Piano Music Transcription and the Realistic Expectations of Artificial Intelligence Research

Ronald Rahenkamp **Nathaniel Nguyen**

Abstract

In this paper, we discuss our accomplishments and limitations in developing a system to transcribe music from an audio file to a readable sheet music format. We aimed to utilize a convolutional neural network to complete this task, but did not progress to the point of fully implementing this system. Instead, we have gleaned experience working with music transcription in the field of AI and also with setting realistic standards for what one can achieve with AI in a limited amount of time.

1. Introduction

Piano music transcription is a task well within the capabilities of a machine learning system, specifically a CNN, or convolutional neural network, being well equipped to handle this sort of task. Unfortunately, this is also a task in which very little is available for user consumption without heavy cost. Data sets are hard to come by, there are no existing open-source models, and even the closed source models are tight-lipped on the amount of information they let the public utilize.

When we set out to complete this project, we were ambitious, a bit too ambitious. We wanted to replicate the work of Klang.io, which is a current, market-level transcription software that utilizes AI to turn audio files into readable sheet music. What we did not realize, however, was that this was a PhD level project that had been undertaken at the Karlsruhe Institute of Technology. As we progressed, this task became daunting, as two undergraduate students with zero funding for a project were trying to recreate doctoral work. Still, we did our best.

In addition to replicating Klang.io, we also set out to add another layer into the mix. We intended to create a system able to recognize patterns in the provided musical works. Unfortunately, this did have to be cut before we reached this stage in our research.

Throughout this research, we have gained a thorough understanding of the limits and difficulties of working with AI, even when it pertains to projects that have been completed in similar fashions before. By exploring the limitations of what we could accomplish while working toward our goal of music transcription, we have made observations about music transcription as well as having realistic expectations when working with artificial intelligence.

2. Related Works

Primarily, our model is based on the aforementioned Klang.io, a web-based music transcription software that claims to use AI when converting the given audio files into sheet music PDF's. Klang.io can take a provided YouTube link or audio file. Our goal was to create a system able to take an Mp3 file as input, but we aimed to have a higher accuracy than Klang.io.

More related was found in the research of Sebastian Murgul and Michael Heizman, who together created a set of research papers concerning the transcription of guitar strumming, singing transcription, and predictive performance.

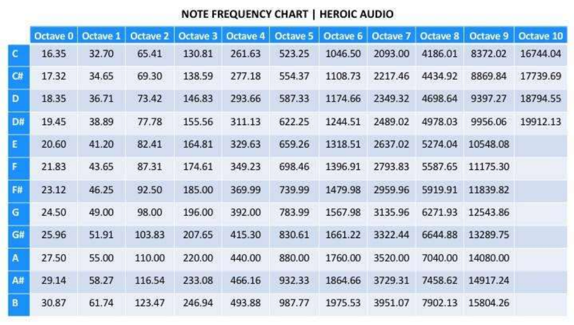
Murgul and Heizman's research into guitar strumming focused not only on the notes of the music, but also on the rhythm and strumming directions utilized by the player. They created a setup where the guitarist would have a device on their wrist that would track their movement. While we did not end up following this line of thinking in our transcription of piano music. A function capable of tracking the location of the key pressed would be useful but would remove some of the proposed functionality of our project, which was aiming to track an existing file rather than a concurrently played set of notes.

Another written piece of research we used while working with the design of our program was a published article by Ian VonSeggern, who wrote a transcription piece that was meant to work with audibly sung notes. The system, coded almost entirely in python, was said to function to output the musical note name of whatever input it received. This system, however, did not have replicable results, as it is one of many directions we attended to take our research in to fully integrate our transcription.

3. Data

In transcribing music, the first vital thing to note is the differences between different notes. Each note in music has its own frequency as well as sub frequencies within it. The difficulty here is separating out the sub frequencies to find the actual note played.

To begin our data, we had to research the frequencies to different notes in octaves for music so that we could utilize these numbers to separate out one note from the next in our system. Figure one outlines the difference in frequencies when working with musical note differentiation. While the specific units of our transcription were written in a different format, the same translations of frequency to note still stand.



NOTE FREQUENCY CHART | HEROIC AUDIO

	Octave 0	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5	Octave 6	Octave 7	Octave 8	Octave 9	Octave 10
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	16744.04
C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92	8869.84	17739.69
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64	9397.27	18794.55
D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03	9956.06	19912.13
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04	10548.08	
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65	11175.30	
F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91	11839.82	
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93	12543.86	
G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88	13289.75	
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00	14080.00	
A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62	14917.24	
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13	15804.26	

Figure 1. Chart of note frequencies, by Huart.

As for the final data we utilized in this project, we downloaded audio files of a piano piece, which we labeled as sample.wav and c5.wav respectively. These audio files are what we finally decided to use in our experimentation to produce the results we will soon discuss.

4. Methods

Our methodology is where problems began to arise in our research. To start, we attempted to use a method similar to VonSeggern's research. This approach sought to utilize pydub to plot the frequency over time of a system. This approach does not directly use AI itself, but we would have incorporated it into the later portions of our project. Unfortunately, we could not replicate the results of VonSeggern's research by utilizing pydub. We spent weeks attempting to alter the code or utilize different systems to achieve the same results, but regardless of what we shifted or changed, pydub would not work for us. Even when using the same sample track as VonSeggern's own program, we could not achieve the same results, let alone using it for other pieces.

With one unsuccessful attempt under our belts, we moved on to a different attempt: breaking apart the file structure of audio files in order to read the different frequencies over time. To begin this approach, we studied the makeup of an Mp3 file. In an Mp3 file, the frequency data is stored in the 21st and 22nd bits, meaning that if the correct conversion is utilized, one should theoretically be able to strip the frequency data from the audio file without too much difficulty. The problem with this approach became interpreting the frequency data.

While in our second approach, we found the location of the frequency data, it became a challenge to strip any meaningful information from these files, as even a second of data in an Mp3 file can contain more captures of audio data than we were able to process in our project. It became impossible for us to strip frequency data in a way that allowed us to consistently grasp the notes played.

After working with Mp3 files in this way, we then moved onto WAV files. While this did not immediately produce any great successes, it did lead us down the path to finding our eventual final method.

Our final, and best, approach came down to the numpy and librosa libraries of python. The way this worked is by utilizing librosa to read the files, we could numpy analyze the frequency data to produce a note and octave from our research.

Half Steps are 440 Hz (A4):

$$\text{half\_steps\_from\_A4} = 12 * \log_2(\text{frequency}/440)$$

$$\text{note\_index} = \text{half\_steps\_from\_A4} \ \% \ 12 - 3$$

$$\text{octave} = (\text{half\_steps\_from\_A4} + 9)/ \ 12 + 4$$

Figure 2: Formulas for frequency calculation.

The above formulas were utilized to pick out a note name from the bank of C, C#, D, D#, E, F, F#, G, G#, A, A#, and B. From there, the octave would be assigned by determining how many octaves away from A4 the actual octave of the note played was. Our final system utilized a function with both of these in order to return the note name and octave at every second or half second depending on the instance of experimentation. This final successful method of taking the names utilized WAV files instead of the intended Mp3, but it was successfully pulling audio data and assigning note names, even if we did not end up able to incorporate the other intended facets of our project.

5. Experiments

The content of our intended experiment changed drastically between when we began our project. At first, our intent was to recapture the sheet music and have a trained musician measure the accuracy of said music. Unfortunately, since the physical sheet music portion of our project was cut before we reached the point of experimentation, we were forced to change directions.

Instead of sheet music, we worked with two different pieces of recorded piano music and input them into our system. We then put each one through our system, one as a baseline to test the functionality of our system, and the other to provide a real dataset for us to compare our results against.

When putting them through the experimentation process, we changed how often each piece was sampled, alternating between one-second-long samples and half second long samples.

First, we attempted our first downloaded WAV files, a file that, at the time we downloaded it, was named c6.wav.

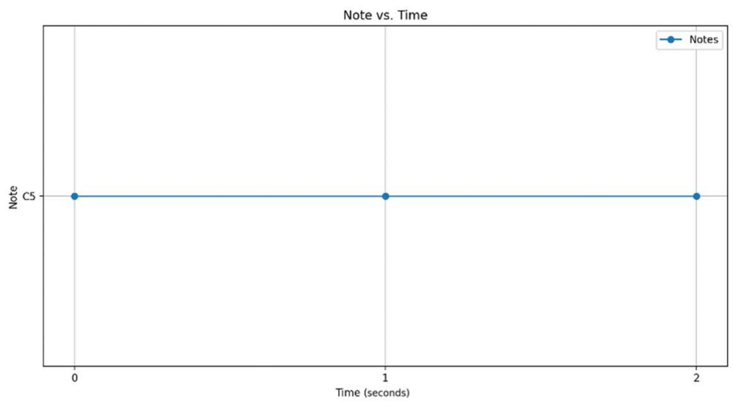


Figure 3. Two-second-long note sample of the named c6 file under the second long sample period.

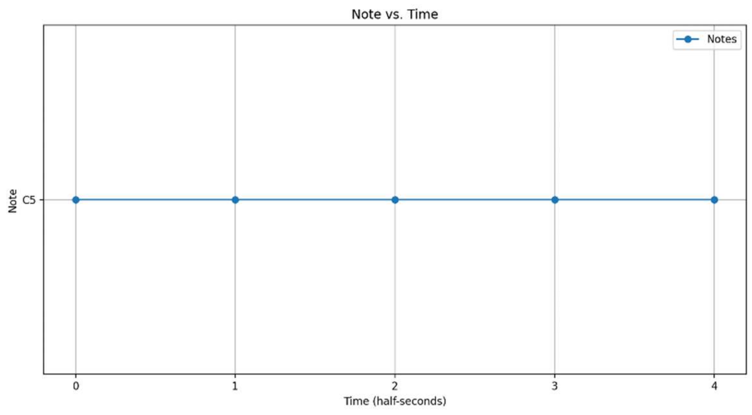


Figure 4. Two-second-long note sample of the named c6 file under the half-second long sample period.

The problem immediately seen is that the c6 file was labeled as a c5. This, however, was not an incorrect attempt by our system, rather an incorrect labeling of the file we had. When downloaded, the file was called c6, but when compared to a c5 note from a raw source, it matched the pitch of the sample we had. Therefore, we changed the name of the c5 sample, confident that both our note naming and our octave recognition were correct.

From here, we utilized another file, this one named sample.wav. Sample.wav was a more complex piece, but like the c5.wav files, we put it in for both the second long and half-second long portions. While this did not change much in the c5.wav files, the differences became more apparent in the sample.wav file.

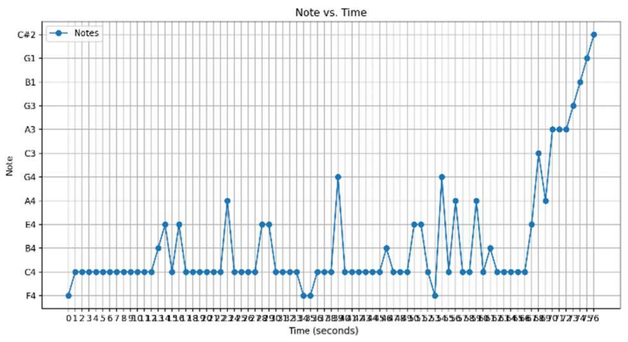


Figure 5. Sample.wav file over the course of 76 seconds, notes marked on the graph over time.

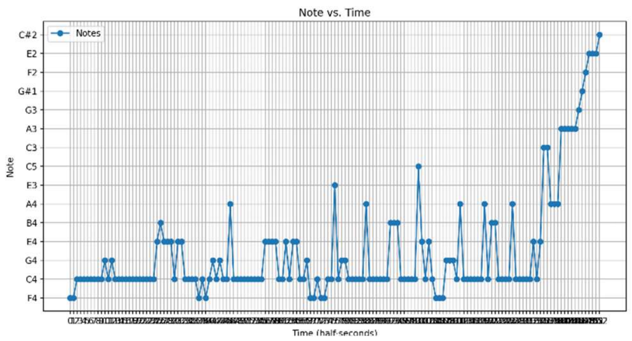


Figure 6. Sample.wav file over the course of 152 half-seconds, notes marked on the graph over time.

Certain key details in the comparison between these two graphs illustrate certain qualities of musical transcription. First, there are more transitions in the half-second variation of the graph, shown in figure 6. Specifically at second 34 in the second graph, shown in figure 5, there is a difference in which the graph in figure five captures the F4 note twice in a row in seconds 34 and 35, whereas the graph in figure six captures a C4 in between seconds 34 and 35, indicating that there was a change in tone, just one that was quick or insignificant enough that the graph in figure five, which only captured once every second, would miss it.

A similar effect occurs at second 17-21, where the F4 note is not visited at all in that range on the graph in figure 5, yet is touched twice in that same time span in the graph in figure 6.

One other notable difference in the graphs is the notes that seem to appear and disappear between the two graphs. Notes that have appeared are tones that appeared in the piece, yet did not linger long enough for the graph in figure 5 to capture. These notes, such as G4, E4, C5, E3, F2, and E2, are not present at all during the span of the graph in figure 5. They seem to come out of nowhere, but it's just representative of a higher accuracy due to the higher capture rate of the graph in figure 6.

Disappearing notes are far more interesting. Given that they way our program captured notes was to capture the average frequency of the audio at the time, notes that disappeared or changed are notes that were not truly in the piece as individual notes but created through the chordic structure of the music. This chordic structure is what caused the graph in figure 5 to note the presence of a B1 when a B1 was never truly played in the music. The graph in figure 6, at the same point, reports a G#1, which is also not an actual note present in the music, instead, a different average of frequencies to come from the same point.

6. Conclusion

The results of our research into piano transcription highlight the importance of an accurate sampling rate in music transcription, as an improper sampling rate will result in incorrect or muddied averages in the research data.

The more prominent conclusion we came to, however, is the necessity of proper establishment of realistic expectations when it comes to research in AI sources. When first taking on this project, researching into music transcription was simply an interesting idea to try chasing. The difficulty level of the research we were delving into came as a shock once we were already halfway into the project, learning that this was the equivalent of a four-year long PhD project taken on and still being refined by students in Germany. Despite this, we did our best to accomplish the task set before us.

Further research would obviously strive to finish what we began in this class. If we were to continue, we would begin structuring our AI system to intake the note data and transcribe it into actual music format. With just a semester, however, the lofty goals we set to accomplish were rendered unrealistic.

7. Supplementary Materials

Video Link:

https://youtu.be/L3hXa8OxZ3Y

The following materials can be found on our project repository, https://github.com/csce585-mlsystems/Melomusic.git:

- Final Melomusic Source Code
- Sample files used in experimentation
- Presentation slides
- ReadMe for the project
- Project proposal

References

"AI Software Tools for Transcribing Music into Notes." *Klangio*, 6 Dec. 2024, klang.io/.

Schwabe, Markus. "Startpage." *KIT*, www.iiit.kit.edu/english/3896.php. Accessed 12 Dec. 2024.

VonSeggern, Ian. "Note Recognition in Python." *Medium*, Medium, 6 June 2020, medium.com/@ianvonseggern/note-recognition-in-python-c2020d0dae24.