ECE324 Final Report

Project: AutoTranscriber

Anthony Lem (1005503992), Richard Yang (1004908870), Katarina Chiam (1004908996)

Word Count: 2000

Penalty: NULL

Permissions

Video Upload

Richard Yang: Wait for upload
Anthony Lem: Wait for upload
Katarina Chiam: Wait for upload

Final Report Public Availability

Richard Yang: Allow
Anthony Lem: Allow
Katarina Chiam: Allow

Code Public Availability

Richard Yang: Allow
Anthony Lem: Allow
Katarina Chiam: Allow

Introduction

Music has become integral to today's entertainment culture, but getting sheet music to play is difficult. Sheet music can be bought, but can be expensive. Finding the right pitches and rhythm is tedious for any individual, especially because when multiple notes are being played simultaneously. Therefore, we are capitalizing on the opportunity to automate the process of music transcription.

Music transcription is the process of taking an audio file and writing it to a format like MIDI, which contains information about the pitches, volume, duration, etc. of the music for each instrument and can be converted into sheet music. In our case, the goal of AutoTranscriber is to turn recordings of solo piano music into MIDI files. By using a neural network, we can expand this task to multiple instruments, instrument transfer and classification, efficiently. We have also seen that machine learning has proved successful in similar projects, solidifying our confidence in utilizing a machine learning based approach.

Illustration / Figure

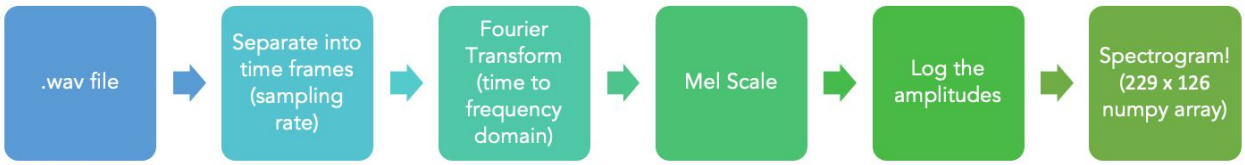


Figure 1: Data preprocessing pipeline

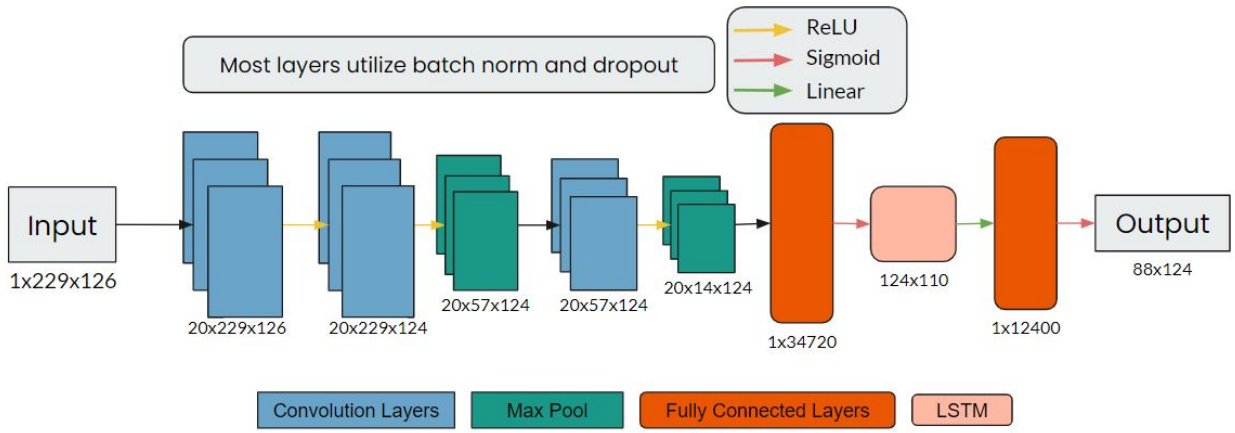


Figure 2: Final network

Background and Related Works

There are several challenges in automating music transcription. These include pitch, note duration, and volume estimation, instrument recognition, and beat tracking, among others [1]. Our project focuses only on the piano, and on pitch and note duration estimation.

Our project's model is inspired by the state-of-the-art neural network architecture that is a mix of CNNs and LSTMs presented in, "Onset and Frames: Dual-Objective Piano Transcription," by Hawthorne et. al [2]. This paper addressed automatic music transcription for piano and by taking in audio spectrograms and producing a corresponding MIDI representation. The architecture is inspired by similar models used in speech recognition tasks. The main insight this work provides is to consider the onset of notes when predicting note activations in a frame of music.

Data and Data Processing

The data we collected consisted of recordings of different genres of solo piano music. To collect audio, we recorded it in .m4a format while listening to the corresponding MIDI file collected from online to attempt to synchronize the data (audio) and label (MIDI) as much as possible. It was then converted into a .wav file. This was then sampled at a rate of 16 kHz, as done in prior papers [2], and partitioned into 4-second long segments. Afterwards, each segment was transformed into a spectrogram using Fourier transforms, and scaled with the mel scale with 229 bins, again as done in prior papers [2]. The mel scale transforms the frequencies into frequency bins spaced evenly according to the hearing of humans. We log the amplitudes to generate the final spectrogram. The spectrogram contains normalized values with respect to the training set.

To supplement our data, we used the MAESTRO dataset, which consists of .wav audio files and corresponding MIDI labels. We sampled from each of the training, validation, and testing sets from MAESTRO by composers to reach a train/validation/test split of 90/5/5 for a total of 8900 samples and ensured equal composer representation (i.e: selecting the same number of songs per composer when possible) to yield a balanced dataset. The process of converting the audio to spectrograms followed the same as above, which are summarized in Fig. 1. A sample of data is found in the figure below (Fig. 3) followed by statistics of each set (Fig. 4).

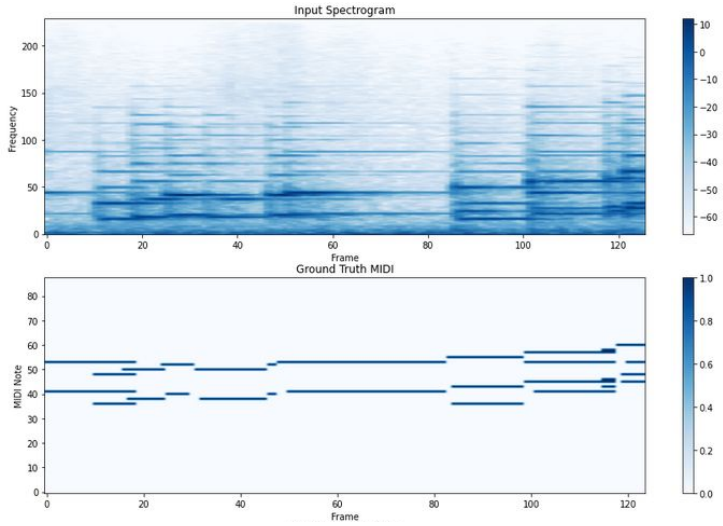


Figure 3: Sample of input spectrogram with corresponding MIDI label. The spectrogram has the time frame on the x-axis, and the frequency on the y-axis. To read the spectrogram, each point represents the loudness of that frequency (according to the colour scale, measured in decibels) at that point in time.

Set	**Duration [h]**
Train	10.00
Validation	0.625
Test	0.625
Total	**11.25**

Figure 4: Duration of recordings from MAESTRO outlying the amount of data used for each set

Architecture

Our model can be seen in figure 2 and utilized a sequence of a CNN, a single fully connected layer, an LSTM and another fully connected layer. Dimensions for each layer are specified in Figure 2.

CNN: The CNN which has three convolutional layers, each with 20 kernels of size 3x3 and followed by a ReLU activation function. The second and third layers have a padding of 1 and are followed by a max pool with a 4x1 kernel, followed by dropout at 25%. The output is flattened to be passed into the fully connected layer. All convolutional layers utilize batch normalization before activation functions.

Fully Connected Layer #1: Dropout is implemented at 50% and the output is normalized using batch normalization and passed through a ReLU activation function. The output is then reshaped for the LSTM.

LSTM: The LSTM has one hidden layer and is initialized to take in data where batch size is the first dimension. The output is flattened for the next fully connected layer.

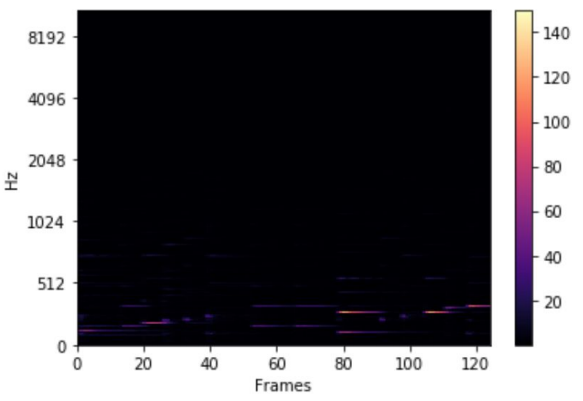
Fully Connected Layer #2: Dropout is implemented at 50% and the output is normalized using batch normalization and passed through a ReLU activation function. Since our loss function is BCElogitsloss, which applies a sigmoid within it's computation, we return the raw predictions after the ReLU to pass into our loss function and then another predictions tensor which is passed through a sigmoid as the actual network prediction.

Loss Function: Utilizes BCElogitsloss.

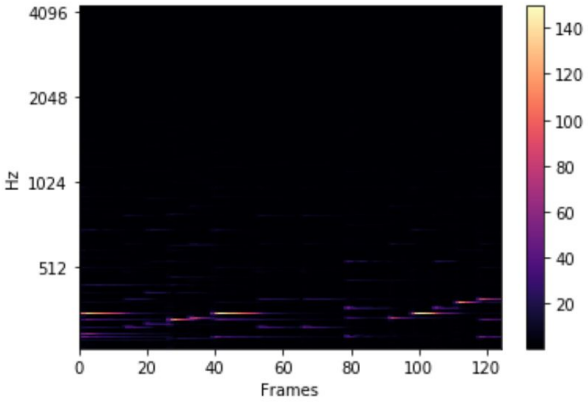
Optimizer: Utilizes RMSProp.

Baseline Model

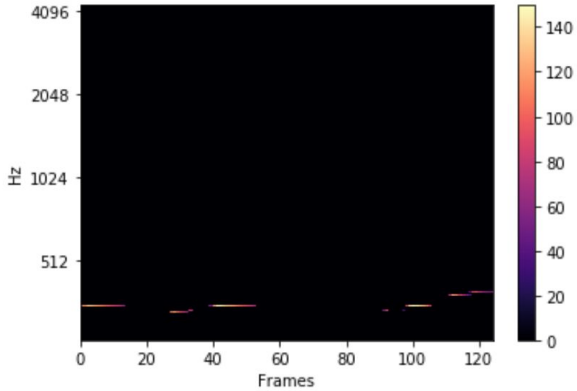
The baseline model is a system that attempts to automatically convert WAV files to MIDI format for piano pieces. The process is shown in the figure below.



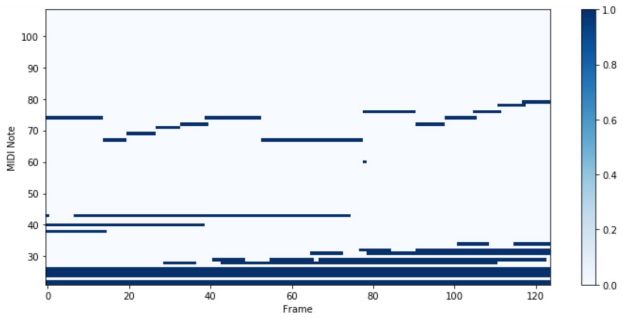
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure 5: Baseline model steps: (a) spectrogram of .wav file, (b) irrelevant frequencies removed, (c) loudest frequencies retained, and (d) conversion to MIDI notes.

The spectrogram in Step 1 is computed from the input .wav file using the Librosa library. Step 2 keeps only the rows in the spectrogram that correspond to frequencies playable on the piano. Step 3 is performed by setting all values less than 75% of the maximum value in each column to 0. This helps reduce noise. Step 4 is performed by mapping the rows in the spectrogram (each frequency) to their corresponding MIDI note.

Quantitative Results & Discussion

We considered 2 measures of accuracy - an element-wise match and a check for matching values within a specified window. The element-wise match was used to check the number of matching elements in our label array with our prediction array. Since the outputs for each value within the array was either a 1 or 0, the values were binary with a 50% chance the model could guess correctly.

We also wanted to determine the correct frames per prediction and thus considered an accuracy metric of finding the number of columns (i.e: pitches being played at each time frame) from our prediction matching each column from our labels individually, as well as matching within a specified window. This is to account for our own collected data inaccuracies as it was impossible to ensure the values would be played perfectly in time with our labels.

Method 1 Results

Below are the accuracy and loss curves for our final model using the first method.

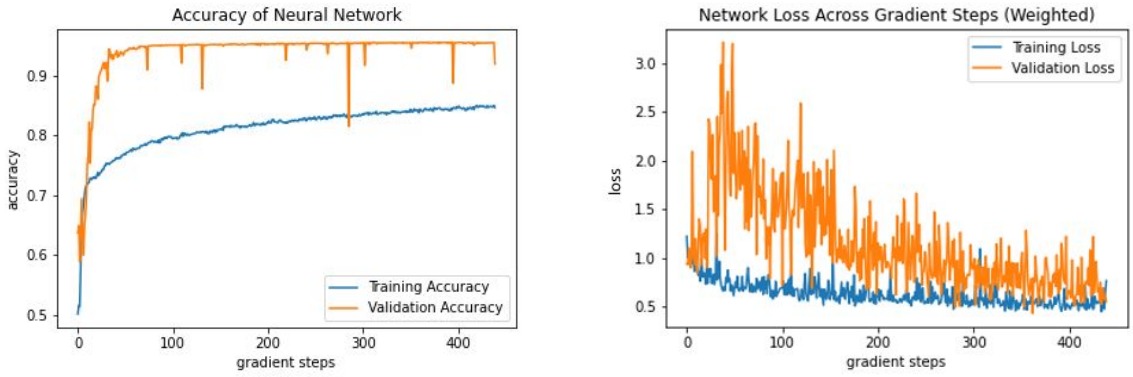


Figure 6: Plots of Element-Wise Accuracy versus Gradient Steps and Loss versus Gradient Steps for Final Model

The model achieves a high validation element-wise accuracy of 94.5% by the end of training with training accuracy of 84.6% and test accuracy of 91.5%. However, this is likely due to the large number of zeros in the ground truth MIDI file (the label), and from the large number of zeros predicted by our model. The most likely cause for the high validation accuracy compared to the training accuracy is due to the usage of

dropout in the model. While the training and validation losses at the end of the training loop are close to each other, to lower the loss, a model with more parameters would be required. Compared to our baseline which achieved an accuracy of 86.7%, our network was able to outperform it.

Method 2 Results

Table 1. Method 2 Results From Network

	Window Size 1	Window Size 5	Window Size 11
Train	0	0	0
Validation	1.3%	5.2%	13.3%
Test	1.4%	1.9%	10.5%

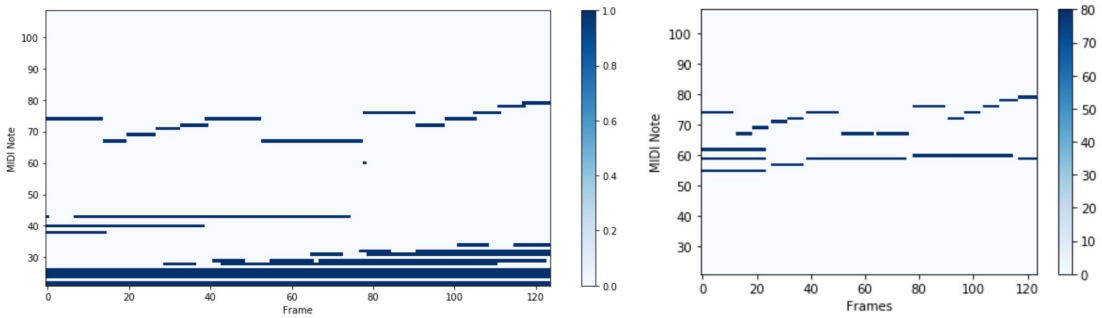
Table 2. Method 2 Results From Baseline

	Window Size 1	Window Size 5	Window Size 11
Result	0.5%	1.1%	1.6%

We see that while element wise accuracy is quite high, column wise accuracy is low and provides a better understanding of the actual quality of our predictions. Again, we see validation and testing accuracies higher than train as a likely result of heavy dropout presence. Further, we notice our network outperforms the baseline again.

Qualitative Results & Discussion

Baseline Model Results



(a) Baseline Model Predictions (d) Ground Truth MIDI

Figure 7: Comparison of Baseline Model Predictions against Ground Truth MIDI

Fig. 7 (a) shows a qualitative result, characteristic of what the baseline model produces. It correctly predicts the melody of the song (MIDI notes 60 to 80), but misses many of the other notes played, and instead falsely predicts many low notes. This occurs because when converting the audio file to a spectrogram, while not easily visible, a lot of the low frequencies have nonzero values which are not removed.

Final Model Results

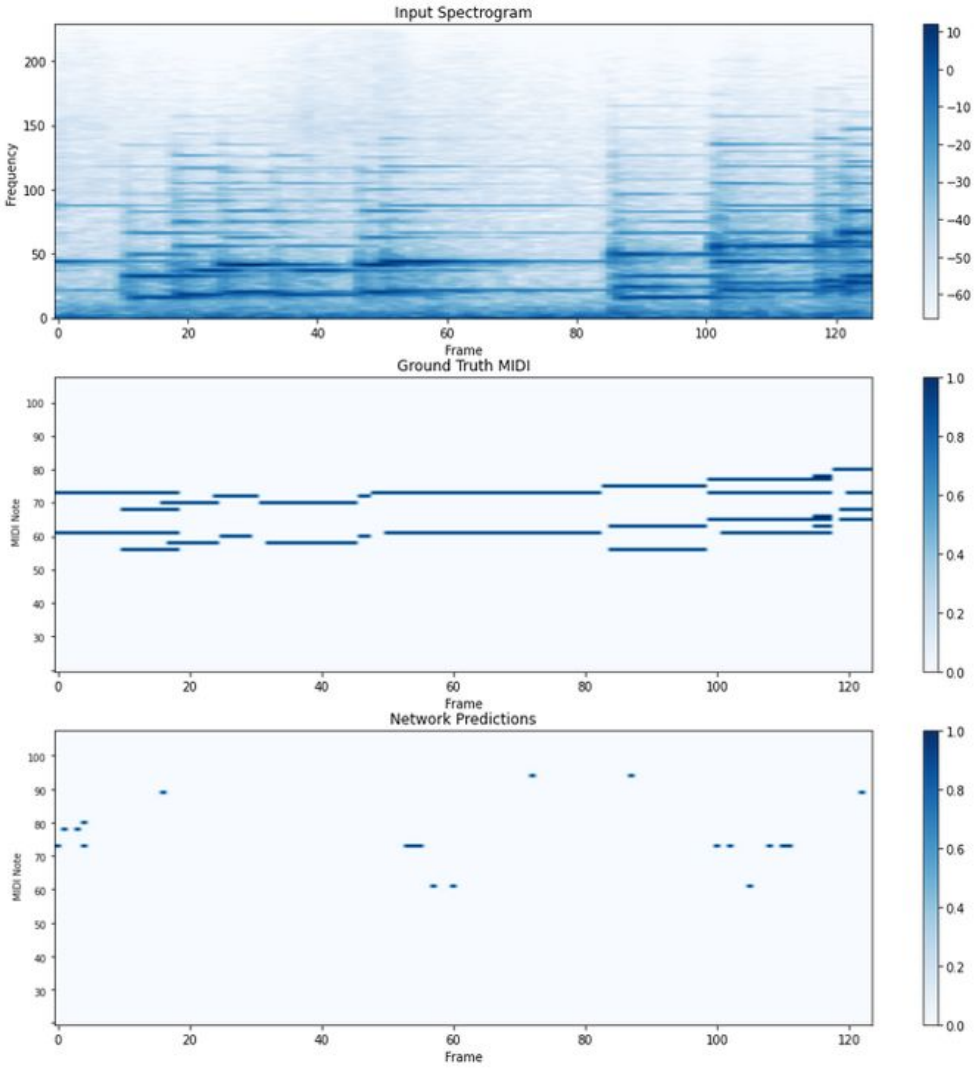


Figure 8: Network Predictions for a Song with Few Notes

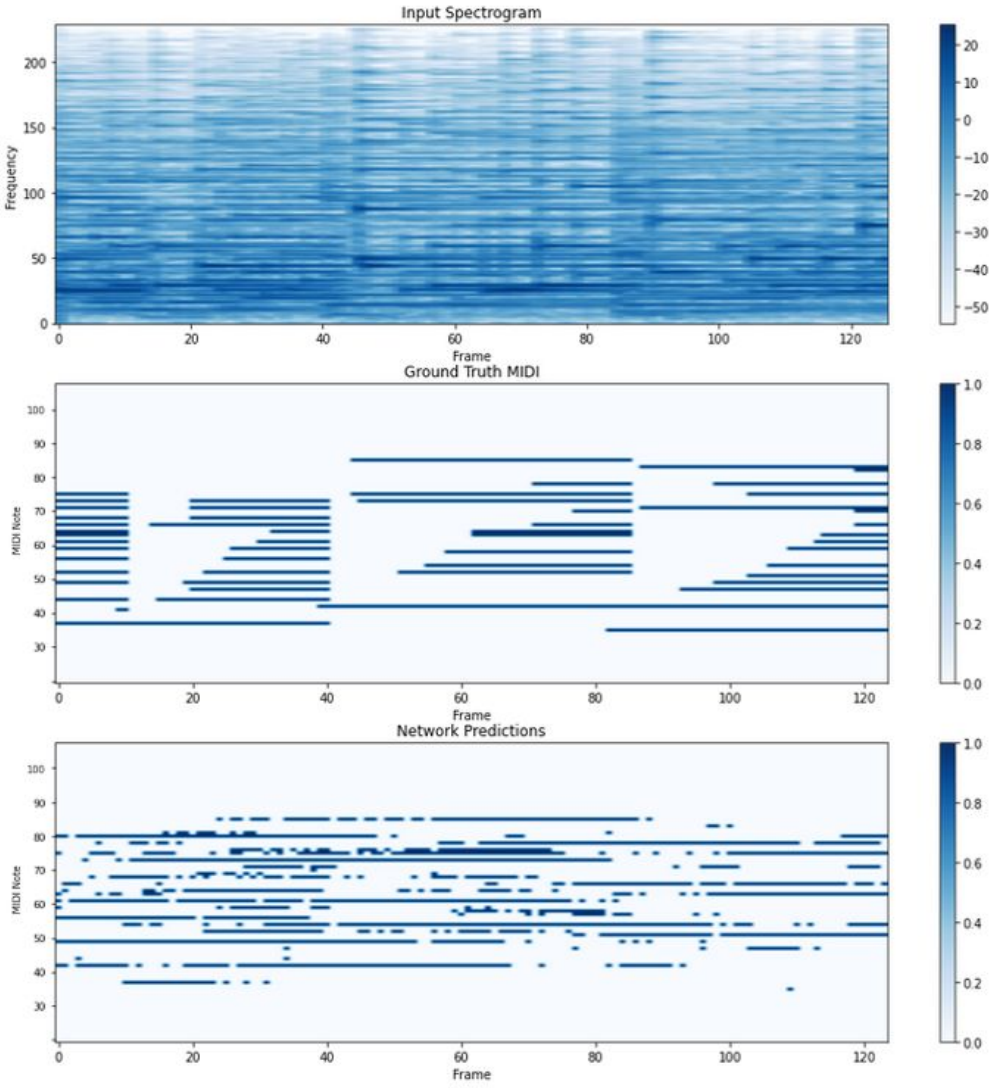


Figure 9: Network Predictions for a Song with Many Notes

Figures 8 and 9 together demonstrate that the network is able to correctly learn when to predict more or less notes. The predictions in Fig. 8 are sparse, while the predictions in Fig. 9 are much denser, reflecting the number of notes present in the input. The predictions themselves are not very meaningful though, implying the network was unable to learn the correct pattern and representations underlying the data. Therefore a larger network with more parameters or a better architecture is required.

Learnings

Originally, the model did not perform well as it was learning to consistently predict 0's, which corresponds to silent notes. This was due to the sparsity of 1's in our dataset. We had a training accuracy of 93% using the element-wise accuracy function, and inferred that about 93% of our dataset consists of 0's and adjusted our weighting to weigh the 1's 9.3x more heavily in our loss function. When starting a similar project, we would visualize the data much earlier to catch the problem and provide ourselves more time to fix our network. Broadly speaking this was an interesting lesson in dealing with inherent class imbalance and applies to a variety of problems beyond just music transcription such as object segmentation.

Although weighted classes helped our network produce more 1's, it was still unable to perfectly recreate the ground truth. If we had another similar project, we would consider adding information about note onsets, which refers to the initial moment a note is played. Other papers such as Google's Onset and Frames have designed models incorporating onset detectionormation and yielded successful results [2].

Ethical Framework

We will not be dealing with sensitive information or personal data as our data is exclusively audio files. The data utilized was from Google's MAESTRO dataset which is publicly available and will bring further awareness to Google's dataset and its research, providing beneficence. Due to the public availability of our data, we will not need to ponder the ethical implications of data collection.

For the user, there are little negative ethical impacts. Our tool is beneficent for users, transcribing various music sources into playable sheet music for them. It also bolsters their autonomy by providing new sources for piano sheet music. The user's individual justice is not impaired, nor is this tool maleficent towards them. However, the sheet music industry would face injustice through a possible loss in sales. Many musicians sell sheet music on platforms like YouTube by performing songs would also face a similar consequence. Our tool would allow users to extract audio files of such performances and generate sheet music for free. Within this context, our tool could be maleficent. In addition to independent musicians, music stores could also have their autonomy diminished as less users may feel inclined to purchase sheet music when they can attain similar versions for free using our model.

References

[1] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert, “Automatic Music Transcription: An Overview,” *IEEE Signal Processing Magazine*, vol. 36, pp. 20-30, Jan 2019.

[2] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck, “Onsets and Frames: Dual-Objective Piano Transcription,” In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, Paris, France.