A Multi-Dimensional Approach to Generative Adversarial Networks for Deep Generative Music

CARLIER Jacinthe
Art & Technology Dpt
Media Technology Major
120180346

Abstract

In this thesis, we combine Iannis Xenakis' sound generation methods with generative neural networks in an adversarial synthesis model. We discuss the microsound scale and granular methods as well as common representations for audio in deep learning to experiment Xenakis' screen generation in a Generative Adversarial Network trained on improvised jazz music, confronting random distribution for the Generator model and human performed randomness in the music dataset provided to the Discriminator model.

Research goals and motivations:

As Generative Adversarial Networks have been a popular topic in the deep learning community for the past few years, most attempts have had the purpose to generate images. Here we focus on generating music, which is also an exciting area of research with results getting better and better, but that still sound very experimental and sometimes noisy.

On another hand, Iannis Xenakis' work focuses on a very mathematical approach of music generation called 'Formalized Music'. His work tries to drift away from the concept of generating music that sounds good to the human ear, that is to say that music that we are already used to hear.

Generating music with deep learning involves using a database and taking the risk (or completely accept) that the generated data might have similarities with the original database fed to the deep learning model, but using data generated with Iannis Xenakis' methods could help generate results that are more different.

Research design and methodology:

An algorithm like GANs can allow us to compare music produced in what we will call a traditional way and music generated from Xenakis' methods using the Discriminator and Generator to their full potential.

To Discriminate against our Generative methods, we will use recordings of Jazz improvisation. Improvisation is a common practice for musicians, that allows randomness into a performance. But as in mathematics, this randomness has rules and limits (for example a chord progression that allows musicians to sound good improvising together). We cannot talk about pure randomness here.

Data was gathered on Archive.org, that contains (among other content) the Boston Public Library 78rpm collection, its description being as follows :

> *"The Boston Public Library (BPL) sound collection includes hundreds of thousands of audio recordings in a variety of historical formats, including wax cylinders, 78 rpms, and LPs. The recordings span many genres, including classical, pop, rock, jazz, and opera – from 78s produced in the early 1900s to LPs from the 1980s. These recordings have never been circulated and were in storage for several decades, uncataloged and inaccessible to the public. By collaborating with the Internet Archive, Boston Public Libraries audio collection can be heard by new audiences of scholars, researchers and music lovers worldwide. "*

Not the whole collection was used in the following experiments, only live albums have been chosen, assuming most jazz live performances have moments of improvisation. The dataset is in that sense not just pure improvisation, as it would be hard to isolate what is improvisation and what is not. In this collection albums being in between the 1900s and the 1980s, quality of recording (especially for live albums) is not optimum, which might result in more noisy data after training.

During experiments, two versions of the dataset were used : a small one composed of 15 tracks, and a bigger version with 50 tracks. All tracks being between 3 and 9 minutes.

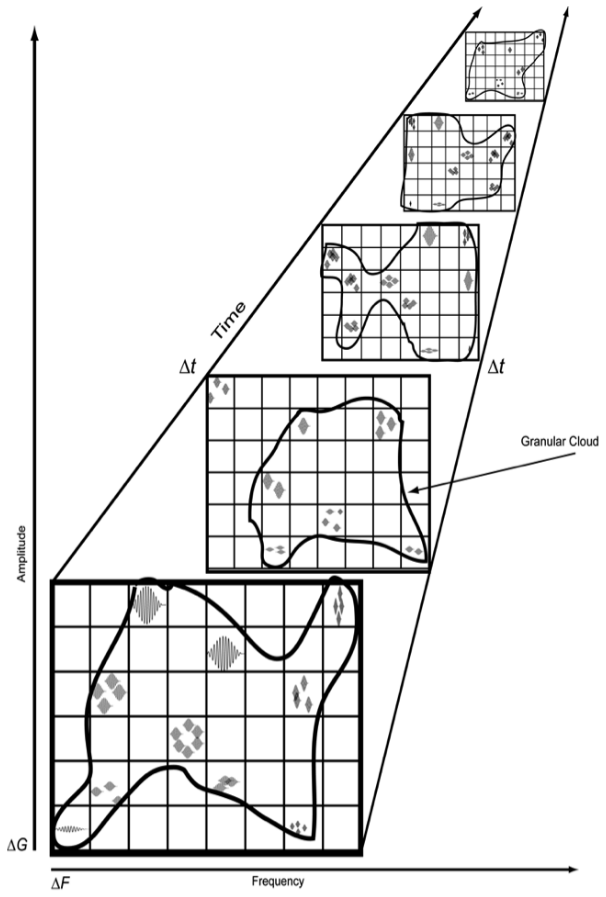
Type of data:

Symbolic data makes the training process easier and faster as for example spectrograms transforming audio into images. But that process, as much as using MIDI data or piano rolls, results in a loss of information compared to raw data. Using symbols means simplifying data and as our goal is to get music that sounds original, we need all the data we can get, even if using raw data requires huge deep learning models and a lot of RAM and GPU power. Google Colab Pro provides higher-RAM and bigger GPUs for faster training at low-cost (10$USD/month).

Raw data means using the sample numbers contained in a raw audio file (often .wav or .flac). This kind of data has usually one (mono) or two (stereo) channels.

For our Generator, instead of feeding it with a vector of numbers generated by a simple random distribution, we will use sounds generated with Xenakis' methods.

Xenakis generates sounds using multi-dimensional data, forming a screen-like image filled with short-duration grains forming clouds. To obtain this, time, frequency, amplitude and density information can be generated with random distributions.



Xenakis' screen

The screen itself doesn't need to be recreated as an image per say and can be fed as audio data into the model, which is what was done in this experiment.

Sounds were generated using a python DSP library named Pyo. With an amplitude table to start with, density, duration and frequency can be modulated too. Different sounds were generated by changing the random distributions' parameters for each dimension (1)(2).

35 tracks of around 30 seconds were made with this method to feed the Generator model.

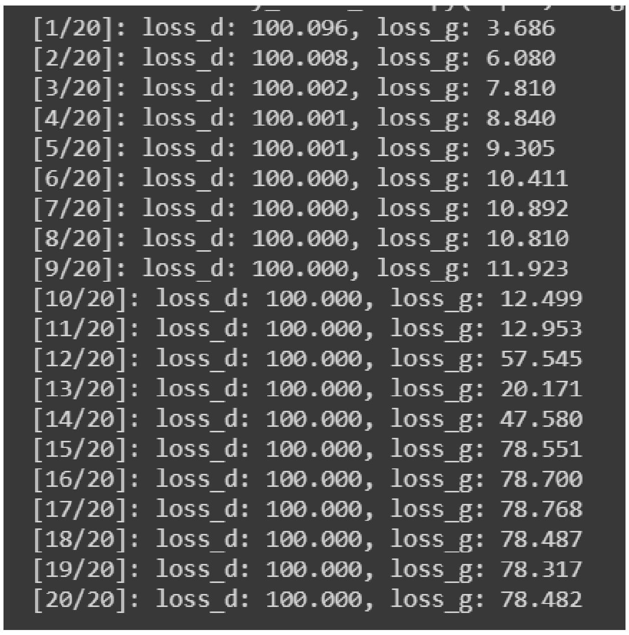
Experiments:

- Train a simple audio GAN with our dataset of jazz improvisation
- Add sounds made from Xenakis' screen in the Generator instead of using a random distribution
- Train a simple audio GAN with sounds made from Xenakis' screen
- Repeat the same experiments with Xenakis' screen method not as audio data but as images

First experiences were made with GAN model made of linear layers, with 1 second slices of audio (for a smaller amount of data processed at once) at a sample rate of 16kHz (standard in raw audio models, the higher the sample rate the more data we have to go through in one file).

generated3.wav (Ligne de commande)

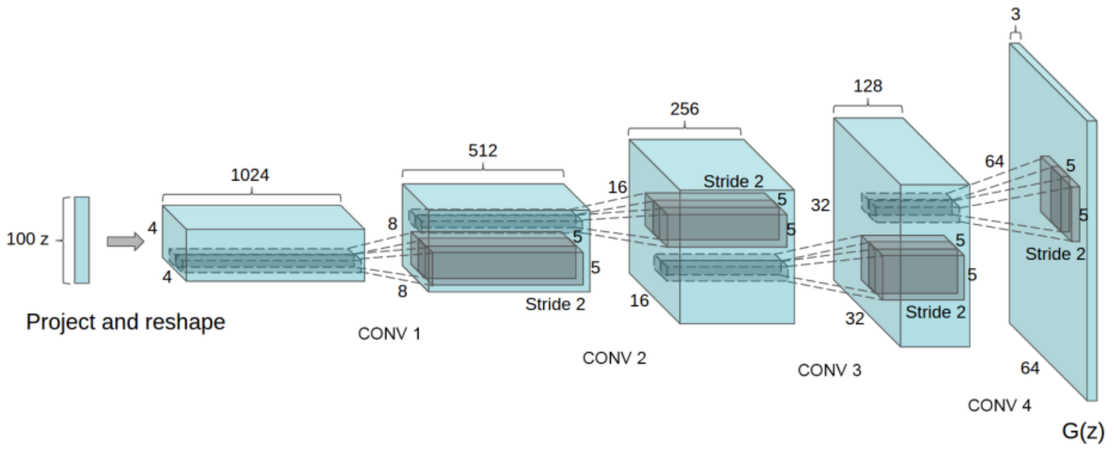
Results were very noisy even though changes in amplitude and frequency can be heard. The loss of the discriminator and generator ended up either at 100 or 0, resulting in convergence failure.



```
[1/20]: loss_d: 100.096, loss_g: 3.686
[2/20]: loss_d: 100.008, loss_g: 6.080
[3/20]: loss_d: 100.002, loss_g: 7.810
[4/20]: loss_d: 100.001, loss_g: 8.840
[5/20]: loss_d: 100.001, loss_g: 9.305
[6/20]: loss_d: 100.000, loss_g: 10.411
[7/20]: loss_d: 100.000, loss_g: 10.892
[8/20]: loss_d: 100.000, loss_g: 10.810
[9/20]: loss_d: 100.000, loss_g: 11.923
[10/20]: loss_d: 100.000, loss_g: 12.499
[11/20]: loss_d: 100.000, loss_g: 12.953
[12/20]: loss_d: 100.000, loss_g: 57.545
[13/20]: loss_d: 100.000, loss_g: 20.171
[14/20]: loss_d: 100.000, loss_g: 47.580
[15/20]: loss_d: 100.000, loss_g: 78.551
[16/20]: loss_d: 100.000, loss_g: 78.700
[17/20]: loss_d: 100.000, loss_g: 78.768
[18/20]: loss_d: 100.000, loss_g: 78.487
[19/20]: loss_d: 100.000, loss_g: 78.317
[20/20]: loss_d: 100.000, loss_g: 78.482
```

Losses for Linear Model, convergence failure

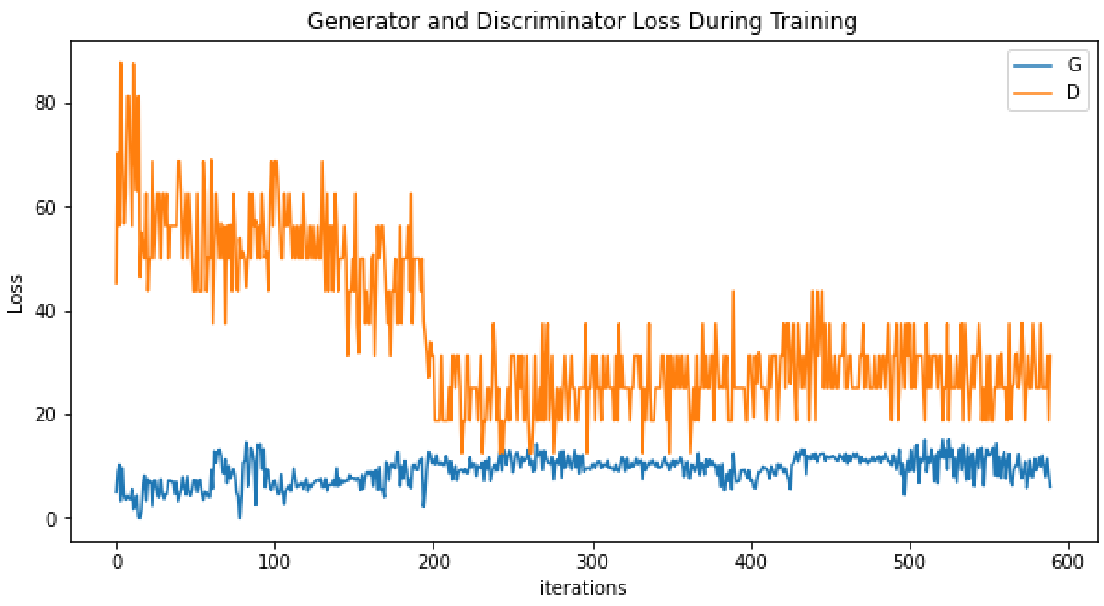
Follow-up experiments were made with a 2D convolutional model, more precisely a DCGAN. This model is known to be very efficient and reliable for image generation, so it was worth trying to see its application to audio. Layers were adapted from the original (below). The size of images in the original model is 64x64, but raw audio slices are bigger than that depending on their duration and sample rate.



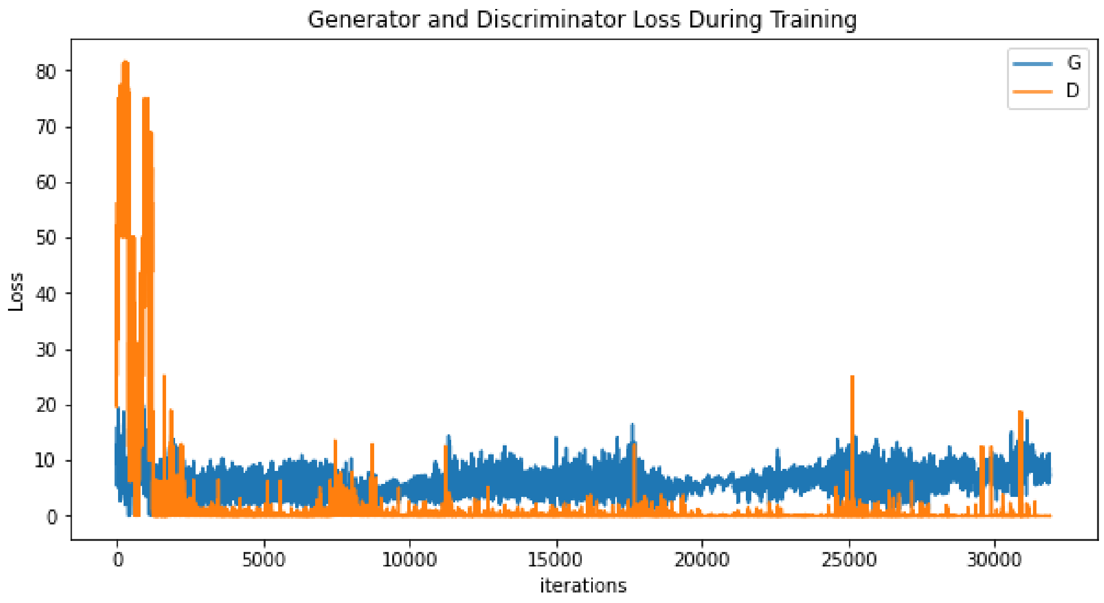
DCGAN Model architecture

DCGAN is also the model that inspired WaveGAN, a raw audio waveform generation model.

Slices of audio were made longer (10 secs) as in audio generation, it can do good to preserve them so. Indeed, a song can be divided in musical phrases (which is a term actually used in music theory) and making the model learn those phrases as a whole can be more interesting than just cutting them in slices, as they have meaning as much as words forming sentences together. Of course, it would be a very meticulous job to separate those musical phrases perfectly, so in that case using audio slices as big as possible is better.



Loss plot for a model with 10 sec slices. Discriminator and generator loss stabilize but remain high.



Loss plot for a model with 1 sec slices. Discriminator loss is stable although stays high, and Generator loss stays a bit unstable

In further experiments, using dropout layers helped as well as additional convolution layers. When using 10 sec slices, the dimension of the layers should also be adapted. Although a bigger model was hard for the Google Colab Pro GPU/RAM to handle.

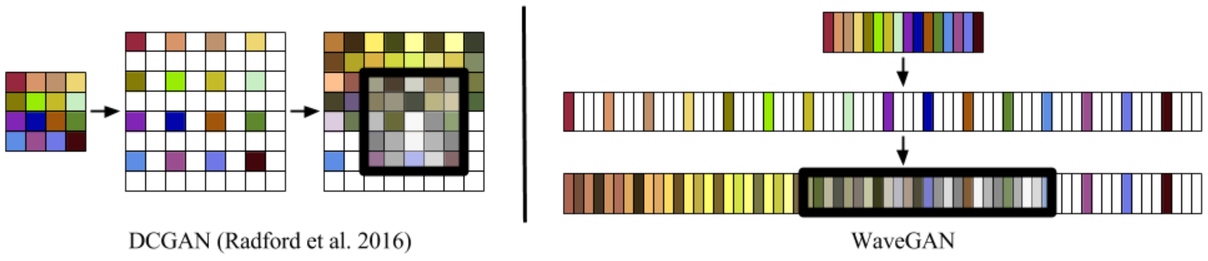
generated47.wav (Ligne de commande)

generated28.wav (Ligne de commande)

Results are still noisy but some kind of rhythm can be heard now.

For future experiments, other cloud services such as Microsoft's Azure can be of help when it comes to GPU limits in order to use more data in the original database and longer slices.

Also, building a 1-dimensional convolutional model like WaveGAN might be more appropriate for audio data, as it does not have height and width like images.



DCGAN vs WaveGAN, 2D vs 1D convolutions

Other parameters such as the learning rate, kernel size and stride of layers can be also modulated to curate different results, but it turned out their settings in the original DCGAN was the most satisfactory.

Main References:

- Briot, Jean-Pierre & Hadjeres, Gaëtan & Pachet, Francois. (2017). Deep Learning Techniques for Music Generation - A Survey.
- Xenakis, Iannis (1971), Formalized Music Thought and Mathematics in Composition, Indiana University Press.
- Goodfellow, Ian J. et al. "Generative Adversarial Nets." NIPS (2014).
- Hoffmann, Peter. (2009). Music Out of Nothing? A Rigorous Approach to Algorithmic Composition by Iannis Xenakis. 10.14279/depositonce-2292.
- Chris Donahue and Julian J. McAuley and Miller S. Puckette (2019), Adversarial Audio Synthesis, ICLR 2019.
- Roads, Curtis (2004), Microsound, MIT Press.
- Soroush Mehri & Kundan Kumar & Ishaan Gulrajani & Rithesh Kumar & Shubham Jain & Jose Sotelo & Aaron C. Courville & Yoshua Bengio, (2016), SampleRNN: An Unconditional End-to-End Neural Audio Generation Model, ICLR 2017.
- https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/