Generative Adversarial Networks for Audio Generation: A Comparative Study

Ernest Vanmosuinck
s3210359

Antonio Mone
s3113159

Alessandro Marincioni
s3442209

Adetunji Ayodele
s3403181

June 20, 2022

Abstract

Recently, more and more attention has been given to the use of Generative Adversarial Networks (GANs) as a mean of synthesizing data. In this work, we consider the potential application of GANs for the difficult task of synthesizing audio and reproducing the rhythmic sound of rock music. We investigate and experiment on four different models: WaveGAN, SpecGAN, MP3net and GANsynth. We obtained promising results and discovered that those models have very high potential in synthesizing audio, however they are rather slow to train.

1 Introduction

So far much of related works have been applied to use cases outside of the data confidentiality domain with a common application being the production of artificial images. Here we consider the potential application of GANs for the purpose of generating different samples of rock musics.
Generative Adversarial Networks (GANs), first proposed by Goodfellow et al. (2014), are the focus of much of the research literature. GANs are a generative deep learning technique that use artificial neural networks. The generative element of a GAN does not access the original data whilst training, and can therefore produce synthetic data without directly interacting with the original data; this is an interesting feature that might in theory reduce disclosure risk. A representation of a typical GAN architecture can be observed in Figure 1.

2 Problem Description

The aim of this project is to investigate different methodologies employing Generative Adversarial Networks to synthesise audio and reproduce the rhythmic sound of rock music. We aim to use WaveGAN, SpecGAN, MP3net and GANSynth, as they are all employing different types of GAN models. While the subject of audio synthesis is not a novel/state of the art subject, it is not a very popular domain of research when investigating Generative Adversarial Networks. Furthermore, we want to investigate if such networks can be utilized to model a more technical music genre in the form of rock/pop-rock music.

3 Related Work

Whilst research into using GANs for mixed-type, or tabular, microdata has so far been limited, GANs are generating much research interest and have been used for various applications. Research has focused on purposes such as: the generation of artificial images of plant seedlings using generative adversarial networks [4] (Simon L, Madsen et 2019); image colorization (e.g. Nazeri et al. (2018)); image-to-image translation (e.g. Huang et al.(2018), Isola et al. (2017), and Zhu et al. (2017)); image inpainting (e.g. Demir and Unal (2018)); super-resolution (e.g. Ledig et al. (2017) and Menon et al.

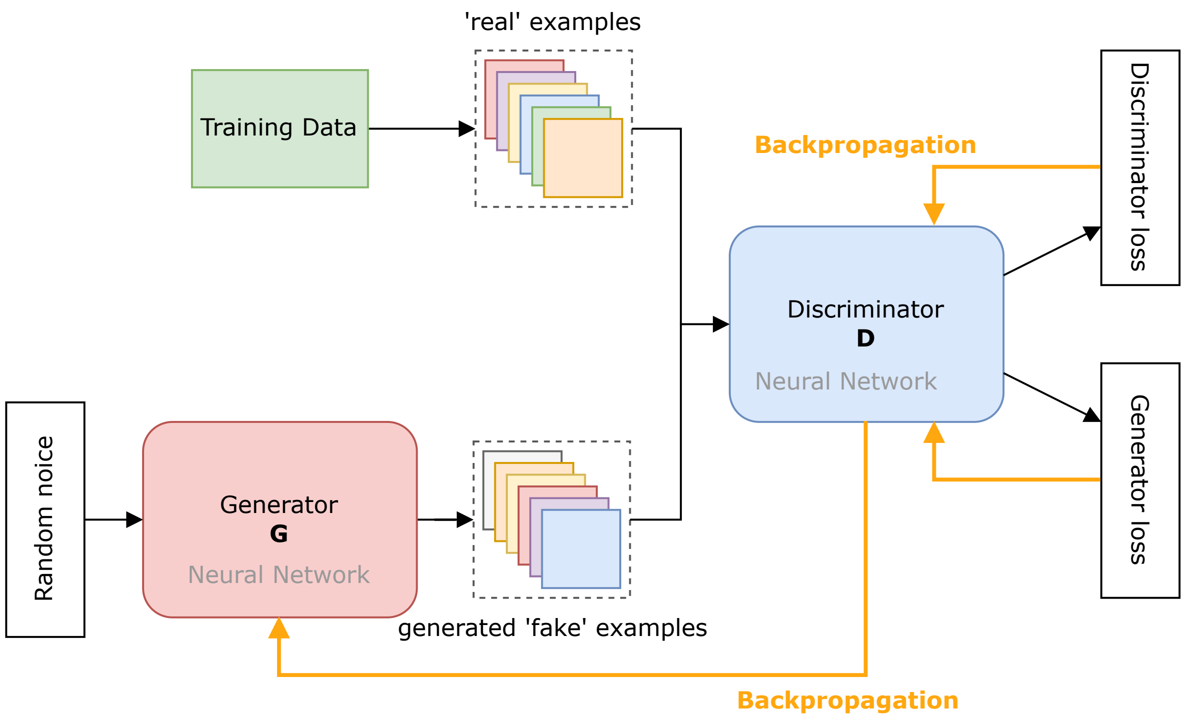

Figure 1: Generative Adversarial Network Architecture

(2020)); and synthetic medical image generation (e.g. Frid-Adar et al. (2018), Iqbal and Ali (2018), Piacentino et al. (2021), and Sandfort et al. (2019), and of course many more. The field of music generation through GANs is however still a rather novel field that is starting to be explored.

4 Implementation

4.1 WaveGan & SpecGAN

WaveGAN and SpecGAN are built around the same structure, which is built off of the architecture of DCGAN [6], which popularized the use of GANs for image synthesis. It uses a transposed convolution operation (as seen in figure 2) to upsample low-resolution feature maps to high-resolution images. Donahue et al. [1] modified the DCGAN model to better handle applications to audio data, for example modifying the two dimentional 5×5 layers of DCGAN to a one-dimensional layer of length 25. Furthermore, DCGAN outputs 16×16 images, representing 4096 audio samples. Donahue et al. added a layer to produce 16384 audio samples, slightly bigger than one second of audio at 16Hz. WaveGAN uses the waveform representation of audio, while SpecGAN uses the spectogram representation of an audio sample using short-time Fourier transform. This extra step for SpecGAN makes the task harder for the machine to learn as it can introduce more noise.
Luckily for us, WaveGAN and SpecGAN implementations were readily available for usage and testing in python, through the GitHub repository *WaveGAN* written by Chris Donahue et al. [2]. The implementations are capable of handling short audio samples as well as longer ones. Since we had two different datasets, it was important for us to have an implementation that would easily allow for the tuning of parameters to test each dataset separately.

4.2 MP3net

MP3net [8], on the other hand, is a deep 2D convolutional GAN that uses techniques from Vorbis/MP3 audio compression and is inspired by the architecture of ProGAN [7] and its concept of progressive training, with the aim of creating coherent minute-long audio samples from raw audio, while using the

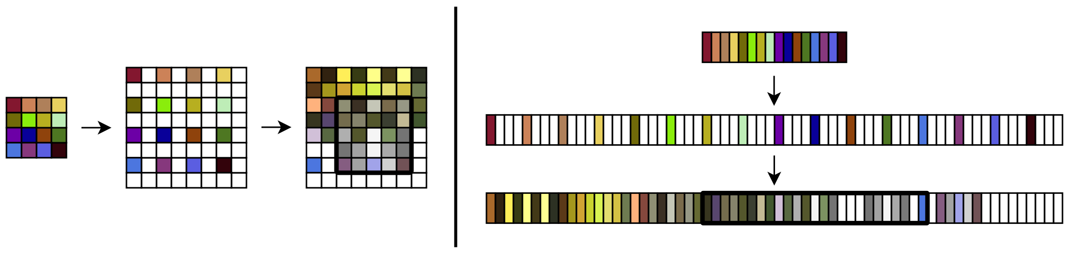

Figure 2: Transposed Convolution Operation of the first layers of the DCGAN [6].

kind of WGAN (a.k.a. Wasserstain GAN) training proposed by Arjovsky et al. [5].
In this model the data is represented through MDCT (Modified Discrete Cosine Transform) amplitudes, and given the fact that it is a real transformation (without the use of complex numbers) it does not have the need to recreate the phase information since all the phase information of the raw audio signal is encoded. Furthermore, in order to improve training stability and convergence, it takes into account also a psychoacoustic filter, a phenomena that takes place into the fact that the human ear is not able to distinguish between all the different possible raw audio signals, it is not capable of hearing under certain frequency thresholds and when there is a loud sound some quiet sounds are inaudible, creating a masking effect. These two elements are applied through some Gaussian noise added to the representation before it enters the discriminator.
Like the ProGAN [7] architecture, in the Mp3Net architecture each model block in the generator takes the activation tensor from the previous layer as input and computes an activation tensor with double the resolution as output. Similarly, the model blocks of the discriminator halve the resolution of the input activation tensor. To double the audio resolution of the MDCT amplitude representation two options are showed: doubling the sampling rate of the signal doubling the number of blocks or doubling the size of the blocks. The author decided to use both, doubling the number of frequency components and quadrupling the number of blocks.
The implementation of MP3net is available on the github repository created by Korneel van den Broek, but there are no pretrained models. Furthermore, the available implementation is not capable of handling short audio samples without modifications and needed modifications in order to deal with mono audio files instead of stereo.

4.3 GANSynth

GANSynth is a GAN model for generating audio samples in the spectral domain. The generator model takes a latent vector and pitch information as input and creates audio samples of the specified pitch. The discriminator uses convolutional layers and downsampling to distinguish fake samples from real ones. It also tries to classify each sample according to its pitch. After training, the latent vectors of the model learn an encoding for pitch information of the audio samples.
Pretrained GANSynth models are available and can be loaded using the magenta python library. We use two pretrained models, one trained on acoustic samples only, and one trained using samples of all instruments found in the NSynth dataset [3].

Our implementations of each model can be obtained on the project repository[1].

5 Experiments

5.1 WaveGan & SpecGAN

Due to time constraints and computation power limitations, we decided to train each model for about 30 000 epochs. WaveGAN was trained using batches of size 32 and SpecGAN with batches of 16. We observed that the SpecGAN model reached that 30000 mark faster than the WaveGAN model, which

[1]https://github.com/ernestvmo/API-project

needed around 40h to reach that mark. We trained all the network on a AMD Ryzen 5 3600 CPU and an NVIDIA GeForce GTX 1660.

5.2 MP3net

In order to work with the available implementation of MP3net, the audiocodec audio encoder and decoder is needed, available on the github repository created by the same author, including building and installation. Furthermore the README of the repository lacks instructions on how to use the network or on the requirements the network needs in order to work.
Before training the audio data files need a preprocessing step, yielded by the `dataprep.py` script, that encodes the `.wav` files in `tfrecords`, based on the `batch_size` defined by the user. Due to time constraints and computation power and memory limitations, a `batch_size` of 4 has been used.
Since the available model by itself is not capable of dealing with short audio samples, there were two options available: restructure the whole network in order to make it capable of dealing with short samples or aggregate the audio files in order to reach the minimum length needed (around 95 seconds) with less modifications needed. Given the time constraints and the aim of this work to experiment with the proposed architecture, the latter option has been chosen, plus the addition of extra songs in the dataset. For the 95s configuration of the model the authors of the original paper trained the network for 250h (750000 iterations on 1 Cloud TPUv2), but this kind of computational power was inaccessible, therefore the network has been trained on a Intel Iris Graphics 6100 integrated in an Apple 2015 MacBook Pro Retina for 120h(2400 iterations). It is clear how computational power is indeed a crucial factor when it comes to training of neural networks. After these 120h of training the successive step would have been to use the resulting weights to infer new samples, but the code had to be modified again in order to work and unfortunately due to the size and the complexity of the project, this was not achieved. Therefore we can state that this network highly depends on the dataset for which it has been built and adapting it to slightly different datasets requires a lot of work, furthermore the training of this network highly depends on the computational power available that can extremely influence the training' speed.

5.3 GANSynth

The pretrained GANSynth models have been used to generate 1024 audio samples for each. Pitch information is provided using a midi file provided as input. Each audio sample uses the same pitch input, but different latent vectors. These vectors are sampled at random. The generated audio samples and corresponding latent vectors are then evaluated using a CNN trained to predict the genre of music of audio samples. The latent vectors are stored with a score that expresses how much the samples sound like rock music.

6 Results

Audio is rather difficult to evaluate due to it's subjectivity. Donahue et al. used a special metric to evaluate the performance of their models on generating audio in the form of an *inception score*. We decided to only use human judgment to evaluate progress in the training. We compare the generated audios at 10k, 20k and 30k steps. We can clearly observe an improvement between each timestep. We were surprised to see how quickly the models were able to learn and generate audio resembling rock music. We also observed that SpecGAN, while faster at reaching the 30k steps was much more impacted with noise. This observation was also noted by Donahue et al. [1], and could be due to the need for SpecGAN to transform the generated spectogram by reversing the pre-processing step of transforming the audio into spectograms.

7 Conclusions

Carrying out this study, the architectures and potential applications of GANs for the synthesizing audio task and reproducing the rhythmic sound of rock music has been analyzed. Four different models have been investigated: WaveGAN, SpecGAN, MP3net and GANsynth. Given the comparative nature of this study, the main conclusive points can be presented as follows:

- We were unable to train the model on GPUs, which would have enabled a much faster training and thus a larger iteration count. Instead, we trained all of our models on CPUs. This lead us to only have about 30k epochs, by comparison to the work by Donahue et al who trained for 200k epochs [1].
- Unfortunately for us, the complexity and lack of documentations (insufficient explanations in the README) for MP3net model [8] made its use very difficult, slowing down the progress and ultimately not producing audio samples even after training.
- The result while performing the experiment on the GANSynth model was not entirely impressive as the output didn't really sound like a rock music. We concluded that the model is not well suited for making rock music as it mostly generates single notes.

Conclusively, during the course of running the experiments, It was clear how computational power is indeed a crucial factor when it comes to training of neural network as a few setbacks were encountered that hampered the training speed. However, we were impressed by the results yielded by WaveGAN (and SpecGAN), as their results were still impressive after only 30k epochs, even if SpecGAN was clouded with noise. We look forward to improve those models further with more testing and tuning in future research.

References

[1] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2018.

[2] Clauss, Christian Donahue, Chris, Marafioti, Andrés. Wavegan: Learn to synthesize raw audio with generative adversarial networks, 2020.

[3] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

[4] Simon L Madsen, Mads Dyrmann, Rasmus N Jørgensen, and Henrik Karstoft. Generating artificial images of plant seedlings using generative adversarial networks. *Biosystems Engineering*, 187:147–159, 2019.

[5] Léon Bottou Martin Arjovsky, Soumith Chintala. Wasserstein gan. 2017.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[7] Samuli Laine Jaakko Lehtinen Tero Karras, Timo Aila. Progressive growing of gans for improved quality, stability, and variation. 2018.

[8] Korneel van den Broek. Mp3net: coherent, minute-long music generation from raw audio with a simple convolutional gan. page 11, 2021.