Strings2Scores: Automatic Music Transcription for the Violin

Jason Castillo
dept. Electrical and Computer Engineering
UT Austin
Austin, TX
jzc248

Carie Navio
dept. Electrical and Computer Engineering
UT Austin
Austin, TX
cen847

***Abstract*—This paper discusses the design and implementation of a machine learning model that uses real violin recordings to generate the corresponding sheet music. The architecture is a Sequence-To-Sequence (Seq2Seq) family type with a Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) model. Constrained data was used to reduce variance between data samples to simplify the model. The result was simple sheet music generated from audio input.**

***Index Terms*—Music transcription, deep learning, end-to-end systems, seq2seq, RNN, LTSM**

I. Introduction

The creation of modern-day sheet music began in the 15th century, after the use of printing presses was more commonplace. This process consisted of passing a sheet of paper through the printing press 2-3 times to get the final score. It was incredibly tedious work since each notation needed to be printed crisply in order to differentiate between itself and neighboring notes. Modern violin sheet music transcription allows for professional music notation but the process contains manual dependencies. Services still exist for musicians to pay for other musicians to manually transcribe their audio files [1].

Digital synthesizers emerged in the 1980s which allowed for straightforward techniques when converting from audio input to sheet music. The synthesizers mimic notes that instruments play. In this time, there was also the development of the Music Instrument Digital Interface (MIDI) music protocol. MIDI allowed for digital instruments such as the keyboard, to automatically create sheet music. While this protocol allowed for straightforward conversion of digital interfaces, analog instruments such as strings and woodwinds, cannot use this tool without extra preparation. Thus, the creation of a machine learning model was designed and implemented which when inputted a waveform recording of a violin, outputs sheet music.

II. Background

Current machine learning models are typically based for keyboard due to the number of readily available samples as well as the simplicity in capturing waveform features from the instrument. A keyboard generates sound in the following way: when a key is pressed on the keyboard, a circuit is closed, and the digital equivalent of that note is outputted through the speaker. In contrast, analog string instruments have a few major components which impact the sound: method of sound generation (i.e. pick, bow), strings, fingerboard, bridge.

The control that violin offers the user in comparison to piano or another fretted instrument like the guitar, means that the regularity of hitting the same note twice is never guaranteed. In fact, Jascha Heifetz, who is regarded as one of the best violinists of all time, when asked how he played each note in tune, famously said ”I don’t. But I adjust the note before you notice it was out of tune in the first place” [2]. The user is also responsible for tuning the violin strings. In group settings such as chamber, orchestras, or symphonies, all string instruments must tune to ”concert A” (440 Hz). However, in pieces which have no accompaniment, the user tunes to a ”concert A” with a variance of frequency from $\pm$ 2 Hz. This means that a note may not always correlate to a specific frequency and thus the design will have to compensate for a band of frequencies to correlate to a note.

A. Data Constraints

Due to the various techniques used on the violin, in order to simplify the model, certain aspects the data needed to be omitted or clearly defined prior to data collection: vibrato, slurring, pizzicato, harmonics, and

bow position. All of these techniques add more unique symbols to music engraving. To simplify this project and to get a proof of concept, many of the techniques were either omitted or simplified. However, choosing to omit or simplify these techniques reduced the ability to pull samples externally.

B. Vibrato

Vibrato is the technique of pulsing the finger pressed on the fingerboard in order to get an oscillating tone that creates an enriched sound. While it creates depth in music, the oscillation aspect of vibrato adds a layer of complexity because vibrato technique is specific to a person.

C. Slurring

Slurring is the technique of using the same bow stroke to play different notes. While slurring traditionally uses different notes, the variance of the amplitude for the frequency generated when slurring adds a level of complexity since the first note played within a slur is strongest.

D. Pizzicato

Pizzicato is the technique of plucking the string – usually with the right hand but can also be done with the left. It creates a sound akin to a guitar and would require training on pure pizzicato notes to ensure that the model can distinguish between bow and plucking.

E. Harmonics

Harmonics is the technique of lightly placing a finger on the fingerboard to generate a sound that resonates on both sides of the string rather than from the finger to the bridge of the violin. There are two different types of harmonics on the violin and each type requires a different technique as well as notation [3]. Natural harmonics are created on the open string and can produce the following: one octave, two octaves, one octave and a fifth above the open string, and two octaves and a third above the open string. Artificial harmonics are created by pressing the index finger (first finger) firmly on the fingerboard and the pinky finger (fourth finger) lightly to produce the note that is two octaves higher than the index finger.

F. Bow Position

The position of the bow impacts the sound generated and the resulting waveform. Bow positioning closer to the tip creates a smaller waveform while box positioning closer to the frog (where the hand holds the bow) creates a much larger waveform with generally more noise.

III. Pulling Data

The data omitted the use of vibrato, slurring, pizzicato, and harmonics. All bow strokes were generated at the middle of the bow – approximately between 1/4 and 3/4 of the length of the bow. Given the imposed constraints for data collection, there existed no large dataset from which to pull violin recordings and sheet music from. Each note was played at a tempo of 100 BPM with a single bow stroke for each note.

Audio files consisted of the following 2-3 octave major scales: A, B♭, B, C, C♯, D♭, D, E, F, F♯, G♭, G, and A♭. The result was 34 files. Recording varied from device to device but each contributor recorded the audio as a Waveform Audio File Format (WAV) file with 44100 Hz sampling frequency, monoaural sound, and 32-bit float recording range.

The sheet music was generated using LilyPond, a music engraving program. It uses simple text notation as the input and outputs the corresponding music score via PDF, SVG, or PNG. LilyPond does not have a graphical interface so it requires a text editor to verify the music generated. Figure 1 shows a version of "*Twinkle Twinkle Little Star*" using LilyPond music engraving. Frescobaldi is a text editor and was used in conjunction to LilyPond to write the sheet music as well as test the output of music generated.

```
\score {
  \new Staff {
    \key d \major

    d'4 d' a' a' | b' b' a'2 |
    g'4 g' fis' fis' | e' e' d'2 |
    a'4 a' g' g' | fis' fis' e'2 |
    a'4 a' g' g' | fis' fis' e'2 |
    d'4 d' a' a' | b' b' a'2 |
    g'4 g' fis' fis' | e' e' d'2 |
  }
}
```

Fig. 1: Example of "Twinkle Twinkle Little Star" using LilyPond in Frescobaldi

Time signature is the notation used to denote the note value in each measure – a measure is denoted by a bar. *Common Time* or **C** is the time notation of 4/4. 4/4 time is the default time signature for LilyPond and was used in this project to reduce memory for the label inputs. 4/4 time means that 4 quarter notes create one measure, with each quarter note equal to 100 BPM. Since musical note

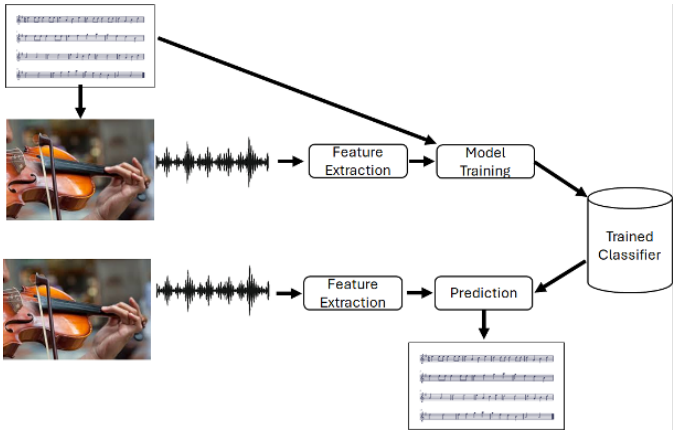

Fig. 2: Simplified model design

time can vary heavily, the model was limited to eighth, quarter, and half notes.

The key signature, pitch, and note duration were captured in the program. LilyPond offers a relative pitch technique which chooses the octave of the note relative to the previous note. Since this relied on previous information, the text generation used a stand alone pitch model.

IV. Methodology

A. Overall Architecture

The model utilizes a Sequence-To-Sequence (Seq2seq) approach with a recurrent neural network (RNN), specifically Long Short-Term Memory (LSTM). Seq2Seq was chosen due to the parameters of the model: given the sequence of notes in an audio file, generate the sequence of notes on a musical staff. Using an RNN allows to utilize information from a previous section to predict the model. In the example of speech recognition, RNNs take into account the words, or tokens, near a words to determine what the phrase will mean. LTSM was chosen as the RNN in order to track time and frequency dependent information. The LSTM model also reduces vanishing and exploding gradient, which occurs with RRNs. Figure 2 is a graphical representation of the model design.

B. Audio Extraction

Librosa was used to extract different data from the WAV files. Multiple features were pulled while determining design: Mel-Frequency Cepstral Coefficients (MFCCs), Constant-Q Transforms, Melspectrograms, and Chromagrams. After testing each feature, the melspectrogram was chosen because it most represents sheet music. Melspectrogram is a series of Fast-Fourier Transforms (FFTs) over sections of time, in which the waveform is reshaped from a time domain to a frequency domain graph as seen in Figure 3. The frequency is then converted from Hertz to Mel using the equation below:

$$mel = \frac{1}{\log(2)} \times \log(1 + \frac{Hz}{1000}) \times 1000$$

Mel scale represents frequency in relation to the human ear. Since notes are pitches at certain frequencies, the output was a close relation to music transcription [7]. Two dimensional array created was with padded to the length equal to max length of all WAVs captured.

C. Music Score Extraction

The LilyPond markup was stripped of formatting information as can be seen in Figure 1. The result was a two-dimensional array with one dimension for the vocabulary and the other padded to a max size for length of a sheet music score. Since music scores have finite options for notation, a predefined vocabulary was used to compare and capture note attributes.

D. One-Hot

The attributes captured from IV-C, were in the form of One-Hot encoding. One-Hot encoding is a common method for dealing with categorical data in machine learning. It converts data from categorical information to more easily read machine learning data – ones and zeros – to improve prediction accuracy. The inverse, One-Hot Decoder, was created and used to generate the output of the model prediction into LilyPond-readable syntax.

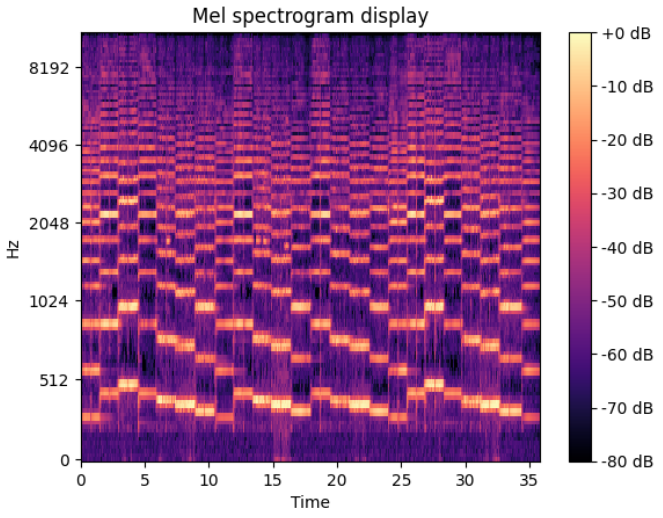

Fig. 3: Melspectrogram Graph for Twinkle Twinkle Little Star

V. Conclusion

A. Final Testing

Many variations of the design were implemented, including but not limited to: epoch value, feature type, compile loss type, compile optimizer, vocabulary choice, model design, LSTM unit size, Tensorflow, and GPU utilization. The best-case solution involved creating a Tensorflow dataset of the generated data, loss type set to ”categorical crossentropy”, optimizer set to ”adam”, and setting the unit size of the LSTM model to 248.

B. Optimization

Initially, the model underwent training iterations over the entire dataset for one epoch at a time, resulting in jagged graphs and failure to converge. Transitioning to TensorFlow's fit method enabled training on the entire dataset for multiple epochs in each iteration, thereby enhancing training efficiency and streamlining the process. Subsequent optimization strategies include:

- **GPU Computation:** This was done to allow the training to be completed faster. Sine not all options are available for GPU computation, if Tensorflow in unable to use the GPU, it would revert back to CPU for those cases.
- **Multi-layer LSTM:** Up to three layers of LSTM were added yet there was no improvement. In actuality, the model performed better as a single layer. Though the impacts were minimal, adding a dropout of 0.25 between layers improved performance.
- **Data Augmentation:** Copied data to increase volume and then applied transformations to the wave forms. Used a pitched shift for one transformation and a time shift for another set. The shifts were random and set to a certain tolerance that won't compromise the sample. We saw some better accuracy using this data than just the limited data we have.
- **Go Backwards:** This is a function that LSTM offers to where it performs the sequence in backwards and returns the sequence. This had the greatest contribution to the training models accuracy by far.
- **Batch Size:** Decreased batch size and saw an increase of accuracy but greatly decreased training efficiency. Through manual checks, the optimal size was determined to be 8.
- **LSTM Units:** These units are the positive integer, dimensionality of the output space. In general, increasing this number improved the model accuracy but can lead to over-fitting if too high.

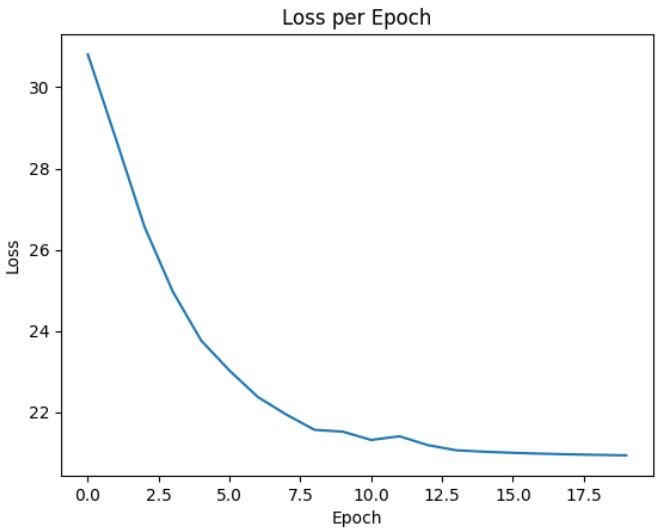

Fig. 4: Best Training Model After Optimizations

C. Results

Though the model was able to accurately predict certain audio files if it was in the trained dataset, it was not able to predict the output of an untrained WAV file. The non-scale file was removed from the training dataset in order to remove another complexity: time between quarter and half-note. Though this had a positive impact on trend of the Loss per Epoch, the results were still incorrect and inconsistent. It is speculated that the lack of varied training data greatly impacted the results.

Figure 4 is the result of the best training model generated after including the optimizations.

D. Future Work

It would be beneficial to expand the size of data being trained to properly test the model. Subsequent work would be to incrementally remove the constraints discussed in Section II-A implement a more complex model.

Acknowledgment

Special thanks to Jamie Spectarr and Olivia Mitchell for their violin recording contributions!

References

[1] “Violin Transcription Service: My Sheet Music transcriptions” My Sheet Music Transcriptions. (2024, February 16). https://www.mysheetmusictranscriptions.com/violin-transcription-service/

[2] “Intonation the “Impossibility” of Playing in Tune” https://www.simonfischeronline.com/uploads/5/7/7/9/57796211/205_july_intonation.pdf

[3] “String Harmonics” https://montgomeryphilharmonic.org/index.php/members/violin-harmonics/
[4] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz- Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[5] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[6] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[7] D Byrd, “A Table of Musical Pitches” https://homes.luddy.indiana.edu/donbyrd/Teach/MusicalPitchesTable.htm
[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[9] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[10] R. G. C. Carvalho and P. Smaragdis, ”Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score,” 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 2017, pp. 151-155, doi: 10.1109/WASPAA.2017.8170013.
[11] Jason Castillo and Carie Navio. *Strings2Scores: Automatic Music Transcription for the Violin*. [Online]. Available: https://github.com/conejoking/Strings2Scores