# Exploring Audio Synthesis using GANs

Aditya Vinod Bhangale
*Electrical Engineering*
*IIT Bombay*
Mumbai,India
210070004@iitb.ac.in

Vinay Sutar
*Electrical Engineering*
*IIT Bombay*
Mumbai,India
21d070078@iitb.ac.in

Vikas Kumar
*Electrical Enginnering*
*IIT Bombay*
Mumbai,India
210070093@iitb.ac.in

***Abstract*—While Generative Adversarial Networks (GANs) have seen widespread success in the problem of generating realistic images, they have received little attention in the field of audio generation, where autoregressive (AR) models such as WaveNets and Transformers dominate by predicting a single sample at a time. While this aspect of AR models contributes to their success, it also means that sampling is painfully serial and slow. Rather than generating audio sequentially, we approach to build a model that can generate an entire sequence in parallel. In this project we attempt 2 ways - PianoGAN and SpectGAN. PianoGAN is capable of generating a sequence of Piano notes and corresponding duration to play synchronized audio clips. SpectoGAN uses the melspectogram of audio clips and learns to generates unique spectograms from these which can be converted to intelligible audio. In both the models the entire audio clip is generated from a single latent vector, allowing for easier disentanglement of global features such as pitch.**

## I. Introduction

Generating audio for specialized domains holds significant practical value in the realm of creative sound design for both music and film. Musicians and Foley artists often sift through extensive sound effect databases in search of specific audio recordings suitable for particular scenarios. This process can be laborious, and the outcome may be unsatisfactory if the perfect sound effect is not available in the existing library.

A more effective approach involves enabling sound artists to navigate a condensed latent space of audio. This method allows for broad exploration to identify general sound categories (e.g., footsteps) and then make nuanced adjustments to latent variables for fine-tuning (e.g., a large boot landing on a gravel path). However, mastering such an approach is challenging due to the high temporal resolution of audio signals, demanding strategies that can effectively operate in these complex, high-dimensional spaces.

This turns out to be the motivation to using Generative Adversarial Networks in this project. It turns out to be an approach for mapping low-dimensional latent-vectors to high-dimensional data (images/audio/videos etc.) The potential advantages of using GAN's are numerous. Firstly, it can be used in Supervised as well as unsupervised setting and GAN's have shown the capability to synthesize audio in both the scenarios.

A simple solution to employ image-generating Generative Adversarial Networks (GANs) for audio involves applying them to image-like spectrograms—time-frequency representations of audio. While this practice is common in discriminative settings, using it generatively presents challenges. The most perceptually-informed spectrograms are non-invertible, making them unsuitable for listening without lossy estimations or learned inversion models.

Recent advancements, demonstrate that neural networks can be trained with autoregression to directly operate on raw audio. This approach is appealing as it eliminates the need for engineered feature representations.
In this forms we investigate both spectogram (image-based) and Waveform (audio-based) stratgies for audio synthesis. In our approach of PianoGAN, we used MIDI-Musical Instrument Digital Interface-data. This format of data makes it easier for generation of notes for instruments like piano. Whereas in SpectoGAN we used the SC09 dataset contains humans audio.

## II. GAN Preliminaries

The fundamental dynamics involve two key components: the generator and the discriminator. The generator is tasked with producing audio—from a latent space. It essentially crafts synthetic musical sequences with the aim of mimicking real, artistically valid compositions. As training progresses, the generator refines its ability to generate sequences that become progressively indistinguishable from those originating in the actual dataset. Conversely, the discriminator functions as a critical evaluator. Its primary role is to discern between authentic sequences from the training data and those crafted by the generator. In a sense, it acts as the "critic" in this two-player Min-Max game.
Generative Adversarial Networks (GANs) learn mappings from low-dimensional latent vectors $z \in Z$, i.i.d. samples from a known prior $P_Z$, to points in the space of natural data $X$. In their original formulation (Goodfellow et al., 2014), a generator $G : Z \to X$ is pitted against a discriminator $D : X \to [0, 1]$ in a two-player minimax game. $G$ is trained to minimize the following value function, while $D$ is trained to maximize it:

$$V(D, G) = \mathbb{E}_{x \sim P_X}[\log D(x)] + \mathbb{E}_{z \sim P_Z}[\log(1 - D(G(z)))].$$

In other words, $D$ is trained to determine if an example is real or fake, and $G$ is trained to fool the discriminator into thinking its output is real. It is demonstrated that their proposed training algorithm for Equation equates to minimizing the Jensen-Shannon divergence between $P_X$, the data distribution, and $P_G$, the implicit distribution of the generator when $z \sim P_Z$.

In the original formulation, GANs are notoriously difficult to train and prone to catastrophic failure cases. Instead of Jensen-Shannon divergence, we use the smoother Wasserstein-1 distance between generated and data distributions:

$$W(P_X, P_G) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_X}[f(x)] - \mathbb{E}_{x \sim P_G}[f(x)],$$

where $\|f\|_L \leq 1 : X \to \mathbb{R}$ is the family of functions that are 1-Lipschitz.

## III. Models

As an attempt to adapt image-generating GANs to audio, we proposed the use of two different approaches to generating fixed-length audio segments based on the DCGAN architecture: PianoGAN and SpectoGAN.

Inspired by the use of convolutional neural networks on spectrograms for audio classification, SpectoGAN operates on imagelike 2D magnitude melspectrograms of audio. However, the process of converting audio into such spectrograms lacks an inverse, as phase information is discarded. Hence, the audio samples generated from SpectoGAN are derived from approximate inversion methods that incur significant distortion. To circumvent the need for such inversion methods, PianoGAN instead operates on raw audio waveforms. This model converts the 2D convolutions over images present in DCGAN to 1D convolutions over time-domain audio signals.

## IV. PianoGAN Model

We primarily envisage our method being applied to the generation of short sound effects suitable for use in music and film.

### A. Dataset and DataProcessing

The dataset is MasteroPianoMidi obtained form Kaggle is used from this project. MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) is a dataset composed of about 200 hours of virtuosic piano performances captured with fine alignment (3 ms) between note labels and audio waveforms.

We made use of pretty-midi library to create and parse MIDI files. The pretty-midi library can accurately extract notes from the piano audio clips. Extract clips with is directly related to the pitch, which can be percieved by Humna ears, is a good strategy as it is easier to consider pitch rather than the conplex time and frequency variation which are more prone to be noisy. Each pitch has a unique Note corresponding to it. We use three variables to represent a note when training the model: ‘pitch‘, ‘step‘ and ‘duration‘. The pitch is the perceptual quality of the sound as a MIDI note number. The ‘step‘ is the time elapsed from the previous note or start of the track. The ‘duration‘ is how long the note will be playing in seconds and is the difference between the note end and note start times. It is also demonstrated that the notes can be combined back to obtained a audio. It forms our motivation to train the GAN to generate pitch, stepsize and duration to obtain short sound effects.

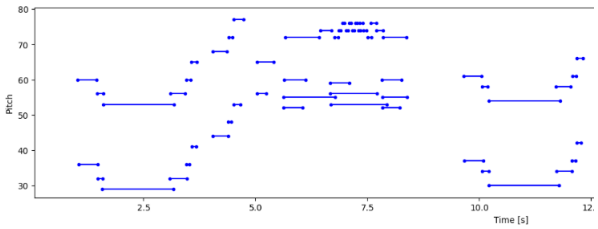


Fig. 1. Demonstration of notes of a piano in a audio clip.

### B. Model Architecture and Training

In a waveform sampled at 16 kHz, a 440 Hz sinusoid (the musical note A4) takes over 36 samples to complete a single cycle. This suggests that filters with larger receptive fields are needed to process raw audio.

*1) Generator and Discriminator:* In the training process, the model operates on batches of sequences, each comprising 256 notes. We base our WaveGAN architecture off of DCGAN which popularized usage of GANs for image synthesis. The DCGAN generator uses the transposed convolution operation to iteratively upsample low-resolution feature maps into a high-resolution image. Motivated by our above discussion, we modify this transposed convolution operation to widen its receptive field. Specifically, we use longer one-dimensional filters of length 25 instead of two-dimensional filters of size 5x5, and we upsample by a factor of 4 instead of 2 at each layer (Figure 2). We modify the discriminator in a similar way, using length-25 filters in one dimension and increasing stride from 2 to 4.

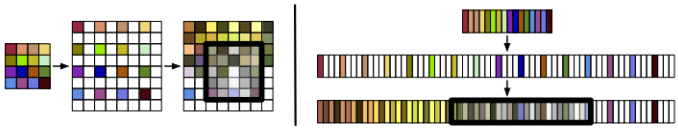

Fig. 2. Depiction of the transposed convolution operation for the first layers of the SpectoGAN (left) and PianoGAN (right) generators. DCGAN uses small (5x5), two-dimensional filters while PianoGAN uses, one-dimensional filters and a larger upsampling factor. Both strategies have the same number of parameters and numerical operations.

*2) Upsampling Procedure and Batch Normalization:* Transposed convolution upsamples signals by inserting zeros in between samples.Transpose convolutional layers with upsampling help generate detailed and fine-grained features by increasing the spatial resolution of the data. This is essential for generating high-quality images or sequences. The generator aims to transform a low-dimensional input (often random noise or latent space) into a high-dimensional output that resembles the target data distribution.

*3) Shuffling:* For audio, analogous artifacts are perceived as pitched noise which may overlap with frequencies commonplace in the real data, making the discriminator's objective more challenging. However, the artifact frequencies will always occur at a particular phase, allowing the discriminator to learn a trivial policy to reject generated examples. This

may inhibit the overall optimization problem. To prevent the discriminator from learning such a solution, we shuffle between the samples in the buffer training set.

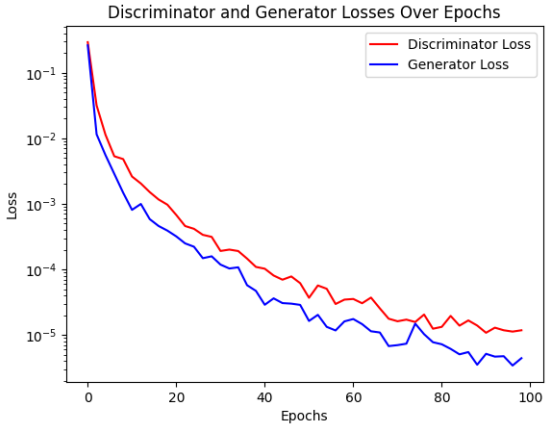


Fig. 3. Generator and Discriminator Loss Variation with number of epochs

### C. Results

The PianoGan model generates a sequence of 256 notes as output. This notes are converted to a audio clip using teh pretty-midi library in python.

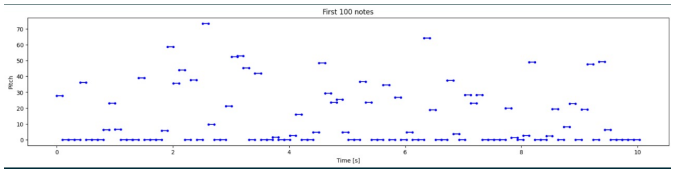


Fig. 4. Generated Piano notes

## V. SpectoGAN

This approach is based on the melspectogram representation of audio. We used the Librosa, library for audio and music processing in Python.

### A. DataSet and DataProcessing

The dataset used here is the SC09 Spoken Digits, consisting of spoken audio clips of digits from 0-9. To process audio into suitable spectrograms, we first perform the short-time Fourier transform with 16ms windows and 8ms stride, resulting in 129 frequency bins linearly spaced from 0 to 8 kHz. We take the magnitude of the resultant spectra and scale amplitude values logarithmically to better-align with human perception. We then normalize each frequency bin to have zero mean and unit variance, and discard the highest frequency bin. These steps were done using the Librosa library in python.

### B. Model Architecture and Training

The model architecture for both the model is similar to that used in PianoGAN except that the transposed convolution operation for the first layers of the SpectoGAN uses small (5x5) two-dimensional filters while PianoGAN uses, one-dimensional filters and a larger upsampling factor. Both strategies have the same number of parameters and numerical operations.

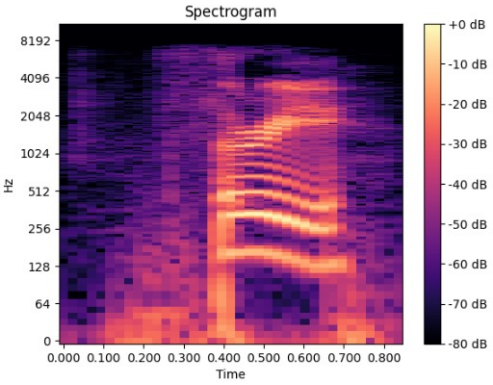


Fig. 5. Melspectogram of a sample audio file

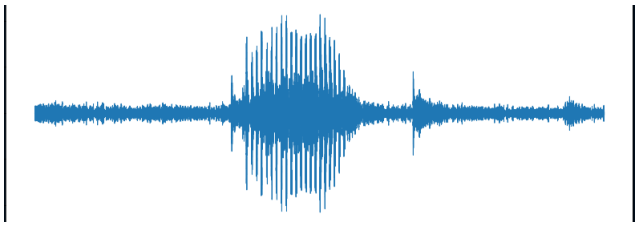

Fig. 6. Audio Waveform Representation

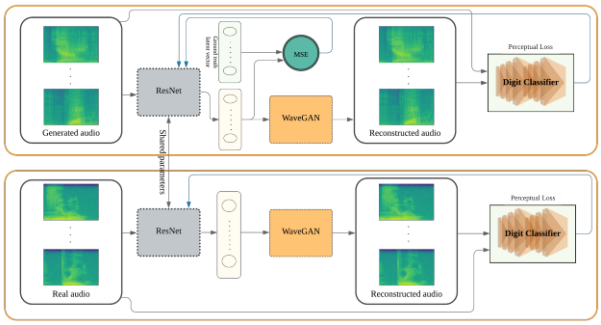


Fig. 7. GAN architecture for SpectoGAN

### C. Results

The trained generator synthesizes unique spectograms, from the labelled training samples. One problem encountered at the end during resonstruction phase (converting the generated spectograms to audio clips) was that melspectogram representations in the discriminative setting aggregate frequencies into logarithmically-spaced bins and discard phase information, rendering them uninvertible.

Very little was found in the literature on how to reconstruct real audio using GAN inversion techniques. A major problem with mapping real audio is that there are no ground truth latent vectors that accurately generate them. Another issue is the limitation of the audio GANs themselves, they are not capable of generating high quality naturalist audio that sound

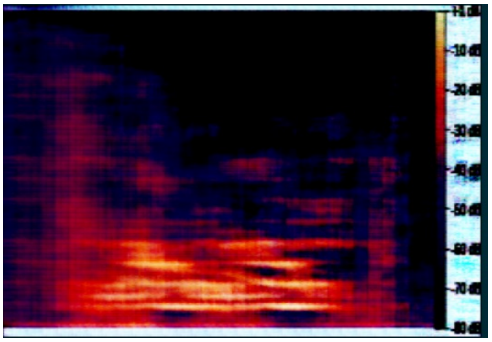

Fig. 8. Generated Spectogram obtained after training on the Spectogram images data

identical to the real peers. Nevertheless, our proposed approach is universal and can later be applied to invert more advanced GANs.

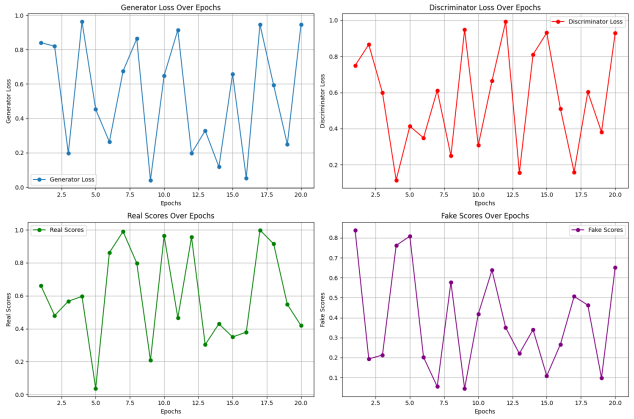


Fig. 9. Variation of various parameters with teh number of epochs.

## VI. Challenges and Future Endeavours

The early approach we were working on involved converting the spoken audio clip (from dataset SC09) into a melspectogram, extracting frequency, time interval and amplitude levels in dB at that particular frequency and time. This 3-dimensional matrix consisting of frequency, time and amplitude will form the input to the GAN architecture.

However, rather than discarding phase information and using a spectrogram representation, we instead transform the audio signal in an invertible manner.

This involves using the short-time DFT or DCT to create a time-frequency representation with phase information. A fixed transformation is applied to all input audio data, and its inverse is applied to the generator output to produce audio signals. GAN training proceeds entirely in the transformed time-frequency domain, using the Wasserstein loss with gradient penalty.

One possible time-frequency representation can be generated with a short-time Fourier transform. An alternative to the discrete Fourier transform is the discrete cosine transform (DCT). We use Hann windows of length 256 with a stride of 128 samples, which allows for constant overlap-add during resynthesis. To ensure the original signal is completely reconstructable, we zero-pad the boundaries. This yields 256 complex values for each of the 129 windows. Since the signal is real, the upper 127 frequencies can be discarded due to Hermitian symmetry, leaving us with 129 frequencies. Treating the time and frequency dimensions as a 2D spatial grid, we have a single-channel complex-valued image of shape (129, 129, 1). Separating the real and imaginary components into individual channels yields an equivalent real image of shape (129, 129, 2). Additionally the amplitude levels in logarithmic scale can also be used to control loudness. This is a rough idea we were trying to implement but instead shifted to other models due to high complexity, and challenges like mode collapse and training instability.

Further work on this has beeb done in the paper Synthesizing Audio using Generative Adverserial Networks by Bowen Zhi, Fan Yang, Nadee Seneviratne and Patrick Owen.

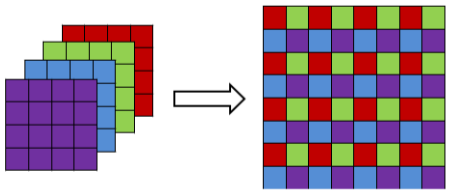

Fig. 10. Visualization of the sub-pixel shuffling operation combining 4 channels into 1 upscaled channel.

Another challenge we faced is in training GANs for audio synthesis, which are stability issues, including mode collapse. The potential solutions we explored for these include experimenting with different GAN architectures, considering conditional GANs, and leveraging pre-trained models or transfer learning.

Designing appropriate loss functions for the GAN, considering the multi-modal nature of audio data with pitch, step size, and duration.

## References


[1] Chris Donahue, Julian McAuley, Miller Puckette, ADVERSARIAL AUDIO SYNTHESIS, 2019.
[2] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts, GANSYNTH: ADVERSARIAL NEURAL AUDIO SYNTHESIS, 2019.
[3] Bowen Zhi, Fan Yang, Nadee Seneviratne, Patrick Owen, SYNTHESIZING AUDIO USING GENERATIVE ADVERSARIAL NETWORKS.
[4] Andrew Keyes, Nicky Bayat, Vahid Reza, Khazaie Yalda Mohsenzadeh, Latent Vector Recovery of Audio GANs, 16 Oct 2020.
[5] https://medium.com/neuronio/audio-generation-with-gans-428bc2de5a89
[6] https://www.kaggle.com/code/fizzbuzz/beginner-s-guide-to-audio-data/notebook
[7] https://github.com/chrisdonahue/wavegan/blob/master/train[underscore]wavegan.py
[8] https://github.com/fyang623/Audio-Synthesis-with-GANs
[9] https://github.com/mostafaelaraby/wavegan-pytorch/tree/master
[10] https://diffwave-demo.github.io/
[11] https://github.com/ilaria-manco/word2wave
[12] https://github.com/ibab/tensorflow-wavenet


Link to Colab notebook: SpectoGAN PianoGAN