CS230

Timbre Transformation using the DDSP Neural Network

Yamaan Atiq
Department of Aeronautics and Astronautics
Stanford University
yamaan@stanford.edu

Kathlynn Simotas
Department of Music
Stanford University
ksimotas@stanford.edu

Abstract

We will implement the process of timbre transformation on large musical datasets consisting of a variety of instruments, including non-traditional or non-Western instruments. We use DDSP as our learning method, employing the Magenta framework for audio processing. We will then compare the output results with existing digital signal processing techniques such as cross-synthesis. Code is found on https://github.com/Yamaan44/CS230.

1 Introduction

This project investigates timbre transformation: a melody recorded on a flute could, for instance, be transformed into a similar sounding one performed on a piano. This process is known in the field of Music Information Retrieval as "timbre transformation". The problem is interesting as it has mostly been studied from a digital signal processing (DSP) perspective, employing techniques from DSP to modify audio spectra, and less from a deep learning perspective. The work which has delved into deep learning approaches has also been focused primarily of transformations of western classical instruments. We find that music generation in this way can be useful for generating entire libraries of songs recorded in different instruments, similar to methods in which voices can be changed, or "faked" by training a network with a dataset of snippets of sounds.

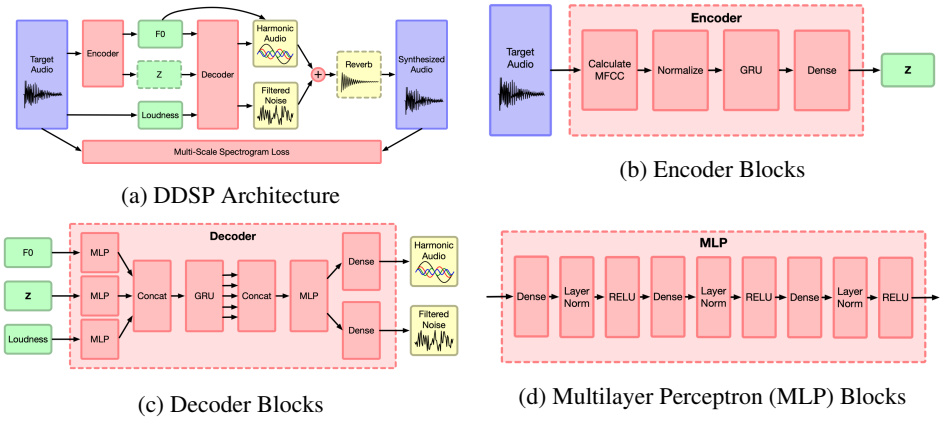
2 Literature Review & Base Architecture

We refer to existing literature (some of which is cited in the references) relating to timbre transformation as a process in signal processing and in deep learning. In particular, we will closely examine the methods presented in the creation of GANSynth and DDSP. Both algorithms, created by the Google Magenta team, are the state of the art methods of timbral transfer to date [1], [2].

Generative Adversarial Networks (GANs) are typically used on more visual applications, where the generation of a result can typically be observed meaningfully (such as an image) and its error quantified. Previous researchers have found audio processing and generation difficult using similar methods, since elements of audio (such as a time series of pitch or frequency may be generated close to a desirable result, but might sound strikingly different). This is in contrast to generating an image, where a slightly different colored or tinted image will still look recognizable, and so error characterization and training is made easier. Google's Magenta team has nevertheless implemented an open source toolbox, GANSynth, to generate audio using GANs [2]. Attempting to use a GAN has its advantages: the main one comes with speed: traditional audio generation toolboxes such as WaveNet have implemented sequence models, where sound is generated in sequence, analogous to many NLP algorithms. This is easier to implement but can result in generated sound clips, especially songs, that

CS230: Deep Learning, Fall 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

do not sound cohesive. Contextual clues from a later time in a song are not used to inform sound earlier. This is also slower, as each sample is used for the next, and so there are specialized kernels required to parallelize generation. GANs tackle both of these issues by generating an entire audio sequence at once, making a clip more cohesive by making use of audio at earlier and later timesteps to look for context clues. Further, samples are not generated sequentially, so the entire clip is generated more efficiently. The Magenta team acknowledges that GANs fall short in cohesiveness on a local timescale, even when the overall clip sounds more cohesive - this decreases overall quality of an audio clip. GANSynth addresses this issue by using instantaneous frequencies and log magnitudes in its model, and yields better results.



(a) DDSP Architecture

(b) Encoder Blocks

(c) Decoder Blocks

(d) Multilayer Perceptron (MLP) Blocks

Figure 1: DDSP Blocks [3]

Differentiable Digital Signal Processing (DDSP) takes a completely different approach to the frequency domain one used by GANSynth [3]. GANSynth still does not necessarily yield subjectively "good" sounding audio - DDSP meanwhile uses the way sound is perceived using vocoders and synthesizers to separate audio generation in to elements typically used to electronically generate music. Signal processing has historically been difficult to integrate with modern machine learning architectures which use autodifferentiation. The DDSP library is built into modules which are compatible with autodifferentiating NNs, allowing the user to focus on specific modules and control elements like pitch and loudness independently. Figure 1 shows the architecture from the original paper, showing integration of deep learning blocks with traditional signal processing blocks. This makes DDSP an ideal choice for timbre transformation, which is an inherently perceiving problem with learned elements.

3 Dataset

To gain access to non-Western instrument data and non instrument sounds, we use the AudioSet dataset which consists of TFRecord representations of more than 2 million 10 second clips from Youtube videos sampled at 16kHz [7]. Each clip is rigorously labeled by instrument and sound type, although many clips contain multiple labels and instruments in them for the same clip. The labels we are most interested in using for our timbre transformation process are the harp, accordion, bagpipes, didgeridoo, and theremin which have for each of them individually between 2,800 and 600 entries (including entries with more than one label). We are also planning to conduct an experiment with timbre transfer using bird songs as well as using more polyphonic instruments and musical settings like the harmonica, orchestra, and choir.

We initially planned to use our Youtube clips in their original TFRecord format, separating them by label with a custom script. However, we quickly discovered that the format of the dataset was incompatible with either our original framework idea, GANSynth, or DDSP. Therefore, we decided to find a way to recover the raw audio data from AudioSet and repackage it as readable data. This does introduce new challenges in our approach, however, namely devising a method for changing data formats such that the original data and labels are maintained as best as possible. There is also an inherent challenge with this dataset that data has multiple labels, which is not ideal for learning for

the purposes of timbre transformation. At best, data for timbre transformation has historically been monophonic, single-instrument data, however, in order to conduct transformation on lesser known or less popular instruments or background noises, we must accept datasets that provide such choices, even if it comes at some cost to the amount of usable data we have or to some extent the quality of the data.

4 Approach

Since NSynth files were used to train the GANSynth model, their weights were optimized to take NSynth input files. The Magenta library does not provide a function or class for pre-processing input mp3 or wav file types in to NSynth TFRecord files, so carrying over some pre-trained weights to apply transfer learning to a new dataset of mp3 files renders the optimized weights not meaningful. In this sense we discovered GANSynth is limiting in its ability to apply transfer learning.

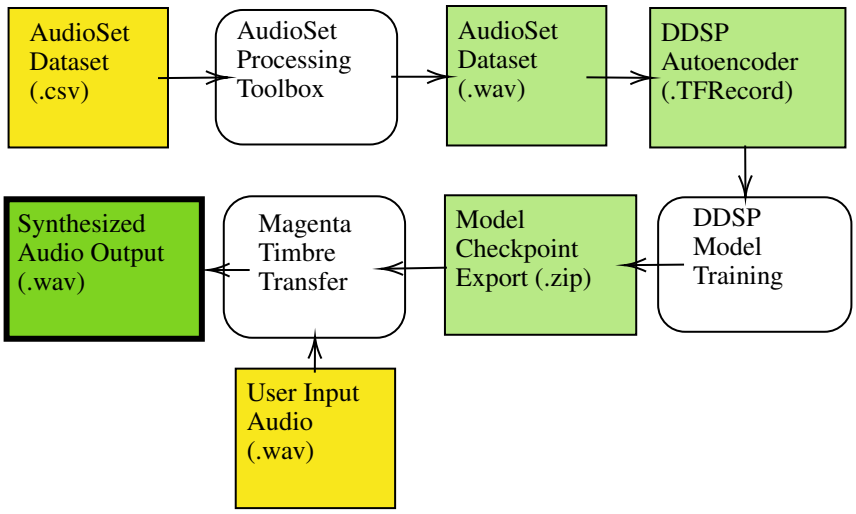
The solution to this was to train the network from scratch, and we have opted for a different model from the Magenta DDSP Library for this training to simplify the scope and avoid retraining on NSynth, which would be too computationally expensive. We use the DDSP Autoencoder, which consists of three separate and distinct neural networks training on different aspects of the data. The pitch or frequency encoder uses a residual convolution neural network, while both the loudness and residual vector encoders use GRUs. Since the overall architecture is differential, the neural networks can be trained together under an SGD optimizer. The losses used by the overall architecture are the multi-scale spectral (which compares the magnitude spectra of the original and reconstructed signals using L1 losses) and perceptual losses for the frequency encoder (which is also calculated using an L1 loss and weighting factor relative to the spectral loss)(3).

Our approach is implemented as follows:

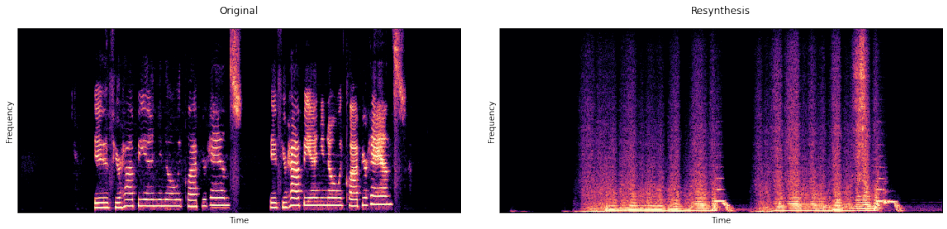
1. The Audioset dataset is a CSV file encoding information about a large collection of Youtube videos, and so there is a preprocessing step required to extract the audio in to some format readable by our model.
2. We employ the AudioSet processing toolbox developed by Aoife Mcdonagh to do access the original audio information via wav files. We use this framework to download audio associated with the video within the set time interval specified in the CSV versions of the data. One of our hyperparameters to tune is the size of the training set - more specifically, we would like to determine a sufficient minimum size for the training set, as when using the entire Theremin set of training clips we did not obtain entirely desirable output. [4]
3. DDSP Autoencoder to preprocesses audio files into a format with meaningful statistics for model to be trained on. wav files must be turned into TFRecord files, similar to the ones used by GANSynth, that NSynth provided.
4. Train the model using the TFRecord dataset. TFRecord files encode all information needed for training, features such as loudness and fundamental frequency, so we do not need to provide any more training examples.
5. We trained the model using Magenta's DDSP solo_instrument gin model. For a dataset of 637 training examples for the Theremin, this took about 2 hours on a GPU. We noted that at around 12,000-15,000 steps the loss was constantly fluctuating around 5-10, the measure at which it says the model is trained. It is possible that our model is overfitting the data here at least in our initial experiment.
6. Export the model as a checkpoint for Magenta's Timbre Transfer skeleton. We have already generated our DDSP model containing weights, so all that is left is to synthesize audio.
7. Upload an input sound clip of sound to use with our synthesizing code. As an example, we show a clip of singing whose characteristics and output are discussed more in the results section.

5 Workflow

Shown below is a flowchart summarizing the workflow steps from the input dataset to the output synthesized audio.



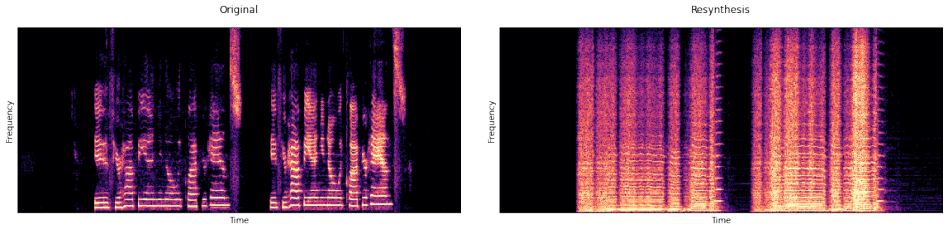
6 Results



(a) Input Singing Spectrogram (b) Output Theremin Spectrogram

Figure 2: Spectrogram Comparison between input and output

The spectrograms between the original and synthesized audio clips look reasonably similar in shape and color, but sound very different. In particular, there are some significant jumps periodically in the synthesized data which seem to be artefacts of the model potentially overfitting and adding extraneous features.



(a) Input Singing Spectrogram (b) Output Bagpipe Spectrogram

Figure 3: Spectrogram Comparison between input and output

6.1 Training and Loss Function

We trained the DDSP solo instrument model on two different instruments, the theremin and the bagpipes both using the default hyperparameters and batch sizes of 16. For the theremin we had 3,000 examples of 4 second clips and for the bagpipes we had 12,000. The DDSP model uses both spectral loss (described in one of the DDSP paper's citations, Wang et. al. 2019) and L1 loss functions. The total loss is calculated using the original and synthesized magnitude spectrograms, S_i and S_{si} respectively using the following equation:

$L_i = ||S_i - S_{si}||_1 + \alpha||logS_i - logS_{si}||_1$

For the theremin we trained over approximately 12,000 epochs and for the bagpipes approximately 30,000.

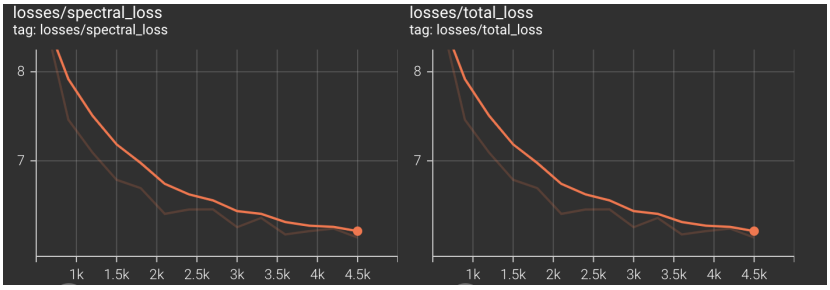


Figure 4: Theremin Loss Function for 4.5k training steps

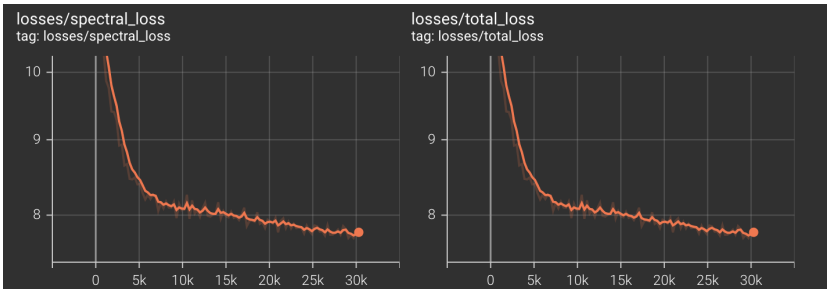


Figure 5: Bagpipe Loss Function for 30k training steps

6.2 Error Characterization and Model Improvement

Timbre in general is difficult to characterize quantitatively. Timbre transfer itself is a highly subjective process to evaluate, and so in this project, we determined the best method of error characterization to use was comparing a time series of fundamental frequency f_0 and computing the root mean square deviation between an input sound clip and the output. We show a plot of f_0 for the bagpipes in Figure 6 and compute the RMS deviation, which we find to be 20.27 for the bagpipes and 19.66 for the theremin. It is worth mentioning that this metric is flawed - a better method of characterizing the timbre transfer effectiveness would be to train another neural network on various input and output clips for the synthesizer to determine how well timbre transfer has worked. But replicating the fundamental of the original input is an essential first step of timbre transfer, so this was the most accessible and illuminating quantitative error measurement in our view.

7 Future Work

Future work would involve model improvement and model evaluation. Firstly, we need to improve the model as our generated results are not showing the synthesis we expect. We need to be careful about the conclusions we make given the inherent background noise in our dataset, as well as decide whether or not to limit our data to only those with single labels (less background noise). It is also possible our model is overfitting, so we need to evaluate the model quantitatively and then improve the model's synthesis by tuning hyperparameters - we believe a large part of this is the input dataset size, so it is possible that by splitting the dataset into minibatches, we might improve our performance.

The model evaluation is coupled with this - we need to define specific metrics to quantitatively evaluate the model. As audio is a highly qualitative process, in getting the model running we have mainly been using playback to check performance. Over the course of a larger dataset, performance metrics would be necessary here.

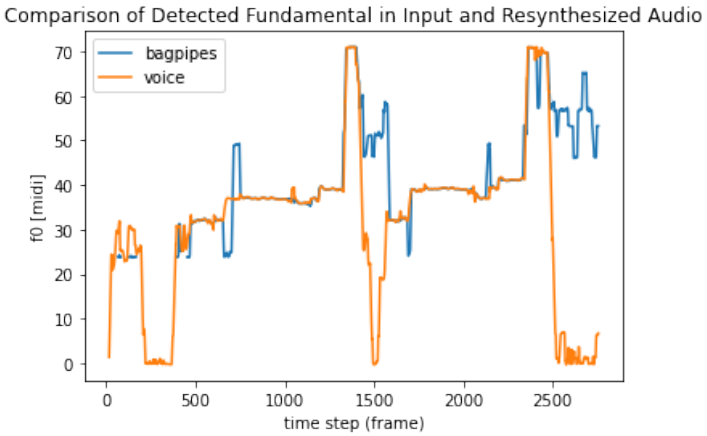


Figure 6: Fundamental Original vs Resynthesized for Bagpipe Model

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[4] Mcdonagh, A. (n.d.). Aoifemcdonagh/audioset-processing: Toolkit for downloading and processing Google's AudioSet dataset. GitHub. Retrieved November 8, 2021, from https://github.com/aoifemcdonagh/audioset-processing.

References

[1] COSMIR. cosmir/dev-set-builder: Boostrapping weak multi-instrument classifiers to build a development dataset with the open-mic taxonomy.

[2] ENGEL, J., AGRAWAL, K. K., CHEN, S., GULRAJANI, I., DONAHUE, C., AND ROBERTS, A. Gansynth: Adversarial neural audio synthesis.

[3] ENGEL, J., HANTRAKUL, L. H., GU, C., AND ROBERTS, A. Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations* (2020).

[4] ENGEL, J., RESNICK, C., ROBERTS, A., DIELEMAN, S., ECK, D., SIMONYAN, K., AND NOROUZI, M. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

[5] GABRIELLI, L., CELLA, C.-E., VESPERINI, F., DROGHINI, D., PRINCIPI, E., AND SQUARTINI, S. Deep learning for timbre modification and transfer: an evaluation study.

[6] GABRIELLI, L., CELLA, C. E., VESPERINI, F., DROGHINI, D., PRINCIPI, E., AND SQUARTINI, S. Deep learning for timbre modification and transfer: An evaluation study. In *Audio Engineering Society Convention 144* (2018), Audio Engineering Society.

[7] GEMMEKE, J. F., ELLIS, D. P. W., FREEDMAN, D., JANSEN, A., LAWRENCE, W., MOORE, R. C., PLAKAL, M., AND RITTER, M. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017* (New Orleans, LA, 2017).

[8] HAORANWEIUTD. Haoranweiutd/chmusic: Baseline solution for chinese traditional instrument recognition on chmusic dataset.

[9] PATERNA, M. Timbre modification using deep learning, Jan 2017.

[10] WENNER, C. Timbre transformation.