CS230

Music Timbre Transfer

Jay Appaji
Department of Computer Science
Stanford University
jappaji@stanford.edu

Arunothia Marappan
Department of Computer Science
Stanford University
arunothi@stanford.edu

Siddharth Saxena
Department of Computer Science
Stanford University
sisaxena@stanford.edu

1 Introduction

In this project, we explore the problem of timbre style transfer of a piece of monophonic melody. Timbre is the unique characteristic of a musical instrument which distinguishes it from another. For example one may be playing the same note on a violin and a piano, but still humans would be able to tell one instrument from another. This is because even when playing the same musical note on different instruments, the sound quality coming from an instrument has other higher harmonics of the fundamental note of varying amplitude. This combination of the fundamental note and higher frequency harmonics truly represents the sound quality of a musical note coming from an instrument. Different instruments have different harmonics of the fundamental, and different amplitude of the harmonic. This results in each instrument's unique sound quality even if musical note is the same. The input to our model is a piece of melody played in one instrument (in .wav format), and the output would be the same melody (in .wav format), but the timbre changed to that of a target instrument. Timbre transfer is a particularly useful application for composers who quickly want to experiment how a melody may sound on multiple instruments. To limit the scope of the project, we focus on timbre transfer between piano and flute.

2 Related work

Music style transfer literature consists of either timbre transfer or music compositional style transfer (for example style transfer from classical to Jazz, or changing the style of Beethoven to Mozart).

TimberTron (5) outlines a network in which an audio signal's Constant Q Transform (CQT) is used as the input to a Generative Adversarial Network (GAN), called CycleGAN. CycleGAN is a network used for unsupervised image-to-image transfer problems originally proposed by (Jun-Yan Zhu et. al) (6). The output from CycleGAN is then used as the input to WaveNet, which serves to recreate the audio signal. One advantage of this approach is that it uses the log-amplitude CQT representation as the input as opposed to the Short Time Fourier Transform (STFT) because the CQT possesses greater frequency resolution at lower frequencies. In the context of timbre transfer, this provides a greater ability to represent lower register instruments such as trombone, and/or preserve time resolution for higher-register instruments such as flute, which in turn makes it easier to preserve the rhythmic structure of the music. A potential limitation of this approach is that when the CQT is computed and input into CycleGAN, the phase information is discarded. The phase information is estimated during waveform reconstruction in WaveNet through the Griffin-Lim algorithm (4). This process randomly generates phase values and then applies the STFT and reverse STFT operations

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

on it until convergence with representation in hand. A potential counter-solution offered is the use of a "Rainbowgram" which represents the time derivative of the phase information using colors. The results were validated via perceptual test, where the authors played generated audio to a set of listeners and asked the listeners to evaluate the audio based on certain musical criteria.

An alternate approach is to use the raw waveform itself instead of using a time-frequency representation of the audio. Engel et. al (3) proposed Wavenet auto encoder model capable of processing raw waveforms and is able to generate new realistic timbre styles. This model is trained on the Nsynth dataset.

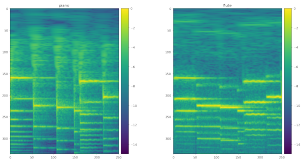
3 Dataset and Features

Our dataset is a subset of the MIDI files downloaded from http://www.jsbach.net/midi/, www.piano-midi.de/chopin.htm and http://www.piano-midi.de/. We further use a software called Timidity to synthesize .wav files from MIDI in the sounds of flute and piano. Using this approach, we are able to create paired data for the same melody on piano and flute. While use a cycleGAN for the task of timbre transfer which does not require paired data on the two domains, but paired data is useful for evaluating our model and listening to the sound quality output of our model versus the original piece of melody. Our train set consists of 21000 samples each of flute and piano wav files.

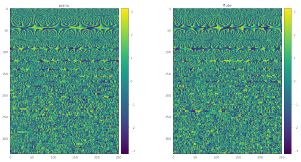
3.1 Audio to Spectograms

From literature analysis, we chose CQT of the wav file to be the best feature representation for this task.

CQT is a complex number containing magnitude and phase information for various frequencies. Figure- 1a shows the log(abs(CQT)) and Figure- 1b shows the phase(CQT) for the same piece of melody when played on a piano and a flute (x-axis is time, y-axis is different frequency bins). As can be seen from the magnitude CQT, piano melodies seem to have more sharp boundaries across time compared to a flute. Also higher harmonics of the fundamental note for flute are more dispersed compared to piano as can be seen towards the bottom section of the magnitude CQTs. On the other hand, the phase plot looks extremely similar. Literature survey also suggests that humans are perceptually not too sensitive to small errors in phase of the audio signals.



(a) Log abs(CQT): Piano (left), Flute (right)



(b) Phase: Piano (left), Flute (right)

3.2 Data Normalization

We normalized the log(CQT Magnitude) value before passing it through our generator network. For this normalization, we used the range of the data distribution (see Table- 1) of piano and flute respectively to shift it to the range (-1,1). This helps preserve the inherent distribution in the data while also making it suited for an optimal gradient descent.

	Mean	Median	Std	Max	Min
Flute	-6.44	-6.11	4.38	1.84	-39.07
Piano	-6.02	-5.30	4.42	1.58	-40.52

Table 1: Statistics of Log(CQT Magnitude)

4 Approach

Our pipeline consists of the following steps:

(1) A pre-processing step to convert raw audio (wav file) to Log-absolute CQT and Phase CQT and normalize them.

(2) Feed the Log-absolute CQT of source and target instruments, unpaired (by random sampling) as separate classes to CycleGAN.

(3) Log-absolute CQT essentially becomes a 1 channel image to the CycleGAN which learns the mapping between the Log-absolute CQTs of piano and Flute as detailed in Figure-3.

(4) Use the target instrument's Log-absolute CQT generated from CycleGAN along with source instrument's phase CQT to generate the raw audio of target instrument.

Refer to Figure- 2 to understand how we generate the target instrument's raw audio from source instrument's raw audio.

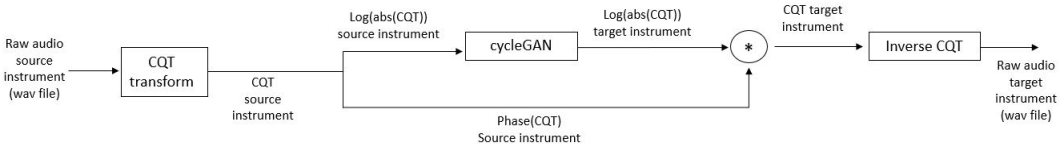


Figure 2: Target Instrument's Raw Audio Generation

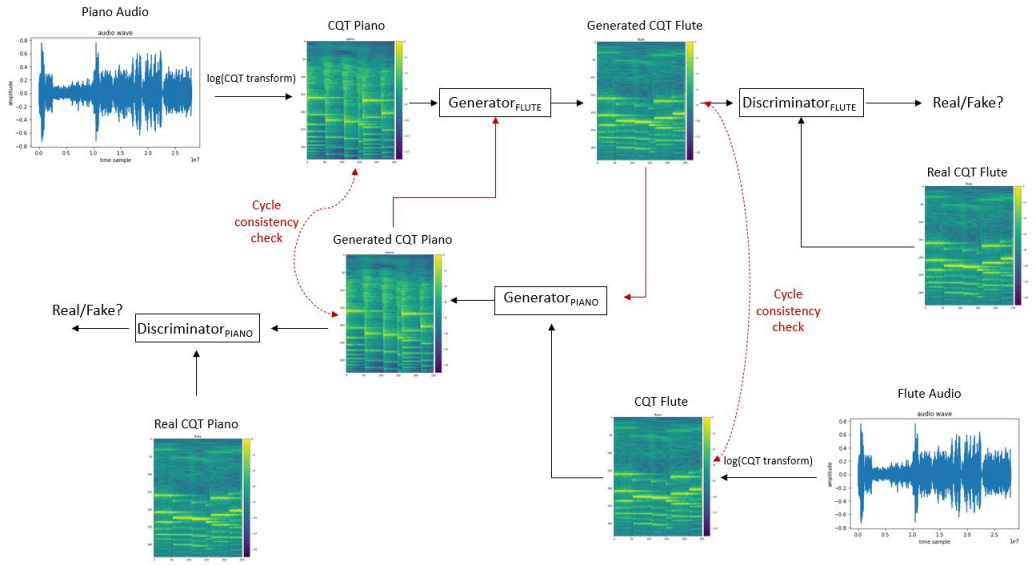


Figure 3: CycleGAN Architecture

5 Methods

5.1 Discriminator Loss

The discriminator loss minimizes the mean square error of discriminator's output with what is expected of it. That is, 0s for fake image input and 1s for real image input.

$$L_{Disc} = \mathbb{E}_{x \sim X}[||D(x) - 1||_2] + \mathbb{E}_{x \sim X}[||D(G(x) - 0||_2] \quad (1)$$

5.2 Cycle Loss

This is the loss that helps estimate whether passing a piano to a flute generator and passing the fake flute generated again to a piano generator produces the original piano or not and vice-versa for flute $\implies$ fake piano $\implies$ flute. We use $\mathcal{L}_1$ Loss to estimate this.

$$L_{cyc}(F, G, \mathcal{X}, \mathcal{Y}) = \lambda_{cyc} \cdot (\mathbb{E}_{x \sim X}[||G(F(x)) - x||_1] + \mathbb{E}_{y \sim Y}[||F(G(y)) - y||_1]) \tag{2}$$

5.3 Identity Loss

This is the loss that helps estimate whether passing a piano to a piano generator behaves like identity and vice-versa for flute. We use $\mathcal{L}_1$ Loss to estimate this.

$$L_{Identity}(F, G, \mathcal{X}, \mathcal{Y}) = \lambda_{identity} \cdot (\mathbb{E}_{x \sim X}[||F(x) - x||_1] + \mathbb{E}_{y \sim Y}[||G(y) - y||_1]) \tag{3}$$

5.4 Generator Loss

The generator loss is basically a combination of Cycle Loss, Identity Loss and the loss of being able to fool the discriminator. This later loss is estimated by mean square loss between discriminator output for a fake image and 1s (basically the opposite of what the discriminator is doing).

$$L_{Gen} = \mathbb{E}_{x \sim X}[||D(G(x)) - 1||_2] \tag{4}$$

5.5 Optimizer

We use Adam Optimizer with β_1, β_2 values as mentioned in Table- 2.

5.6 Gradient Penalty (GP)

In the CycleGAN original paper (6), they propose a Patch Discriminator, which basically looks at a patch (70 x 70) grid of the input image to identify whether it is fake or not, instead of doing it for the whole image. We used this method in our vanilla CycleGAN model as mentioned in Table- 2. In Timbertron paper (5), they suggest to avoid using a patch discriminator for analysing time series data like audio spectograms. And when a whole image discriminator is built, they suggest to use gradient penalty as a way of regularizing the discriminator to avoid the discriminator from becoming too good at the start.

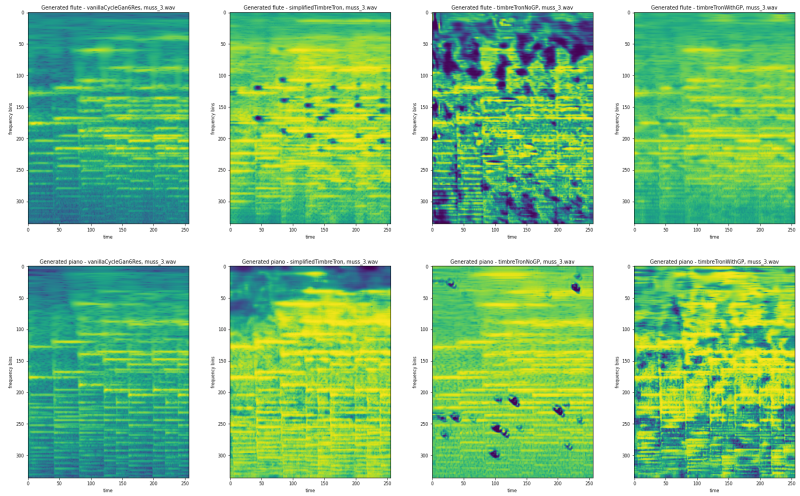
$$L_{GP}(G, D, \mathcal{Z}, \hat{\mathcal{X}}) = \lambda_{GP} \cdot (\mathbb{E}_{\hat{x} \sim \hat{X}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]) \tag{5}$$

6 Experiments/Results/Discussion

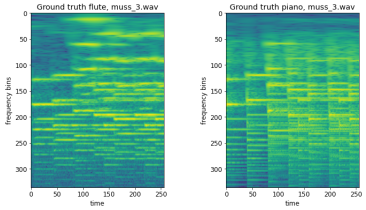
- Our generated audio samples are displayed on our project website (2)
- Our code is located on github (1)

We started experimenting our approach with what we call a Vanilla CycleGAN. This uses the original CycleGAN architecture and the hyper-parameter values recommended in CycleGAN paper (6). We then built CycleGAN model described in Timbertron paper (5), we initially restricted the model to $1/4$ of the feature sizes mentioned in the paper, assuming that translating amongst only two timbres will work fine with a smaller architecture itself. We call this model Simplified TimbreTron and this performed worse than Vanilla CycleGAN as you can see in the loss curve 7. We then trained a full TimbreTron both with and without Gradient Penalty. Due to the lack of time, we were unable to make these models work better than our Vanilla CycleGAN model. The details of all our model architectures are given in Appendix.

As this project is a very perceptual one and due to the limited time, we did not perform quantitative analysis on the models other than looking at their loss curves. For qualitative analysis, we did human evaluation amidst ourselves by looking at random samples generated by each model. We looked at both the Spectograms generated as well as the audios reconstructed and observed that our Vanilla CycleGAN model worked best for flute to piano and vice-versa translation, similar to what we observe in the Loss curve- 7. Spectograms generated by our models is shown in Figure- 4a and the corresponding original spectogram is shown in Figure- 4b.



(a) Generated Log(Abs CQT): Vanilla CycleGAN, Simplified TimbreTron, TimbreTron without GP, TimbreTron with GP (left to right)



(b) Original

Model	Batch Size	Epochs	Learning Rate	β_1	β_2	Residual Blocks
Vanilla CycleGAN	4	10	2e-4	0.5	0.999	6
Simplified TimbreTron	4	50	1e-4	0	0.9	6
TimbreTron without GP	4	10	1e-4	0	0.9	9
TimbreTron with GP	1	6	1e-4	0	0.9	9

Model	λ_{cyc}	$\lambda_{identity}$	λ_{GP}	Norm	Workers	Patch Discriminator
Vanilla CycleGAN	10	1	0	uniform	2	yes
Simplified TimbreTron	10	5	0	uniform	2	no
TimbreTron without GP	10	5	0	uniform	2	no
TimbreTron with GP	10	5	10	uniform	2	no

Table 2: Parameter values of different Model Architectures

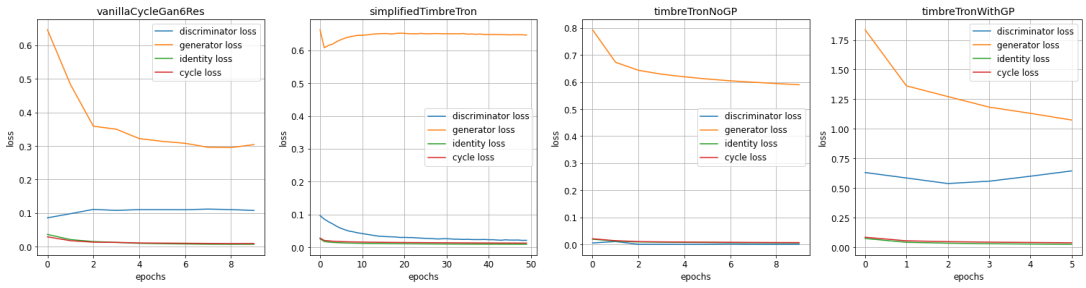


Figure 5: Loss Curves obtained from running our different models

7 Conclusion/Future Work

We conclude that vanilla CycleGAN implentation also works reasonably well for spectogram translation between flute and piano. We also note that even though we could not build a WaveNet like the authors of TimbreTron (5) for the reconstruction of the generated spectogram into audio, out method of reconstructing the audio from source audio's phase CQT and generated abs CQT using invert CQT works reasonably well as well for the problem of translation between flute and piano.

As future work, we would like to build a WaveNet for the reconstruction of generated spectogram into audio. This we believe will remove the noise we currently hear in our output audio.

8 Contributions

Arunothia worked on setting up AWS compute, vanilla CycleGAN implentation, training, evaluation, report, setting up project page and video. Jay was looking into Wavenet implementation for the spectogram reconstruction into audio. Siddharth worked on Timbretron implementation, gradient penalty, training, evaluation, output generation and video.

References

[1] Code Repository. `https://github.com/Arunothia/CS230-project`

[2] Project Webpage. `https://arunothia.github.io/CS230-Project/index.html`

[3] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., Norouzi, M.: Neural audio synthesis of musical notes with wavenet autoencoders (2017)

[4] Griffin, D.W., Jae, Lim, S., Member, S.: Signal estimation from modified short-time fourier transform. IEEE Trans. Acoustics, Speech and Sig. Proc pp. 236–243 (1984)

[5] Huang, S., Li, Q., Anil, C., Bao, X., Oore, S., Grosse, R.B.: Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer (2019)

[6] Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR **abs/1703.10593** (2017), `http://arxiv.org/abs/1703.10593`

9 Appendix

Figures below show the generator architectures we used for our various experiments.

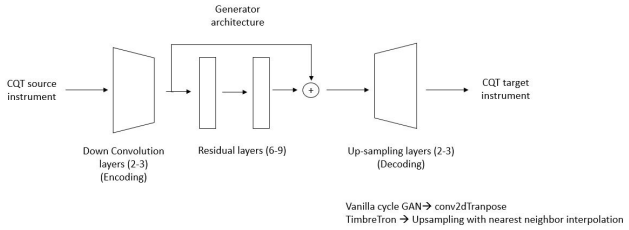


Figure 6: Generator architecture used in various experiments

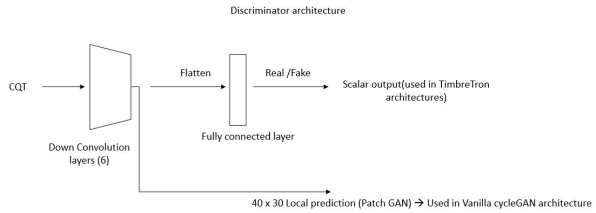


Figure 7: Discriminator architecture used in various experiments