PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EXPLORING SYMBOLIC MUSIC GENERATION TECHNIQUES USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

MANUEL CARTAGENA HERRERA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
DENIS PARRA
RODRIGO CÁDIZ

Santiago de Chile, July 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EXPLORING SYMBOLIC MUSIC GENERATION TECHNIQUES USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

MANUEL CARTAGENA HERRERA

Members of the Committee:

DENIS PARRA

RODRIGO CÁDIZ

HANS LÖBEL

FELIPE BRAVO

RICARDO GIESEN

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, July 2021

Gratefully to my parents, siblings,
girlfriend and pets.

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisors Denis Parra of the Department of Computer Science at Pontificia Universidad Católica de Chile and Rodrigo Cádiz of the Department of Music at Pontificia Universidad Católica de Chile. They were always available to answer my questions or to guide me whenever I had a problem. They consistently steered me in the right the direction whenever they thought I needed it, even when the experiments and results I got weren't satisfactory and it frustrated me, they made me look at the glass half full and encouraged me to continue.

I would like to deeply thank my girlfriend for all the times she helped me push through this, for all her support and kind words of encouragement, especially during this hard times in confinement.

I would like to thank my family, specially my parents, that even though we were mostly apart in this difficult times, they supported me in every way possible and encouraged me to go on.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

LIST OF FIGURES viii

LIST OF TABLES x

ABSTRACT xi

RESUMEN xii

1. INTRODUCTION 1
   1.1. Motivation 1

2. BACKGROUND THEORY 3
   2.1. Generative Adversarial Network (GAN) 3
   2.2. Symbolic Music Representations 4
      2.2.1. MIDI 4
      2.2.2. Pianoroll 5

3. RELATED WORK 7
   3.1. Recurrent Neural Network (RNN) methods 7
      3.1.1. DeepBach 7
      3.1.2. Music Transformer 8
   3.2. Variational Autoencoder (VAE) methods 8
      3.2.1. MusicVAE 9
   3.3. Generative Adversarial Network (GAN) methods 9
      3.3.1. MidiNet 9
      3.3.2. MuseGAN 9

3.4. Differences to Previous Research 10

4. OBJECTIVES 12
4.1. Contribution 12
4.2. Research Questions 13
4.3. Hypothesis 13

5. METHODOLOGY 14
5.1. Datasets 14
5.1.1. MAESTRO 14
5.1.2. LAKH 18
5.2. Models 19
5.2.1. Deep Convolutional Generative Adversarial Network (DCGAN) . . . 20
5.2.2. StyleGAN 2 21
5.2.3. Evaluation 24
5.2.4. Training 26
5.2.5. Class conditions 26
5.2.6. Generation 27

6. RESULTS 31
6.1. DCGAN 31
6.2. StyleGAN 2 ADA 31
6.2.1. Use Cases 33
6.2.2. Exploring the latent space 36
6.2.3. Projecting samples 37
6.2.4. Interpolating samples 39
6.3. Comparison 40
6.4. Models availability 41

6.5. Audio examples 42

7. LIMITATIONS 43

7.1. Hardware 43

7.2. Image Size 43

7.3. Pianoroll representation 44

8. FUTURE WORK 45

9. CONCLUSIONS 47

REFERENCES 49

APPENDIX 54

A. DCGAN Architecture 55

A.1. Generator 55

A.2. Discriminator 56

LIST OF FIGURES

2.1 Basic representation of a GAN architecture. 4

2.2 Pianoroll example, the Y-axis represents note pitches and the X-axis represents the timing for the musical events. The information stored on the matrix represents velocity, which is a value from 0 to 127, but in this example the values are normalized. 6

5.1 Number of compositions from each composer available on the MAESTRO dataset. 15

5.2 Single-track pianoroll of a performance from the MAESTRO dataset. 16

5.3 Nº of perfomances distribution after processing the Maestro dataset. 17

5.4 Number of songs from each genre after preprocessing. 19

5.5 DCGAN generator architecture. 21

5.6 DCGAN discriminator architecture. 22

5.7 StyleGAN 2 mapping network. FC stands for fully connected layer. 23

5.8 Google Colab interface showing how to select a class condition for generation, particularly for composers of the MAESTRO dataset. 27

5.9 Output image from StyleGAN 2 model trained on the Maestro dataset, with input condition to be a piece from Johann Sebastian Bach. Analysing the pitches used, which are (D, E, F, G, G#, B), correspond to a G diminished wholetone scale. Also, the first group of 3 notes is a G major chord. 29

5.10 The StyleGAN 2 model is able to generate a variety of musical ideas (A to F). The latent space can also be interpolated between 2 outputs to generate a musically-meaningful sequence. In this case, the generated sequence exhibits how the network morphs from sample A to sample F in 4 steps, visually divided by a red line for easier differentiation. 30

6.1 DCGAN image generated using the Maestro dataset. 32

6.2 FID metric during StyleGAN 2 training with the Maestro dataset. 33

6.3 FID metric during StyleGAN 2 training with the Lakh dataset of piano. . . . 34

6.4 FID metric during StyleGAN 2 training with the Lakh dataset of guitar. . . . 34

6.5 FID metric during StyleGAN 2 training with the Lakh dataset of bass. 35

6.6 FID metric during StyleGAN 2 training with the Lakh dataset of strings. . . . 35

6.7 FID metric during StyleGAN 2 training with the Lakh dataset of drums. . . . 36

6.8 Polyphony metric for the Maestro Dataset 37

6.9 Empty timesteps metric for the Maestro Dataset 38

6.10 Metrics for the Maestro Dataset using unconditional models. 38

6.11 Google Colab interface for exploring the latent space of one of the StyleGAN 2 models trained on the Lakh dataset. 39

6.12 Target image to project on the latent space on the left, and resulting projection on the right. 40

LIST OF TABLES

6.1 Comparison table of the best FID results for each model on different datasets. 50000 images were sampled for calculating this metric. 41

ABSTRACT

Generative models have become an area of utmost importance in recent times, due to their ability to learn a probabilistic data distribution from an input data set. Currently these models have been explored mainly in the generation of images, but not so much in the musical field, where the use of these models makes sense since music is rich in structured information which can be learned by these models. In this paper we present the analysis of two case studies of generative models based on deep convolutional networks. We study their ability to generate symbolic music for one or more instruments in the pianoroll format, and whether it is possible to condition the output to display characteristics of different composers or genres. Also we study how controllable are the results generated. We evaluate both models using the Fréchet Inception Distance (FID), a metric for generative image models, in addition to musical metrics defined by us. One of these cases is the use of the recently developed StyleGAN 2 model. Using this type of architecture in a non-visual domain is novel and we present interesting results in terms of FID and in qualitative musical terms. Despite this model was designed for a visual domain to generate high quality images, it can be adapted to a totally different context. In addition, it has properties that are of interest to the area of musical composition, such as having a disentangled latent space, where it is easy to explore different musical ideas, and conditional input to further control the output of the model. We believe that the results we show in this work are a step forward in understanding how to create better generative models in the symbolic music domain, taking into account the concepts of conditionality and controllability to develop better tools for the end users.

Keywords: generative models, symbolic music generation, machine learning.

RESUMEN

Los modelos generativos se han convertido en un área de gran importancia en los últimos tiempos, debido a su capacidad para aprender una distribución probabilística de los datos de entrada. Actualmente estos modelos han sido explorados para la generación de imágenes, pero no tanto en el ámbito musical, donde la música es rica en información estructurada que puede ser aprendida por estos modelos. En este trabajo presentamos el análisis de dos casos de estudio de modelos generativos basados en redes convolucionales profundas. Estudiamos su capacidad para generar música simbólica para uno o más instrumentos en el formato pianoroll, y si es posible condicionar la salida para mostrar características de diferentes compositores o géneros. También estudiamos hasta qué punto son controlables los resultados generados. Evaluamos ambos modelos utilizando Fréchet Inception Distance (FID), una métrica para modelos generativos de imágenes, además de métricas musicales definidas por nosotros. Uno de estos casos es el uso de StyleGAN2, donde por primera vez se utiliza este tipo de arquitectura en un dominio no visual, adaptándolo a un contexto distinto con resultados interesantes tanto en FID como en términos musicales cualitativos. Además, tiene propiedades que son de interés para el área de la composición musical, como tener un espacio latente desenredado, donde es fácil explorar diferentes ideas musicales, y la entrada condicional para controlar aún más la salida del modelo. Creemos que los resultados que mostramos en este trabajo son un paso adelante en la comprensión de cómo crear mejores modelos generativos en el dominio de la música simbólica, teniendo en cuenta los conceptos de condicionalidad y controlabilidad para desarrollar mejores herramientas para los usuarios finales.

Palabras Claves: modelos generativos, generación de música simbólica, aprendizaje de máquina.

1. INTRODUCTION

1.1. Motivation

Today, the area of artificial intelligence is one of the fastest growing areas of study, which has permeated practically in every other area in one way or another. Specifically, the deep learning branch has long made it possible to obtain excellent results in many tasks such as image classification, anomaly detection, language processing, among others (Sengupta et al., 2020). The advancement of deep learning has influenced the arts, either in the way certain tasks are performed, or the tools available to create new pieces of art. The availability of new and better models in the music domain have made it possible to improve tasks such as genre classification of music, instrument separation from audio files, text-to-speech, among others.

Although machine learning started to be used long ago in music mainly for information retrieval tasks rather than for the generation of new content, probabilistic approaches existed. These early approaches were mainly handcrafted algorithmic tools for composing new pieces.

Due to the appearance of large volumes of data, increasingly deeper models have been developed, with millions of trainable parameters that can solve the most complex tasks. These models can also learn a data distribution instead of solving a task, being able to have a representation of these immense volumes of data in a neural network. This learned distributions can be used to generate new examples that resemble our input data, allowing this type of models to contribute to the creation of artistic content,

and also to nurture the creativity of its users to compose music or find new sounds through the use of these new tools.

2. BACKGROUND THEORY

In this chapter, we introduce the music and machine learning concepts necessary to understand this work. We explain how generative adversarial networks work, which are the main architecture used throughout this work. We also explain the MIDI representation and how it translates to the pianoroll representation used as input for the models described in section 5.

2.1. Generative Adversarial Network (GAN)

Generative adversarial networks (Goodfellow et al., 2014) were introduced in 2014 as a new framework for generating new synthetic instances of data using two neural networks: the generator and the discriminator. The generator consists of a network that produces new samples that tries to resemble the training data, while the discriminator is in charge of classifying this samples as real samples from the training data or fake samples that do not belong to the distribution of the input data. In Figure 2.1, there is a general representation of the architecture used to train a GAN. The generator is fed with some input noise, usually sampled from a normal distribution, and from this input transforms it to generate an example that the discriminator classifies.

This new approach of training both networks simultaneously, where one of them tries to "fool" the other corresponds to a minimax game that can reach a global optimum when the discriminator can not tell whether a sample from the generator is real or fake, implying that the generator has

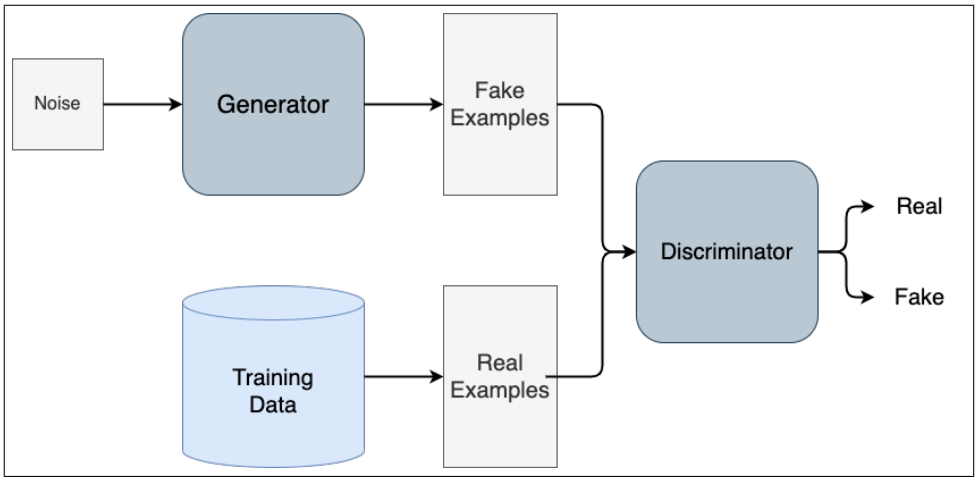

Figure 2.1. Basic representation of a GAN architecture.

learned the training data distribution and that is capable of generating new samples that seems similar to the original data.

2.2. Symbolic Music Representations

2.2.1. MIDI

The MIDI (Music Instrument Digital Interface) Protocol (Association et al., 1983) was introduced in 1983 to synchronize and communicate different electronic music instruments. At its core, MIDI is a standard for communicating musical events, which consist of multiple controls whose values are quantized as 7-bit integers, therefore, allowing to pass 128 different values per message. In 1988 it was defined the MIDI file standard, which to this day is one of the simplest ways to store a musical score that can be played in different instruments that implement this standard. This 8-bit binary file standard stores a score using one or more MIDI streams

with timing information for each event. These streams allow having multiple instrument information for a song. Other information such as tempo, time signature information, and track names, among others, can be stored.

2.2.1.1. Pitch

This term refers to which of the 12 notes of the traditional western tuning an instrument is playing and in which octave. When two or more notes are played simultaneously, it is called polyphony.

2.2.1.2. Velocity

In the MIDI standard, velocity can be interpreted as the force used to play a note, meaning that higher velocity results in a note played harder on a keyboard than a note with lower velocity. This feature was implemented to incorporate dynamics.

2.2.2. Pianoroll

A pianoroll is a symbolic music format, inspired by old self-playing pianos that used rolls of paper, which represents a music piece by a 2-dimensional matrix. The vertical axis represents note pitch information and the horizontal axis represents the timing of when these notes are played. The values stored on this matrix represent the velocities of the notes. In Figure 2.2, there is an example of a single-track pianoroll, showing a piano performance from the Maestro dataset (Hawthorne et al., 2019), which only has information from a single instrument.

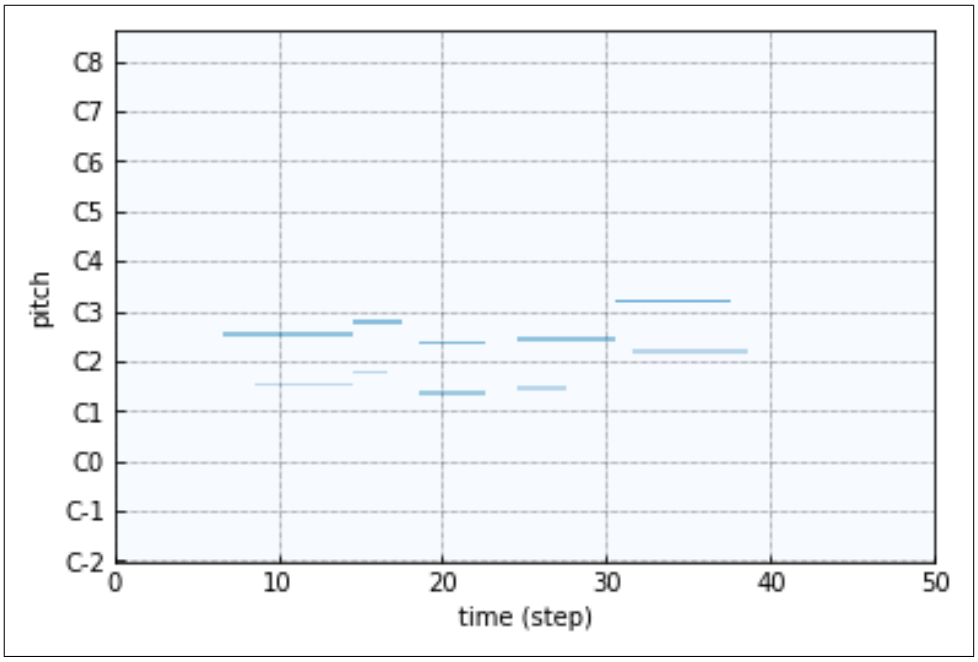

Figure 2.2. Pianoroll example, the Y-axis represents note pitches and the X-axis represents the timing for the musical events. The information stored on the matrix represents velocity, which is a value from 0 to 127, but in this example the values are normalized.

3. RELATED WORK

In this section we provide an overview of relevant related work. The section is split into three parts: Recurrent Neural Network (3.1), Variational Autoencoder (3.2) and Generative Adversarial Network (3.3). All the sub-sections are important to better understand the different approaches to the problem we are targeting on this work, and their main differences are the type of model architecture. A final section Differences to Previous Research (3.4) highlights what we add with our work to the already existing literature in the area.

3.1. Recurrent Neural Network (RNN) methods

Recurrent neural networks area a type of neural network for handling sequential data and can have an internal state to process and "remember" dynamic behaviour of the input data, which is a feature desired for modelling structured temporal data like music.

3.1.1. DeepBach

DeepBach (Hadjeres, Pachet, & Nielsen, 2017), is a neural network that models 4-voice polyphonic music and specifically hymn-like pieces using a Deep-RNN (Pascanu, Gulcehre, Cho, & Bengio, 2013) architecture to generate Bach-like chorales in a special encoding of the MIDI format proposed in this work for taking advantage of the chorales metadata. The model uses two Deep-RNN to encode past and future information from the chorale, and a third neural network for the information of the current played

notes. These three outputs are merged into a fourth neural network which makes the prediction.

3.1.2. Music Transformer

Classic RNN-type models have trouble learning long sequences because of the vanishing gradient problem. The transformer architecture first introduced by Vaswani et al. (2017) fixed this problem using self-attention, achieving to learn coherent long structures, although this models are mainly used for natural language processing, in music there is a heavy use of repetition in their structures, like in language, which these models are suited for. In the work of Music Transformer (Huang et al., 2018), utilizing a new algorithm to alleviate the complexity of modelling very long sequences for music, the authors managed to generate new music in the MIDI representation with a transformer architecture.

3.2. Variational Autoencoder (VAE) methods

These kinds of models use an autoencoder architecture to learn a reduced embedding space to represent the training data. By replacing this encoded representation space with a prior probability distribution, these models can learn how to represent the input data as a probability distribution.

3.2.1. MusicVAE

Roberts, Engel, Raffel, Hawthorne, and Eck (2018) used a hiearchical RNN style encoder-decoder in order to produce a latent representation of sequential data in the Midi representation. The input was encoded through a bidirectional RNN encoder to obtain a latent code that serves as the conductor, which the decoder transforms into a sequence.

3.3. Generative Adversarial Network (GAN) methods

Goodfellow et al. (2014) introduced this new framework for training a generative network that could represent rich data distributions. The following works use this type of network for generating symbolic music.

3.3.1. MidiNet

MidiNet (Yang, Chou, & Yang, 2017) uses a deep convolutional network GAN to generate melodies as 2D matrices, with additional conditions to the input, like chords or previous melodies that the model should continue, and can be extended to multitrack melody generation. They used a Midi dataset of pop songs with 2-channels one for melody and the other for accompaniment.

3.3.2. MuseGAN

In the work of Dong, Hsiao, Yang, and Yang (2018), the authors approached the problem of music generation for multi-instrumental tracks. Three types of models were developed with different hierarchical types of

generation. The first one uses a common generator and noise input to represent a conductor that coordinates the piece. The second uses 5 different generators and noises to resemble independent "jamming" of the instruments. Finally a hybrid model that have separate noise inputs and a shared one for all the generators. In this work, there is also a RNN component for generating multiple bars with a coherent structure between them.

3.4. Differences to Previous Research

Compared with past research, our work focuses on how to use existing generative methods with the capacity to condition its result, in the domain of symbolic music. Taking into account that methods based on recurrent networks are predominant in this field and widely studied, we wanted to analyze how GANs, which have a completely different approach to generation, can contribute in terms of variability and controllability of the results.

In GANs, MidiNet uses a DCGAN architecture for melody generation, while MuseGAN uses a mixture of RNN and GAN to produce sequences and uses conditionallity only on a track-basis, meaning that they input a piece that the network has to continue. We tried to condition on the type of output generated rather than on a per-example approach. MidiNet has conditioning but they approach a different problem, which is melody and chord progression generation, where they condition based on musical knowledge of chords key and type. This limits the datasets available to use because of the prerequisites to obtain the labels. In the field of VAEs there is a work called MidiMe (Dinculescu, Engel, & Roberts, 2019) that forces conditionality on a previously trained model of a MusicVAE network, constraining

its output to a desired condition like tonality or note density. We wanted to train a model that did not need extra training to allow for this type of conditions, that also enforces to have separate models for each of the conditions.

4. OBJECTIVES

We proposed to implement and study generative models that could generate single or multi-track new pieces of music with and without input conditioning, meaning, that a user could generate a random excerpt of music, or explicitly tell the model to generate music with different characteristics as part of the input.

4.1. Contribution

In this work we present an exploration of deep generative models for symbolic music generation, specifically we analyze different GAN models available for music generation using the pianoroll representation, in particular we studied the capability of conditioning and control of the output of these models.

We trained different models that could generate musical excerpts from different classes and for different instruments using a state-of-the-art model for image generation, finding a new use for type of models in a completely different domain that they were designed to.

To the best of our knowledge, there are no other works to this date that use the StyleGAN 2 (Karras et al., 2019) architecture to generate symbolic music using for input pianorolls as images. We tested this new approach in two well known datasets in this area, with interesting results both musical and in terms of potential to develop new creativity tools for music composers. We made available web based tools to explore the results and

generate new pieces of music, along with the datasets generated to train these models.

4.2. Research Questions

To drive our research, the following questions were defined:

- **RQ1**. Are image-based deep generative models capable of changing domains and producing symbolic music?
- **RQ2**. Are image-based deep generative models trained with symbolic music capable of producing musically similar pieces to training ones?

4.3. Hypothesis

(i) **A Deep Convolutional GAN (DCGAN) could be trained in order to generate symbolic music controllably and conditionally.**

We want to test if a DCGAN architecture (Radford, Metz, & Chintala, 2015), designed for generating images, can be adapted for a different domain and have the same results.

(ii) **The StyleGAN2 architecture will be capable of generating controllable and conditional symbolic music.**

We want to test if this state-of-the-art GAN architecture, mainly designed for generating images of faces, can be adapted for a different domain and have the same results and controllability of the output.

5. METHODOLOGY

In this section we will present the materials used for this work, how we processed them to be used with the methods selected to answer the hypothesis presented.

5.1. Datasets

5.1.1. MAESTRO

We used the MAESTRO V2.0.0 dataset (Hawthorne et al., 2019) which gathers over 200 hours of piano performances of classical pieces from the International Piano e-Competition. This source offers the raw audio data of each performance along with a direct MIDI capture of the recording. There are 1282 performances in version 2.0.0 of the dataset, from 61 different composers, Figure 5.1 shows the distribution of this performances per composer(s).

5.1.1.1. Preprocessing

Each MIDI file was transformed to a pianoroll, which is a 2D representation for music, where one axis represents time and the other represents which notes are played on a given time, the library used to make this transformation was pypianoroll (Dong, Hsiao, & Yang, 2018), the performances were binarized to remove the velocity of the played notes. We decided to binarize the input to facilitate learning in the different models tested in

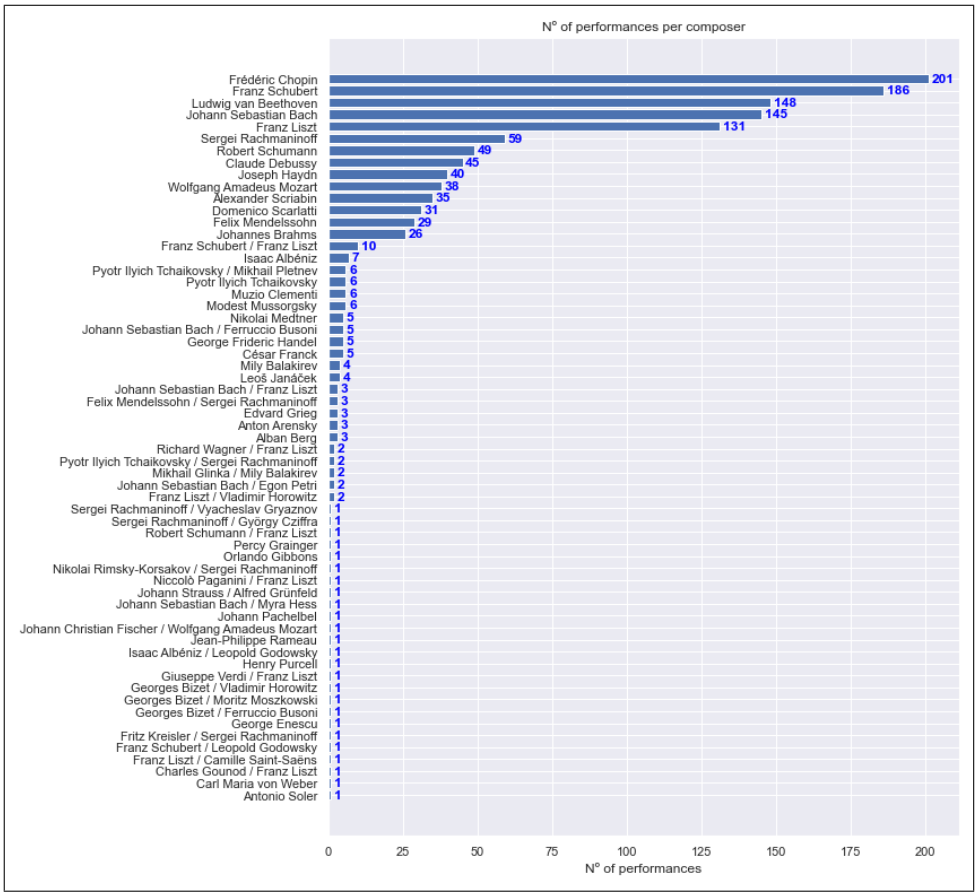

Figure 5.1. Number of compositions from each composer available on the MAESTRO dataset.

the hope of obtaining better results. Because, without dynamics, the information of the notes played is less diffuse. The plan was to incorporate dynamics after obtaining and analyzing these results. In Figure 5.2 there is an example of a single-track pianoroll from the MAESTRO dataset.

Because the majority of composers has less than 10 performances in this dataset, and also there are compositions written by more than one person, these were grouped into a single class keeping the first composer and

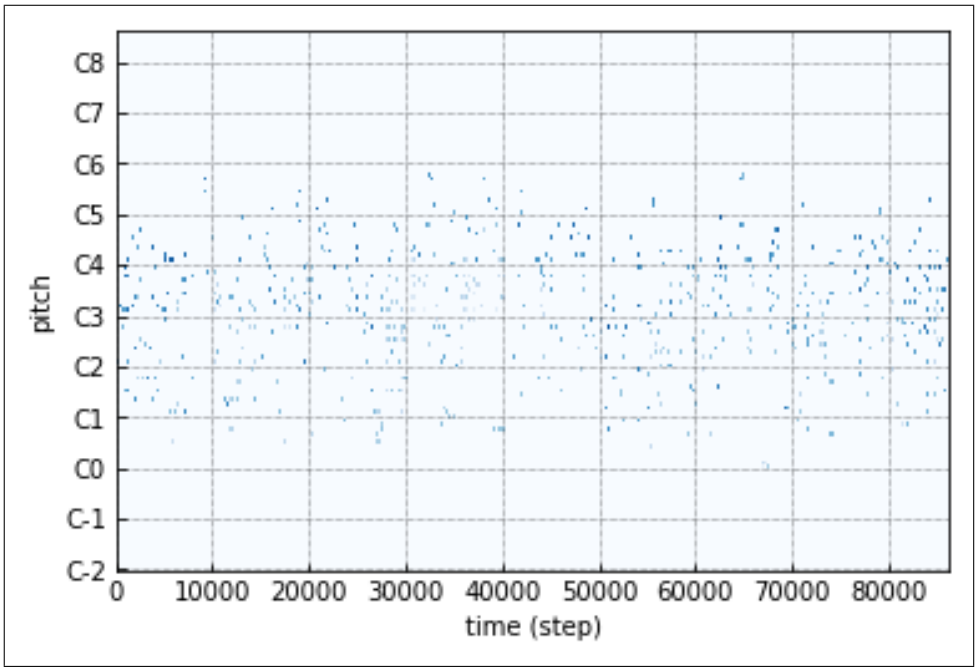

Figure 5.2. Single-track pianoroll of a performance from the MAESTRO dataset.

arbitrarily chose the 6 composers with the most performances after this procedure.

After transforming each MIDI file and sorting them by the reduced composer classes, the performances were quantized to a resolution of 24 time steps per quarter note, this is in order to cover common temporal patterns such as triplets and 32th notes, and split into chunks of 128 time steps, resulting in matrices of size $(128, 128)$, because the MIDI protocol can represent up to 128 notes. In Figure 5.3 there are the resulting performances per composer after processing.

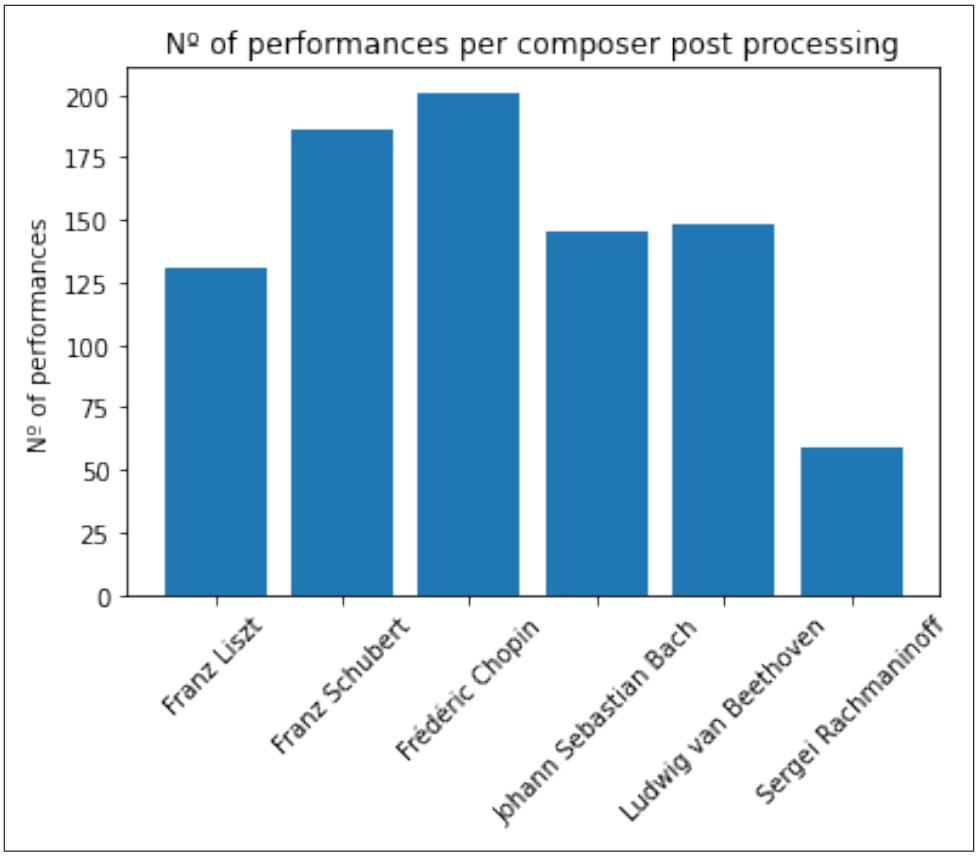

Figure 5.3. Nº of perfomances distribution after processing the Maestro dataset.

5.1.1.2. Availability

The processed dataset previously described and used for the experiments in this work is available at `https://zenodo.org/record/4747698`.

5.1.2. LAKH

We used the LAKH pianoroll dataset (Dong, Hsiao, Yang, & Yang, 2018), which is a derivative of the LAKH MIDI dataset (Raffel, 2016), specifically we used the cleansed LPD-5 variation of the dataset which consists of 5-track pianorolls of the match between the LAKH MIDI dataset and the Million Song dataset (Bertin-Mahieux, Ellis, Whitman, & Lamere, 2011) considering additional rules to ensure consistency between the songs. The variation used has 21425 multitrack songs from all genres derived from the Million Song Dataset, the specific tracks included are piano, guitar, bass, strings and drums.

5.1.2.1. Preprocessing

Each pianoroll file was binarized and separated per instrument in 5 different single track pianorolls, and like with the MAESTRO dataset, each track was quantized to a 24 time step resolution and split into chunks of 128 time steps resulting in matrices of size $(128, 128)$.

The dataset is divided in 13 genres from, the Million Song Dataset Allmusic Top Genre Dataset (Top-MAGD), Figure 5.4 shows the distribution of songs per genre.

5.1.2.2. Availability

The processed dataset previously described and used for the experiments in this work is available at `https://zenodo.org/record/4768780`.

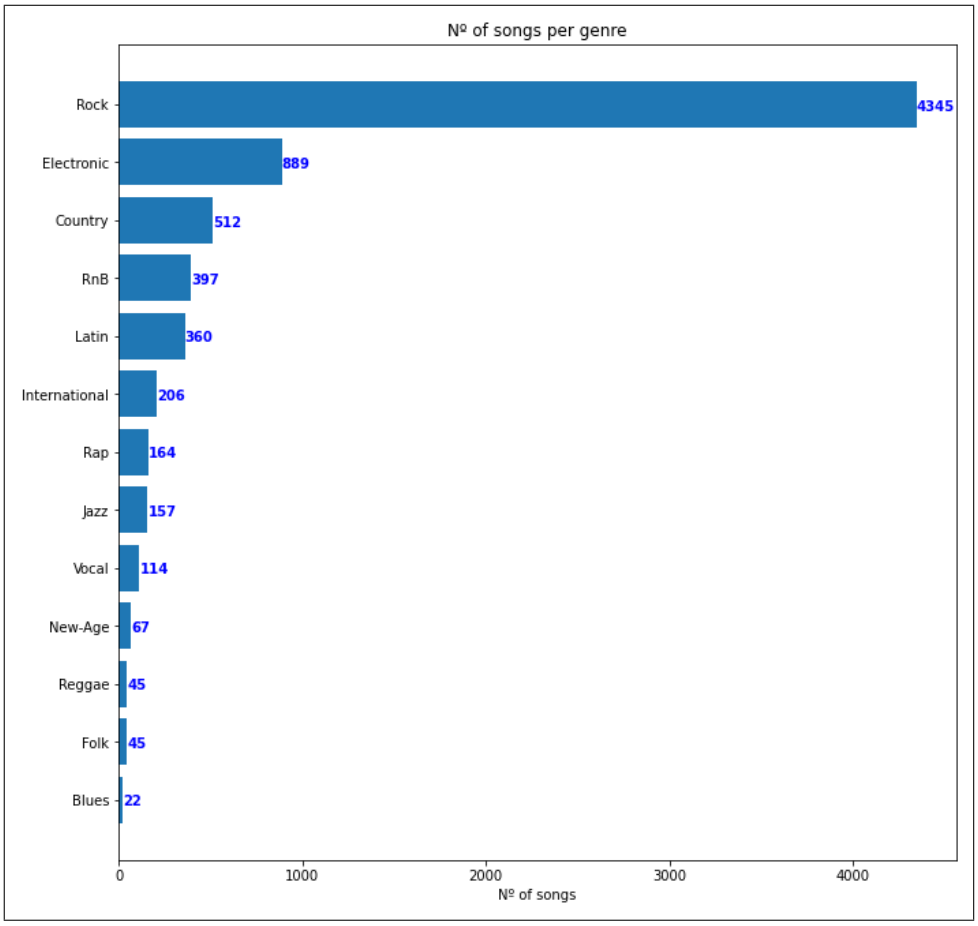

Figure 5.4. Number of songs from each genre after preprocessing.

5.2. Models

We tried different methods to achieve results that would meet the objectives set on the beginning of this work, and that generated the best results according to our subjective judgement.

5.2.1. Deep Convolutional Generative Adversarial Network (DCGAN)

Our first approach was to develop a DCGAN (Radford et al., 2015) model that could generate pianoroll images unconditionally, meaning that the output of the network does not have a predetermined class or feature. The goal of this approach was to have a baseline model to improve adding conditionallity as a next step.

Using MuseGAN (Dong, Hsiao, Yang, & Yang, 2018) as inspiration, we developed both the discriminator and generator so that they could work with 5-channel matrices to generate multi-instrumental music, using the LAKH pianoroll dataset (Dong, Hsiao, Yang, & Yang, 2018). A version with one channel was also developed for the MAESTRO dataset (Hawthorne et al., 2019). One major difference between our model and the MuseGAN one (Dong, Hsiao, Yang, & Yang, 2018), is that they also take into account the temporal aspect of music using a RNN Network to generate longer pieces of music, which we planned to implement once we had a working baseline.

For the addition of conditional information, based on the MidiNet (Yang et al., 2017) architechture, we used a 1-dimension vector as extra input for both the generator and discriminator networks, where information like genre was encoded in a one-hot vector to force the model to generate music according to this extra input.

In Figures 5.5 and 5.6, we have the generator and discriminator architectures respectively, for the generator, we feed the model with a one dimensional noise vector sampled from a normal distribution into 6 transposed

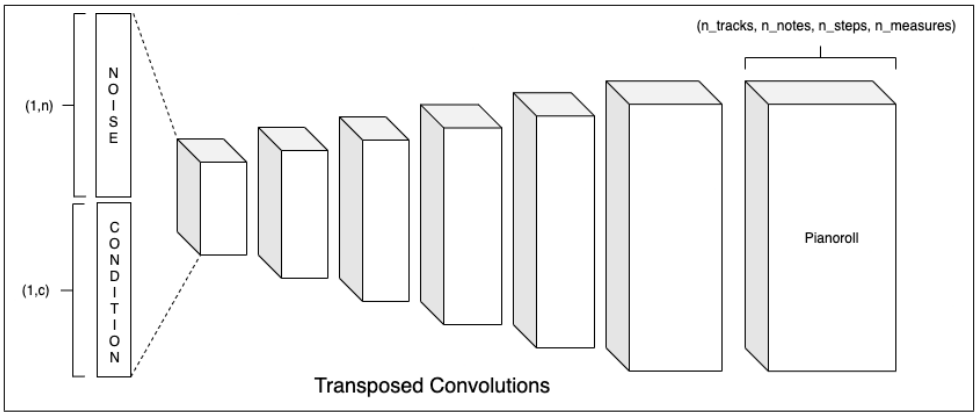

Figure 5.5. DCGAN generator architecture.

convolution layers and finally reshaping the output to the desired shape of (number of tracks, number of notes, number of timesteps per measure, number of measures) which can be then transformed into a MIDI file. For the discriminator, we used a similar layer architecture, but using conventional convolutions, we used 8 layers until a flatten layer follower by a dense layer of 1 neuron to obtain the model decision over the input images. We tested different losses, and ended up using the Wasserstein loss (Arjovsky, Chintala, & Bottou, 2017) which had the least convergence problems and which we could see some learning qualitatively. Further information of the layers and the values used are in Appendix A.

5.2.2. StyleGAN 2

Our second approach was based on the newly developed StyleGAN 2 (Karras et al., 2019), this network achieved state-of-the-art results in image generation, specially for creating new human faces that look highly realistic. Considering the amazing results for generating images, which are a

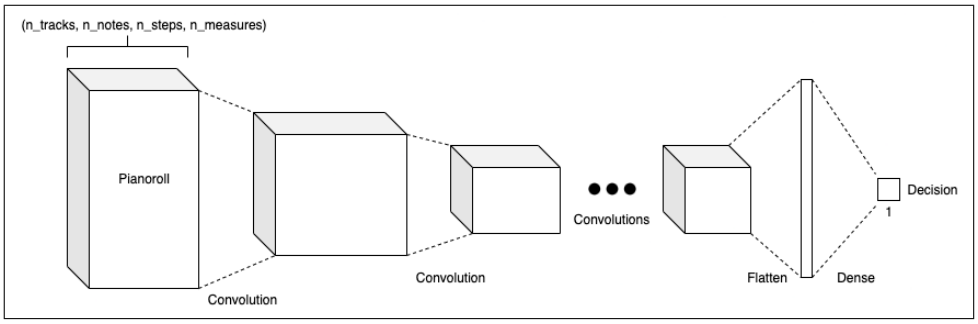

Figure 5.6. DCGAN discriminator architecture.

2D representation of visual information, we experimented to see if this network could generate pianorolls, which are also a 2D representation, but in this case of a musical composition.

Our hypothesis was that even though pianorolls are conceptually very different from the image of a human face, we wanted to study if certain properties captured from visual information can be useful for training a musical generator model from 2D "images" of music.

One of the most interesting aspects of using this network is the amount of control over the latent space. This network does not use the most typical setting of sampling from a normal distribution as the noise input, but rather use this noise to further train a series of fully-connected layers mapping this noise to a new space that they call the w-space. Figure 5.7 shows how the mapping network is built. This new space is much more disentangled, meaning that moving in this space does not change many features of the output, e.g. altering the input would only change the color of the hair of a generated person, instead of changing the color and gender for example. Although in music this type of changes are difficult to measure even in a

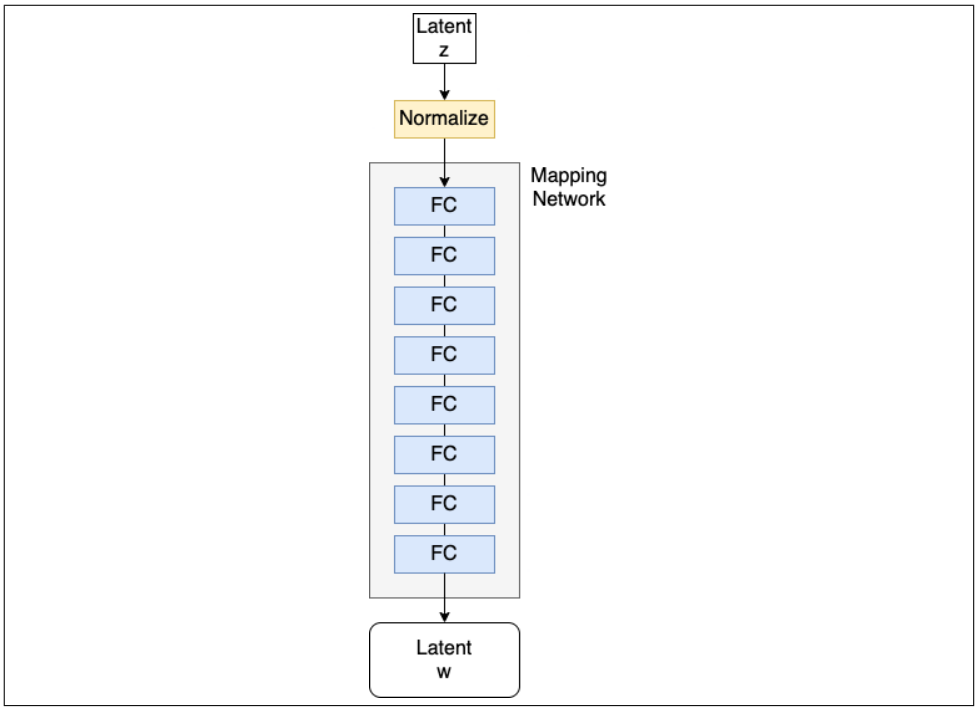

Figure 5.7. StyleGAN 2 mapping network. FC stands for fully connected layer.

qualitative manner. Because it requires a high level of musical training and knowledge to differentiate and identify between the musical characteristics of different composers and genres.

This disentanglement is highly desired in music to modulate the output of the generated pieces, in tonality, density of notes, polyphony, among other types of features that can be of interest to composers.

Some of the improvements of StyleGAN 2 with respect to the original StyleGAN architecture can be found in the synthesis network, where the progressive growing is discarded because of the phase artifacts it produced

on the resulting images. Instead they propose a high-resolution image generator similar to MSG-GAN (Karnewar & Wang, 2019). Also, the Adaptive Instance Normalization (AdaIn) blocks were changed for a modulation and normalization block to remove water-like droplet artifacts while still retaining control over the style mixing.

5.2.3. Evaluation

One of the main difficulties of working with generative models, specially in the music domain is that there is no objective method of evaluating if a generated piece of music is better than other because of the inherent subjectivity of music appreciation and lack of objective quality metrics.

We used the Fréchet Inception Distance (Heusel et al., 2017) metric, which uses the Fréchet distance to improve the Inception score (Salimans et al., 2016), to evaluate the generated images of pianorolls in comparison with the real examples seen by the network during training. This metric helps to discern between different models and implementations which generate the most similar images to a real pianoroll, therefore, the models that best capture the input data distribution. Because the Fréchet distance compares the real data distribution with the fake data distribution, it exhibits a better performance in assessing image quality. In equation 5.1 we can see how this metric works by measuring the Fréchet distance between two multivariate Gaussians, $X_r = \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g = \mathcal{N}(\mu_g, \Sigma_g)$ where r stands for real and g for generated.

$$FID = \|\mu_r - \mu_g\|^2 + tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{5.1}$$

We also calculated additional metrics derived from the C-RNN-GAN (Mogren, 2016) work, in order to have a better understanding of the generated pieces in comparison with the input data. This metrics consist in polyphony, number of silences, and used pitches. This metrics are an attempt to find an objective way of measuring the musical output of the developed models.

- **Polyphony**: Percentage of timesteps where there‘s more than one note played simultaneously.
- **Number of silences**: Percentage of empty timesteps.
- **Used pitches**: List of used pitches, used for determining known musical scales.

Although this metrics can not be directly optimized, they can be easily interpreted, for example, for different applications a person would want higher polyphony in the output and with lower number of silences, or the used pitches can be used to determine if the output belongs to a certain scale.

5.2.4. Training

5.2.5. Class conditions

Both architectures were trained from scratch using both datasets. For the MAESTRO dataset (Hawthorne et al., 2019), the class conditions consisted of the 6 composers with the most performances, according to Figure 5.1. For the LAKH dataset (Dong, Hsiao, Yang, & Yang, 2018), the class conditions consisted of the 13 genres derived from the Million Song Dataset (Bertin-Mahieux et al., 2011), which can be seen in Figure 5.4. Figure 5.8 shows how to condition the generation based on a composer.

5.2.5.1. DCGAN

This model was implemented in Tensorflow 2 with Keras and trained on 2 Nvidia GeForce GTX 1080.

5.2.5.2. StyleGAN 2

We used a modified version of the Tensorflow 1.14 implementation of StyleGAN 2 Ada (Karras et al., 2020), developed by NVidia, to use conditional classes. The Ada version of StyleGAN 2 was designed to improve training with limited data, they introduce an adaptive discriminator augmentation that stabilizes training. The conditional classes are encoded with both the generator and discriminator inputs. All the different variations of the model were trained on a Tesla V100 GPU in Google Colab Pro (Bisong, 2019), for approximately 40 hours which correspond to approximately 3000

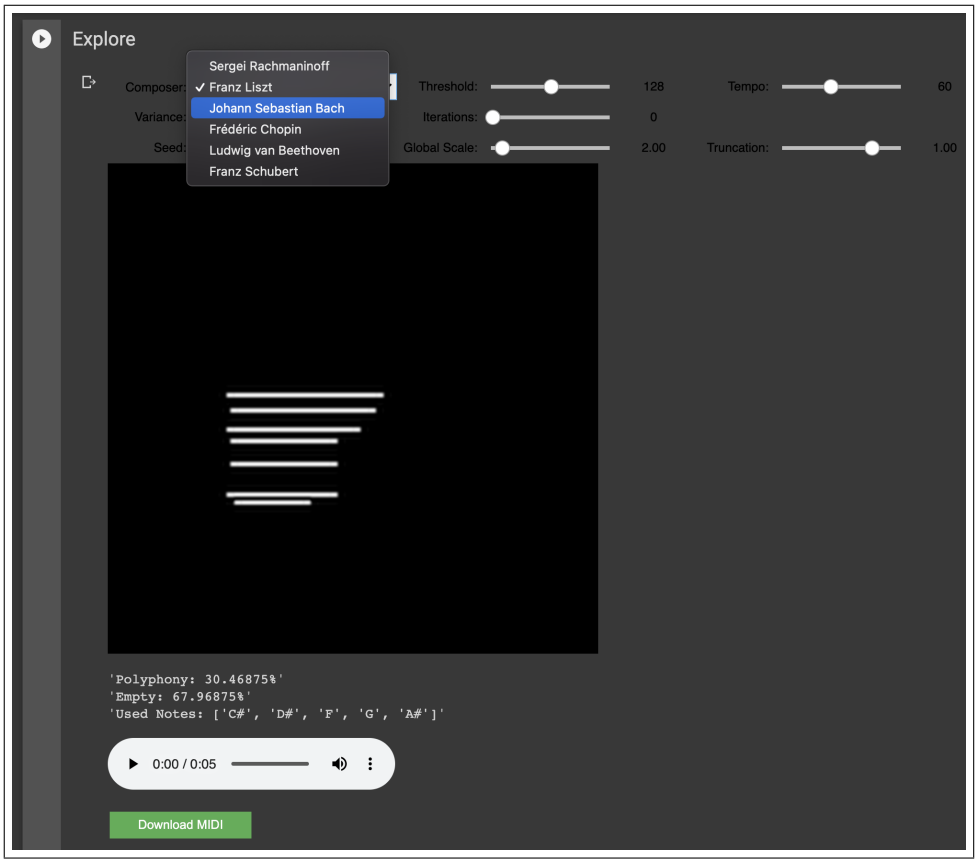

Figure 5.8. Google Colab interface showing how to select a class condition for generation, particularly for composers of the MAESTRO dataset.

kimg (3 million images) seen during training. Longer training time did not significantly improve the FID score for all the models.

5.2.6. Generation

To produce a pianoroll image utilizing the DCGAN model, the network is fed with an input noise (z) sampled from a normal distribution. If the

DCGAN model uses conditionality as input, the one-hot encoding of the desired class is concatenated.

For the StyleGAN2 models, the class identifier is embedded into a 512-dimensional vector that is concatenated with the latent code input (z) after normalizing each one (Karras et al., 2020), and passed as input to the generator.

The process of generating an audio file from the output image of any of these models has two parts: (1) defining a threshold and tempo for the generated piece, and (2) transforming the image to a numerical matrix. The first step is needed because the models returns a grayscale image, where the pixel values are between 0 and 255, and has 3 color channels. Figure 5.9 shows an image generated by StyleGAN 2 (Karras et al., 2019). With the threshold defined, we took the mean of the 3 channels and binarize the image using this threshold to determine which pixels correspond to the played notes, thus, obtaining the numerical matrix which can be transformed to a pianoroll using the pypianoroll package developed by (Dong, Hsiao, & Yang, 2018) to later convert it to a MIDI file. For listening to these files we used Timidity++ to convert them to a wave file in the preselected tempo.

We can generate new musical excerpts using this GAN models through exploration of the latent space, changing the input values to get new pieces. Another interesting musical application is to interpolate between two examples generated by the network, defining the number of steps we can generate a sequence of concatenated outputs while moving from one point in the latent space to another, as shown in Figure 5.10. In the supplemented folder

Figure 5.9. Output image from StyleGAN 2 model trained on the Maestro dataset, with input condition to be a piece from Johann Sebastian Bach. Analysing the pitches used, which are (D, E, F, G, G#, B), correspond to a G diminished wholetone scale. Also, the first group of 3 notes is a G major chord.

there are examples of different sequences from two random sampled points in the latent space, showing how the trained model evolves one excerpt into another in a series of steps.

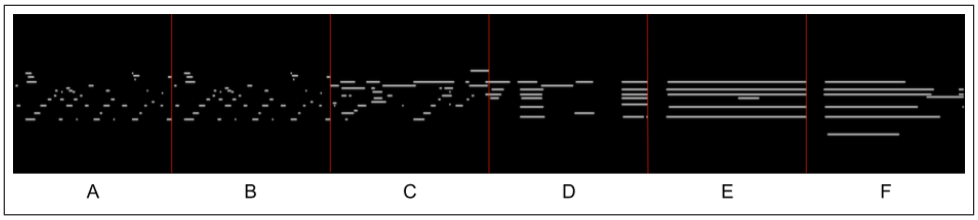

Figure 5.10. The StyleGAN 2 model is able to generate a variety of musical ideas (A to F). The latent space can also be interpolated between 2 outputs to generate a musically-meaningful sequence. In this case, the generated sequence exhibits how the network morphs from sample A to sample F in 4 steps, visually divided by a red line for easier differentiation.

6. RESULTS

6.1. DCGAN

The results obtained with the DCGAN model (Radford et al., 2015), with or without conditioning, were not good enough to provide subjectively acceptable pieces of music. Mainly because of common problems with GAN training, such as mode collapse, convergence problems, among others.

We tried many of the tricks seen on the literature to improve on this aspects (Salimans et al., 2016) (Mescheder, Geiger, & Nowozin, 2018) (Miyato, Kataoka, Koyama, & Yoshida, 2018), but the results showed that the symbolic music domain is more difficult than the image domain for this type of generative model. Even when the loss decreased during training, the qualitative results showed no signs of learning from the model. In Figure 6.1 we can see that the output from the model is very noisy and when transformed to audio is not a desirable outcome.

6.2. StyleGAN 2 ADA

Seven models were trained based on this work, two with the Maestro dataset (Hawthorne et al., 2019), with and without composer conditioning, and the other five for each of the instruments available in the Lakh dataset (Raffel, 2016) previously shown, all the Lakh models used conditional information as input. The evolution of the FID metric during training can be seen in Figures 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, most of this models converged

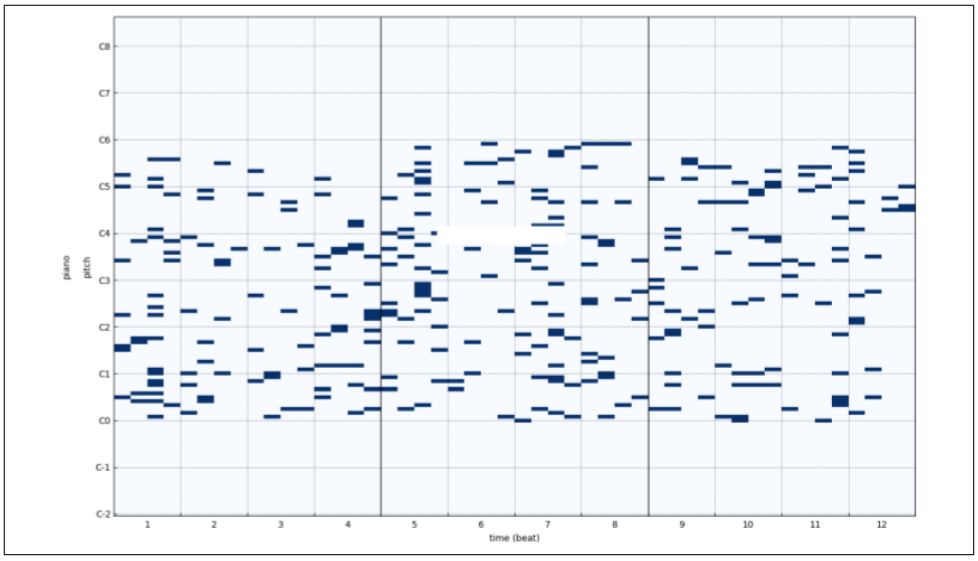

Figure 6.1. DCGAN image generated using the Maestro dataset.

with a similar FID score, except for drums which achieved the lowest FID of them all, we believe this is due to the stronger use of repetition in popular music drum tracks.

The Lakh models were trained for approximately 3000 kimg while the Maestro model was trained for almost twice as much time, but the model apparently converged on a value of around 30 in FID and did not improve on this score while training for longer time.

Listening to the results, it is observed that all models generate pieces similar to the input data, and although the FID metric is not as low as in what is seen in the literature for other types of images, we believe that this may be due to a bias in using an ImageNet (Deng et al., 2009) trained Inception network (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016). We attribute this bias in that the InceptionNet does not fully capture the

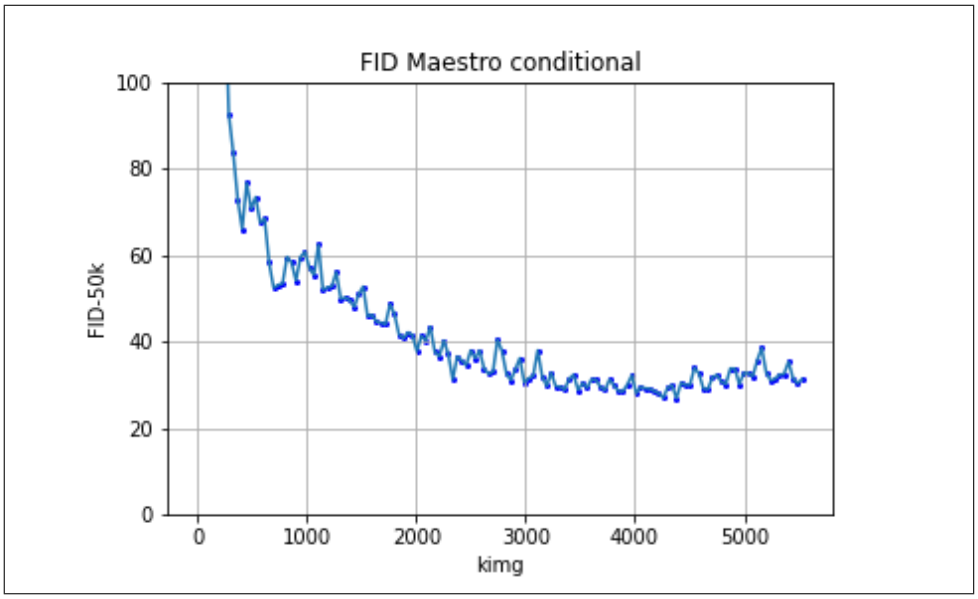

Figure 6.2. FID metric during StyleGAN 2 training with the Maestro dataset.

needed features for simpler type of images like the pianorolls. In section 8 we talk about ways to improve on this aspect.

To further compare the learned distribution, we calculated additional musical metrics which can be seen in Figures 6.8 and 6.9, we sampled 50k images for calculating this metrics.

6.2.1. Use Cases

To the best of our knowledge, we are the first ones to use the StyleGAN 2 model to generate symbolic music in the form of pianoroll images. This work can be seen as a creativity tool for composers, sound designers and the general public who wants to experiment with musical excerpts generated by an AI model. There are many use cases for this work that we investigated,

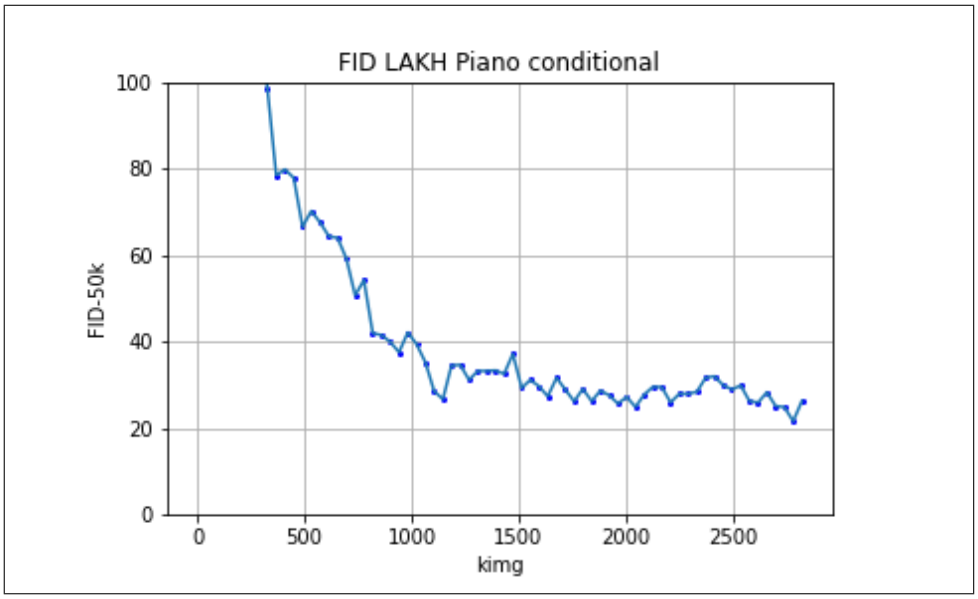

Figure 6.3. FID metric during StyleGAN 2 training with the Lakh dataset of piano.

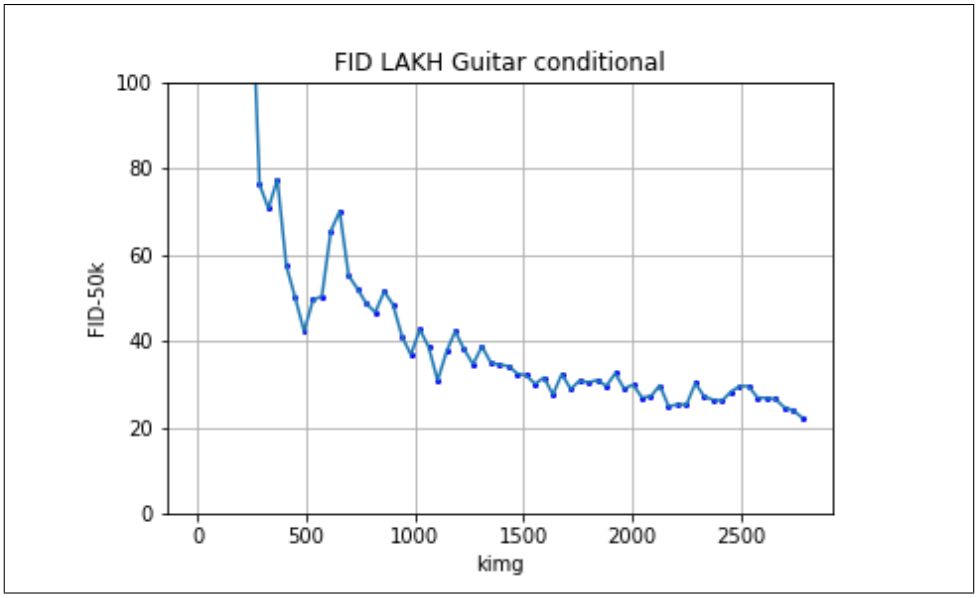

Figure 6.4. FID metric during StyleGAN 2 training with the Lakh dataset of guitar.

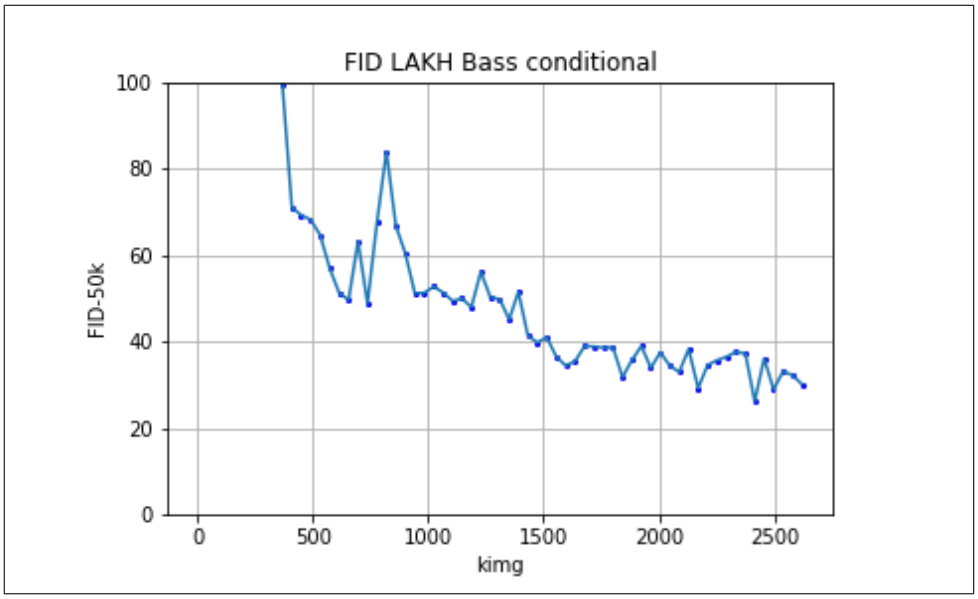

Figure 6.5. FID metric during StyleGAN 2 training with the Lakh dataset of bass.

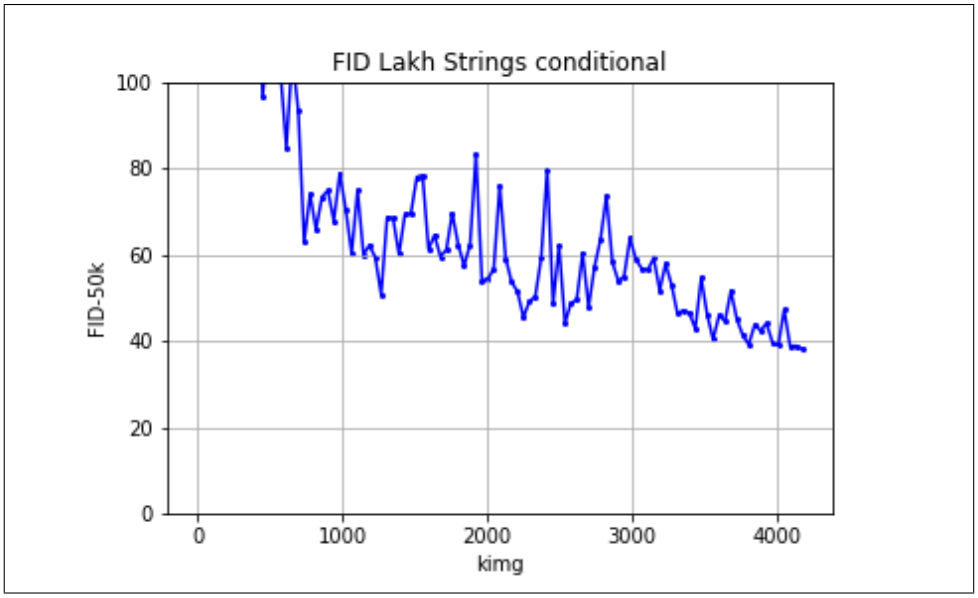

Figure 6.6. FID metric during StyleGAN 2 training with the Lakh dataset of strings.

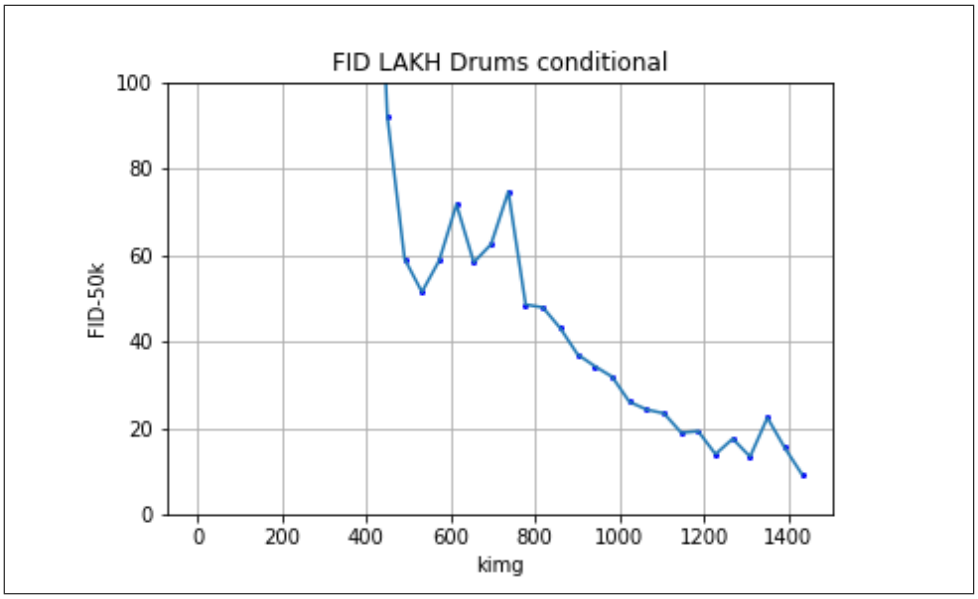

Figure 6.7. FID metric during StyleGAN 2 training with the Lakh dataset of drums.

and this results can be seen on multiple Google Colabs (Bisong, 2019) that are publicly shared as part of this work on section 6.4, and Figure 6.11 shows the interface of the available tools.

6.2.2. Exploring the latent space

The most interesting aspect of using generative models such as GANs is that we have a very large latent space to sample and obtain new outputs. Because StyleGAN uses a disentangled w-space rather than directly using the z-space sampled from a normal distribution to synthesize a new image, it is easier to find directions in this new latent space that change different features, without changing more that one. This is easier to understand in practice looking to how an image changes, but due to the lack of good

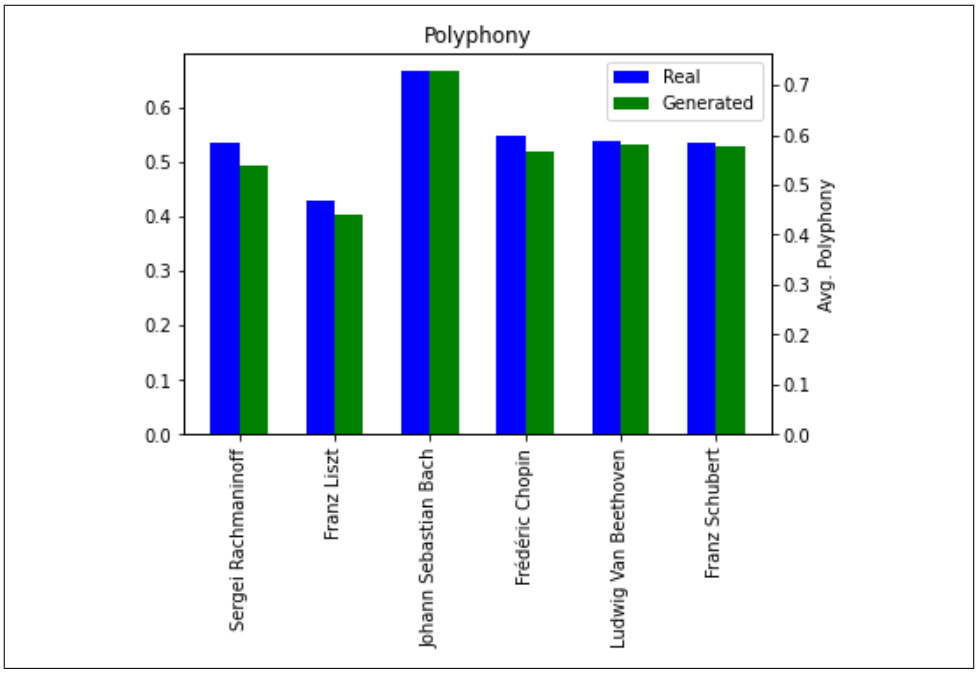

Figure 6.8. Polyphony metric for the Maestro Dataset

musical metrics is hard to understand which features are being changed. In the work of Ganspace (Härkönen, Hertzmann, Lehtinen, & Paris, 2020) there is an approach using Principal Component Analysis (PCA) to find this directions automatically rather than experimentally, in the supplemented audio material there are different directions generated by Ganspace in the MAESTRO dataset.

6.2.3. Projecting samples

Another use case is to find in the latent space the most similar sample to an input pianoroll image, whether to analyze if there is over-fitting of the network in the case that it learned the exact same pieces from the input data, but also to find a starting point similar to a previously defined piece

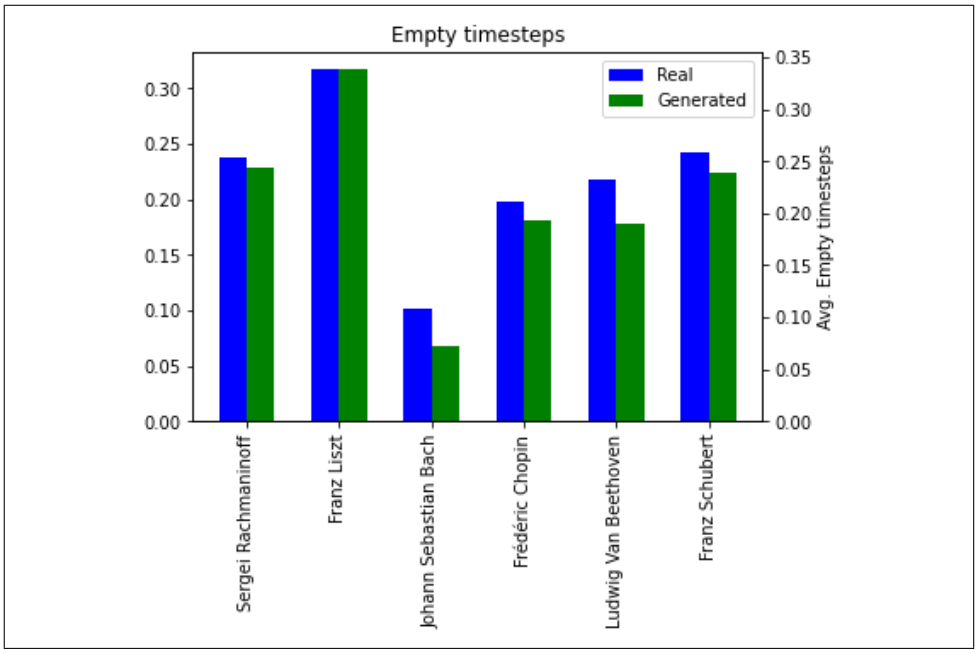

Figure 6.9. Empty timesteps metric for the Maestro Dataset

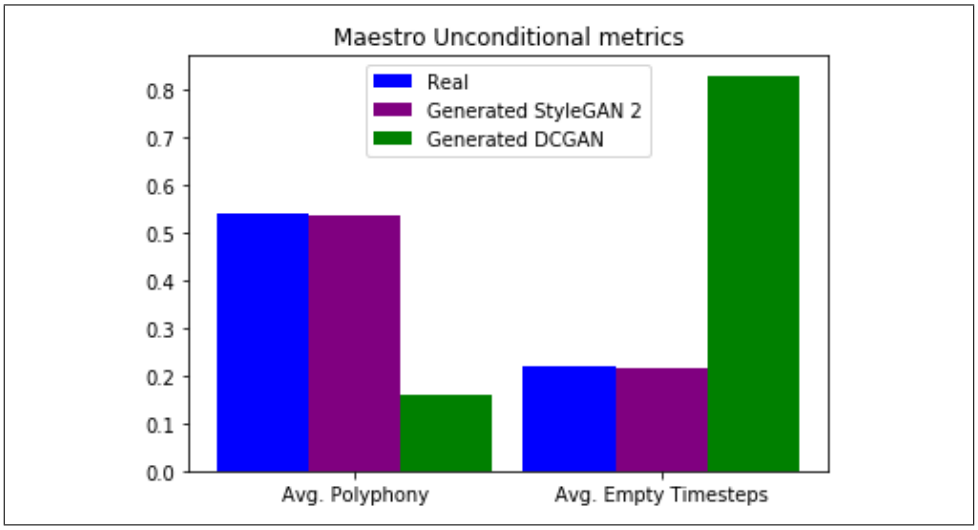

Figure 6.10. Metrics for the Maestro Dataset using unconditional models.

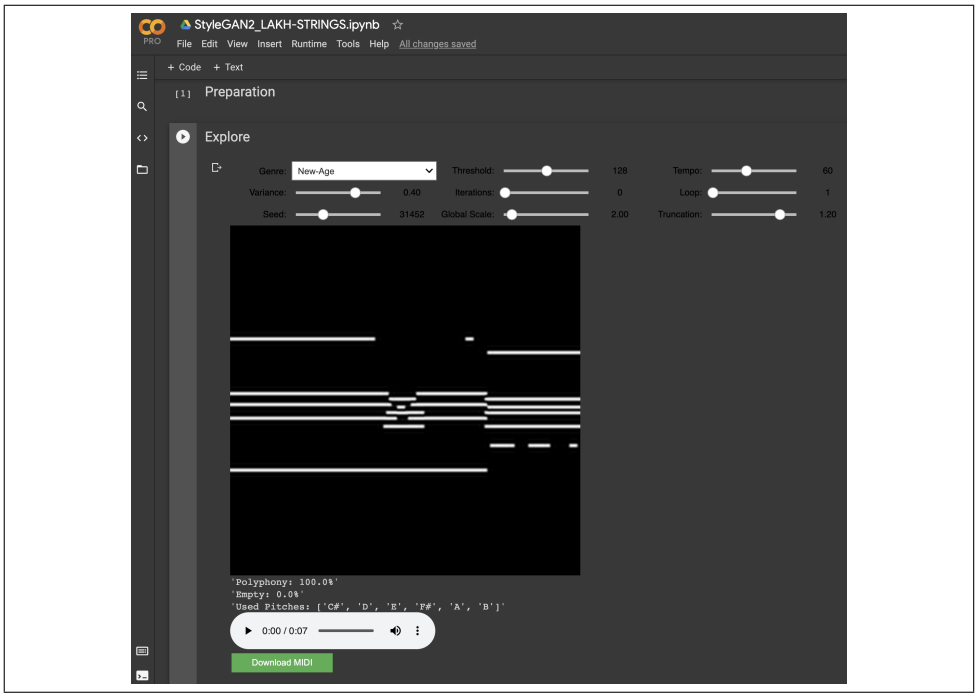

Figure 6.11. Google Colab interface for exploring the latent space of one of the StyleGAN 2 models trained on the Lakh dataset.

from the user. The projection can be done using any type of optimization technique to find the point in latent space that's closest to the target image according to a distance metric. In Figure 6.12 there is an example of a target sample and the projected result in the w-space, qualitatively, it can be seen that there are similar structures between both images, in addition to having a similar temporality of the note events.

6.2.4. Interpolating samples

Because of the fact that the disentangled w-space can change different features according to the direction of movement. We can interpolate

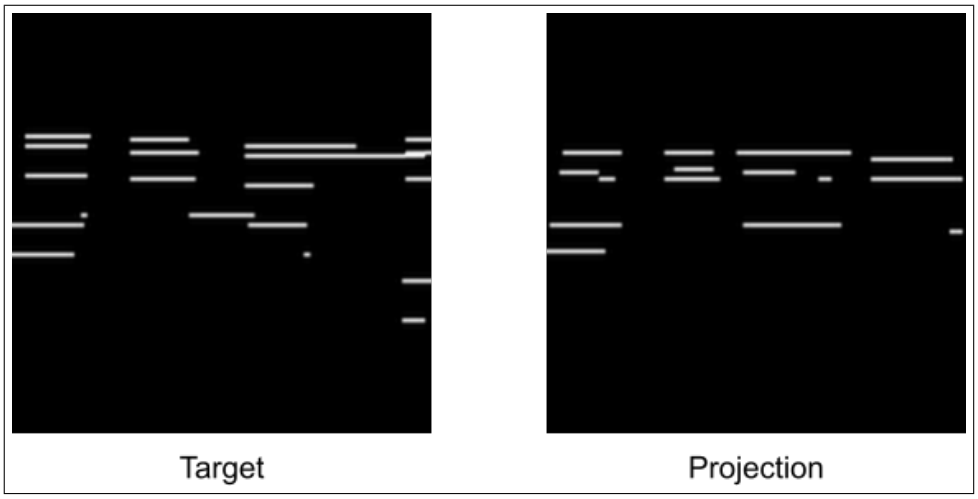

Figure 6.12. Target image to project on the latent space on the left, and resulting projection on the right.

between 2 points in this latent space in a determined number of steps to generate musical sequences that morph from one to the other. In Figure 5.10 there is a sequence generated by interpolating linearly between two points in the latent space, getting a way to convert one sample to another. There are other types of interpolation methods that can achieve new ways of moving between samples.

6.3. Comparison

In table 6.1 there is a comparison of the best FID score for both models on different datasets. The DCGAN model could not achieve results that were not similar to noise to be able to compare with the StyleGAN 2 model, although, in Figure 6.10 we still calculated the musical metrics for all the unconditional version of the models to compare it to the original distribution. The DCGAN results differ a lot from the original data distribution,

while the StyleGAN 2 model is almost the same to the input data, showing a correlation between FID and the musical metrics.

Table 6.1. Comparison table of the best FID results for each model on different datasets. 50000 images were sampled for calculating this metric.

FID 50k Samples						
Model	Maestro	Lakh Piano	Lakh Guitar	Lakh Bass	Lakh Strings	Lakh Drums
DCGAN	259.9782	192.6182*	192.6182*	192.6182*	192.6182*	192.6182*
StyleGAN2	26.8372	21.5397	22.2263	26.0752	38.1092	9.2536

*Same value because it was multi-instrumental generation.

6.4. Models availability

We published the StyleGAN 2 models for exploration on Google Colab which can be used in the following links:

- Maestro
  (i) `https://colab.research.google.com/drive/1bueGBYpBX-M4dX1y4FdY3mChocCI3Qa4?usp=sharing`
- Lakh
  (i) Bass `https://colab.research.google.com/drive/1ejmpgtI6woanbTK04xp31DHfkBCt0IHs?usp=sharing`
  (ii) Drums `https://colab.research.google.com/drive/1M54m4JfX-shEX-IVHreanADFRNacuiZB?usp=sharing`

(iii) Guitar https://colab.research.google.com/drive/1_IAyo-YStHYnvUDnyrtv-ThxFz2UeV2c?usp=sharing

(iv) Piano https://colab.research.google.com/drive/1t-wcqUpSSfsrWr2-5oZW1VuxGJTi-xum?usp=sharing

(v) Strings https://colab.research.google.com/drive/13x9kofba2IzaT_p8ktDuaimBnrK0T9u7?usp=sharing

6.5. Audio examples

We collected different samples from both models and made them available in the following link: https://drive.google.com/drive/folders/1p2H_jTl5vGdeZLuP9H9qT7dL_lLhj4Rg?usp=sharing

7. LIMITATIONS

7.1. Hardware

The StyleGAN 2 model used in this work has approximately 30 million trainable parameters, because this and the type of operations the hardware needed is very demanding. As previously stated, this models were trained on a Tesla V100 which has 16Gb of GPU RAM, video cards with lower RAM had more problems when training, needing to change training configurations to diminish the requirements, resulting in difficulties when training.

7.2. Image Size

The official implementation of StyleGAN 2 from Nvidia only accepts squared images, which for this application is a limitation of the length of pianorolls capable of producing. In the vertical axis of the images we have the different notes available to play, which are 128 in the MIDI protocol, and on the horizontal axis of the images there is the temporal placement of these notes. With a longer horizontal axis we could represent more time from a given performance or song, resulting in a model that can generate longer pieces. There are two solutions for this problem, one is to use a larger squared image size like 256, 512 or 1024, and only use a predefined portion of the vertical axis to represent the notes and the rest as black pixels, or scale the notes in the vertical axis, letting them use more than one pixel of height. The other solution to this problem is to modify the implementation

to accept rectangular images, although there are some implementations that do this, none of them have the conditional input implementation.

7.3. Pianoroll representation

One of the main advantages of using the pianoroll representation, is that it is easy to pass this information as an image to convolutional networks. But there is a loss of information in this MIDI to pianoroll transfer. Unlike the MIDI format where there are specific messages to communicate when a note start and stop playing. Depending on the quantization used, if there are 2 consecutive notes without silence in between, in the resulting matrix these will become a single long note. Therefore, these small details of the performance are lost in the transformation.

8. FUTURE WORK

The field of generative models is rather new, and has been increasing in popularity on this recent times because of the huge array of applications to different areas, but in music there is still a wide gap in what this models can do and what they are generating to this day.

We believe that the approach shown in this work can be a first step to explore the use of the StyleGAN 2 (Karras et al., 2019) architecture in the music realm, both in its symbolic form and in the raw audio domain.

For future work, we want to conduct a user study to compare our results against other symbolic music generation models. Also, the velocity information was discarded in this work, so a logical next step is to use this information to have better note articulation of the generated pianorolls. Another idea is to extend the StyleGAN 2 model to work with multi-instrumental pieces of music, using each channel of the input image for a different instrument, hoping that the network is capable of learning to generate musical pieces that work together rather than having little to none interaction between instruments.

For future research we believe that this models can be trained on audio spectrograms to generate new sounds that can be modulated using the disentangled w-space, as can be seen on how easy is to modulate the output of the StyleGAN 2 model, which can be a powerful tool for composers and sound designers.

Addressing one of the limitations of the StyleGAN 2 model, we want to train versions with larger inputs, not only constrained by the square input, there are variations on this model that can achieve this, but they need more iterations to work with different conditional classes and arbitrarily shaped inputs. Other possibility for further research is to use an extra recurrent network to model the implicit temporality of music and generate multiple images that form a sequence.

Finally, we want to train an InceptionNet (Szegedy et al., 2016) specifically for symbolic music like pianorolls, hoping that a specific model better represents this type of images and test if it correlates with better audible results. Another approach to this would be to experiment with different layers of an ImageNet (Deng et al., 2009) trained network, because in convolutional networks the first layers tend to capture more general features like textures and shapes. Finding a better way to evaluate this type of models is important to have better tools to compare different symbolic music models across the literature and with a musical foundation to interpret the results.

9. CONCLUSIONS

In this work, we explored how to generate music using deep generative models, presenting two models based in GANs (Goodfellow et al., 2014). One of these models did not perform as desired, despite efforts made to make it work in this new domain. The second model had innovative and interesting results for the area in terms of a new approach to how it is possible to generate music using methods designed for high quality image generation, specifically realistic object generation.

The final results, which answer **RQ1**, show that it is possible to use image-based models to generate symbolic music, as is the case with both DCGAN (Radford et al., 2015) and StyleGAN 2 (Karras et al., 2019), despite the fact that the DCGAN model did not have the expected results in the proposed metrics or in the auditory evaluation, there is still room to improve this type of model and obtain better results, as seen on the literature on different variations of this type of GAN architecture.

In StyleGAN 2, the results exceeded the expectations set at the beginning of this work and showed a lot of potential for creating tools to enhance creativity on composers. Since the generated latent space is disentangled and allows its exploration, it remains to be able to find a simple way to connect different directions within the space with musical characteristics, although Ganspace (Härkönen et al., 2020) is an alternative, further investigation and better metrics are needed to adapt it to the symbolic music domain.

With respect to **RQ2**, the results show that the generated musical pieces with or without the input conditions, are very similar to the training data distribution, according to the FID metric and the proposed musical metrics, as seen on the comparison in Figure 6.10.

Finally, we have found a new domain where the architecture type of StyleGAN 2 can also stand out and generate new uses and applications for the field of musical composition, where it can be highlighted how this type of models can expand the creativity of its users, by providing multiple ideas in the form of a pianoroll image that can be altered moving through the latent space. Allowing to interpolate between images to create moving sequences or to project an existing idea to see alternatives provided by the model. This is a first approach on symbolic music generation using a StyleGAN 2 architecture, and more research on this topic is promising for the future of music and artificial intelligence.

REFERENCES

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214–223).

Association, I. M., et al. (1983). Midi musical instrument digital interface specification 1.0. *Los Angeles*.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011).*

Bisong, E. (2019). Google colaboratory. In *Building machine learning and deep learning models on google cloud platform* (pp. 59–64). Springer.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dinculescu, M., Engel, J., & Roberts, A. (2019). Midime: Personalizing a musicvae model with user data.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2018). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Dong, H.-W., Hsiao, W.-Y., & Yang, Y.-H. (2018). Pypianoroll: Open source python package for handling multitrack pianoroll. *Proc. ISMIR. Late-breaking paper;[Online] https://github.com/salu133445/pypianoroll*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Hadjeres, G., Pachet, F., & Nielsen, F. (2017, 06–11 Aug). DeepBach: a steerable model for Bach chorales generation. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1362–1371). PMLR. Retrieved from http://proceedings.mlr.press/v70/hadjeres17a.html

Härkönen, E., Hertzmann, A., Lehtinen, J., & Paris, S. (2020). Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., ... Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International conference on learning representations.* Retrieved from https://openreview.net/forum?id=r1lYRjC9F7

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., ... Eck, D. (2018). Music transformer. *arXiv preprint arXiv:1809.04281*.

Karnewar, A., & Wang, O. (2019). Msg-gan: multi-scale gradient gan for stable image synthesis. *arXiv preprint arXiv:1903.06048*.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020). *Training generative adversarial networks with limited data.*

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2019). Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*.

Mescheder, L., Geiger, A., & Nowozin, S. (2018, 10–15 Jul). Which training methods for GANs do actually converge? In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 3481–3490). PMLR. Retrieved from `http://proceedings.mlr.press/v80/mescheder18a.html`

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *CoRR*, *abs/1802.05957*. Retrieved from `http://arxiv.org/abs/1802.05957`

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

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

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

Raffel, C. (2016). *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching* (Unpublished doctoral dissertation). Columbia University.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018, 10–15 Jul). A hierarchical latent vector model for learning long-term structure in music. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 4364–4373). PMLR. Retrieved from `http://proceedings.mlr.press/v80/roberts18a.html`

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). *Improved techniques for training gans.*

Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F., ... Peters, A. (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*, *194*, 105596.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016, June). Rethinking the inception architecture for computer vision. In *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr).*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez,

A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

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

APPENDIX

A. DCGAN ARCHITECTURE

A.1. Generator

Layer	filters	kernel_size	strides
Conv3DTranspose	512	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Conv3DTranspose	256	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Conv3DTranspose	128	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Conv3DTranspose	64	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Conv3DTranspose	16	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Conv3DTranspose	4	(1, 1, 4)	(1, 1, 4)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Dropout	-	-	-
Reshape	-	-	-

A.2. Discriminator

Layer	filters	kernel_size	strides
Conv3D	128	(1, 1, 2)	(1, 1, 2)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	128	(1, 1, 3)	(1, 1, 3)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	128	(12, 1, 1)	(12, 1, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	128	(7, 1, 1)	(7, 1, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	128	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	128	(1, 2, 1)	(1, 2, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	256	(1, 4, 1)	(1, 4, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Conv3D	512	(1, 3, 1)	(1, 3, 1)
BatchNormalization	-	-	-
LeakyReLU	-	-	-
Flatten	-	-	-
Dense (1)	-	-	-