{3}$$

where:

λ = penalty coefficient

Eq. 3 highlights the basic training procedure where a sample from x and $\mathbb{G}_\theta$ is chosen and the Wasserstein distance $\mathbb{W}$ is minimised between the two points [5].

2.4 Challenge

"A challenge is one of the qualities (requirements) that may be desired for music generation" [3]. This research is focused on developing creative and original music samples by using accompaniments that have been learned from the original dataset. Indeed, quantitative analysis of the musical output is hard to define and out of the scope of this research, but qualitative analysis which although is subjective, does allow for perceptual evaluation of the musical tones and pitches.

2.5 Strategy

2.5.1 Dataset

The `JSB-Chorales` dataset[2] in piano roll format was used to train the GAN system. The piano roll format is a digitalised discrete version of music than be represented as a two-dimensional grid with pitch and time as the axis members [3]. The research dataset consists of 229 chorale piano-only snippets, where each chorale is comprised of one melody and three undertones of harmony. Each sample is a 32 timestep snippet with 128 different pitches, as shown in Fig. 1. The dataset was converted into a `NumPy` array to correctly parse each sample into its corresponding piano roll format which was then used for training the GAN system.

[2] Available to download from here.

2.5.2 Implementation[3]

Algorithm 1 GAN with Wasserstein Loss and Gradient Penalty [5]

Require: α (learning rate), c (clipping parameter), m (size of each batch), n_c (the number of iterations for the discriminator per generator iteration), w_0 (initial discriminator parameters), θ_0 (initial generator parameters).

while $\theta \not\to 0$ **do**
  for $t = 0, \ldots, n_c$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim p_x$ a batch of real data
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of priors
    $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
    $w \leftarrow w = \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
  $\theta \leftarrow \theta - \alpha \cdot \text{RRMSProp}(\theta, g_\theta)$
end while

Algorithm 1 highlights the training steps taken by the GAN system to produce new musical samples. The dataset is loaded as a `TensorFlow` object and the two competing models, $\mathcal{G}$ and $\mathcal{D}$, run in a conditional loop which does not terminate until convergence i.e., $\theta \not\to 0$. The $\mathcal{G}$ (based on the popular U-net architecture [12]) takes a sample from the real data and synthesizes several new multitrack piano roll samples with added accompaniments. The $\mathcal{D}$ model takes the new samples and predicts how much tonal deviation exists between the real and synthetic samples. This information is fed back to the $\mathcal{G}$, which updates the weights (using the Wasserstein Loss function with Gradient penalty explained in Section 2.3.2) to bring the next synthetic samples closer to the real samples. This process continues until convergence at which point $p_x = p_z$, and the model cannot differentiate between the real and synthetic samples.

Table 1: GAN parameters

Parameter	**Value**
Batch size	64
Shuffle buffer size	$1e^3$
Input size	(32, 128, 1)
Latent noise vector size	(2, 8, 512)

Table 1 displays the parameter values for the system. The data is fed into the system in batches to ensure learning stability and the shuffle buffer size represents the variability in the input sample to negate possible overfitting. The input size highlights the time step, number of pitches, and number of tracks respectively, while the latent noise vector is emblematic of the deviation of the synthetic sample from the original one to ensure originality of the generated content.

[3]The complete system has been made **open-source** and is available for perusal here.

3 Analysis

3.1 Results

(a) 1000 iterations; 1 update to $\mathcal{D}$ per $\mathcal{G}$ output

(b) 2000 iterations; 2 updates to $\mathcal{D}$ per $\mathcal{G}$ output

(c) 5000 iterations; 5 updates to $\mathcal{D}$ per $\mathcal{G}$ output

Figure 6: Total $\mathcal{G}$ & $\mathcal{D}$ loss with various configurations of the training parameters

Fig. 6 displays the total iterative loss for the $\mathcal{G}$ and $\mathcal{D}$. Fig. 6a displays the loss for a 1000 iterations with 1 update where a mild oscillating behaviour is observed as the models *almost* converge. The adversarial nature of the models is evident as the $\mathcal{D}$ loss remains stable around the convergence limit whilst the $\mathcal{G}$ loss displays oscillation, albeit not severely. Fig. 6b highlights the adversarial nature of the models with 2000 iterations and 2 updates for $\mathcal{D}$ per $\mathcal{G}$ output. The oscillation in the $\mathcal{G}$ loss is highly observable until $\approx$ 1000 iterations at which point the models converge. At $\approx$ 1250 iterations, the models diverge slightly and the trend continues till the iterative end. The oscillation for the $\mathcal{G}$ loss is attributable to the number of updates, as the $\mathcal{D}$ consistently gives feedback on the generated samples, whereby the $\mathcal{G}$ model modulates the weights of the network according to the incoming feedback. Therefore, severe oscillation is observed until the $\approx$ 1250 iteration mark from whereon the models converge and diverge slightly, as previously mentioned. Fig. 6c shows the loss for 5000 iterations with 5 updates for $\mathcal{D}$ per $\mathcal{G}$ output. The result is similar to the previous test where severe oscillation is observed before the $\approx$ 1000 iteration mark. Convergence and divergence of the two models is yet again observed at $\approx$ 1250 iterations, but the models do show promising convergence at again at around the $\approx$ 5000 iteration mark. Two samples[1] were generated from the final test of the 5000 iterations and, as mentioned in Section 2.4, the quantitative analysis of the samples is beyond the scope of this research, yet if the reader were to listen to the music snippets, noticeable variations in the tones and pitches can be observed thereby validating the experimental methodology. Moreover, the complete tonal metrics of the generated musical samples can be investigated in Appendix A.

3.2 Further work

The original motivation for the project was derived from Gino et al.'s research, "Symbolic Music Genre Transfer with CycleGAN" that implies the usage of two $\mathcal{G}$'s with a Cycle Consistency Loss function to improve the fidelity of the output [13]. In lieu of this information, the first suggested improvement is centred on using different loss functions that can minimise the divergence of the $\mathcal{G}$ and $\mathcal{D}$ models. Secondly, due to time and computational limitations (generating music requires extra memory allocation), the author could not run the experimentation for $>$ 5000 iterations as the GPU provided by `Google Colab` kept timing out. Future research should consider investing in a private GPU or upgrading the `Google Colab` account to increase the availability of the computational resources. Lastly, different datasets and genres should also be considered for this research to enable multi-domain genre transfer that exudes originality and creativity.

3.3 Audio deepfakes

A pertinent high-level issue that arises from this research is the generation of audio deepfakes that are increasingly becoming a mainstream issue due to recent breakthroughs in speech synthesis and voice conversion techniques [14]. In a world where one does not need to travel much to face hostility, and where cyberwarfare is not only obfuscating the truth, but also enabling pernicious manipulation of personal biometrics, it would be prudent to carry out research in the domain of audio generation as a viable deterrent.

[1] Available as audio files here (Section 6.2).

4 Conclusion

The aim of the research was to develop a GAN system that could parse given musical samples in the piano roll format and generate original musical snippets by mixing learned tones and pitches. The GAN was integrated with a Wasserstein Loss with Gradient Penalty function to offset the deviation of the generated samples from the original samples. The results display a promising trend where the generated samples have been analysed qualitatively ensuring the validity of the proposed methodology. Additionally, various improvements and high-level issues have also been suggested.

References

[1] Sanidhya Mangal, Rahul Modak, and Poorva Joshi. "LSTM-based music generation system". In: *arXiv preprint arXiv:1908.01080* (2019) (page 3).

[2] Rishi Madhok, Shivali Goel, and Shweta Garg. "SentiMozart: Music Generation based on Emotions." In: *ICAART (2)*. 2018, pp. 501–506 (page 3).

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. "Deep learning techniques for music generation–a survey". In: *arXiv preprint arXiv:1709.01620* (2017) (pages 3, 4, 6, 7).

[4] Jean-Pierre Briot. "From artificial neural networks to deep learning for music generation: history, concepts and trends". In: *Neural Computing and Applications* 33.1 (2021), pp. 39–65 (pages 3, 5).

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html (pages 3, 7, 8).

[6] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. "MidiNet: A Convolutional Generative Adversarial Network for symbolic-domain music generation". In: *arXiv preprint arXiv:1703.10847* (2017) (page 3).

[7] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (page 3).

[8] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016) (page 4).

[9] Aashiq Muhamed, Liang Li, Xingjian Shi, Suri Yaddanapudi, Wayne Chi, Dylan Jackson, Rahul Suresh, Zachary Lipton, and Alexander J Smola. "Transformer-GAN: Symbolic music generation using a learned loss". In: *4th Workshop on Machine Learning for Creativity and Design at NeurIPS*. 2020 (page 4).

[10] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. "Generative Adversarial Networks: An Overview". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65 (pages 4, 5).

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (pages 6, 7).

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241 (page 8).

[13] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao. "Symbolic Music Genre Transfer with CycleGAN". In: *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE. 2018, pp. 786–793 (page 10).

[14] Tianxiang Chen, Avrosh Kumar, Parav Nagarsheth, Ganesh Sivaraman, and Elie Khoury. "Generalization of audio deepfake detection". In: *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*. 2020, pp. 132–137 (page 10).

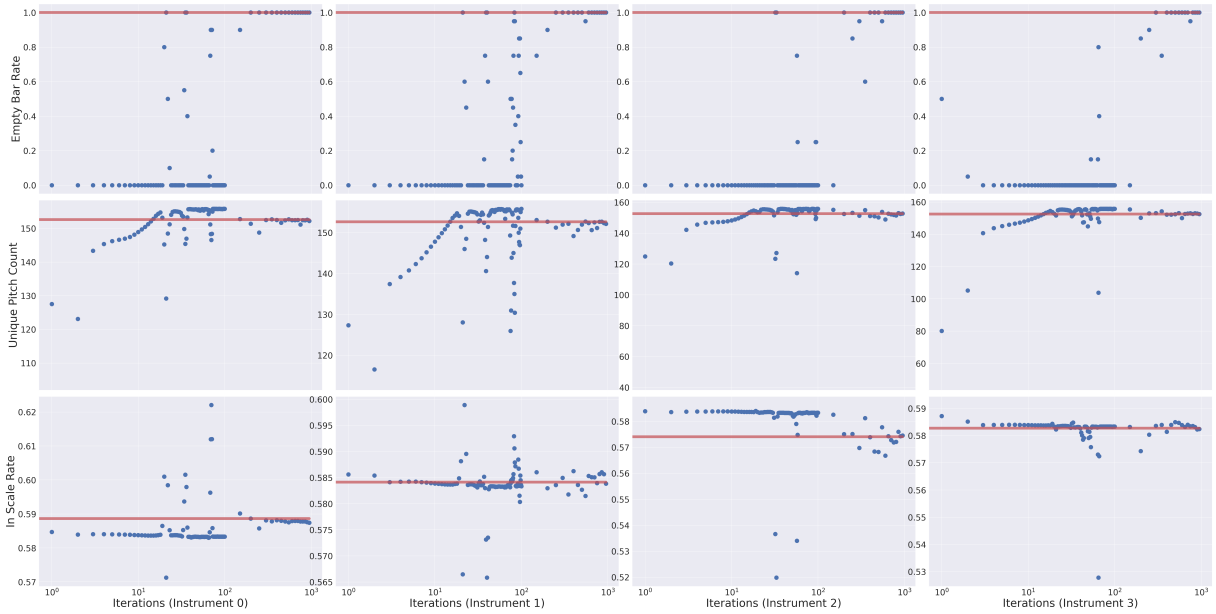
Appendix

A Output metrics

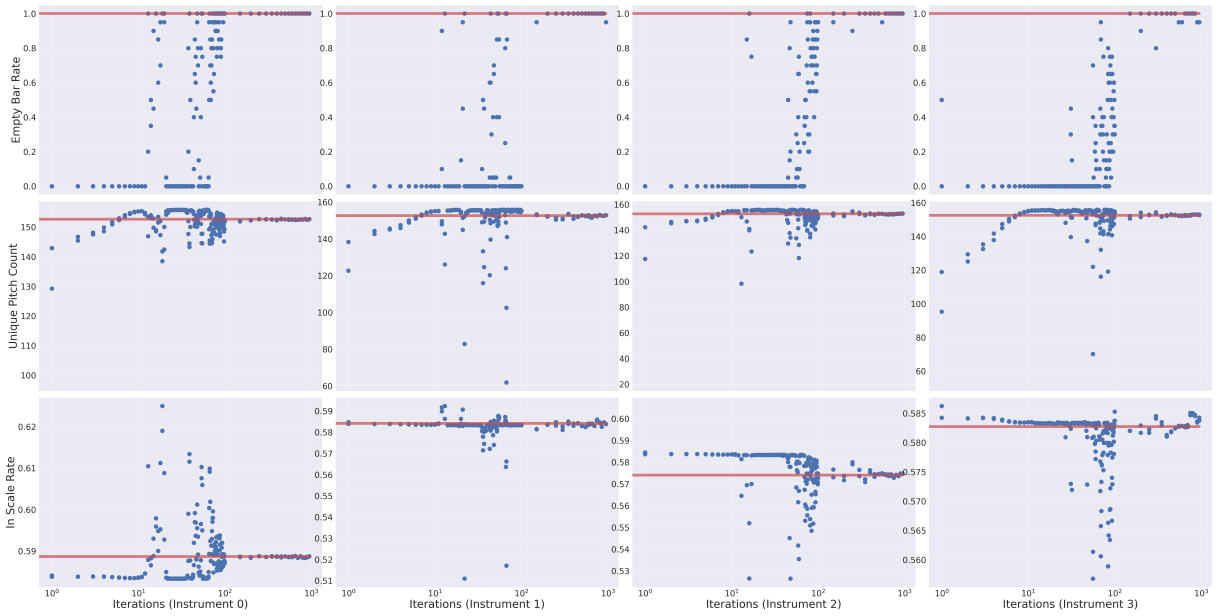
The 3 quantitative measurements that have been reported include:

1. **Empty bar rate**: Ratio between total number of bars and empty bars.
2. **Unique pitch count**: Represents the number of unique pitches that have been produced.
3. **In-scale ratio**: Ratio between total number of notes and C-major key notes.

A.1 1000 iterations; no update to $\mathcal{D}$ per $\mathcal{G}$ output



A.2 2000 iterations; 2 updates to $\mathcal{D}$ per $\mathcal{G}$ output



A.3 5000 iterations; 5 updates to $\mathcal{D}$ per $\mathcal{G}$ output