AN EVALUATION OF METHODS FOR INSTRUMENT STYLE TRANSFER

Sachinda Edirisooriya **Heidi Cheng** **Payal Pandit** **Ahmed Hussaini**
University of California San Diego
{sediriso,h8cheng,ppandit,a5hussai}@ucsd.edu

ABSTRACT

In this work, we tackle the problem of instrument style transfer. More concretely, the task consists of converting an input piano audio clip to a guitar audio clip that has the same musical properties (pitches/notes) as the piano audio clip, but sounds like a guitar. To solve this, we first create a dataset of piano and guitar audio clips. We then propose three approaches to solve the problem: convex optimization, a waveform-based neural network, and a spectrogram-based neural network. Through our experiments, we find that all of these methods are capable of outputting reasonable guitar audio clips given a piano audio clip, with the spectrogram-based neural network performing the best on our human evaluation metric. All source code used for our experiments can be found at `https://github.com/HeidiCheng/wi22-203-FinalProject/`

1. INTRODUCTION

1.1 Motivation

When a music producer is writing a melody, they often try having several different instruments play it before deciding which instrument fits the song the best. Sometimes there may be a specific sound from another song that the producer wants to use, however they are unsure how to recreate it using a synthesizer. Instead of trying to recreate it, what if the producer could just convert the sound from a basic instrument to the sound of some other instrument? Towards a solution to this problem, we tackle the simpler task of instrument style transfer using paired data.

1.2 Previous Works

In the past few years, style transfer has become quite popular in the computer vision community. Recently, researchers involved in music have begun to apply these techniques to music. Some examples of this are in [1, 2] where the authors use CycleGAN-inspired techniques [3] to perform transfer from one instrument to another with unpaired audio samples. Additionally, Bitton et al. proposed using a VAE for one-to-many instrument style transfer using unpaired audio samples [4]. Unlike these works, we use paired audio samples, and find that this allows us to solve this task in a simpler way, resulting in a more stable training process than something like a CycleGAN or VAE.

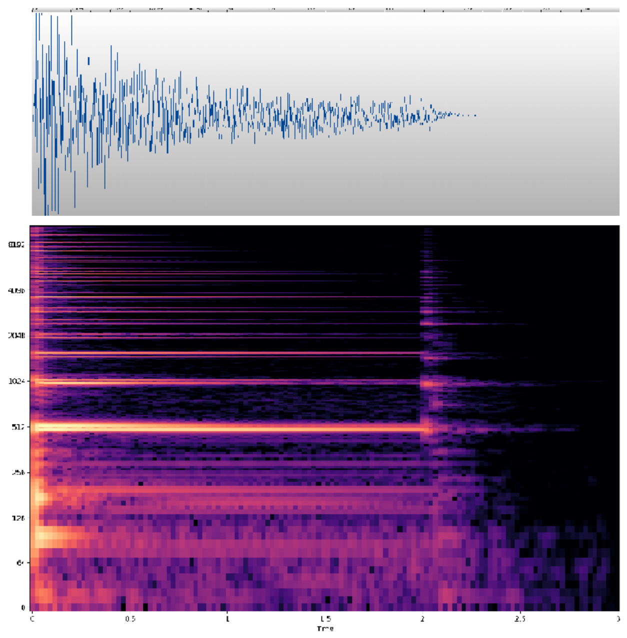

Figure 1. Visualization of the two kinds of audio representations we work with – (top) waveform (bottom) spectrogram

We even find that convex optimization methods can give decent results in this setup.

1.3 Our Contributions

We try out three different methods for this task. First, we frame the problem as a convex optimization one and pair the results from a convex solver with gradient descent to learn a generalized function that transforms a piano sound to a guitar sound. Since this does not produce strong results, we instead choose to find a closed form least-squares solution to use instead. Second, we design and train a neural network that uses the raw audio sampled at 44100 Hz as input and output for our network and learn this conversion. Lastly, we design and train a neural network that uses the Short-time Fourier Transform of each audio clip sampled at 22050 Hz as input and output of the network.

1.4 Organization

The rest of our paper is organized as follows. First we discuss the statement of our problem in Section 2. Then we talk about the dataset we use in Section 3. After that we present our methods in Section 4. Finally, we discuss the results of our experiments in Section 5.

2. STATEMENT OF THE PROBLEM

2.1 Problem setup

The problem that we solve is converting an audio clip **x** of a piano to an audio clip **b** of a guitar, where $\mathbf{x}, \mathbf{b} \in R^n$. To do this, we want to solve for a function **F**, such that $\mathbf{F(x)} = \mathbf{b}$. For our approach to solve this problem using convex optimization, we have **F** be defined as a matrix $\mathbf{A} \in R^{nxn}$. In our neural network approaches mentioned later, **F** is the neural network.

Given our dataset of 5000 paired piano and guitar audio clips, we choose to use 4000 to learn this matrix **A**, and use 500 each for validation and testing. In our initial attempt for which we describe the primal and dual formulations, we constrain **A** to be a diagonal matrix. The objective we optimize is the squared norm of the error of an individual audio clip. Since the convex solvers struggle with solving **AX = B** where **X** and **B** are matrices corresponding to the whole training set, we individually solve the optimization problem for each sample, giving us 4000 different **A** matrices, and then find a convex combination of these matrices **M** that minimizes the Mean Absolute Error of the validation set using gradient descent.

2.2 Primal formulation

$$min_{\mathbf{A}}||\mathbf{Ax} - \mathbf{b}||_2^2$$
$$s.t.||\mathbf{Ax}||_2^2 \leq ||\mathbf{b}||_2^2$$

where **A** is a square, diagonal matrix, and $\mathbf{A} \in \mathrm{R}^{nxn}$, **x**, $\mathbf{b} \in \mathrm{R}^n$

2.3 Dual formulation

Lagrangian:

$$L(\mathbf{A}, \lambda) = ||\mathbf{Ax} - \mathbf{b}||_2^2 + \lambda(||\mathbf{Ax}||_2^2 - ||\mathbf{b}||_2^2)$$
$$L(\mathbf{A}, \lambda) = ||\mathbf{Ax} - \mathbf{b}||_2^2 + \lambda||\mathbf{Ax}||_2^2 - \lambda||\mathbf{b}||_2^2$$

Rewrite first term:

$$L(\mathbf{A}, \lambda) = ||\mathbf{Ax}||_2^2 + ||\mathbf{b}||_2^2 - 2 < \mathbf{Ax}, \mathbf{b} > + \lambda||\mathbf{Ax}||_2^2 - \lambda||\mathbf{b}||_2^2$$
$$L(\mathbf{A}, \lambda) = (1 + \lambda)||\mathbf{Ax}||_2^2 + (1 - \lambda)||\mathbf{b}||_2^2 - 2 < \mathbf{Ax}, \mathbf{b} >$$
$$L(\mathbf{A}, \lambda) = (1+\lambda)(\mathbf{Ax})^T(\mathbf{Ax}) + (1-\lambda)(\mathbf{b}^T\mathbf{b}) - 2 < \mathbf{Ax}, \mathbf{b} >$$

Let **Ax** = $< [a_1 a_2 \dots . a_n], \mathbf{x} >= a_1x_1 + a_2x_2 + \dots a_nx_n$, where a_i is the nonzero element of the ith column vector of **A**:

$$L(\mathbf{A}, \lambda) = (1 + \lambda)(a_1x_1 + a_2x_2 + ...a_nx_n)^T$$
$$(a_1x_1 + a_2x_2 + ...a_nx_n)$$
$$+(1 - \lambda)(\mathbf{b}^T\mathbf{b}) - 2(a_1x_1b_1 + a_2x_2b_2 + ...a_nx_nb_n)$$

Gradient w.r.t. arbitrary a_i:

$$\frac{\partial L}{\partial a_i} = (1 + \lambda)2a_ix_i^2 - 2x_ib_i = 0$$

Solve for a_i:

$$a_i = \frac{b_i}{(1 + \lambda)x_i}$$

Plug back into Lagrangian:

$$g(\lambda) = (1 + \lambda)(\frac{b_1}{1 + \lambda} + \frac{b_2}{1 + \lambda} + ...\frac{b_n}{1 + \lambda})^T$$
$$(\frac{b_1}{1 + \lambda} + \frac{b_2}{1 + \lambda} + ...\frac{b_n}{1 + \lambda}) + (1 - \lambda)(b^Tb)$$
$$-2(\frac{b_1^2}{1 + \lambda} + \frac{b_2^2}{1 + \lambda} + ... + \frac{b_n^2}{1 + \lambda})$$

$$g(\lambda) = (1 + \lambda)(\sum_{i=1}^{n} \frac{b_i}{1 + \lambda})(\sum_{i=1}^{n} \frac{b_i}{1 + \lambda}) +$$
$$(1 - \lambda)(\sum_{i=1}^{n} b_i^2) - 2\sum_{i=1}^{n} \frac{b_i^2}{1 + \lambda}$$

$$g(\lambda) = (\frac{1}{1 + \lambda})(\sum_{i=1}^{n} b_i)^2$$
$$+(1 - \lambda)(\sum_{i=1}^{n} b_i^2) - \frac{2}{1 + \lambda}\sum_{i=1}^{n} b_i^2$$

$$g(\lambda) = (\frac{1}{1 + \lambda})(\sum_{i=1}^{n} b_i)^2 + \frac{-\lambda^2 - 1}{1 + \lambda}\sum_{i=1}^{n} b_i^2$$

Dual:

$$max_{\lambda} g(\lambda)$$
$$s.t.\ \lambda \geq 0$$

2.4 KKT Conditions

1.) $||\mathbf{Ax}||_2^2 \leq ||\mathbf{b}||_2^2$
2.) $\lambda \geq 0$
3.) $\lambda(||\mathbf{Ax}||_2^2 - ||\mathbf{b}||_2^2) = 0$
4.) $\frac{dL}{da_i} = (1 + \lambda)2a_ix_i^2 - 2x_ib_i = 0$

3. DATASET

To create our dataset consisting of piano and guitar samples, we first created an audio sample for each possible note that a guitar and piano can play, resulting in 45 audio clips that are 3 seconds long for each instrument. Then we used Pydub [5], an API for manipulating audio, to create audio samples of chords, a musical term meaning multiple notes play at once. Since a guitar can play up to 6 notes at once, we had chords ranging from having 2 notes playing at the same time up to 6, and had 1000 of each. In total, this left us with 5000 paired audio samples of piano and guitar.

	Error Metrics			
Models	Mean Absolute Error (MAE)	Mean Squared Error (MSE)	Human Sound	Human Pitch
ConvexOpt	849.25	47.07	1.85	2.65
WavNet	**620.33**	**32.30**	2.7	2.7
SpectroNet	6007.79	2261.22	**3**	**3**

Table 1. Performance of our three approaches on the test set. The best result of each metric is bolded.

3.1 Data representations

Audio samples can be represented in several different ways. Due to this, we decided to try out multiple different representations to see if there is any significant differences in the performance of our systems. In our experiments, we tried out two different ways to represent audio: waveform and spectrogram.

The waveform representation of audio is relatively straightforward as it is the format that audio is stored in lossless file types such as WAV, FLAC, etc. Essentially, these formats just store the amplitudes at each sample of the audio. In the case where the sample rate of the audio clip is 44.1 kHZ, it would store 44100 amplitudes for each second of an audio clip. For our experiments we used sample rates of 22050 and 44100 for our neural network models and 3000 for convex optimization methods due to the convex solver having trouble with extremely high dimensional matrices and vectors.

The spectrogram representation of audio is a 2-dimensional representation of sound in terms of frequencies and time, showing how the amplitudes of different frequencies change over the course of an audio clip, as shown in Section 1. To compute this, we use the discrete Short-time Fourier Transform as in Equation (1). The parameters we use for this are a sample rate of 22050 Hz, frame size of 2048, and hop size of 512. To reconstruct a waveform representation from the spectrogram, we use the Griffin-Lim algorithm.

$$\begin{aligned} spectrogram &= ||STFT(audio\_clip)|| \\ audio\_clip &= GL(spectrogram) \end{aligned} \tag{1}$$

4. METHODS FOR INSTRUMENT STYLE TRANSFER

4.1 Convex Optimization with Gradient Descent

Our initial approach using convex optimization was to solve the problem we mentioned in Section 2.2.

$$min_{\mathbf{A}} ||\mathbf{Ax} - \mathbf{b}||_2^2$$
$$s.t. ||\mathbf{Ax}||_2^2 \leq ||\mathbf{b}||_2^2$$

where $\mathbf{A}$ is a square, diagonal matrix, and $\mathbf{A} \in R^{nxn}$, $\mathbf{x}, \mathbf{b} \in R^n$

More specifically, we obtained a diagonal matrix $\mathbf{A}$ that optimized the problem above for each piano-guitar pair in the training set, a total of 4000. Let $\mathbf{x_v}$ and $\mathbf{b_v}$ be piano and guitar audio clips from the validation set. We then used gradient descent to find the weights $c_1, c_2, ..., c_n$ that minimized the square error between $\mathbf{Mx_v}$ and $\mathbf{b_v}$ for all validation samples, where $\mathbf{M} = w_1\mathbf{A_1} + w_2\mathbf{A_2} + ... + w_{4000}\mathbf{A_{4000}}$ and $\mathbf{w} = Softmax(\mathbf{c})$, thus $\sum_i w_i = 1$. In other words, M is a convex combination of the 4000 $\mathbf{A}$ matrices that were obtained from the training samples.

While this idea made sense to us intuitively, when we implemented it in practice, the results were very poor and the learned matrix $\mathbf{M}$ would just convert each piano audio clip to a very noisy audio clip completely different from the ground truth guitar audio clip.

4.2 Convex Optimization Closed Form

After the previous approach mentioned above gave us poor results, we instead attempt to learn an $\mathbf{A}$ matrix which minimizes the objective function stated in Equation (2)

$$||\mathbf{AX} - \mathbf{B}||_2^2 \tag{2}$$

where $\mathbf{A} \in R^{9000x9000}$, and $\mathbf{X}, \mathbf{B} \in R^{9000x4000}$. $\mathbf{A}$ corresponds to a transformation matrix, $\mathbf{X}$ corresponds to a matrix of the piano training set, and $\mathbf{B}$ corresponds to a matrix of the corresponding guitar training set.

The goal is to find an $\mathbf{A}$ matrix that when multiplied with the $\mathbf{X}$ matrix (the training set of piano audio) will result in a matrix that is the closest to the $\mathbf{B}$ matrix (the desired guitar audio).

Since the convex solver fails to solve this problem due to the very high dimensionality, we instead look for a closed form solution. Additionally, in contrast to the matrix mentioned in Section 4.1 which was a diagonal matrix, we let $\mathbf{A}$ be an arbitrary matrix in this case.

Once we have our matrix $\mathbf{A}$, we use it to transform our individual piano test audio clips into their corresponding predicted guitar audio clips.

4.3 Objective Function

In the next two subsections, we will discuss our neural network approaches to this problem. The objective function we chose to use for our neural models is Mean Absolute Error (MAE). For WavNet, we calculated it with respect to the waveforms. For SpectroNet, we calculated it with respect to the spectrograms.

4.4 Waveform Neural Network (WavNet)

We design two models using waveform as input to learn the instrument transformation. The first model has one linear layer as the instrument transformer. However, since

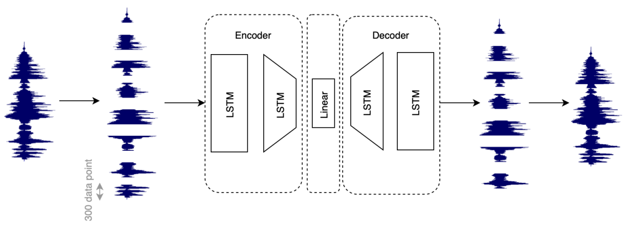

Figure 2. Architecture of Waveform Model

the size of parameters for a linear layer which uses original size waveform as input cannot fit into our computing memory, we downsample the waveform with 5000 as the new sample rate using API from torchaudio [6]. Although the model is simple and effective for the transformation, it requires approximately 2.6 GB space for saving one checkpoint; that is to say, significant amount of parameters is needed in this model.

To deal with this problem, for the second model, we add an encoder and a decoder to reduce the dimensionality of an audio file sampled at 44100 Hz before performing any transformations. Since the transformation of two instruments is similar to sequence-to-sequence translation in Natural Language Processing, we use the Long Short-term Memory [7] architecture to encode and decode sequences, which is well-suited to process time-series data. For the transformation layer, between a convolutional neural network and linear model, the linear model had a better performance in our case. Architecture of our final waveform model is shown in Fig. 2. Although the second model seems more complex than the first model, the size of its checkpoint is actually 30 times smaller than the first model.

4.5 Spectrogram Neural Network (SpectroNet)

For our spectrogram neural network, we took inspiration from the famous U-net architecture [8] that was used on images since a spectrogram is somewhat like a visual representation of audio. As an encoder, we use four convolutional blocks with pooling to downsample the spectrogram. Then we perform a transformation on the sampled spectrogram using four convolution layers. Finally we upsample to return to the original sized spectrogram. The architecture can be seen in Fig. 3.

In addition to learning to convert a piano clip to a guitar clip, we found that by adding a channel to the spectrogram input to the neural network, we could condition the model to either convert piano clips to guitar clips or guitar clips to piano clips, a simple way to have the model do both as opposed to needing to train two completely separate models to achieve the following.

5. EXPERIMENTS

5.1 Implementation Details

5.1.1 Convex Optimization Closed Form Approach

We use 4,000 sample pairs corresponding to our training set to solve for the $\mathbf{A}$ matrix. To load each audio clip with

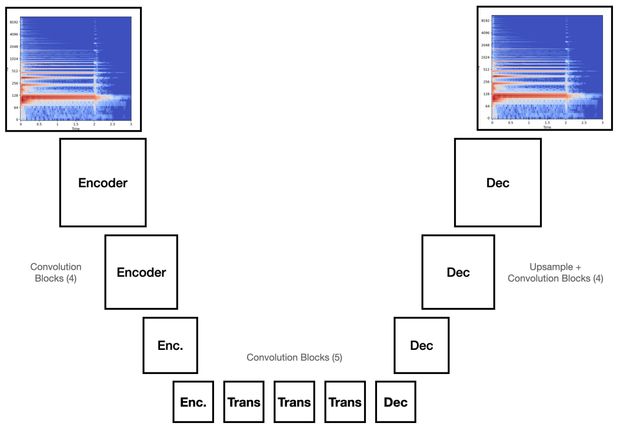

Figure 3. Architecture of Spectrogram Model

a sample rate of 3000, we use librosa.load [9] and arrange our 3 second piano clips into a 9000 by 4000 matrix, $\mathbf{X}$. Each column of matrix $\mathbf{X}$ contains 9000 feature values that are associated with a singular piano sample's WAV audio file. There are 4000 columns in matrix $\mathbf{X}$, because we have 4000 piano samples in our training set.

Then, we repeat the same process but with the 4000 guitar samples in the training set in order to create matrix $\mathbf{B}$. Similar to matrix $\mathbf{X}$, matrix $\mathbf{B}$ is a 9000 by 4000 matrix, where each column is associated with a singular guitar sample and contains the 9000 feature values associated with that guitar sample's WAV audio file.

We then want to solve Equation (2) to find the $\mathbf{A}$ matrix that minimizes the objective function L seen in Equation (3). To do this, we first take the derivative of the objective function with respect to $\mathbf{A}$. We then set the derivative equal to zero and solve for $\mathbf{A}$. We find that the optimal solution for $\mathbf{A}$ is the matrix $\mathbf{B}$ (guitar samples) multiplied by the pseudo inverse of $\mathbf{X}$ (piano samples). $\mathbf{A}$ will be a 9000 by 9000 matrix. For the notation below in Equation (3) and Equation (4), $\mathbf{x^k}$ refers to the kth column of a matrix $\mathbf{X}$, and $\mathbf{x_k}$ refers to the kth row of a matrix $\mathbf{X}$.

$$\begin{aligned} L &= \sum_{k}^{4000} ||\mathbf{A}\mathbf{x^k} - \mathbf{b^k}||_2^2 \\ &= \sum_{k}^{4000} (\mathbf{A}\mathbf{x^k} - \mathbf{b^k})^T(\mathbf{A}\mathbf{x^k} - \mathbf{b^k}) \\ &= \sum_{k}^{4000} \mathbf{x^{kT}}\mathbf{A^T}\mathbf{A}\mathbf{x^k} - \mathbf{x^{kT}}\mathbf{A^T}\mathbf{b^k} - \mathbf{b^{kT}}\mathbf{A}\mathbf{x^k} + \\ &\mathbf{b^{kT}}\mathbf{b^k} \\ &= \sum_{k}^{4000}(\sum_{i}^{9000} <\mathbf{a_i}, \mathbf{x^k}>^2 - 2\sum_{i}^{9000} <\mathbf{a_i}, \mathbf{x^k}> b_{ik} + \\ &\mathbf{b^{kT}}\mathbf{b^k}) \end{aligned} \tag{3}$$

	Chord Size (Number of Notes Playing Simultaneously) MAE				
Models	2	3	4	5	6
ConvexOpt	596.35	751.24	840.82	973.59	1048.19
WavNet	**365.80**	**480.77**	**623.51**	**721.76**	**871.65**
SpectroNet	3870.36	5180.48	6461.79	6553.31	7698.20

Table 2. Comparison of our three approaches evaluated on different chord sizes

$$
\begin{aligned}
\frac{\partial L}{\partial a_{ij}} &= \sum_k^{4000} (< \mathbf{a_i}, \mathbf{x^k} >^2 - 2 < \mathbf{a_i}, \mathbf{x^k} > b_{jk}) \\
&= \sum_k^{4000} (2a_{ij}x_{jk}(< \mathbf{a_i}, \mathbf{x^k} > - b_{jk})) \\
\frac{\partial L}{\partial a_{ij}} &= 0 \\
&= \sum_k^{4000} (2a_{ij}x_{jk}(< \mathbf{a_i}, \mathbf{x^k} > - b_{jk})) = 0 \\
&= \sum_k^{4000} (< \mathbf{a_i}, \mathbf{x^k} > - b_{jk}) = 0 \\
&= \mathbf{a_i X} = \mathbf{b_j} \\
&= \mathbf{a_i} = \mathbf{b_j X}^\dagger
\end{aligned} \tag{4}
$$

This gives us the following matrix $\mathbf{A}$ that minimizes the expression $\sum_k^{4000} ||\mathbf{Ax^k} - \mathbf{b^k}||_2^2$.

$$
A = \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_{9000} \end{bmatrix} = \begin{bmatrix} b_1 X^\dagger \\ b_2 X^\dagger \\ \cdot \\ \cdot \\ \cdot \\ b_{9000} X^\dagger \end{bmatrix} = BX^\dagger
$$

Now that we have the $\mathbf{A}$ matrix, we are able to generate predicted guitar samples. To do this, we can just put a single piano sample into a 9000 by 1 vector by making use of librosa.load with a sample rate of 3000. We then multiply $\mathbf{A}$ and that vector together in order to get the predicted guitar sample. Finally we use the scipy.io.wavfile library [10] in order to create the WAV file of the predicted guitar sample.

5.1.2 Waveform Neural Network Approach

Since we used torchaudio to load our wave file, we chose the default sample rate, 44100Hz, as the sample rate. As it has shown in Fig. 2, we group 300 data points as one time stamp in the input sequence and the input size of our 3 sec file would be 441 for the first LSTM layer. We chose 441 and 256 as the output size for the two output layers respectively. For the linear transformation layer, the input and output size are both 256. The model is trained with 10^{-4} as the learning rate and updated parameters with the Adam optimization algorithm [11].

5.1.3 Spectrogram Neural Network Approach

We first used librosa to perform the Short-time Fourier Transform on each input audio file. Full details on parameters used in the convolution blocks can be found in our code. In terms of hyperparameters, we used a batch size of 10, and a learning rate of 10^{-4} with Adam optimization.

5.2 Experiment Setup

To evaluate our different methods for solving the problem of converting a piano audio clip to a guitar audio clip, we conduct both a quantitative evaluation and a qualitative evaluation. For the quantitative test, we measure the mean absolute error and mean squared error between the converted piano audio clips resampled to a standard 22050 Hz and the ground truth guitar audio clips. For the qualitative test, each of us went through a sample of different chord size (ranging from 2-6) and evaluated the sound (how guitar-like our prediction is) and the pitch (how similar the pitches sound to the target). We rank each sample for each of these metrics on a scale of 1-3 with 1 being the worst, and 3 being the best, and get the average scores across each model.

5.3 Experiment Results

Our results are summarized in Table 1 and Table 2. We found that overall, quantitatively, the waveform-based neural network was the best in our experiments. However, qualitatively, the spectrogram-based neural network was the best. To listen to some example results from our different models, please check the README file at this link: `https://github.com/HeidiCheng/wi22-203-FinalProject/blob/main/README.md`.

5.3.1 Convex Optimization Approach (ConvexOpt)

As can be seen from Table 1, out of the approaches we compared, the convex optimization approach resulted in the second lowest mean absolute error, 849.25. It performed only slightly worse than the WavNet approach in terms of mean absolute error. However, when looking at the results from the qualitative test, we can see that the convex optimization approach had the worst performance out of the three approaches compared. Compared to the other two approaches, the convex optimization approach led to converted piano samples that sounded the least guitar-like. That being said, the convex optimization approach did produce converted piano samples that were decently accurate in terms of pitch.

We believe that one reason that the convex optimization approach may have performed the worst out of the three approaches in the qualitative test could be due to the fact that we sample the input training piano and guitar samples at a sample rate of 3000, and then later resample the converted piano sample to 22050 Hz. This could potentially make the predicted guitar sample sound stranger to the human ear, thus causing it to perform worse in the human evaluation.

Additionally, an observation we notice is that at larger sample rates, the pseudo inverse convex optimization solution does not work as well as it does at lower sample rates. The MAE seems to increase when using higher sample rates. Therefore we chose to use a sample rate of 3000 for loading in our samples, because it allows us to obtain decent converted piano samples and returns better MAE results than what we get when we use higher sample rates.

5.3.2 Waveform Neural Network Approach (WavNet)

For all quantitative tests, MAE, MSE, and MAE of different chord sizes, WavNet performs the best among the three approaches. We suggest the model might be able to learn more details of the audio by using raw audio. However, based on human judgement, SpectroNet outperforms WavNet. The pitch and sound of WavNet's transformation are relatively accurate, but the outputs have some noise especially on chords which contain high notes. We believe the reason for this is the same as what is discussed in Jukebox [12]; the reconstruction of mid-to-high frequencies is hard for the model to learn using only raw data.

5.3.3 Spectrogram Neural Network Approach (SpectroNet)

As can be seen in Section 2.3 above, the MAE of SpectroNet was significantly higher than all of the approaches, at 6007.79. Based on human evaluation, we all agreed that the outputs (converted piano clips) produced by this were quite accurate to the human ear both in terms of sound and pitch. We suspect this high MAE is due to the fact that this model works with spectrograms, and the reconstruction of an audio clip from a spectrogram is not perfect largely due to the lack of phase information.

6. CONCLUSION

In this work, we first introduced a paired dataset of piano and guitar audio clips. Then we proposed three different approaches to solve the problem of converting a piano audio clip to a guitar audio clip: convex optimization, a waveform based neural network, and a spectrogram based neural network. Lastly we conducted experiments using these three approaches, and used both quantitative and qualitative metrics to analyze their effectiveness.

In the future, it would be interesting to develop a one-to-many instrument transfer model that could provide more features to a user. Additionally, it would be useful to use data augmentation so that our approach could better generalize to arbitrary piano and guitar sounds.

6.1 Individual Contribution

All four of us worked together to write this report. **Sachinda** helped with the dataset creation and worked on the spectrogram neural network using PyTorch. **Heidi** worked on the dataset creation and built the waveform neural network using PyTorch. **Payal** worked on the convex optimization approach. **Ahmed** worked on the primal and dual formulation as well as the KKT conditions.

7. REFERENCES

[1] C.-Y. Lu, M.-X. Xue, C.-C. Chang, C.-R. Lee, and L. Su, "Play as you like: Timbre-enhanced multi-modal music style transfer," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1061–1068.

[2] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer," *arXiv preprint arXiv:1811.09620*, 2018.

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[4] A. Bitton, P. Esling, and A. Chemla-Romeu-Santos, "Modulated variational auto-encoders for many-to-many musical timbre transfer," *arXiv preprint arXiv:1810.00222*, 2018.

[5] "Pydub," accessed: 2022-03-12. [Online]. Available: https://pypi.org/project/pydub/

[6] "Torchaudio," accessed: 2022-03-12. [Online]. Available: https://pytorch.org/audio/stable/torchaudio.html

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[9] "librosa," accessed: 2022-03-12. [Online]. Available: https://librosa.org/doc/main/generated/librosa.load.html

[10] "Scipy.io.wavfile.write." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.write.html

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341v1*, 2020.