AUTOMATED MUSIC TRANSCRIPTION

R. Faran, Y. Wu, S. Xu

ECE 208/408 - Machine Learning, Spring 2025, University of Rochester

ABSTRACT

Audio transcription as a field has been important since audio first began being written down and shared. In music, the ability to write down music and share amongst musicians has been important for the prevalence and spread of individual songs or pieces. Historically, this process was done by hand and by trained individuals, but the need for automated transcription has become more and more prevalent in recent decades.

To address the increasing needs for transcription, we have developed a machine learning model that uses a Convolution Neural Network which accepts a .wav music file input and converts it to a MIDI output, mapping the note values and timing constraints based on the STFT and chroma features we extracted from our input.

1. INTRODUCTION

Audio transcription as a field has been important since audio first began being written down and shared. In music, the ability to write down music and share amongst musicians has been important for the prevalence and spread of individual songs or pieces. Historically, this process was done by hand and by trained individuals, but the need for automated transcription has become more and more prevalent in recent decades.

Musical notation has been around in one form or another as far back as 1400 BCE in Babylonian cuneiform, but the term "automated music transcription" was not first researched and published until the 1970's, when two papers were released in the November 1977 issue of the Computer Music Journal in separate papers by Moorer (Stanford) [1] and Piszczalski and Galler (University of Michigan) [2].

One of the main areas of exploration in regards to audio transcription is polyphonic materials, which we tackle in our approach. This became the main focus of our model in part because of the robust nature of the curated dataset.

This early work began by approaching the problem through the use of band-pass filters on restricted data sets to locate fundamental frequencies and a series of tests to find the relative harmonics. As a starting point, great strides were made, but there were great limitations when it came to polyphonic material with similar overtone structures, such as octaves and perfect fifths. The limited understanding of both signal processing and psychoacoustic perception of polyphonics were the main limiting factors to early attempts, such as Moorer.

More modern approaches tend to use Moorer's original approach at least in part for locating fundamentals and signal energy, however the methodology for actually separating melodic content has been completely changed.

Many models such as that of Ryynänen and Klapuri use Hidden Markov Models to help map note values [3], however we chose to use a CNN because the research using neural networks as a methodology is more limited and therefore has more room for advancement.

There have also been issues with overfitting when using Hidden Markov models to predict future notes, and by using a CNN we hoped to avoid this issue.

One of the biggest flaws in publicly-available transcription models is their small scope. Many models will try to predict next notes, leading to overfitting as mentioned above, but most models also focus on monophonic transcription [4]. The separation of note values from their overtone harmonics has been a continued issue for researched models, making our polyphonic transcription aspirations even steeper.

Despite being far less studied and more limited in scope, CNN's have shown potential in how it limits the need for human intuition, removes the need for intermediate representation, and can withstand some of the issues which continue to plague other automated audio transcription methodologies.

The issues of transcribing polyphonic music typically simplify into one of two camps: tracking a pitch and deciding whether that pitch is relevant to the specific instrument or melody being transcribed. This second issue is particularly an issue because many models are monophonic in nature and/or train on intentionally simpler music with fewer instruments.

The first problem has several approaches that have been shown to work well, but the latter issue continues to be an issue as models continue to improve and new models developed. We hope to address both issues using our machine learning model.

2. DATASET

Our dataset included 330 curated pieces of classical music as supplied by MusicNet for the purpose of training machine learning algorithms such as ours.

We chose classical music for several reasons. First was the robustness of our dataset. Many pop music datasets lacked the lossless audio files or even any audio at all. One reason for this was the limitations due to copyrights. We also took into account the prevalence of classical music for scholastic endeavors and musical training regimens and the depth of music that fits this style compared to more modern styles such as rock or jazz.

Each piece of our dataset consisted of a .wav file containing the lossless audio, a .mid file with the MIDI formatted performance, and a .csv file with the information of each note, including onset and offset time, MIDI value, and note values.

For our model, we used the .wav files as our inputs and the .csv information as the respective labels for our dataset. The MIDI files were used in the evaluation of our model. Our model intends to take a .wav and convert to a .mid, hence our label and evaluation assignments. The dataset also consisted of pre-split training and testing datasets.

3. METHODS

Our model is a CNN, or convolutional neural network. However, before we advanced to our model, there were several pre-processing steps we implemented.

After properly importing and storing the label data from the respective .csv files, we made sure to extract features from our input .wav files. This included both the Short-Time Fourier Transform (STFT) and chroma features, allowing us to extract both an STFT and a chromagram of each music file. The STFT is helpful in identifying note values and pitch data for small time values, identifying note onset and offset times as well as pitch values. Chromagrams are better for analyzing sections of a piece, collecting the pitch information over one octave for visualization. This gives us a better idea of harmonic structure, even if it lacks exact timing data. These features were extracted and stored in arrays for all pieces in the dataset to be passed onto the model.

However, we ran tests both using the STFT data and trying to account for the information in our CNN, and found that the benefits were negligible. Thus, we removed this section of pre-processing and re-formatted the initial layer input of the CNN.

We downsampled and filtered the audio waves to 22050 Hz. Because we are interested in the pitch and not the timbral content, this allowed us to limit the active bandwidth without losing important pitch information. The reason for this is because we expect all fundamental pitches to be below the new Nyquist frequency, and only overtones and harmonics to be above this frequency.

For the model itself, we utilized split our audio files into usable time blocks or frames for processing. Next, we used four convolution encoding layers to extract the features of the waveform including the time-domain features. This information is helpful in establishing an easily referenced hierarchy for the structural progression of the piece. Our model architecture begins with two one-dimensional convolutional layers, each designed to progressively reduce the temporal resolution of the input while extracting meaningful local features. Both layers use a kernel size of 9 with padding of 4 and a stride of 2, preserving alignment while halving the input length at each stage. Each convolutional layer is followed by a ReLU activation function to introduce non-linearity and a MaxPooling layer that further reduces dimensionality and emphasizes dominant activations. The combined effect of these layers is to transform raw waveform input into a compressed representation that retains critical information about temporal patterns, such as note onsets and transient dynamics, while discarding irrelevant or redundant data. This compact, hierarchical encoding is then passed on to the recurrent layers for further temporal modeling.

Following the convolutional stack, the resulting compressed features are reshaped into a sequence format and passed into a two-layer bidirectional Long Short-Term Memory (LSTM) network with a hidden size of 128. This recurrent architecture enables the model to capture both past and future temporal dependencies within the musical sequence, which is critical for learning musical structure such as phrasing, rhythm, and note transitions. The bidirectional configuration ensures that each time step benefits from context in both directions, helping the model distinguish overlapping or closely spaced notes more effectively—particularly in polyphonic material. This sequence modeling stage refines the temporal coherence of the extracted features before classification.

In our output layer, a softmax layer helps to create a probability distribution of the 128 possible MIDI values (127 notes or note off) to help organize the pitch structure of the model.

The output from the LSTM is passed to a fully connected output layer with a softmax activation function, producing a probability distribution over 128 possible MIDI note values for each time step. This design allows the model to interpret the temporal sequence as a series of discrete note predictions, where the highest-probability class corresponds to the most likely note being played—or silence, depending on implementation. During training, we apply a padded binary mask to ignore inactive frames in each batch and compute the Binary Cross-Entropy (BCE) loss only over the active segments. This masked loss function ensures that the model focuses on musically relevant predictions, especially in variable-length sequences, and avoids learning from zero-padded or silent regions. The output predictions are later thresholded or post-processed to generate MIDI events, forming the basis for transcription evaluation.

We ran into several setbacks while working, but these will be discussed at greater length in our conclusion section.

Once training was complete, we took the output of our model and converted to a MIDI format.

4. EXPERIMENTS

In our original attempts, we used a small subsection of our dataset to attempt multiple variations of input for our CNN model. These included the full STFT and chromagram extraction, only a chromagram extraction, and now pre-processed feature extraction at all. Based on our results, we found that the features were already being processed effectively in the convolution layers of our neural network on small-scale tests. Because of our issues with GPU access, we chose to eliminate this step to reduce the computational cost of our model and the incremental benefits of these steps.

We also tested using different numbers of convolution layers, but given the robustness and complexity of our dataset, we settled on our current model.

At one point, we planned to split up our predicted notes into instruments, but given time constraints and the issues in training, we felt it more important to move to our large-scale model rather than spend time on this approach, thus we abandoned this idea.

One other approach we experimented with was using a sigmoid layer instead of a softmax layer in our model.

5. CONCLUSIONS

In most publicly available papers with a focus on automating music transcription, the model will be trained on pop music. This is for several reasons, one of which is the simplicity. Pop music tends to have a very distinct melody, making it easier to transcribe.

Many of these models also focus on either chord recognition or a single line, such as the melody or bass. Both of these limitations on the training model give the proposed algorithms a better chance of success. This in itself is a sign of how ineffective the public study of music transcription is.

Our model took neither of these shortcuts, training on polyphonic music from a myriad of classical composers with varying instrumentation and depths of complexity.

Based in part by the work of Carvalho and Smaragdis [5], we chose to implement a neural network to address the complexities and structural hierarchy of our dataset. While a less popular approach, we felt that it was an interesting methodology to pursue in how it better addressed polyphonic issues on small-scales compared to HMM's.

In hindsight, our intentions were extremely lofty and ambitious. The robustness of our dataset combined with issues in accessing a proper GPU that could handle our training process greatly reduced our work efficiency.

Issues with GPU access included being locked out of CIRC for over a week, being limited from using Google Colab for using too much strain on their servers, all students having Mac laptops without the ability to directly assign to a GPU core, and multiple online services crashing, resulting in the need to restart the training processes completely from scratch after nearly a day of training.

These continued setbacks extremely limited our ability to tune hyperparameters and optimize the efficiency of our system by changing our parameters or methodologies.

A neural network model shows promise in automated music transcription, including polyphonic music, but it requires far more research and dedicated resources before being considered for wider application.

In general, automated music transcription is an interesting field of study, with many uses from sheet music transcription for musicians to music information retrieval for study or analysis. But as it stands, accuracy in pitch tracking and part separation continue to cause issues. Our model was unable to adequately address the current flaws in publicly-available models due to time constraints, lofty goals, and issues with proper computing resources.

6. REFERENCES

[1] J. A. Moorer, “On the Transcription of Musical Sound by Computer,” Computer Music Journal, vol. 1, no. 4, pp. 32–38, Nov. 1977. Accessed: Mar. 12, 2025. [Online]. Available: https://www.jstor.org/stable/40731298

[2] M. Piszczalski and B. A. Galler, “Automatic Music Transcription,” Computer Music Journal, vol. 1, no. 4, pp. 24–31, Nov. 1977. Accessed: Mar. 12, 2025. [Online]. Available: https://www.jstor.org/stable/40731297

[3] M. P. Ryynänen and A. P. Klapuri, “Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music,” Computer Music Journal , vol. 32, no. 3, pp. 72–86, 2008. Accessed: Mar. 26, 2025. [Online]. Available: https://www.jstor.org/stable/40072648?seq=2

[4] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gomez, S. Streich and B. Ong, "Melody Transcription From Music Audio: Approaches and Evaluation," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 15, no. 4, pp. 1247-1256, May 2007

[5] 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. 151-15