Convolutional Neural Network-Based Approach for MIDI Transcription from Raw Audio

1st Alexander Fisher
Computer Science and Technology
Kean University
Union, NJ, USA
fisheral@kean.edu

2nd Kevin Parra-Olmedo
Computer Science and Technology
Kean University
Union, Nj, USA
parraolk@kean.edu

3rd Kuan Huang, Ph.D.
Computer Science and Technology
Kean University
Union, NJ, USA
khuang@kean.edu

I. Introduction

Transcribing musical audio to musical notation of any kind is a complex task that typically requires a trained musician's ear to produce accurate results. Constituents of a musical piece include (but are not limited to) pitch, pitch bends, temporal note placement, construction of chords with several notes being played simultaneously, individual note amplitude, and overall tempo of the song. All the while, any number of instruments can also be playing simultaneously. To supplement the grueling work of such a process, an AI-integrated web application is proposed; an application that receives musical audio as input and outputs a prediction of note placement along a MIDI piano roll.

II. Data set

The MAESTRO data set contains 100GB, or 200 hours worth of musical audio paired with its ground-truth MIDI transcriptions. It was constructed over the course of ten years, all data captured from the International Piano-e-Competition. For this reason, the entire training and validation data set comprises of monophonic music, that is, music with only one instrument: piano. The final model was trained on a small fraction of this data set, about 100 songs, and validated on about 27, creating an 80-20% training/validation split.

III. Literature Review

A. Automatic Music Transcription: An Overview [1]

In the context of building an application for transcribing raw music audio to MIDI, the reviewed paper "Automatic Music Transcription: An Overview" [1] is pivotal. It discusses advanced methods in automatic music transcription (AMT), with a focus on deep learning and non-negative matrix factorization techniques. These methodologies are essential for accurately transcribing polyphonic music, a core challenge in AMT. The paper also explores future directions like music language models and context-specific transcription, which are crucial for developing robust and versatile audio-to-MIDI transcription applications. This paper is a great introduction for anyone interested in learning about automatic music transcription and its feasibility.

B. Music transcription using a convolutional neural network [3]

In their research article titled "Music transcription using a convolutional neural network" [3], Verma, D. discusses his exploration of musical transcription using a convolutional neural network (CNN). The methodology involves segmenting audio input into eighth-second intervals. Each interval's frequency data is represented through a Constant Q-Transform (CQT) spectrogram, leveraging its logarithmic scale to accurately reflect audio frequencies. The CNN processes these spectrograms to predict notes within each sample. However, a limitation noted is the assemblage of the final MIDI output from numerous short segments, which impacts the transcription's overall fidelity in aspects such as note timing, BPM, and note length.

C. A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation [2]

The paper "A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation" by Rachel M. Bittner et al. [2] explores a neural network-based method for AMT, and their free-to-use audio-to-midi web application, "Basic Pitch" is a result of their research. The Spotify research team highlighted a transcription method significant for its lightweight design, facilitating efficient execution on low-resource devices. It introduces a model that can transcribe polyphonic music from a range of instruments, including vocals, without needing retraining for each new instrument. This approach addresses the challenge of creating versatile, low-resource AMT systems and sets a benchmark for future research in this field. The paper's findings are particularly relevant for developing applications that transcribe raw music audio to MIDI, offering insights into efficient, instrument-agnostic transcription methods.

IV. Methodology

For this research, our model architecture was taken from Spotify's research [Figure 1]. Their public GitHub repository enabled us to fork their code-base to our own project repository, and explore audio to MIDI transcription ourselves. The methodology for implementing our web application included: initializing python package dependencies to create a virtual environment capable of running our modified Basic Pitch code, creating an on-the-fly processing method for formatting audio data into a shape that the model would accept, processing ground-truth midi files into binary matrices to be directly compared with model output for calculating prediction loss, training the model using tensorflow, saving the weights of the trained model to our file system, building our web application front-end user interface, coding a python script to respond to a POST request with the model's predicted MIDI transcription of submitted audio, and finally, packaging the python back-end into a Docker container to be easily hosted on a web server.

A. *Initializing Dependencies*

Because we forked Spotify's Basic Pitch model architecture, we had to also create a python environment capable of running the pre-written code. Thankfully, the forked repository included a setup file with dependency compatibility information, and dependencies were installed into a conda virtual environment using pip.

B. *Data Pre-processing*

During the pre-processing stage, the MAESTRO data set was downloaded from magenta.tensorflow.org, which contained .wav song files and with each one, a paired MIDI representation as a .mid file. A JSON file was included for traversing the data set file structure, which allowed us to programmatically pair each audio file with their ground-truth MIDI file. A processing pipeline was written in python to process audio samples into intervals of 2 seconds with a sample rate of 22050Hz each, and then taking a 2 second interval from the paired ground-truth MIDI file using the same time offset to maintain synchronicity between sample and the paired MIDI representation. Each audio sample had a shape of (2, 43844, 1) and each paired MIDI representation had a shape of (172, 88), where 172 is the resolution of frames captured from the MIDI representation in a two second interval, and 88 is the number of possible notes that could be played on a piano.

C. *Training the model*

Next, we trained a model using Spotify's Basic Pitch neural network architecture using the pre-processed data. Because a note onset is either true or false at any given frame in a song, the model outputs a binary classification matrix, with each value equivalent to the predicted chance that a note is at that particular location in the song. In music, space between notes is crucial to achieve a memorable and enjoyable melody. Following the nature of music, its representation as a binary matrix subsequently remains largely empty, with few frames actually containing a note onset. For this reason, we used a heavily class-imbalanced weighted binary cross entropy loss function during training, where the existence of a note onset contributed to 99% of the calculated loss, where the other 1% was contributed by the absence of a note. We used an Adam optimizer with a learning rate of 0.001, and trained using 2-second interval samples taken from 100 songs for 1 epoch. The validation loss value finalized at 0.0026.

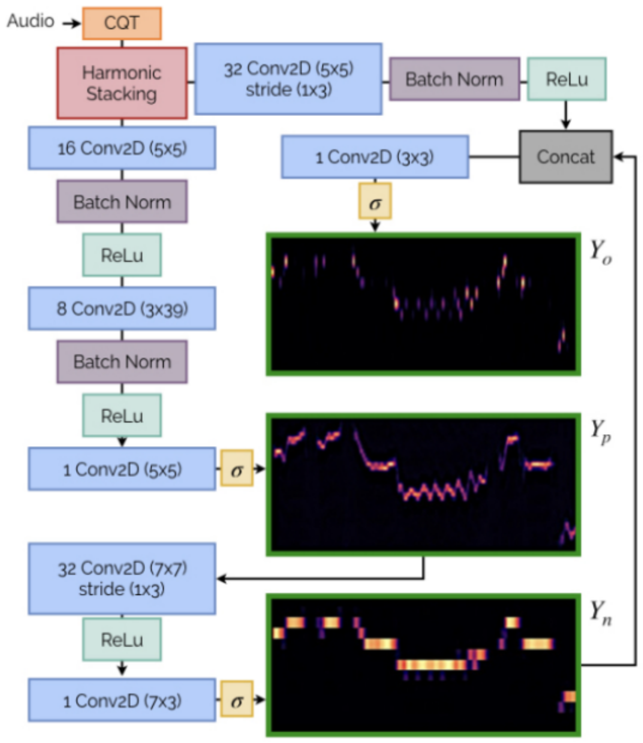

Fig. 1. Spotify's Basic Pitch Neural Network Architecture

V. Evaluation

After training the model, evaluation was done qualitatively by ear, by listening to the predicted MIDI output from the model for a song familiar to us. This was done by taking the post-processed model output in the form of a midi file and dragging it into a digital audio workstation for playback. Based on our analysis, the resulting model could successfully produce MIDI sequences with close resemblance to the input audio. Qualitative analysis of music is tricky because it can often be dismissed as subjective. Nonetheless, we believe our model can produce MIDI that is indeed usable in a practical scenario.

VI. Applying our model to an interactive application

The purpose of training an AI model for this task was for it to be implemented into a practical application that could be used by musicians and music producers. So, we built a web application to act as a user interface for the model. The technologies utilized for the web application include HTML, CSS, JavaScript, jQuery, WaveSurfer, Tone.js, and Python.

A. *Implementation*

The front-end UI [Figure 2] was built using the Bootstrap CSS library among our own custom CSS to make the HTML page look nicer. Upon loading the website, the user will see a drag-and-drop file upload input field, where they may upload any .wav or .mp3 file to be transcribed by the AI model.

The WaveSurfer JavaScript library was used to display an interactive audio waveform representation of the audio file uploaded by the user, enabling them to play, pause, or reset the audio playback in-browser on the webpage. Another HTML form field is found below these two UI elements, which prompts the user to select a model to transcribe with from a dropdown menu. We allowed the user to select between two trained models; our trained model, or Spotify's ICASSP 2022 trained model.

To interface with an AI model written in Python, we used an HTTP POST request using jQuery's AJAX function to call the python script hosted on the server upon HTML form submission. The python script accepted two POST data fields, one being the uploaded audio file, and another being the chosen AI model for transcription. Depending on which model was chosen to be used, the python script retrieves the model's trained weights file path, whose weights have been previously saved using tensorflow's built-in model-saving function. Afterwards, the basic pitch package's "predict_and_save" function was imported and called, using the uploaded audio file and the saved model weights as input. The output of the function was the inferred midi representation of the audio file, whose file path we returned as a response to the POST request. The browser was then able to retrieve the MIDI file using the file path given by the server's response and display it on the page for the user to view and interact with using the tone.js library, allowing the user to listen to the playback audio of the midi file, and follow along with the midi notation as it is being played. The web application was then packaged into a Docker container to be easily hosted on a web server without the hassle of re-initializing all dependencies and environment variables.

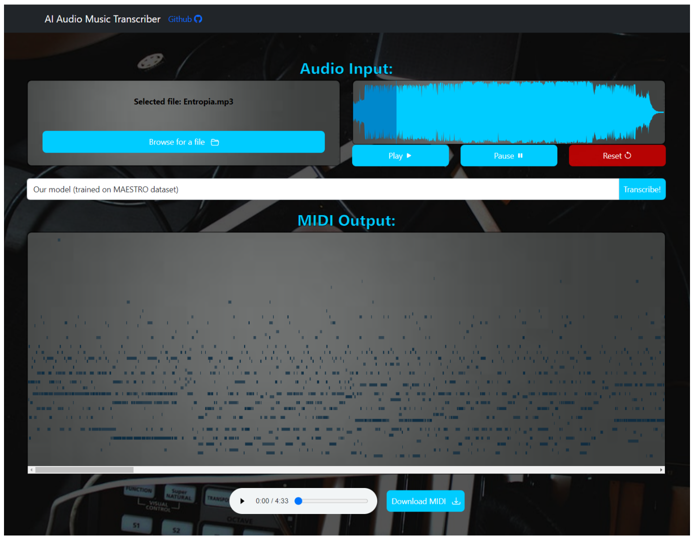

Fig. 2. Webpage User Interface

VII. Conclusion

All in all, our custom-trained basic pitch model did a decent job at transcribing a piece of raw musical audio into its predicted midi representation. However, the more data we trained the model on, the worse its predictions became. We believe this was due to the heavy class imbalance between the absence of note onsets and the existence of them, as the model has a tendency to produce more emptiness in the resulting midi respective to how many more samples the model is trained with. Nevertheless, the overarching goal of attempting to supplement the tedious process of musical transcription has been achieved with our project results.

VIII. Future Work

Improvements could be made for this research in several areas. For one, the UI could be more interactive, allowing the user to fine-tune model parameters to receive a MIDI transcription generated by a more customized model, such as the note prediction threshold. Another improvement for the research could be made within the training pipeline of the model because of the pitfall discovered in receiving inferior results in relation to how much data the model is trained on. An over-fitting problem is surely afoot, where the model becomes more likely to predict emptiness than actual note onsets after training on a larger data set.

References

[1] E. Benetos, S. Dixon, Z. Duan and S. Ewert, "Automatic Music Transcription: An Overview," in IEEE Signal Processing Magazine, vol. 36, no. 1, pp. 20-30, Jan. 2019, doi: 10.1109/MSP.2018.2869928. https://www.researchgate.net/publication/330068609_Automatic_Music_Transcription_An_Overview

[2] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal and S. Ewert, "A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation," ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, Singapore, 2022, pp. 781-785, doi: 10.1109/ICASSP43922.2022.9746549.

[3] Verma, D. (2020, January 10). Music transcription using a convolutional neural network. Medium. https://medium.com/@dhruvverma/music-transcription-using-a-convolutional-neural-network-b115968829f4