NoisyGAN: Using Adversarial Networks to Generate Music as Raw Audio

Tripp Gordon and Tom Hanlon

Introduction

Generative art can serve as a sort of litmus test for the current state of AI. It allows algorithms to express their learned abstract representations of data in a way that humans can understand very intuitively, if difficult to precisely quantify. Algorithms like CycleGAN, when applied to the image style transfer problem, demonstrate that modern deep learning strategies can be applied to domains that previously defied quantitative definition, like the process of emulating a particular artist's style[1].

While a great and growing body of research exists in the domain of generating image data, progress has been more sparse in the audio domain, making it a very interesting and compelling area of research. Research in this area includes WaveNet[2], a neural autoregressive generative model that manipulates probability distributions to generate speech, SampleRNN[3] which uses dilated recurrent layers to generate new audio to generate speech, and Databots[4], a band that uses an implementation SampleRNN to generate novel samples of music in genres like metal and math rock.

A recent paper from Google's Magenta team, "DDSP: Differentiable Digital Signal Processing", introduces Differentiable Digital Signal Processing, a library of tools that allow users to perform DSP operations on tensors within an end-to-end trainable deep neural network. The researchers demonstrated that these tools can be used to perform audio style transfer by use of an autoencoder network[5].

Our hypothesis is that traditional GAN approaches can be applied to the domain of audio generation, given that the right priors have been encoded into the network. We demonstrate that DDSP components, along with Complex Pattern-Producing Networks (CPPNs) are good candidates for these priors. We achieve results that are impressive by today's standards both in quality and computational efficiency.

Related Work

Other work has been done on generating audio with deep learning. WaveNet, from some of the same researchers behind PixelCNN, uses a similar methodology of manipulating probability distributions to generate new audio. While the initial goal of the project was to create voice samples, they found very interesting success when training their algorithms on classical music[2]. However, our approach using a GAN is substantially different than theirs.

SampleRNN is another algorithm that was designed to produce audio of voices[3]. As its name implies, it is a recurrent neural network. However, similar to dilating convolutions, SampleRNN uses multiple recurrent elements, with some seeing activations from more than one iteration in the past. This sort of temporal dilation allows the network to capture sonic information at various scales. While our method implements some recurrent elements, we do not currently plan to dilate them. Our generative model is distinct in that, instead of continually asking an RNN "what comes next?" to generate arbitrarily long samples of audio, we will attempt to create a generator that creates convincing audio samples. While we will lose out on the ability to create indefinitely long samples, we gain the ability to represent our network's abstractions through a latent space.

Problem Definition and Algorithm

Problem Definition

In this paper we propose a deep learning architecture that takes a latent vector as an input, and outputs a sample of raw audio that emulates our training dataset. Our primary goals are for this generated audio to emulate the sonic texture, or timbre, of the training data, and that it also emulates some melodic and rhythmic qualities too.

Algorithm Definition

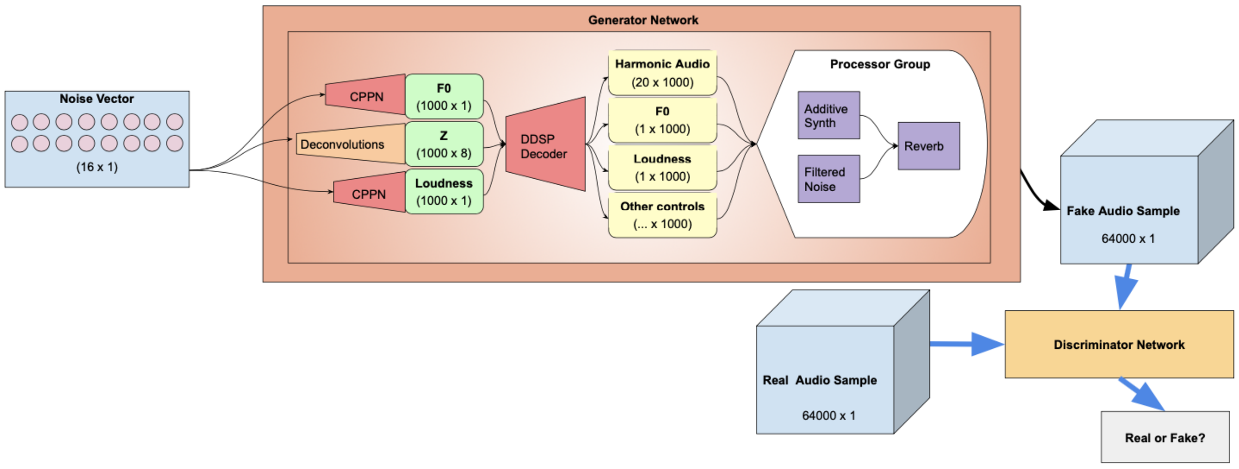
An integral part of our algorithm is an implementation of parts of the DDSP library. This library implements DSP tools like synth modules, effect modules, and wave tables, and allows them to be seamlessly integrated into a differentiable model. A DDSP synth takes three main inputs: A fundamental frequency vector, a loudness vector, and a harmonic distribution vector. Using these three inputs, which can either be static or learned as parameters, the module outputs synthesized raw audio. The GAN that we are designing includes a generator that uses DDSP synthesizers to generate audio, and a discriminator that utilizes the library's encoders to prepare features for classification.

Our generator upscales an initial latent vector into three vectors that are used as inputs to a group of DDSP synths. These synths also have other internal controls which are trainable. The output of these synths is generated raw audio that is sent to a discriminator network. The discriminator takes a sample of raw audio, and uses DDSP preprocessors to encode the audio into mel-frequency cepstral coefficients. It then runs the encoded data through a three-layer convolutional network before making a binary classification.

When generating fundamental frequency (F0) and loudness (pictured below), we used Complex Pattern-Producing Networks (CPPNs)[7]. These essentially model a function that takes in a coordinate (in this case time) and outputs a value for that coordinate. For example, if your coordinates were x-coordinates of a black and white picture, and you had no hidden layers with a single node and a linear activation, you would essentially achieve a gradient from dark to light along the x-axis. When adding activations and hidden layers, you can begin to achieve complex patterns and combinations of gradients. These outputs can be tuned using backpropagation to produce a huge number of patterns, all of which are biased toward regular patterns.

The reason we decided to try this in our case can be illustrated nicely if we think about real musical instruments being played in real life. When striking a note on a guitar, there is a continuous fade in volume that can be thought of as a gradient through time. Similarly, if one considers vibrato being played on a violin, there are elements of sinusoidal patterns. We felt that this was a better way of encoding F0 and loudness when compared to 1D deconvolutions, as it encodes an important prior into the architecture of the network. This was verified with improved results.

Our general architecture is pictured below. The discriminator can be thought of as roughly the inverse of the generator.



Experimental Evaluation

Methodology

Automating the evaluation of GAN's can be very difficult, especially in the domain of emulating an artistic style. Our evaluation relies heavily on manual evaluation of audio generated from periodic checkpoints of our generator algorithm and by monitoring the individual networks' reported losses.

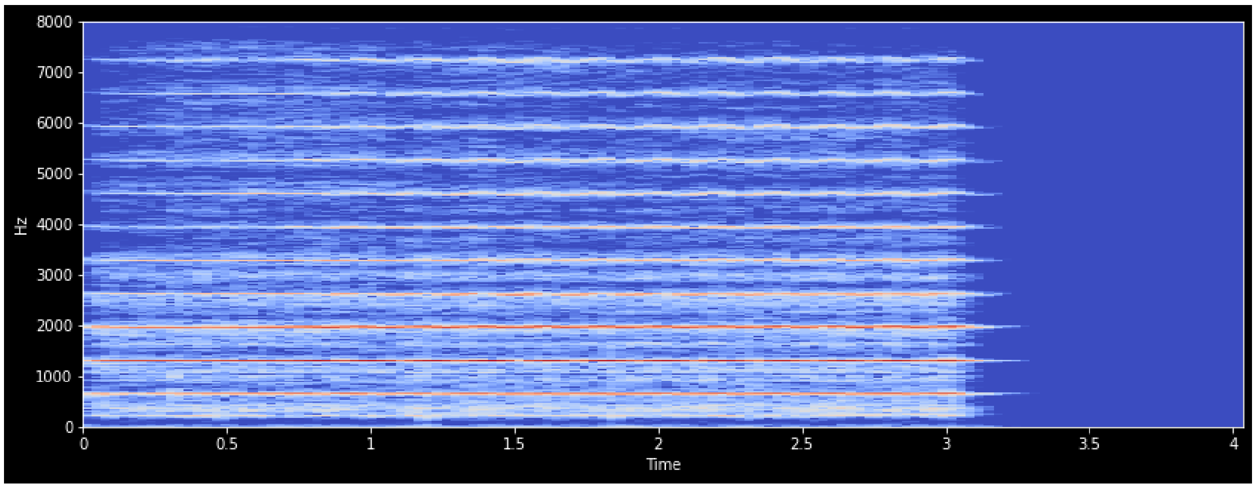
Results

The interesting thing about GANs is if you had a better mathematical way of determining how "realistic" something was, you would use it as your loss function. Measuring the success of this system is highly subjective, which is why we have included the following links to listen to our progress:

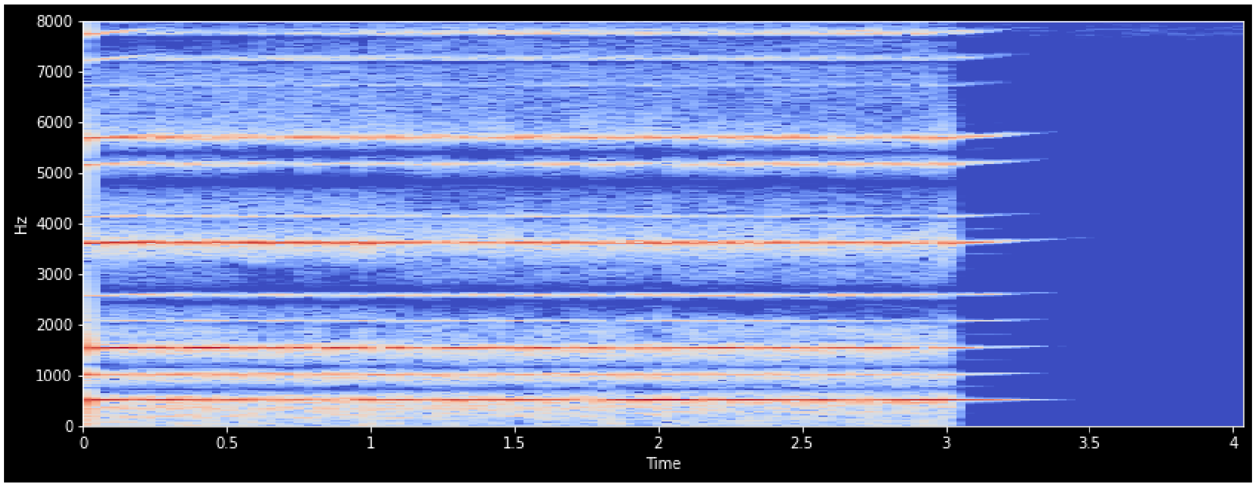
Listen real
Listen random initialization (no training)
Listen fake

Additionally, we have included the following spectrograms and wave plots of our generated sample next to a real sample of a flute from our dataset.

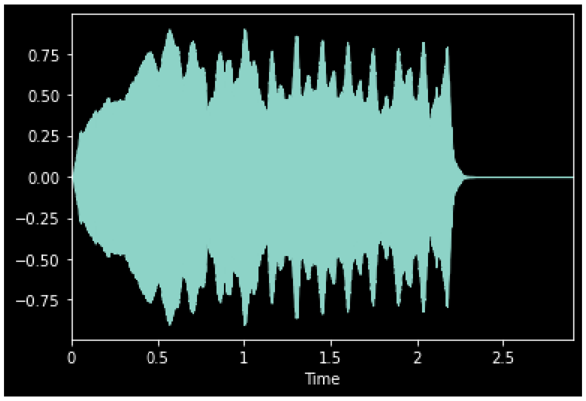
Real



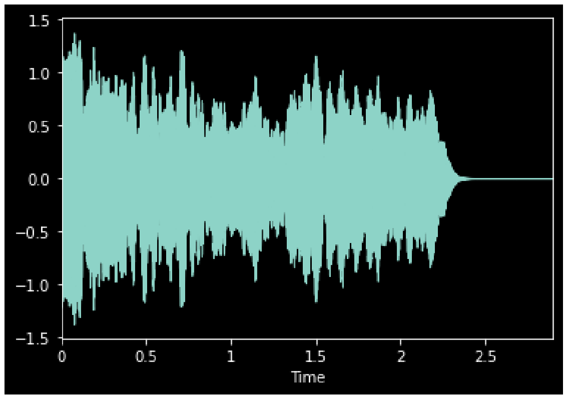
Fake



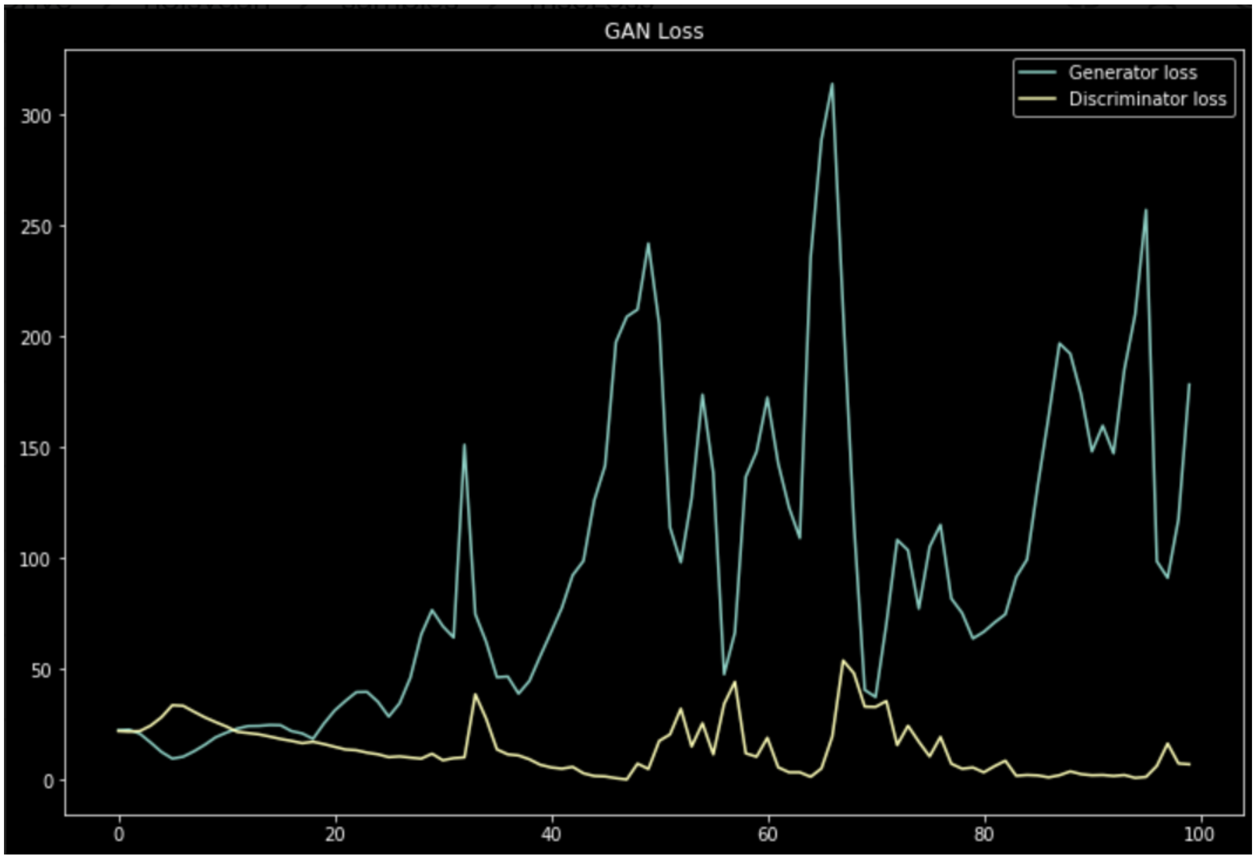
Real



Fake



For sake of completeness, here is a graph of the loss functions for the generator and discriminator. It should be noted that they are involved in a zero-sum game, so these loss values are relative and arguably arbitrary.



Discussion

When considering computational efficiency (only having to generate & classify F0, loudness, and texture as opposed to a full 16k sample rate audio clip) as well as qualitative results, these results are substantial.

Code and Dataset

Code is in a private repo. Please reach out if you'd like it shared.

Our dataset is Google's NSynth dataset. It consists of a large (300k+ samples, 70+ GB) collection of labelled single-instrument & single-note samples. Each sample is 4 seconds long at a sample rate of 16k samples per second.

Conclusion

We conclude that traditional GAN principles that have been tried and true for image generation can in fact transfer to the audio domain. However, model architectures in and of themselves encode priors into a given problem. The classic image architectures (i.e. very large CNNs) are inherently encoding priors about that domain, whether conceptually or through trial and error. Similarly, we have taken priors about audio generation such as synths, effects, and other DSP components, and embedded them into our audio GAN architecture. This has yielded impressive results by today's standards, and has allowed a huge amount of additional computational efficiency when compared to previous attempts.

Bibliography

- [1] Zhu, Jun-Yan, et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, doi:10.1109/iccv.2017.244.
- [2] Van den Oord, Aaron, et al. "WaveNet: A Generative Model for Raw Audio." 8 Sept. 2016.
- [3] Mehri, Soroush, et al. "SampleRNN: An Unconditional End-To-End Neural Audio Generation Model." *ICLR 2017*, 2017.
- [4] Carr, CJ, and Zach Zukowski. "Generating Albums with SampleRNN to Imitate Metal, Rock, and Punk Bands." *Proceedings of the 6th International Workshop on Musical Metacreation (MUME 2018)*, 2018.
- [5] Engel, Jesse, et al. "DDSP: Differentiable Digital Signal Processing." *International Conference on Learning Representations*, 2020.
    - Used code available at: https://github.com/magenta/ddsp
- [6] Brownlee, Jason. "How to Evaluate Generative Adversarial Networks." *Machine Learning Mastery*, 12 July 2019, machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/.
- [7] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life, 15 (2): 185--212, 2009. http://axon.cs.byu.edu/~dan/778/papers/NeuroEvolution/stanley3**.pdf .
- [8]Brownlee, Jason. "How to Implement GAN Hacks in Keras to Train Stable Models." *Machine Learning Mastery*, 12 July 2019, machinelearningmastery.com/how-to-code-generative-adversarial-network-hacks/.
-