Project: Multi-Instrument Audio Separation

Amey Dhamgunde, Rory Gao
Github: ameydhamgunde/ECE324_project (github.com)

Abstract

This paper aims to produce a model for Multi-Instrument Audio Separation. We explore prior approaches to this problem, such as 1D/2D convolution and attention based transformer networks. 2 approaches are attempted in this paper: audio-to-audio (convolution) and sequence-to-sequence (transformer). The convolutional architecture goal was to separate an audio file into its constituent instruments by separating the file itself by converting it into a spectrogram and performing a series of 1D and 2D convolutions using a model architecture inspired by Demucs. We created a synthetic dataset for this task, by overlapping different instruments playing different pieces together and training the model, which was ultimately unsuccessful. For the attention-based transformer model, we implemented and fine-tuned a pre-trained model known as MT3. We adapted a pytorch implementation of MT3, trained it on our synthetic dataset, which was generated by using piano and guitar playing the exact same notes but spaced apart by two octaves. We successfully trained the implementation of the MT3 model for 50 epochs, reaching a lowest validation loss of 0.81296 at epoch 17. We recognize the issue of overfitting, and attribute this to our lack of compute resources to train the model on a larger dataset and propose freezing some model layers. A qualitative assessment of the model output indicated moderate success, despite some extra/missing notes.

I. Introduction

Miserere mei, Deus, by Italian composer Gregorio Allegri was a piece performed only in the Sistine Chapel in Rome, with the Papacy in the Vatican keeping the score secret for more than a century. In 1770, the young prodigy Mozart ‘pirated’ the song by listening to it and transcribing it to sheet music from memory. Inspired by this legend, we want to approach audio separation in this project - given a polyphonic multi-instrumental sound sample, we aim to return separate tracks containing the contributions from each individual instrument.

Our project aims to train a deep NN to identify the contributions of all musical instruments in an audio track containing an ensemble of musical instruments into their respective components.

II. Background

The problem of musical instrument separation lies in identifying the aural “signature” of each instrument, and separating the contributions from each known instrument from a given piece. As you may have heard before, musical instruments can sound vastly different when playing the same note! The reason lies in the instrument’s *timbre.* An instrument playing a frequency may also be playing many more frequencies at integer multiples of the fundamental frequency of the note (C,D,E,F,G,A,B). In addition, instruments have distinct *attack*, *sustain,* and *decay* patterns, which dictate how fast the envelope of the sound amplitude grows, varies while held, and dies

down, respectively. So intuitively, any approach taken to learn a representation of musical timbre must learn temporal and spectral representations of its audio.

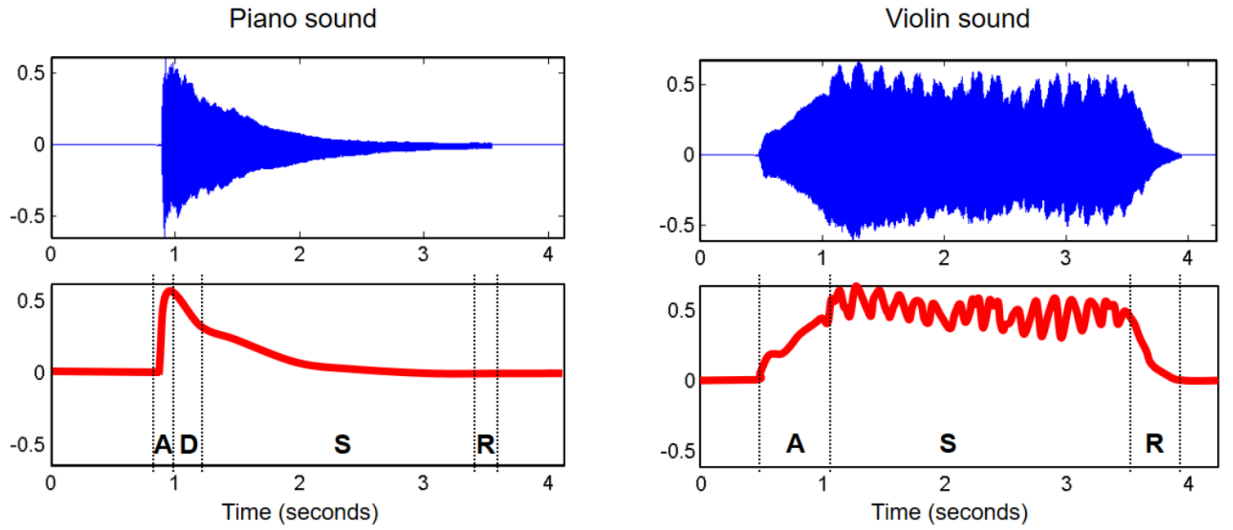

Figure 1. Audio envelope sample for one note played by a piano and violin. From [1].

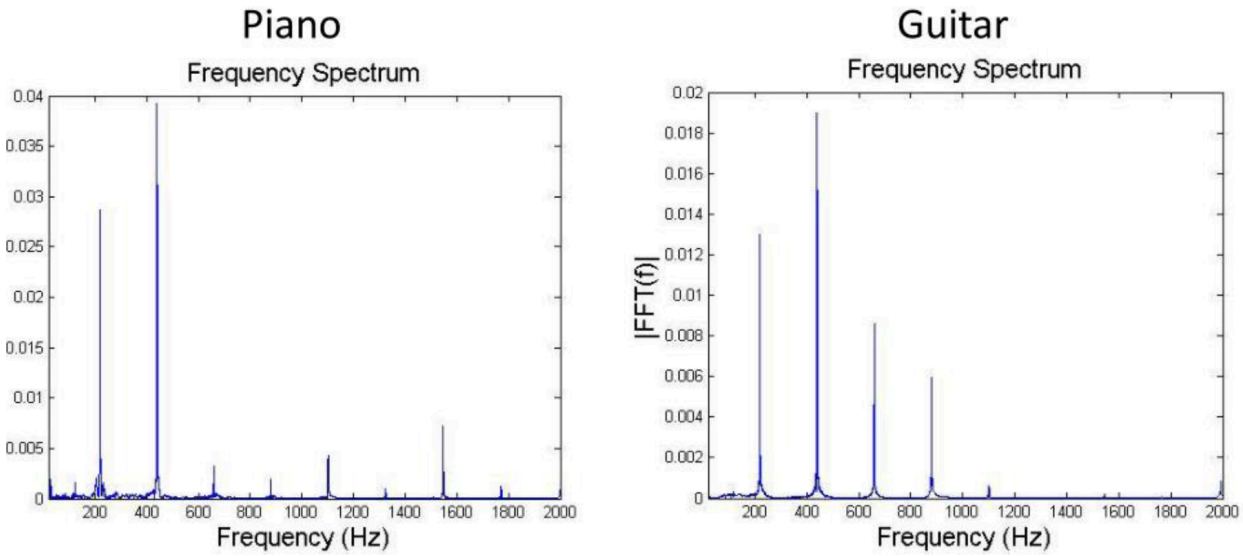

Figure 2. Frequency Spectrum of Piano and Guitar playing the same note, C4 at 440 Hz.

The task of musical instrument separation can be approached two ways. One approach is to directly separate the audio input into individual audio tracks of each instrument. This is an audio-to-audio approach. Another approach is to go from audio input to a note sequence of each instrument (i.e transcribe the notes played by each instrument), then regenerate the instrument audio from the note sequence. This is a sequence-to-sequence approach.

III. Prior work

Many models that use the audio-to-audio approach for general audio separation deal with the task of separating vocals from songs. These architectures tend to be 1D convolutional on the audio waveform or 2D convolutional on spectrograms [2,3,4]. Not many have been applied to instrument separation, as instrument audio is more “mixed” and “harder” to tell apart, especially when two instruments are playing the same note and potentially have the same medium in generating sound (i.e. string instruments, like piano, guitar, etc. and wind instruments).

Current state of the art sequence to sequence approaches for audio transcription are transformer models that take an audio sequence as input and produce output as sequence of notes for each instrument [5,6].

In this project, we try both approaches. We develop a 2D convolutional architecture to perform audio separation, and also fine tune an existing audio transcription model for mixed instrument transcription using a proprietary dataset.

IV. Convolutional architecture (audio-to-audio)

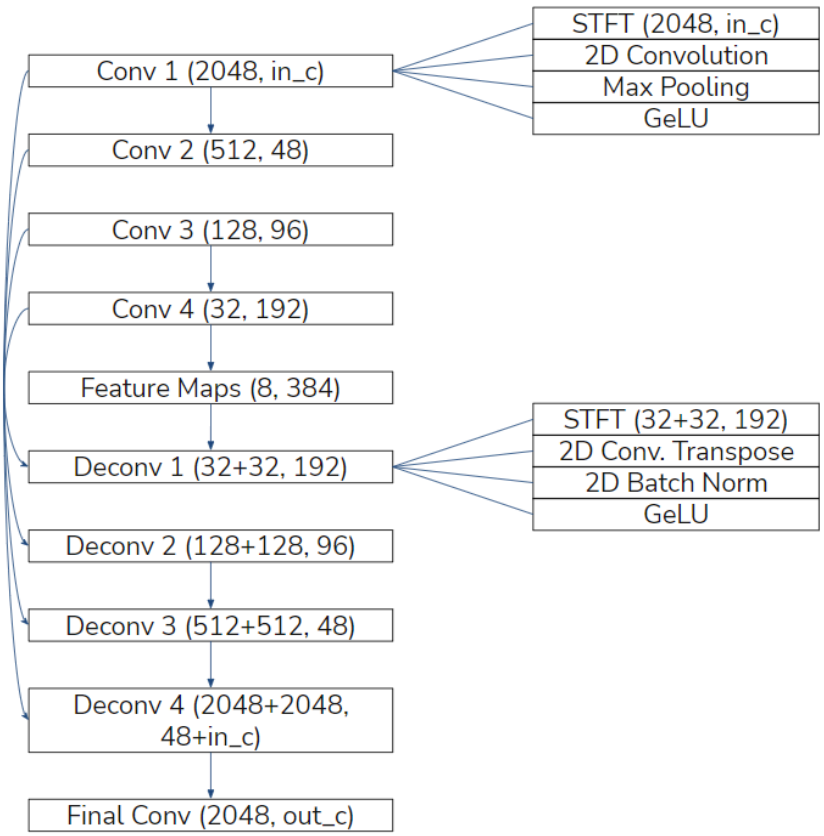

Figure 3. Proposed architecture of convolutional audio separation network.
Our convolutional architecture is heavily inspired by a leading audio separation model called Demucs [7].

The workflow of our convolutional audio separation network is as follows:

1. Get **STFT** representation of waveform with 2048 frequency bins (image of H: 2048 W: t)
2. **Convolutions** with **frequency max pooling**, until we arrive at layers of size (H: 8, W: t) with multiple feature maps
3. Re-expand layers using **transposed convolutions** with concatenated output of previous layers **(skip connections)**
4. Arrive at N channels, (H: 2048, W: t), each channel representing spectrogram of an instrument
5. Loss of each function is **Mean-Squared Error** between predicted spectrogram and ground truth spectrogram

Data generation for convolutional architecture:

Due to the unreliability of existing datasets we chose to generate our own synthetic music dataset.

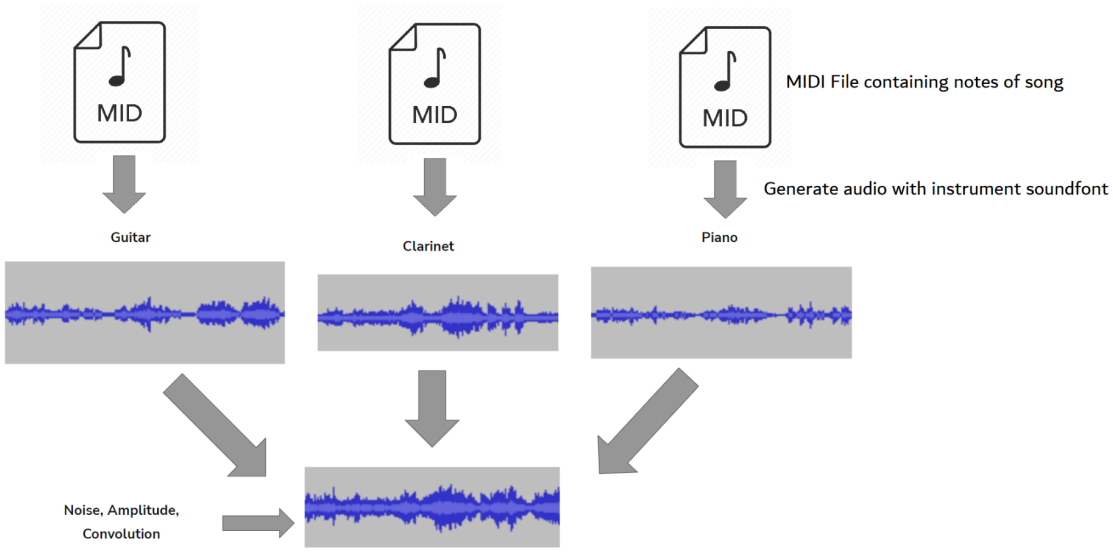

Figure 4. Illustrated workflow of synthetic data generation

From a bank of 50 classical pieces [8] (~15 hours of audio) we generate ~40 hours of training data.

1. For each instrument in our bank {clarinet, guitar, piano}, save the audio signal of the MIDI file played by the instrument.

2. With all the tracks containing all our songs played by all our instruments, create mixed tracks by:
    a. Randomly mixing different pieces played by different instruments
    b. Mixing the same piece played by different instruments

The idea behind (a) is that the model should be able to separate instrument noises even if the track sounds like gibberish. We hope to encourage the model to learn the signatures of individual instruments to perform effectively at this task. (b) approximates orchestral musical pieces, where many instruments play the same note in unison which causes more entangled frequencies.

3. Perform data augmentation in these forms:
    a. Convolution with impulse response of {concert hall, slightly echoing room, random response}. These impulse responses were also synthetically made by us.
    b. Additive white noise
    c. Multiplicative amplitude adjustment

As far as we know, the method of convolution data augmentation is novel. Many papers do not do this, instead just doing volume, pitch, and speed augmentation

Implementation and training for convolutional architecture:

Using pytorch, we implemented the model as outlined by figure 3, which can be found in the Separator.ipynb file on our Github. We trained this model on our synthetic generated dataset using the mean squared error loss for 5 epochs (due to limited compute resources) and a learning rate of 1e-3. We then attempted to convert the multi-instrument audio file into its component instruments using the model, and visualized it as a spectrogram.

Results of convolutional architecture:

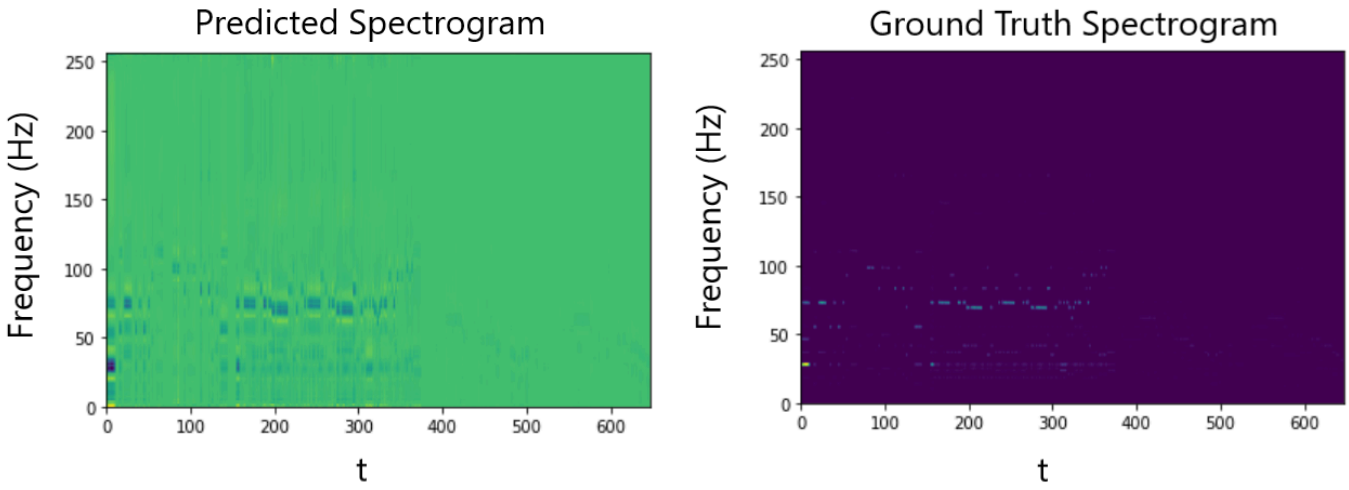

Figure 5: Convolutional architecture model output vs ground truth spectrograms

As seen in figure 5, the model produces the spectrogram on the left, and qualitatively there is a vague resemblance to the ground truth. However, the prediction fails to produce meaningful audio because:

- The spectrogram is inverted (green when supposed to be black, black when supposed to be green) indicating the need for a post processing layer or function to
- Predicted notes are not “isolated” enough. The ground truth shows distinct peaks of green which represent the distinct frequencies of musical notes, while in our prediction there is a broad peak where the note should be. Translated into audio, this sounds unintelligible, unclear and ‘fuzzy’.

After receiving feedback on this approach, we decided to change our approach and fine-tune a pretrained model for a transformer-based sequence-to-sequence architecture. We made this decision for a few reasons:

1. We lack the compute resources to successfully train a model of this size on a large enough dataset for long enough to reach convergence.
2. In the event that the model does not converge, we do not have enough time to experiment with changing the model multiple times until we reach a version that converges.

V. Attention-based transformer architecture

Our sequence-to-sequence approach to instrument separation is by fine tuning a pretrained model called MT3 [5]. MT3 is a transformer neural network that is capable of doing multi instrument audio transcription, and we would like to adapt it to our dataset. MT3 is a transformer model which uses the T5/T5X framework by Google as its main building block - specifically, T5ForConditionalGeneration. T5 itself is an encoder-decoder model for text-to-text formatted tasks. MT3 uses a variety of python libraries (librosa, note_to_seq, pretty-midi, etc.) in order to

preprocess the audio files (.mp3, .wav, etc) into tokenized sequences (tensors) which are used as input to the model. The model output is then postprocessed from tensors into MIDI sequences (one for each instrument), which fulfills the task of transcription. Furthermore, these MIDI files can then be played through a soundfont to obtain separated audio files.

Data generation for transformer architecture:

The data here is also synthetic data, very similar to the data generation for the audio-to-audio architecture. We used 10 classical pieces, amounting to ~1.5 hours of audio. Implementing feedback received, we decided to use piano and guitar playing the exact same notes but spaced apart by two octaves. This is more realistic than our previously generated dataset since it fits within the discrete nature of orchestral music where instruments are generally playing at the same tempo and key while harmonizing, rather than randomly overlapping. Note that this is not a trivial task because though the two instruments are playing two octaves apart, their frequency spectrum of each note mixes in others, which can make it a more difficult task for a machine learning model to separate and return the contribution of each individual instrument.

Implementation and training of transformer architecture:

We based our implementation on a pytorch implementation of the MT3 architecture from https://github.com/kunato/mt3-pytorch. With most of the groundwork laid out for us, we used pytorch-lightning to load the MT3 model (with pretrained weights) in MT3_net.py (on our Github). We then used a pytorch-lightning trainer to train the model on our dataset which we preprocessed and postprocessed separately in the training_code.py file - exactly as we would process the model inputs/outputs when generating inferences. Using an 80/20 training /validation split, We trained the model for a maximum of 50 epochs with 50 batches per training epoch, and used checkpoints to save the 5 lowest validation losses computed after every epoch.The loss function used is the cross entropy loss of token predictions by the decoder.

Finally, after training, we loaded the model with the lowest validation loss from the saved checkpoint before overfitting, and ran inference.py on our validation set to get MIDI output files, which we played back using the same soundfont to gain a qualitative insight into the success (or failure) of our trained model.

Results of transformer architecture:

This model performs well on our validation tasks after being trained on similar tasks, and shows convergence on our dataset. Using checkpoints and varying hyperparameters like learning rate and batch size, our best performing model had a validation loss of 0.79238.

Learning Rate	Batch Size	Min. Validation Loss	Epoch (50 steps/epoch)
2e-5	4	0.81296	17

Figure 6: Training loss of sequence to sequence model.

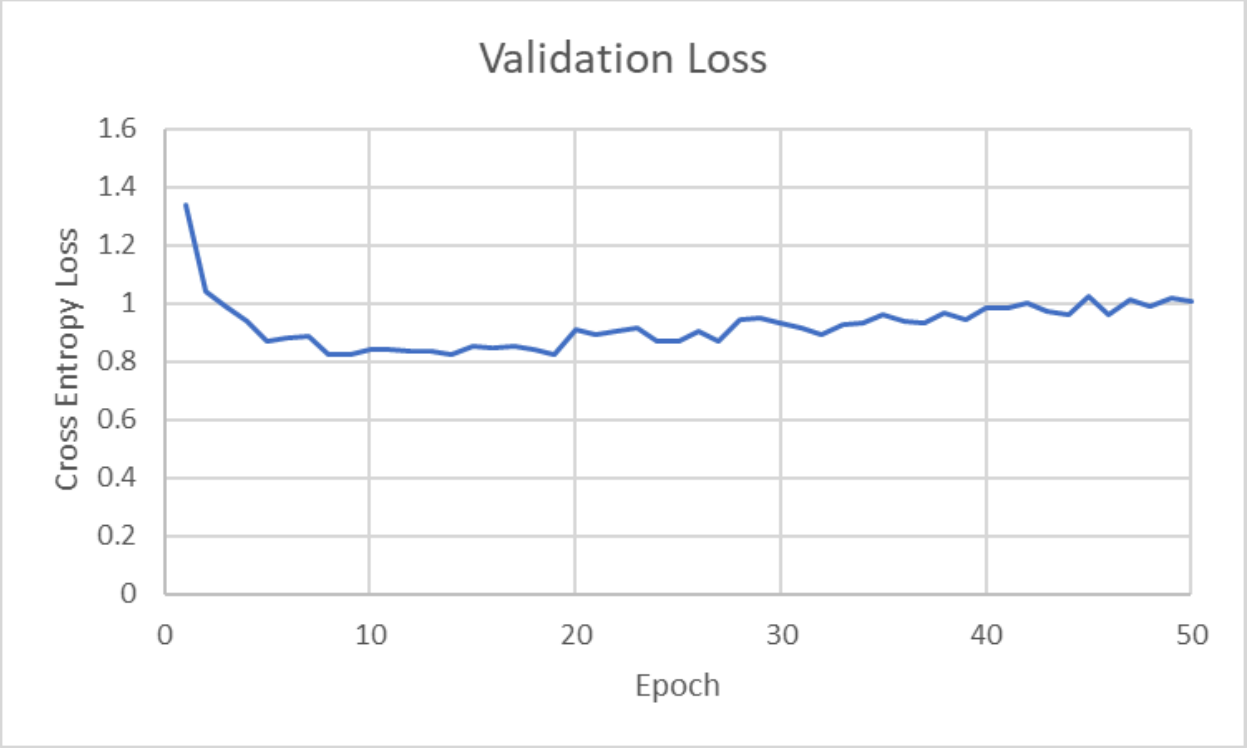

Figure 7: Validation loss of sequence to sequence model.

The loss curves show convergence on training loss but quickly overfits based on validation loss. As a result, we used the model checkpoint at the lowest validation loss level located at epoch 17. We likely ran into the overfitting issue very quickly because the model is quite large, with ~45.9 million trainable parameters, and our training dataset was too small. Unfortunately, we don't have the compute resources in order to train this model for an adequate length of time on a larger dataset, but the results of this paper indicate that we were successful in improving model performance over a few epochs on a small dataset, which proves the validity of our procedure which could be repeated on a larger dataset with more compute resources. Additionally, it would be worth trying to freeze some of the layers in the model to preserve some of the previously learned weights while also combatting the overfitting issue. Due to time constraints, we were unable to do this ourselves for this project.

Qualitative results:

The transcript predicted by the model for a test sample (A rondeau by Beethoven, don't know exact code) can be visualized compared to the ground truth. Errors in transcription include false positives (notes which do not exist in ground truth), and false negatives (omission of note present in ground truth). Our general results show that the model tends to make more false positives than negatives, which is reasonable if the dataset is very mixed (if all you hear is a jumble of sounds, you are likely to attribute it to more instruments than are actually producing the sound).

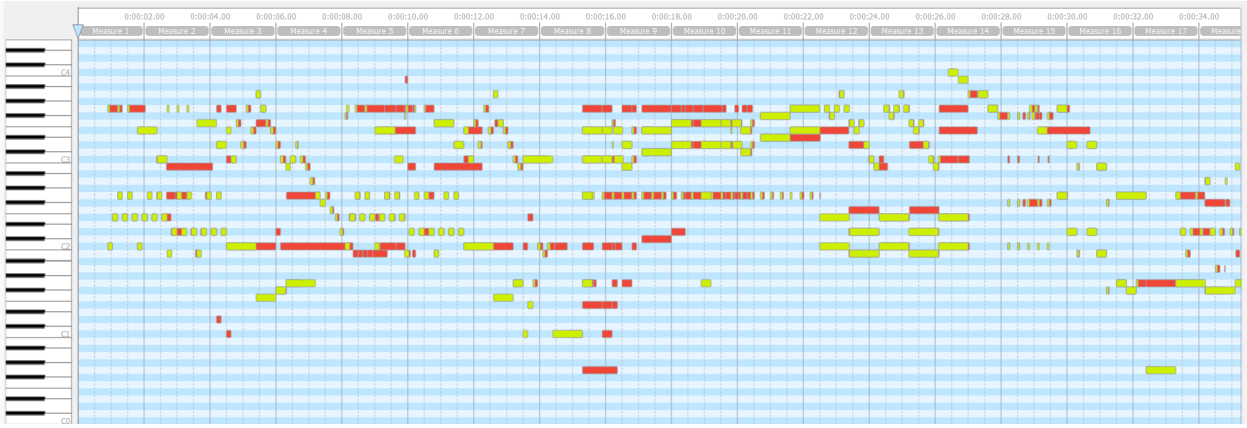

Figure 8: False positives of model prediction for Guitar notes. Predicted notes are overlaid on ground truth notes. Red notes are false positives.

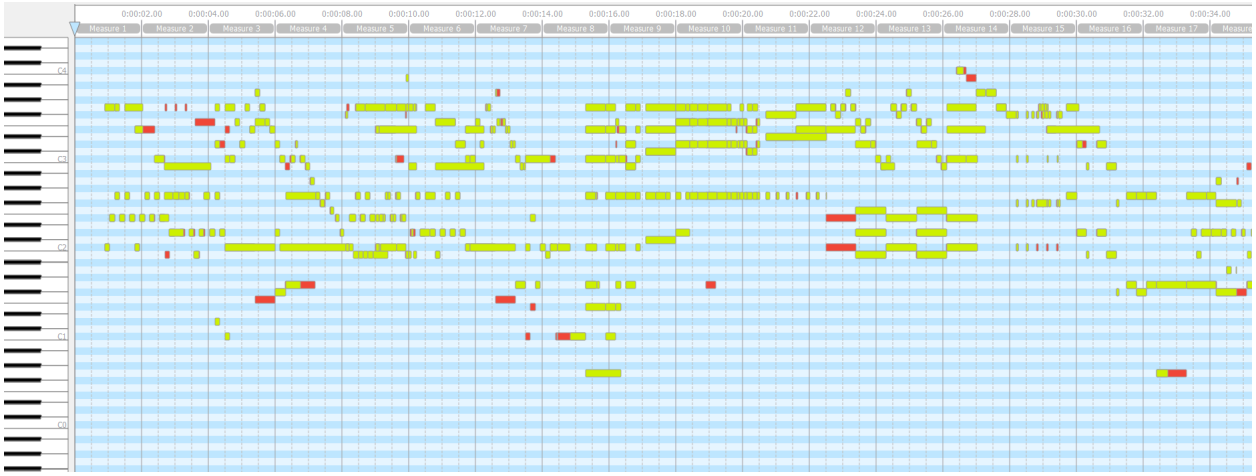

Figure 9: False negatives of model prediction for Guitar notes. Predicted notes are overlaid on ground truth notes. Red notes are false negatives.

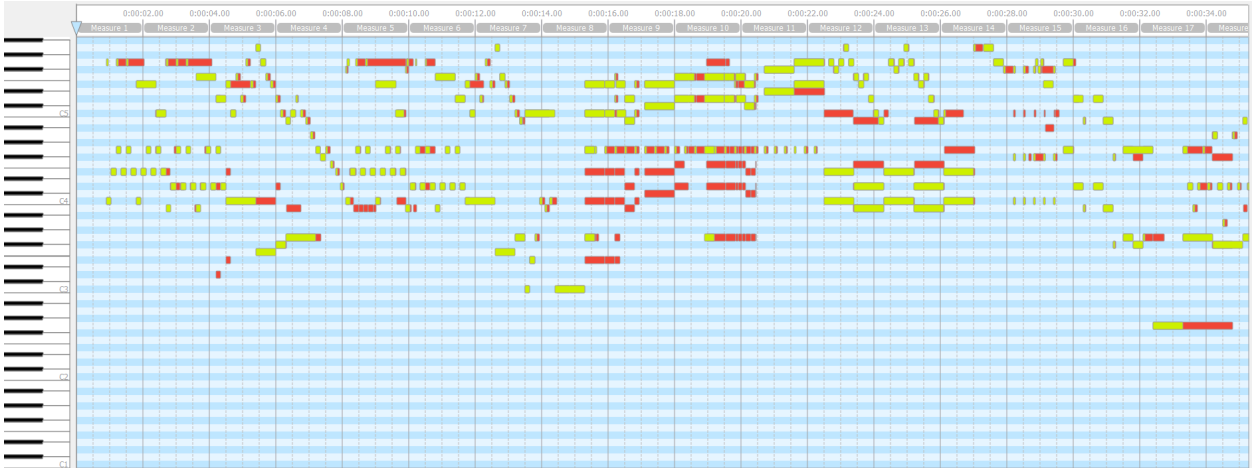

Figure 10: False positives of model prediction for Piano notes. Predicted notes are overlaid on ground truth notes. Red notes are false positives.

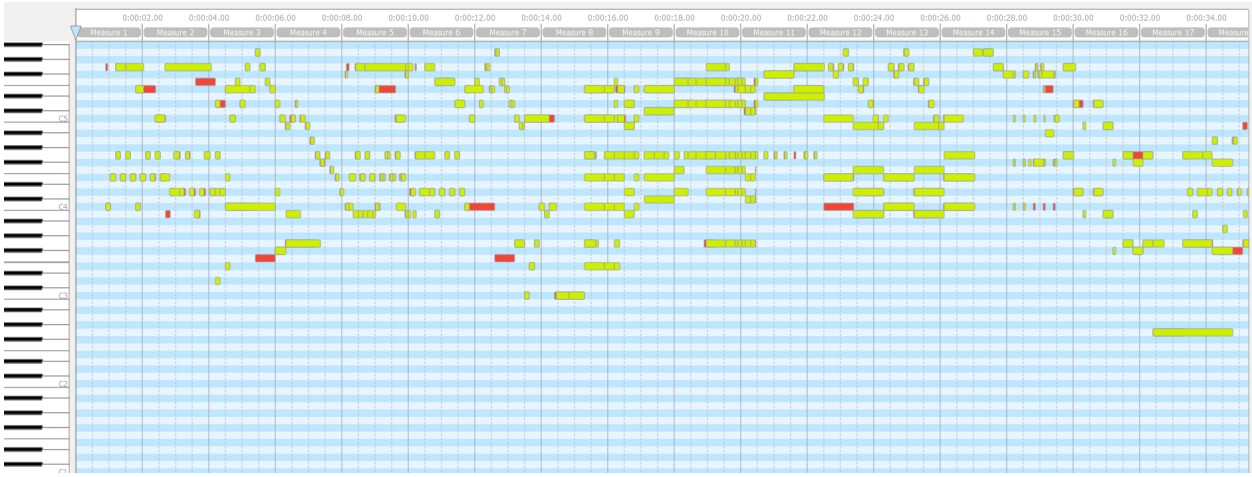

Figure 10: False negatives of model prediction for Piano notes. Predicted notes are overlaid on ground truth notes. Red notes are false negatives.

Instrument	Melody Preserved? (RH)	Accompaniment preserved? (LH)	Chords preserved?	Notes
Piano	Mostly	Has extra notes	Mostly	Model tends to make more false positives than false negatives, but general structure of piece is recovered
Guitar	Somewhat	Yes	Somewhat	Repetitive elements of music preserved very well

Samples are available to listen on our Github under the"Samples" folder.

VI. Ethical Framework:

While deep musical transcription has been greatly boosted by transformers and can achieve almost humanlike accuracy on single and multi instrument transcription, one major limitation is that fundamentally, they are designed to transcribe western classical pieces. "Western music" means notes are assigned to 12 pitch classes (C, D, E, F, G, A, B, and sharps and flats). This concept of "notes" are not universal, as non-european genres such as the blues, Indian ragas [9], Indonesian gamelan music [10], etc, have microtonality (i.e. notes in-between western notes) in their music that all defy the traditional 12-note transcription scheme. This can be addressed using alternate frequency discretization schemes, or using non-discrete frequency representations.

In addition, transcription performance is likely to suffer on non-western music due to lack of diversity in training data. The ethical implications include disparity in effectiveness for non-western cultures who would like to use our tool. This can be addressed by enforcing music diversity in the training set.

Finally, the tokenization step of our model also places an inductive bias on the rhythmicity of the input piece. Most transformer-based transcription models, ours included, rely on a time discretization of output tokens that fits most tempo scales (i.e one token per 0.04 s). However, a wide variety of cultural music features continuous intensity change, continuous tempo change, and continuous pitch change which are features that become arguably more truncated through tokenization than their western music counterparts. One could argue that cultural music transcription should be approached with a different model, but it is important to note that these cases still fall under "music transcription" and any model designed to do this task must make sure it is generalizable to other "flavours" of music. As we did not test our model on

non-western music, it is uncertain how our model performs. Does it fail altogether at giving a coherent transcript, or does it output a transcript that predicts the correct notes, but "westernizes" the piece, by truncating its rhythmicity/tempo to fit within the western musical framework? The latter may be dangerous as it has the potential to erase musical heritage if users are not careful in checking the quality of transcriptions. What can be done to alleviate these concerns is to explore alternate tokenization of input audio/output note sequence that is robust to changes in tempo, intensity, and pitch.

VII. Conclusion

In this paper, we tackled the task of Multi-Instrument Audio Separation. We started with an audio-to-audio (convolutional) approach, where the goal was to separate an audio file into its constituent instruments by separating the file itself by converting it into a spectrogram and performing a series of 1D and 2D convolutions using a model architecture inspired by Demucs. We created a synthetic dataset for this task, by overlapping different instruments playing different pieces together and training the model (found on our Github under Separator.ipynb) using it.

We were unsuccessful in this venture, and decided to switch to a sequence-to-sequence (transformer) model, where we implemented and fine-tuned a pre-trained transformer model known as MT3 (Gardner et al, 2022). We adapted the pytorch implementation of MT3 found at https://github.com/kunato/mt3-pytorch, which can be found on our Github with our key changes in files mt3_net.py, training_code.py, and inference.py. The pretrained model was trained on our synthetic dataset, which was generated by using piano and guitar playing the exact same notes but spaced apart by two octaves.

We successfully trained the implementation of the MT3 model for 50 epochs, reaching a lowest validation loss of 0.81296 at epoch 17. We recognize the issue of overfitting, and attribute this to our lack of compute resources to train the model on a larger dataset and provide an alternative solution of freezing some of the model layers. We also performed a qualitative assessment of the model output which indicated moderate success, despite some extra/missing notes.

References

[1] Müller, M. (2021). Fundamentals of Music Processing. In M. Müller, *Fundamentals of Music Processing* (pp. 26-30). Springer.

[2] Daniel Stoller, S. E. (2018). WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION. *arXiv*.

[3] Lu Zhang, C. L. (2021). MULTI-TASK AUDIO SOURCE SEPARATION. *arXiv*.

[4] Naoya Takahashi, Y. M. (2017). MULTI-SCALE MULTI-BAND DENSENETS FOR AUDIO SOURCE SEPARATION. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.* New Paltz, NY: IEEE.

[5] Gardner, J., Simon, I., Manilow, E., Hawthorne, C., Engel, J. (2022, March 15). MT3: Multi-task multitrack music transcription. arXiv.org. Retrieved April 14, 2023, from https://arxiv.org/abs/2111.03017v4

[6] Prafulla Dhariwal, H. J. (2020). Jukebox: A Generative Model for Music. *arXiv*.

[7] Simon Rouard, F. M. (2022). HYBRID TRANSFORMERS FOR MUSIC SOURCE SEPARATION. *arXiv*.

[8] Classical MIDI Files, from https://www.midiworld.com/classic.htm

[9] Demystifying Indian Classical Music, from Different Kinds of Ragas by Scale (thaat) - Raag Hindustani (raag-hindustani.com)

[10] Gamelan, from Gamelan - intangible heritage - Culture Sector - UNESCO