Midi Shark: For Piano Transcription

Jonah Chen, QiLin Xue, Joe Hattori, Khanatat Thangwatthanarat
University of Toronto
{jonah.chen,qilin.xue,joe.hattori,k.thangwatthanarat}@mail.utoronto.ca
December 8, 2021

Abstract

This project uses transformer encoder to transcribe piano music in a dual-objective fashion similar to the Onsets-Frames model[1]. Our model was able to achieve a slightly higher F1-score of 79.01% with significantly less computational power, demonstrating that once again, transformers were able to outperform LSTMs. On real world data, a recording of the *anime* song なんでもないや. played by Jonah Chen was given to the model. We observed the model was able to capture the key elements of the song, although there were still a few mistakes in the transcription.

1 Introduction

Transcription of music is the process of determining the pitches and timing of notes from recorded audio files. Transcription has always been a specialized task that requires years of musical training. Transcription is even more challenging for polyphonic music, such as piano, which features the simultaneous production of two or more tones. The majority of traditional transcription models focus on extracting all of the notes from the recording using the node onset. This, however, is not the way a trained musician approaches the problem[2].

We developed a model that is more accurate at transcribing piano recordings by analyzing the recording with a neural network and focusing on both the onsets and offsets of the node. Moreover, since images and audios both have common two-dimensional time-frequency input representations, the fact that CNN performs well in image classification problems suggests that CNN could potentially be used for music transcription[1].

We also used techniques that are usually used in natural language processing tasks. Not only is this because music transcription is a sequence-to-sequence task, but there are also patterns in music theory that describe whether a sequence of notes "sounds right," similar to how only certain combinations of words "sounds right"[1].

The code is open source and available here.

2 Illustration

The model architecture is shown in figure 1.

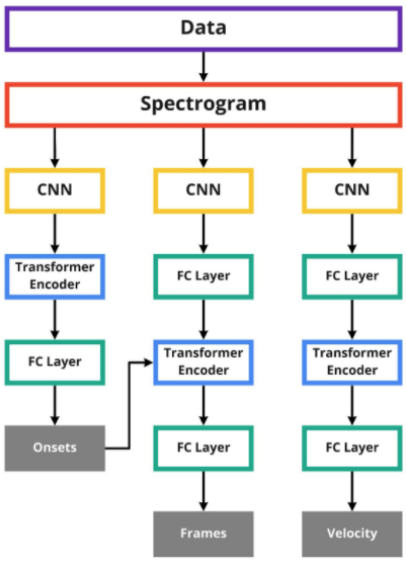

Figure 1: Model architecture

3 Background

Polyphonic automatic music transcription is a difficult task due to the potential of having multiple notes played simultaneously, causing the harmonics of each note to overlap with the others. Therefore, it is a non-trivial task to transcribe the music using the spectrogram.

Since the advent of machine learning, there have been models developed for solving this problem. A relatively successful model is the Onsets and Frames (OF) model developed by the Google Brain Team using TensorFlow in 2017 and implemented into the open-source "magenta" framework[9]. This model uses a two-headed encoder-decoder architecture that predicts onsets (when the notes are played), and frames (what notes are played at the onset). This model uses convolutional neural networks as encoders and bidirectional LSTM as decoders and was able to achieve success on the MASTERO dataset, which is arguably the largest dataset for music transcription task.

4 Data Processing

The MAESTRO dataset consists of pairs of WAV and MIDI files of piano audio separated by year. The years consists of $2004, 2006, 2008 - 2018$. The audio in 2017 served as the validation set and the

audio in 2018 served as the test set. In total, there are around 200 hours of piano performances.

To process the WAV files to act as inputs, we converted them to frequency spectrograms by applying a Short-Time Fourier Transform using a sampling rate of 16000, and used the Mel scale for the frequency in order to generate 229 bands. Since songs can be up to 30 minutes, and transformers have a time complexity of $\mathcal{O}(n^2)$, we are motivated to keep sequence lengths short. As a result, we partitioned the spectrogram into 20 s non-overlapping segments such that each segment has a shape of 229×862. An example spectrogram is shown in figure 2, and is heavily inspired from the onsets and frames baseline model[1].

The MIDI files acted as the ground truth, but the files only contained information about the onset and offset of each note. We preprocessed each midi file into three arrays of shape 88×862, where 88 is the number of notes. Each entry (t, n) denotes the value of the array at a specific time t and a note n.

- Onset Array: We have $(t, n) = 1$ if the note n is played at time t but not played at time $t-1$.
- Frames Array: We have $(t, n) = 1$ if the note n is played at time t.
- Velocities Array: We have $(t, n) = v/127$ where $v \in [0, 127]$ is the velocity of the note, which is an indicator of the dynamics.

These arrays act as the labels for each of the three sub-tasks described in figure 1. Our data processing can be broken down to two major parts. One is converting WAV files to spectrograms, and the other task is converting MIDI files to a certain kind of form which is easier to train the model.

The downside of this approach is that we might have a clip at the end of the song that is shorter than 20 seconds. This can be resolved by zero-padding, however, since most songs were several minutes long, we did not think this would make a noticeable difference and opted to simply ignore these last few seconds.

5 Architecture

The OF model[1] had seen success in the task of automatic music transcription, but since 2017 when the model was published, there have been several publications that have brought forth improvements in areas like sequence processing.

In the OF model, the processed mel-spectrogram data is encoded using a stack of convolutional layers[3]. Furthermore, the OF model uses bidirectional LSTM models as decoders. Since 2017, the transformer architecture has revolutionized the processing of sequential data[4]. Many of today's state-of-the-art models in fields like natural language processing are based on the transformer. As audio is sequential data, we think it will be advantageous to use a transformer encoder in place of the bidirectional LSTM.

Following similar data flow to OF, we will have a two-headed model with an onset and frame head. The onset prediction is passed as an input to the transformer encoder that decodes the frame predictions. We use the same loss functions for the two heads from OF (a generalization of cross-entropy loss, see eq.1-6 in[1]). For the velocity model, we use a similar approach to the frames model but without the input from onset prediction. Apart from that, we use the modified version of mean square error instead of a cross-entropy loss as a loss function. For all the models, we use Adam optimizer as our optimization function. For all the models, we also use fully connected layers in between. For a clearer picture of the model structure, please refer to figure 1 in the introduction section.

6 Baseline Model

The baseline model we compare our model to is the LSTM described in the *Onsets and Frames* (OF) paper by Google[1]. The model architecture is very simlar to figure 1, except transformers are used instead of LSTMs, and the onsets do not feed back into the LSTM.

This is a reasonable choice since our hypothesis is that introducing a transformer will improve the performance of the model. Since Google researchers likely have far greater computing resources than us and better data processing, to create a better comparison, we will write and train the LSTM model on the same processed data and machine used for the Transformer model.

7 Qualitative Results

The results of one of the songs in the test set is shown in figure 3. At first glance, the frames model seems to be very accurate. However, many notes are broken up into smaller segments, leading to a lot of false negatives. This can be addressed by changing the threshold for positive predictions, which is analyzed further in the quantitative results section.

Furthermore, the lower frequencies were not as accurate as the higher frequencies. These characteristics are consistent with other predictions in the test set. The reason for these inconsistencies is explained in the discussion section.

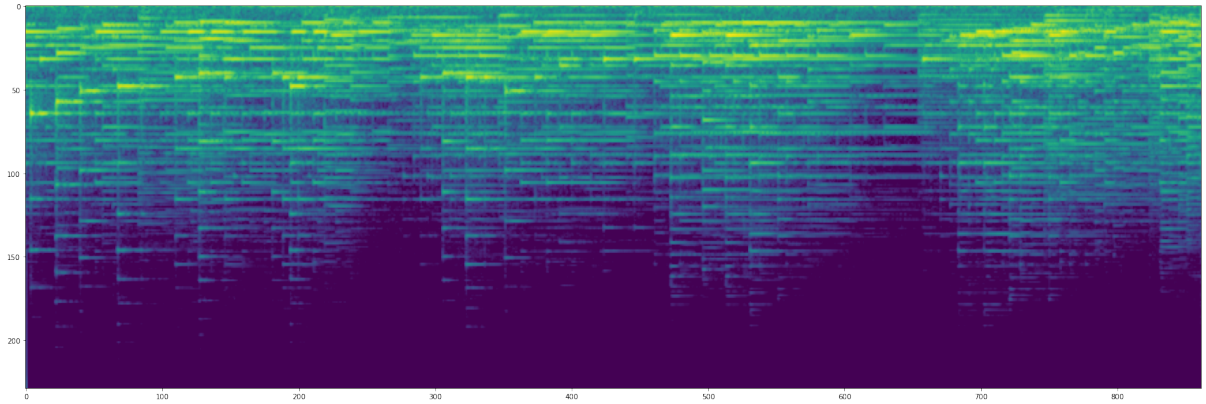

Figure 2: Example spectrogram of one of the songs in the MAESTRO dataset.

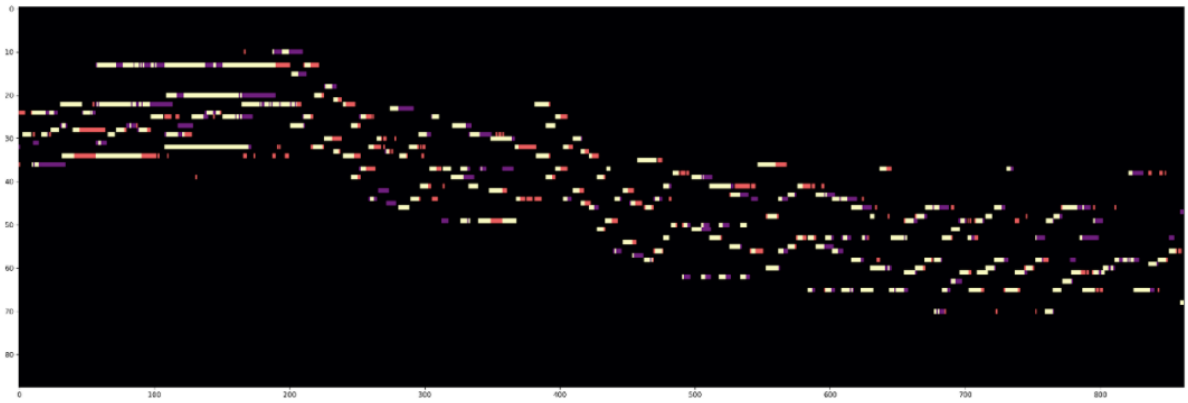

Figure 3: The frame predictions the model made on one of the songs in the test set. Yellow represents the true positives, purple represents the false negatives and the red represents the false positives.

8 Quantitative Results

A piano has 88 keys but a human only has 10 fingers. On average, less than 5% of the notes are pressed at any given time. If we were to use accuracy as our metric, a model that predicts no notes at all would achieve over 95% accuracy. Clearly, accuracy is not a good indicator of the performance of a model. Instead, we opted to use the F1 score

$$F1 = \frac{2PR}{P+R} \tag{1}$$

where P is the precision, R is the recall. We used the test split of the MAESTRO dataset which was never used in training or hyperparameter tuning.

Model	Frames F1 Score
Sigtia et al., 2016	72.22%
Kelz et al., 2016	71.60%
Melodyne[a]	58.57%
OF	78.30%
Google's 2021 Model	**82.18%**
Baseline Model (LSTM)[bc]	74.44%
Midi Shark (Us)[c]	79.01%

[a]Commercial software, not machine learning model.

[b]Same architecture as OF, but trained by us with the same hyperparameters.

[c]Tested on the test split of the MAESTRO dataset, because the MAPS dataset is unavailiable due to website failure.

For the velocity model, we used a different metric. As the velocities represent the intensity of the notes we hear, we track the number of notes (onsets) that are predicted to be played, which are within $\{10\%, 3\%, 1\%\}$ of the ground truth velocities. Our velocity model achieves $\{95.78\%, 59.46\%, 21.48\%\}$ on these metrics respectively. The baseline model achieves ..., again indicating the transformer encoder was able to outperform the LSTM.

9 Testing

One concern is that the music in the MAESTRO dataset share similar features. They are mostly classical pieces from the 17th to early 20th century[5], and there could be inherent biases the creators of the dataset might have when selecting the pieces and the type of recording device used.

We decided to record our own music. In the spirit of having a much different style than the music in the dataset, we recorded an *anime* song played in なんでもないや.

Since we don't have the ground truth, and there were some mistakes made while playing it, we decided to use a holistic evaluation instead, which is what matters at the very end. The generated audio is linked in the Github repository.

In general, the generated audio sounds very good and captures the main melody, harmony, and dynamics. However, it fails to capture some lower notes. Furthermore, there are a few false positives that is exactly one octave above the melody. These are explained in the Discussion section and steps are described to address them.

10 Discussion

When testing on both the test set and completely new data collected independently, we found that the model fails to capture some lower notes. This makes sense as it is difficult to pick up lower frequencies through a microphone, and the microphones the MAESTRO dataset used to record was likely calibrated to better capture the lower frequencies.

Furthermore, there are a few false positives that is exactly one octave above the melody on our own recording. This corresponds to the first overtone of the melody, which is the most dominant one. These same characteristics were found when testing the model on other music found on YouTube, though the quality heavily depended on the quality of the recording device. As our microphone was not calibrated, the higher frequencies are likely to be more intense than those used in the MAESTRO dataset, which caused a model trained on the dataset to produce false positives.

To fix this in future models, we can augment our training data by adding in random noise that is consistent with poor microphone quality. We can also adjust the weight of each frequency to better represent the behavior of real-world recording devices. This will allow the model to perform better on new data that is not recorded with a professionally calibrated microphone.

We also noticed in Figure 3 that it is common for the model to split up a note into smaller pieces. This behavior could be explained with the pixel-wise cross-entropy loss function. If the model mispredicts only one pixel, it only incurs a small loss but it will split a note into two, which sounds unpleasant. It is possible to combine the onsets predictions with a classical algorithm to combine these segments.

Another way to resolve this issue is to change the threshold for positive predictions. It is common to consider a "positive" prediction of the model outputs a probability greater than 0.5 (or a positive logit). However, we discovered using figure 4 that using a threshold of -0.4 was able to achieve the highest F1 score on the validation set of just under 79%. Moreover, much fewer notes were split up.

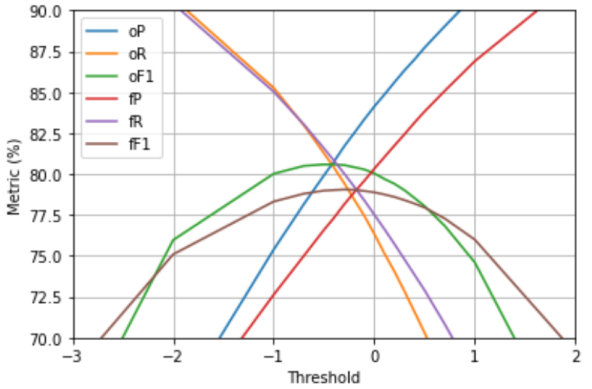

Figure 4: The metrics (**P**recision, **R**ecall, **F1** score) for **o**nsets and **f**rames with different threhold logits.

Finally, we noticed that when training the model, the model would stagonate at zero accuracy for up to one hour before it would start to improve for small learning rates ($lr < 10^{-4}$), and the model never improved for large learning rates. We think this may be caused by the initialization of the parameters of the model. The default initialization for pytorch

11 Ethical Considerations

In the United States, the music sheet industry is worth around 1 billion USD[6]. A technology that can transcribe music may render purchasing music sheets useless. Many of these sales come from music books, which often contain several scores that can be readily found online, so big publishers will likely not be greatly affected by this technology.

However, this technology will negatively affect small composers, who may make a living by selling sheet music to their music, for example on services such as MusicSpoke[6]. If this technology is accurate and readily available, it may encourage musicians to automatically generate the sheet music to the songs they like, instead of purchasing and supporting artists. Fortunately, transcribing isn't just about having the right notes, but the style in which it is presented. This is why two transcribers will not produce the same sheet music for the same song.

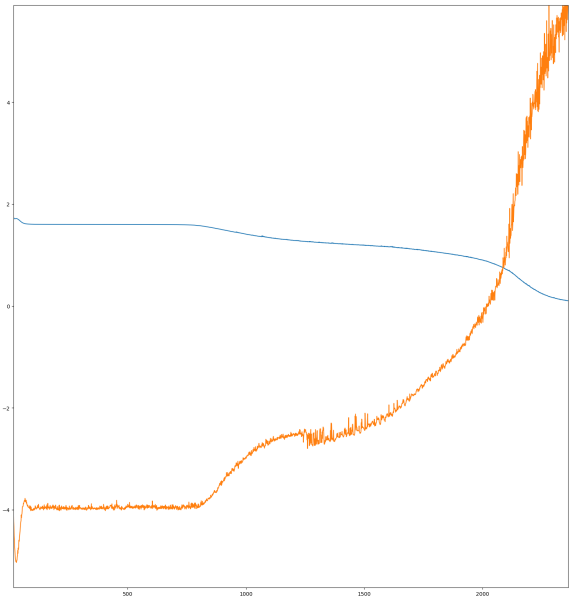

Figure 5: The models progress tracked every minute of training. Blue indicates the loss and orange indicates the **maximum** logits the model predicted throughout all batches of the training data it saw. Note if that the orange line is below 0 (the orange line reaches 0 slightly before crossing the blue line), the model would not predict any notes being played. This plot was made during debugging so the tick labels are not very clear. What is important is the general behavior.

The MASTERO dataset and the scope of our project consist of working solely with music, so there is a low risk of discriminating against humans from pre-processing to training and post-processing. However, a large majority of the music from the dataset is Western music, so the results would likely be more reliable towards Western music. Thus, people from other cultures may not find the same success in this model as people in Western cultures do. To address this, we will attempt to find other similar datasets that were trained on music from other cultures.

12 Conclusion

We are able to demonstrate that our model, while not perfect, is able to transcribe audio that is at least competitive with the existing state-of-the-art model in music transcription. We have shown that switching from a LSTM to a transformer model can drastically improve the performance. Future work involves improving the model to recognize lower notes through data augmentation.

References

[1] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *CoRR*, vol. abs/1710.11153, 2017. arXiv: 1710.11153. [Online]. Available: http://arxiv.org/abs/1710.11153.

[2] S. Hainsworth and M. Macleod, *The automated music transcription problem*, Department of Engineering, University of Cambridge, 2004.

[3] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, "On the potential of simple framewise approaches to piano transcription," *CoRR*, vol. abs/1612.05153, 2016. arXiv: 1612.05153. [Online]. Available: http://arxiv.org/abs/1612.05153.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[5] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," *CoRR*, vol. abs/1810.12247, 2018. arXiv: 1810.12247. [Online]. Available: http://arxiv.org/abs/1810.12247.

[6] M. Hunckler, *Musicspoke looks to disrupt 1 billion sheet music industry with marketplace for artist-owned scores*, Forbes, 2017.