Adversarial modelling of electronic music loops
ATIAM ML Project 2021

Antoine Caillon, Philippe Esling

Abstract

Recent advances in neural audio synthesis have made possible the creation of exciting new tools for composition and sound design. Early work on unconditional audio generation using autoregressive networks [1] has yielded impressive results, at the cost of expensive training and long synthesis time. Other approaches show the benefits of modelling audio using amplitude spectrograms [2] extracted from the raw waveform in terms of synthesis speed, however they rely on phase estimation algorithm that degrades the overall sound quality. The recent GANSynth [3] model address this problem by generating both the amplitude spectrograms and the corresponding instantaneous frequencies. They show that their approach is superior to previous ones in terms of audio quality, while synthesizing faster than real-time. In this project, you will adapt this approach to electronic music modelling, and will evaluate the effect of generating instantaneous frequencies instead of relying on phase estimation algorithms.

1 Introduction

Deep learning applied to audio signals proposes exciting new ways to perform speech generation, musical composition and sound design. Recent works in deep audio modelling have allowed novel types of interaction such as unconditional generation [1, 4, 5] or timbre transfer between instruments [6]. However, these approaches remain computationally intensive, as modeling audio raw waveforms requires dealing with extremely large temporal dimensionality.

To cope with this issue, the recent GANSynth model [3] uses mel scale spectrograms as an alternative to the raw waveform in order to perform class conditional neural audio synthesis. They show that modelling instantaneous frequencies (see figure 1) gives better performance than the previous phase estimation algorithm [7].

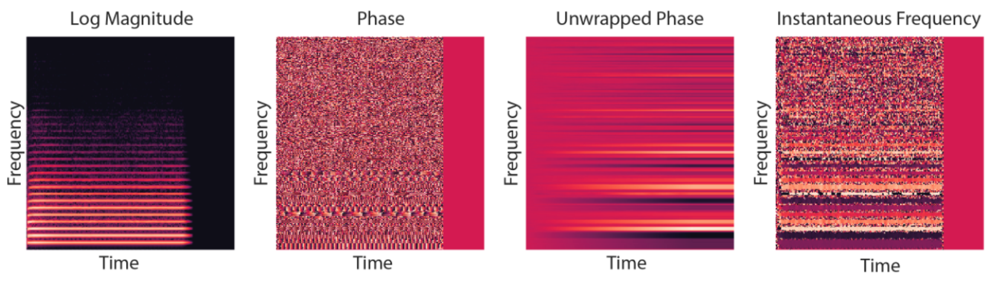

Figure 1: Amplitude and phase of a spectrogram.

During this project you will extend GANSynth to the unconditional neural audio generation task. More specifically, you will create an autonomous system for electronic music synthesis based on a dataset we provide. You will compare several adversarial training strategies, and demonstrate how your approach can be used for creative purposes.

2 Getting started

We will use the *pytorch* deep learning library for this project. Many tutorials are available on their website[1], and it is strongly advised to become familiar with its use before continuing the project.

2.1 Building a simple GAN

Generative Adversarial Networks [8] are infamously difficult to implement and train, thus we start with a simpler problem: adversarial digits generation. The MNIST dataset is composed of approximately 70,000 examples of hand written digits (see figure 2), and is often used as a playground to test new methods. You will start by implementing a simple convolutional GAN composed of a

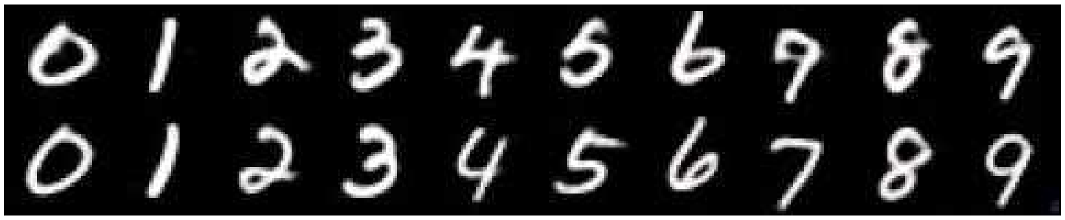
Figure 2: Examples from the MNIST dataset. Each digit is a 28×28 pixels grey scale image.

discriminator and a *generator*. Even if you are free to propose your own architectures, we suggest that you draw inspiration from DCGAN[2] since it has been extensively tested, and is quite easy to implement.

2.2 Benchmarking adversarial formulations

The discriminator is trained to classify true examples from generated ones, while the generator is trained to make the discriminator fail (thus the *adversarial* term). There are several ways to define this adversarial game from a computational point of view, from the very first GAN formulation [8] to more complex variations [9]. Each method has its benefits and disadvantages, and you will compare the following training strategies in term of stability and sample quality:

- Original GAN
- Least Square GAN
- Hinge GAN
- Wassertein GAN

We suggest that you write your code in a *modular* way so that you can reuse it for the rest of the project.

3 Electronic music modelling

Now that you have gained experience with the creation and training of GANs, you will address the electronic music generation task.

3.1 Spectral representation

As proposed in [3], we will leverage mel-scale spectrograms to build our generative model. The first step is therefore to build an invertible chain of transformations in order to compute both the log

[1] `https://pytorch.org/tutorials/`

[2] `https://bit.ly/3wuLAwS`

scale amplitude and instantaneous phase. We suggest that you build each transform as a separate block with *forward* and *inverse* methods (see figure 3).

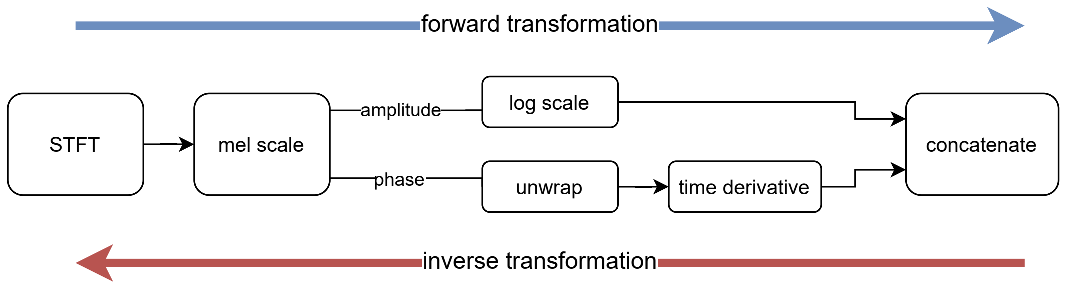

Figure 3: Data transformation graph. Each block has its forward and inverse method so that the overall analysis / synthesis process is simpler to implement.

3.2 Adapting GANSynth

Re-implement the architecture proposed in [3] without the class conditioning. We provide a large scale electronic music dataset in the form of a memory mapped numpy array[3].

We expect you to follow this roadmap

1. Model only amplitude spectrograms (synthesis with the griffin lim algorithm)
2. Add instantaneous frequency, compare with previous model
3. **BONUS:** add progressive growing procedure, compare with previous model

3.3 Demonstration

Randomly sample a *smooth* trajectory inside your high-dimensional latent space, and listen to your generative model gradually evolve. This is the perfect time to try and create something nice! Plot the corresponding evolution of several acoustical descriptors such as loudness and centroid to demonstrate the *smoothness* of your latent space.

References

[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. 9 2016.

[2] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton. Generative timbre spaces: regularizing variational auto-encoders with perceptual metrics. *DAFx 2018 - Proceedings: 21st International Conference on Digital Audio Effects*, pages 369–376, 5 2018.

[3] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial Neural Audio Synthesis. *arXiv*, 2 2019.

[4] Sean Vasquez and Mike Lewis. MelNet: A Generative Model for Audio in the Frequency Domain. *arXiv*, 6 2019.

[5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A Generative Model for Music. *arXiv*, 4 2020.

[6] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A Universal Music Translation Network. *arXiv*, 5 2018.

[3]You can listen to some samples randomly extracted from it here `https://nubo.ircam.fr/index.php/s/zJ2K76ko7ZTZsWz`

[7] Daniel W. Griffin and Jae S. Lim. Signal Estimation from Modified Short-Time Fourier Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *Communications of the ACM*, 63(11):139–144, 6 2014.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv*, 1 2017.