Guitar Tone Converter

An Independent Research Project
By Luke Abram
2022-2023

Abstract

The Guitar Timbre Converter can be used in post-recording production to achieve a specific tone with any guitar's source audio. The converter can save guitarists and producers money by simulating a specific guitar tone. Datasets from high-end guitars can bring a studio quality instrument to anyone's computer. The final iteration of the converter utilizes a short time fourier transform (STFT). This transform uses the audio signal of a wav file and converts it into a complex number spectrogram array. The array is then converted to an image file. The training dataset the model was trained on consisted of an acoustic and electric guitar playing individual notes. The model was able to convert acoustic audio to electric audio with minimum accuracy. This project provides a proof of concept that CycleGANs are able to convert audio timbres without the use of WaveNET, but instead standard STFTs. CycleGANs proved to be a viable solution to timbre conversion. Additionally, in a preliminary test on chord modification, the model was able to convert a perfect fifth of a sine wave to its corresponding major triad with great accuracy. In order to bypass the generator loss plateau which the model experienced, more training data is needed. The additional data should not be notes which have been formatted to fit spectrograms. About 60 minutes of training data (30 minutes A, 30 minutes B) should be sufficient to achieve a high level of accuracy as seen in similar projects.

Introduction

As of now, there is no good way to convert the tone of one instrument to another. Given a recording of an acoustic guitar, it is extremely hard to make that recording sound like an electric guitar instead. Though there are existing solutions, they often result in low audio quality for the output file. This approach utilizes a short time fourier transform to convert audio into spectrograms, or 2D frequency vs time arrays. Once the audio is converted into a spectrogram, it is fed into a CycleGAN neural network. CycleGANs perform image to image translation with unpaired datasets. The CycleGAN in this implementation, converts the audio of one instrument to another.

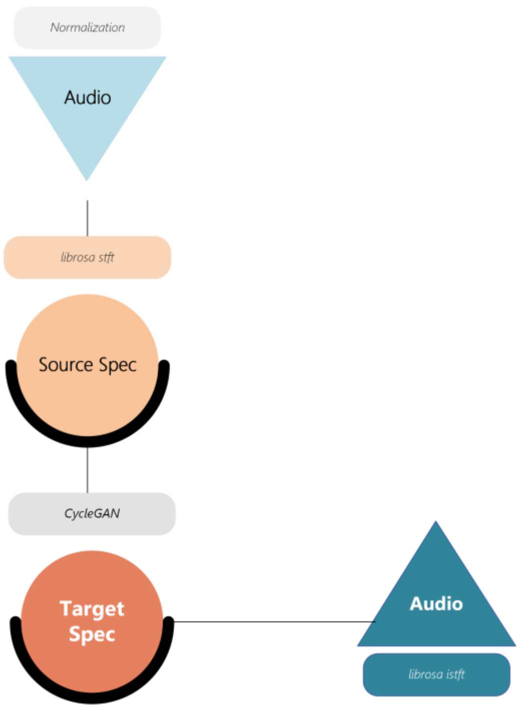


Figure 1.1 A flowchart of the proposed system

This research, if successful, would be instrumental in studio environments. Producers who have recorded audio of one instrument and decide they would like it played by a different instrument would be able to quickly convert the audio. Additionally, recordings of entry-level instruments could be converted to those of higher quality instruments given datasets of both. This would significantly cut costs for musicians.

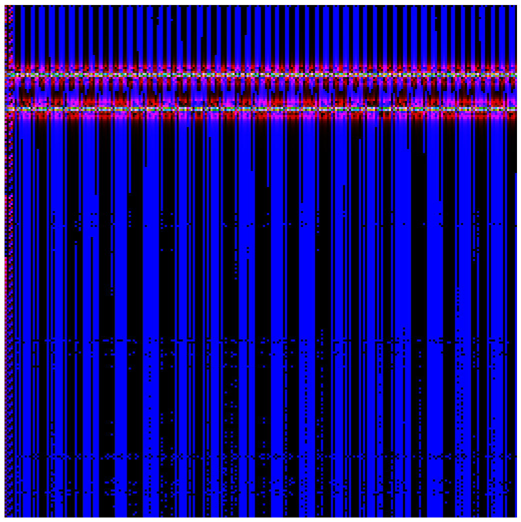
The concepts presented in this research, if further refined, could possibly be packaged as a product and hosted on a server, allowing people to use the computation-intensive software from any device. Additionally, since the actual conversion takes significantly less time/computation than the actual training of the model, a plugin could be made for DAWs (digital audio workstations) allowing users to convert audio using pre-trained models.

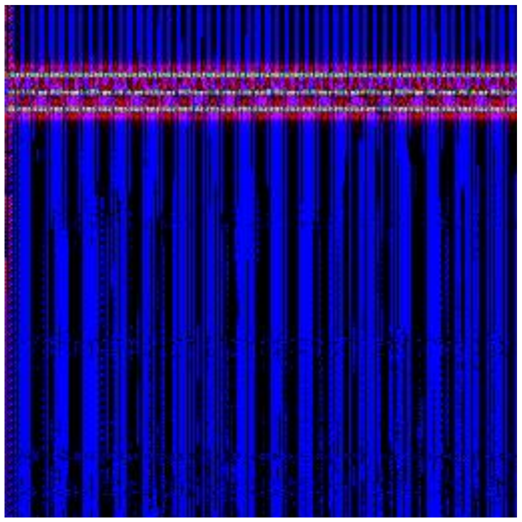
In order to classify this project as successful, the final software will need to be capable of translating instrument tones while maintaining pitch and the loss of both discriminators in the model should reach between 40-50%. A discriminator having a loss of around 50% means it cannot distinguish between generated and real samples, meaning the model is perfectly creating the target audio.

Methods/Approach

In order to complete the goal of instrument tone conversion, the project has been broken up into multiple steps. The first step is optimizing the conversion of audio to an image (spectrogram) and back to audio until there is no noticeable difference between the source and output audio. The spectrogram creation is done with the librosa STFT package. Then, the CycleGAN, which will originally be directly based off of the original CycleGAN paper (Zhu et al., 2020), will be trained and tested on datasets of instruments. These instruments will start off as simple as possible and gradually become more complex in their frequencies as the model is modified to better accommodate images. For example, they will start off as pre-sampled acoustic and electric guitars and eventually be real pianos and guitars.

Before testing the CycleGAN on timbre conversion, the model was tested on chord modification with sine waves as a proof of concept. Figure 2.1 shows spectrograms for the input interval and output chord. The model was able to consistently make accurate transformations on test data.

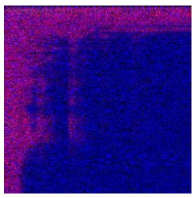




Perfect Fifth (input) → Model (A→B) → Major Triad (output)

Figure 2.1 A perfect fifth (left) and the output major triad (right). The model was able to convert between the two with exceptional accuracy.

Below, the first spectrogram in Figure 2.2 is an electric guitar playing a single note. The spectrogram was originally 1024x1024 pixels when created with Librosa, however, due to memory limitations and time constraints, these spectrograms are cropped to 256x256px.



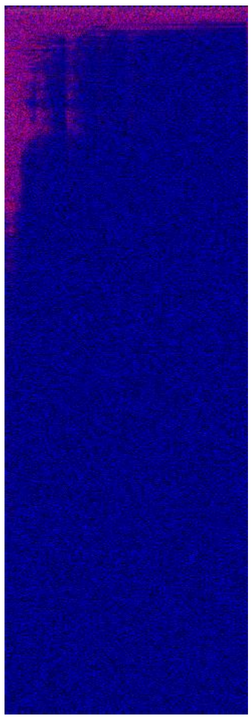


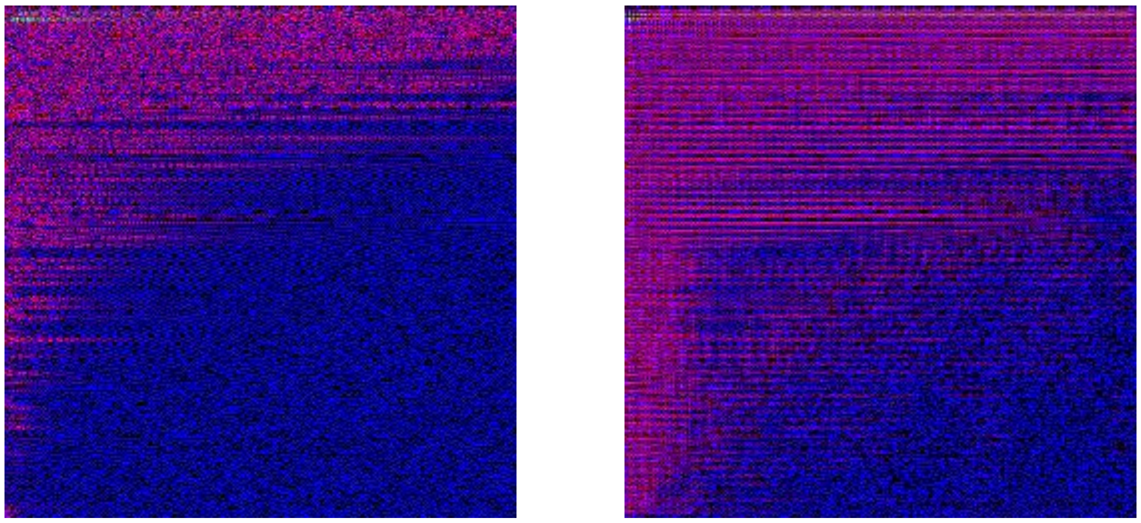
Figure 2.2 A cropped 256x256px spectrogram of an electric guitar playing one note for 2 seconds and the uncropped 1024x346px spectrogram of the same audio file. (This image is not 1024x1024 because the audio file was not long enough).

The second spectrogram in Figure 2.2 is of the same audio yet, as shown, only a small portion of the audio data is cut off when cropped. For this reason, sufficient translations can be made with ¼ of the data, however, when dealing with sources such as piano, there are much more overtones and harmonics in the upper frequencies.

STFTs were chosen for this project due to their ability to create spectrograms fairly quickly with a near-perfect conversion back into audio. The CycleGAN was selected due to its ability to use unpaired datasets of images. This way, datasets have less constraints and are much easier to make.

Each trial of this project will be tested solely on the ability of the model to create high quality output audio. Once the model's output audio is deemed sufficiently close enough to the target audio for all instruments, the model will be optimized to achieve 50% loss in the discriminator.

Figure 2.3 includes spectrograms from one of the early tests of acoustic to electric guitar conversion. The model is able to simulate the difference in tone through a purely mathematical function (the trained model).



Acoustic (input) → Model (A→B) → Electric (output)

Figure 2.3 Spectrogram input and output for acoustic/electric guitar dataset. This is an early trial yet the output is closer to the desired sound than the input.

Project Management

- Deliverables
  - Must deliver
    - Program capable of converting one instrument to another while maintaining pitch
    - Program's target audio is significantly closer to the target dataset than the source dataset
  - Time dependent
    - Packaged software as an application
    - Software hosted online for public use
    - Realtime conversion of audio
    - Hardware for software/raspberry pi module

Timeline (January 2023)

Planned Completion Date	Deliverable
Completed in 2022 (already done)	Model capable of adding major 3rd to chord
End of January 2023	Acoustic ←→ Electric Model

End of February 2023	Model for more complex sounds/higher resolution images
End of March 2023	Model packaged into application
End of April 2023	Hardware?

Results and Conclusions

Results

In order to verify that image to image CycleGANs can be used in musical applications a model was developed to convert sine waves in a perfect fifth interval to a major triad as well as the inverse conversion. This conversion was completed with extremely high accuracy such that the resulting audio retained its timbre as well as intonation or pitch accuracy. In order to create a working model with minimum accuracy, as much data as possible had to be saved in the output image file from the short time fourier transform spectrogram. The most promising iteration of the converter is capable of translating a single note played by an acoustic guitar into the bare minimum electric guitar tone through short time fourier transforms and aforementioned CycleGAN architecture. The accuracy of the most successful iteration was much less than expected when starting this project however it fulfilled the minimum viable product requirements. Going into the project, a higher degree of accuracy was expected much earlier in the process. After being extremely successful in interval to chord conversion, the model failed to produce similarly impressive results with timbre conversion.

The model's timbre conversion problems can be attributed to a lack of consistent training data. In the case of the interval and chord conversion, the dataset only had sine waves which were perfectly consistent in each image. However, in the case of guitar tone conversion, each strum of a note had a different signal due to differences in volume, picking style, and velocity of picking which caused the model to underfit the data.

The short time fourier transform creates a complex number array. For each cell in the spectrogram, there is a real and imaginary number each with a sign. One approach to this problem was to take the absolute value of the array, converting each complex number to a single positive number. Testing this method yielded subpar results which were attributed to the lack of phase data contained in the resulting images as well as the loss of information from combining and scaling each complex number. In order to bypass this I created two adjacent pixels for each cell from the spectrogram. Each pixel contained either the real or imaginary number scaled up to have a max of 0xffff and its sign. When testing the conversion of audio into a spectrogram and back to audio, this new method yielded near-perfect conversion with no audible difference in input and output audio.

In order to optimize the most successful version of the model, Constant-Q Transforms (CQT) were employed instead of short time fourier transforms. The underlying theory behind this decision was that CQTs use a logarithmic scale for their frequency bins unlike the linear scale for STFTs. The logarithmic scale aligns the frequency bins with the frequencies of the 12 tone equal temperament scale. This approach also created spectrograms which were 200x200 pixels instead of the 256x256 spectrograms from the short time fourier transforms resulting in quicker training times. However, due to unknown reasons, training on CQTs resulted in a worse translation. One theory is that the lower resolution of the CQTs resulted in less room for error in the model.

Much more testing and development is necessary in order to bring the Guitar Timbre Converter to a product level research project. This development would consist of training the model on a dataset of a single A and B image in order to get a perfect timbre conversion on an overfit model as a baseline for more advanced applications. Additionally, training the model on about 30 minutes of A and B data would seem to yield better results as it would solve the lack of data the model was facing and has proven accuracy in similar research.

References

librosa.stft—Librosa 0.10.0.dev0 documentation. (n.d.). Retrieved September 20, 2022, from https://librosa.org/doc/main/generated/librosa.stft.html

STFT_Notes_ADSP.pdf. (n.d.). Retrieved September 20, 2022, from https://course.ece.cmu.edu/~ece491/lectures/L25/STFT_Notes_ADSP.pdf

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2020). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* (arXiv:1703.10593). arXiv. http://arxiv.org/abs/1703.10593

Appendix

How do you translate unpaired images?

(Zhu et al., 2020)

Summary: In the paper “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks” researchers from the UC Berkeley AI Research Laboratory present a new way of performing image translation without a paired training dataset. The paper is therefore intended for those doing research in image translation where paired datasets do not exist. The CycleGAN model contains a function G so that G(X) → Y. However since the function is underconstrained it can easily lead to an unrelated and random output image which resembles the Y dataset. So, the CycleGAN model has an inverse function F(G(X)) → X or F(Y) → X to ensure the resultant images resemble the input images. The model was unable to make convincing output images for baseline tests, however it was comparable to the results of pix2pix, a GAN paired image to image translator.

Reflection: In my project I plan on exploiting the current advances in unpaired image to image translation. Unpaired image to image translation could mean that I do not need an identical audio sample from each guitar which would speed up the training of my model. I learned that the inverse loss is necessary to make sure there is a correlation between G(X) and X. When I build my machine learning model for image translation I’ll be incorporating the exact model described in the article to convert heatmaps of audio. I am wondering if image translation of a heatmap would produce useful audio output as heatmaps all look the same and can have large similarities between each other.

How can you convert audio to a heatmap?

(*Librosa.Stft — Librosa 0.10.0.Dev0 Documentation*, n.d.)

Summary: Librosa is a python package for audio analysis. It includes functions capable of spectral representation of audio samples. One function in particular is librosa.stft which performs a short time fourier transform and displays audio as a spectrogram or heatmap. The function provides multiple options for modifying the spectrogram through passing arguments. One can change the number of audio samples between STFT columns or the length of a window of audio. Librosa is an open-source library and commonly used across python audio projects today. The documentation is designed for developers who are working on projects requiring audio analysis and transformation. These functions are available in other packages such as scipy however the only difference seems to be that scipy scales the spectrogram.

Reflection: The librosa stft documentation provides useful information on functions which I will be using to create the training dataset and convert the output of the machine learning model back to audio. I gained new information on what the parameters of the stft function do and the requirements for each of them. I will apply the information from this documentation to optimize the short time fourier transform for audio from a guitar in which case audio fidelity is extremely important. I'm wondering how a change in the hop length does not affect the resolution of the audio.

How to improve the quality of the audio as you convert a heatmap to an audio file?

(*STFT_Notes_ADSP.Pdf*, n.d.)

Summary: The notes on short time fourier transforms from Carnegie Mellon University's electrical and computer engineering course, provide a thorough explanation of the math behind a short time fourier transform. The notes also go over different parameters for a stft such as the window size and shape. When explaining window size of an stft, the paper explains that a lower window size improves the audio quality of In addition to short time fourier transforms, the notes explain inverse short time fourier transforms which convert a time-frequency spectrogram back to audio. Since these notes are coming from a CMU class there is reason to believe they are accurate and reliable especially on a topic with a great amount of math involved. These notes are directly related to the Librosa.stft documentation as they are the same function. These notes provide a deeper overview of how the librosa short time fourier transform works and an example of how window size affects the quality of the spectrogram.

Reflection: I will be using the information from the notes on short time fourier transforms to optimize the stft function for guitar conversion. I learned that window length directly affects the resolution of the audio. I will be reducing the window length in order to increase audio quality for my project. I'm wondering how hop length and window length are related to each other. I thought the hop length would be synonymous with the window length but it's not.