Audio Texture Synthesis Using Convolutional Neural Networks in PyTorch

Jacob Lamadrid
Cognitive Science, Machine Learning
A16282531

Abstract

The overall goal of this project is to generate audio textures using conventional techniques found in image texture synthesis, image in-painting and style transfer. This idea of texture synthesis in sound is made possible through the use of Spectrogram visualizations in which it is possible to implement a Convolutional Neural Network across points. This method has been proposed by researchers at Sorbonne Université and has been explored by others through the use of RI Spectrograms and Neural Style Transfer to name a couple of reimplementations. This project will only focus on the basic texture synthesis and aims to use all native Pytorch libraries.

1 Introduction

The main motivation for this problem is in the extension of audio beyond what is recorded initially. Through the use of audio texture synthesis it is possible to attach audibly realistic sound to the end of a prerecorded object and "extend" the recording. However, this problem has utility beyond audio extension such as various artistic applications, new audio generation, etc.

To approach texture synthesis, a massively popular approach is through the use of Convolutional Neural Networks which was originally proposed by researchers at University of Tubingen, Germany in which they proposed that generation is possible through the correlation, in Gram Matrices, between a target image and white noise image across convolutional layers. This can then provide gradients which will alter the white noise image to generate a new photo realistic texture while remaining different from the original.

The ability to execute these convolutions and apply gradients is made possible by our ability to convert audio to a visual form and retrieve information from this form, all of which being a basic Spectrogram image displaying frequency (Hz) vs time (S). Much effort has been made in this realm of audio visualization within Python, namely in TorchAudio, which provides great tools for these conversions and provide minimal loss in audio quality relative to other packages experimented with.

Figure 1 displays the audio processing tools provided by TorchAudio and displays ways in which one is able to take audio out of the time domain and into a visual frequency vs time domain. After experimentation, this project remains within upper left quadrant of the visual, utilizing the Raw Spectrogram with a Complex Spectrum. Despite notable drawbacks in frequency granularity, this method provides the most accurate audio to the generated Spectrogram image compared to the other methods shown in figure 1. In future iterations of this project I would like to explore new audio visualizations tools which can provide more insight into the target audio to eventually generalize to all sounds which may be input, including meter, rhythm, etc.

Overall, this use of convolutions over an input image is the approach which I have taken within this project with notable differences in network architecture and loss calculations. This resulted in audibly realistic outputs in terms of "natural" audio textures such as birdsong, cricket ambience, etc, but fails to generate audio with less granularity across frequencies.

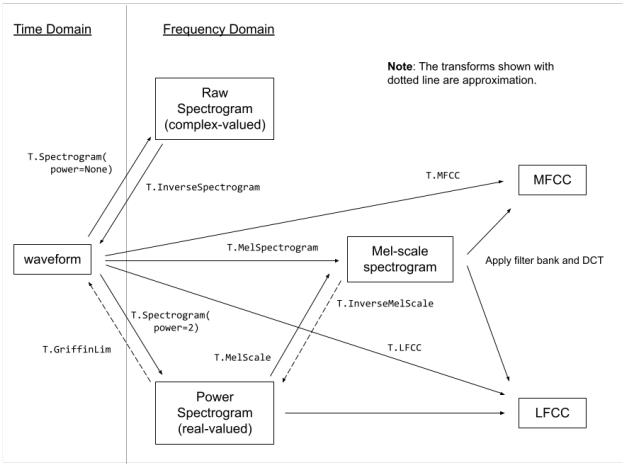

Figure 1: TorchAudio Spectrogram Tools

2 Related Work

As mentioned previously, works in image texture synthesis were the predecessors of this audio texture synthesis problem. The most notable work in this field is "Texture Synthesis Using Convolutional Neural Networks"[1]. This work displayed possibilities in image generation using Neural Networks by calculating loss across convolutional layers, effectively creating an image which displays characteristics of the target image without copying the original due to this compression factor across convolutions. This method is the most popular method within works of Neural Style Transfer and other synthesis tasks.

However, the main motivation for this project comes from "Sound texture synthesis using convolutional neural networks"[2], in which the original image texture synthesis method is applied to audio through manipulations of this method. The most significant difference comes in how correlations across convolutions are calculated as a new method is required in order to preserve relics across frequencies. Without this method the generated Spectrogram image would be overtaken by the features of the most prevalent frequencies. The proposed function is as follows (formula from [3]):

$$H^l_{ijm} = \sum_n F^l_{imn} F^l_{jmn}$$

Figure 2: 3-Dimensional Gram Matrix Calculation

This method takes the original Gram Matrix correlation which remains in a 2D space and adapts it to a 3D space in which frequency features are maintained and accounted for in computing the loss. This change in correlation computation is what made audio texture synthesis from Spectrogram images possible, but is very limited by these Spectrogram representations and as seen in "Sound texture synthesis using RI spectrograms"[3], efforts have been made to expand on this.

3 Method

As seen in figure 2, the cross correlation matrix is computed across convolutions. This correlation will then be compared through the formula as in figure 3 (formula from [2])

The loss is summed over all convolutions (k), in which the gradient will be computed. In this project, the Adam optimizer is used as this is the most common and stable optimizer in texture synthesis and style transfer techniques. The optimization phase is terminated at the point in which across 100 epochs, a difference in loss of less than 0.0005 occurs.

$$\ell = \sum_k \frac{||\tilde{H}^k - H^k||_2}{||\tilde{H}^k||_2}$$

Figure 3: Loss Function

The architecture of the network also follows standard architecture as it is a single layer Neural Network comprised of 8 CNNs of varying filter sizes [3, 5, 7, 11, 15, 19, 23, 27]. Due to computational limitations, the network utilized in this project is limited to only 32 filters per CNN, $\frac{1}{4}$ of the standard practice. This overall reduces the audio quality of the outputted texture, but still provides decent results. This computational limit also reduced my capabilities in experimenting beyond the standard practice. However, additional CNNs would likely prove to be counter productive as granularity must be determined through smaller filter sizes and adding filters beyond the size of 27 would ultimately take too large of an area in the Spectrogram into account, leaving incoherent sound instead of recognizable patterns. The most significant improvement which could be made in this method would be the use of visualizations and potentially combining multiple visualizations to create a very holistic image of the sound and improve granularity.

Overall, this project takes the standard approach as seen in other audio texture synthesis works in both its computation and network architecture. However, this implementation is expanded to the ability of processing 2 channels due to TorchAudio Spectrogram translation.

4 Experiments

All input files are WAV files of either one or two channels. Experiments include two "natural" audio recordings set in open space (one from original paper), both with bird recordings included within to get a reasonable assessment of performance. Also include were two instrumental recordings, one guitar and one synth.

The natural audio recordings performed very well in terms of realism and their ability to essentially "randomize" the audio with sounds being put at different points yet maintaining their same patterns, such as in the bird recordings. The granularity of natural recordings lends itself to this success.

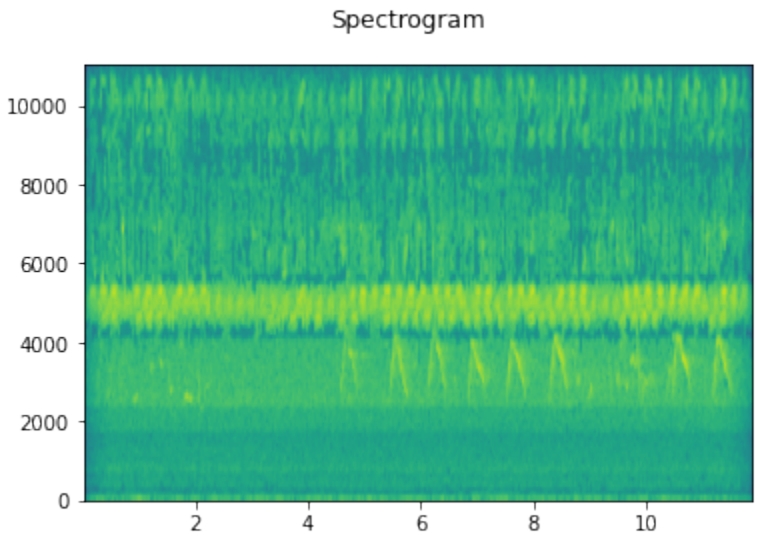

Figure 4: Crickets Original Recording

The instrumental recordings performed very poorly in comparison to the natural audio recordings as mentioned above. The Spectrogram image of the guitar recording for example is nearly unrecognizable upon viewing relative to the natural recordings. This leads to the failure of audio texture

synthesis using CNNs as a CNN necessitates an image upon which convolutions are able to be performed. So even though we see a lower loss in the musical recordings, the output is significantly poorer than other, more granular recordings.

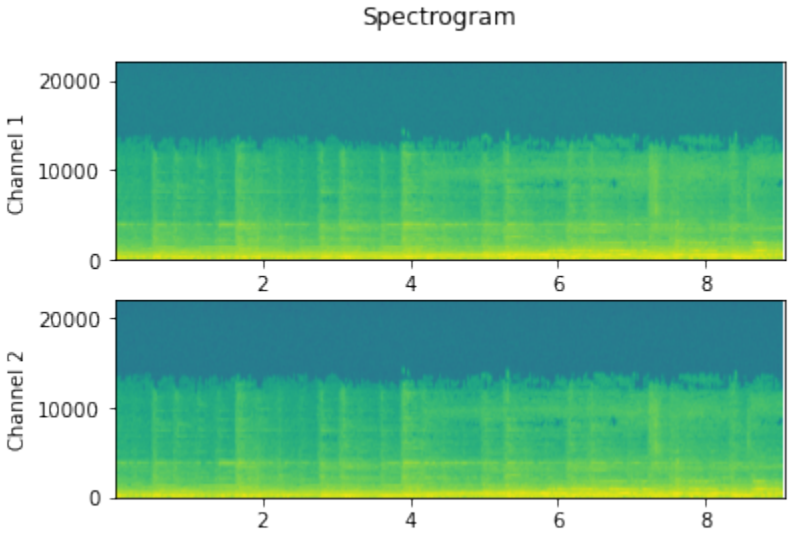

Figure 5: Grouper Original Recording

The effect of granularity is the most significant factor in this project as not only does the models ability hinge on this, but the performance has a direct correlation with increased granularity. In the following figure we see a somewhat granular recording of a bird call generated in which the relatively defined sections in which the bird is audible, the generated image is able to reproduce very successfully, but the surrounding area leaves the model unsure of what to produce.

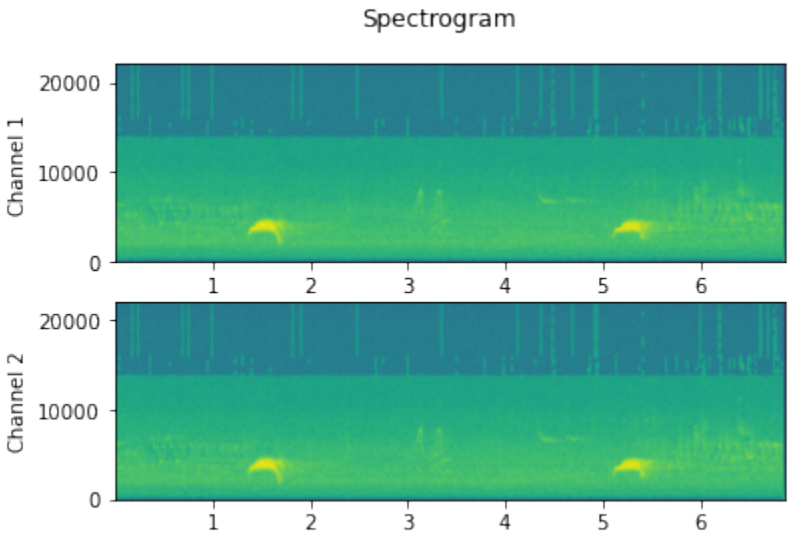

Figure 6: Original Bird Recording

In figure 6 we see noticeable patterns of the bird recording with the chirps representing the small arcs between 1 and 2 seconds as well as between 5 and 6 seconds. This granularity in recording is what makes it possible to detect certain textures and in the following figures you can see how this then gets interpreted by the Convolutional Neural Network and how it is outputted in the generated audio.

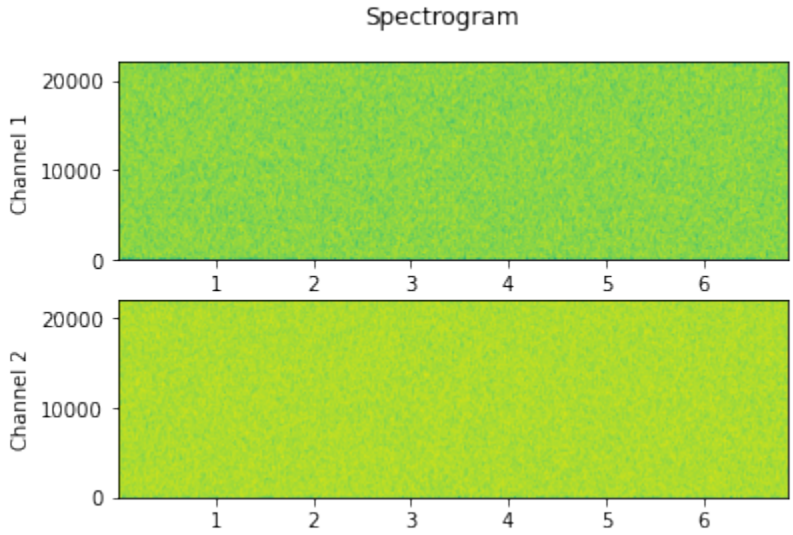

Figure 7: Input White Noise

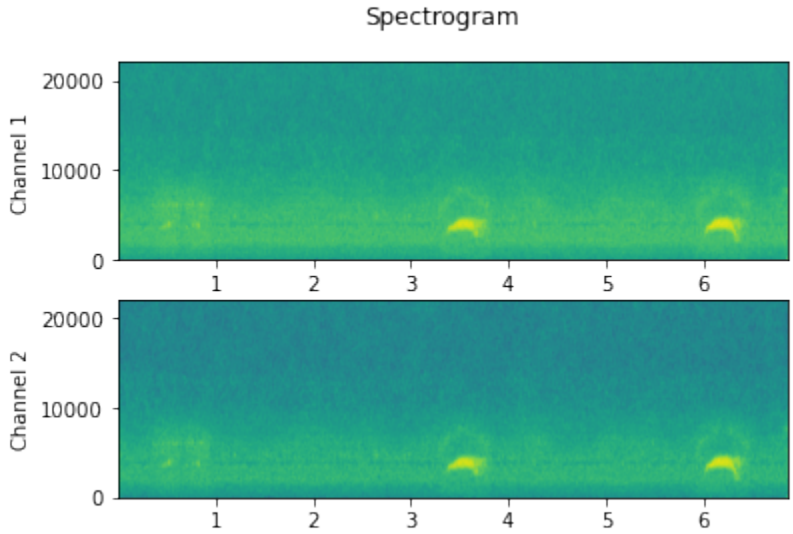

Figure 8: Generated Bird Recording

Going from figure 7 to figure 8, we see how non-granular sections are poorly represented as it essentially concedes to a low white noise across frequencies. However, the noticeable spikes in amplitude where the birds are present are perfectly synthesized according to the input sound. This result has major implications for the process of audio extension, for example an endangered bird of which we have limited recordings of. The results are promising and I believe that with better audio representations, this issue of granularity will become less prevalent.

5 Supplementary Material

Audio, code, and demo can be found here: github.com/j-lamadrid/audio-texture-synthesis

References

[1] 1505.07376 Leon A. Gatys, Alexander S. Ecker and Matthias Bethge. Texture Synthesis Using Convolutional Neural Networks, 2015; arXiv:1505.07376.

[2] 1905.03637 Hugo Caracalla and Axel Roebel. Sound texture synthesis using convolutional neural networks, 2019; arXiv:1905.03637.

[3] 1910.09497 Hugo Caracalla and Axel Roebel. Sound texture synthesis using RI spectrograms, 2019; arXiv:1910.09497.

[4] 2110.15018 Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair and Yangyang Shi. TorchAudio: Building Blocks for Audio and Speech Processing, 2021; arXiv:2110.15018.

[5] torchaudio.transforms - torchaudio 2.0.1 documentation Torchaudio.transforms¶, https://pytorch.org/audio/stable/transforms.html, torchaudio.transforms - Torchaudio 2.0.1 documentation