Deep Learning for Audio and Music:

Joshua Ryan Lam

April 2020

1 Introduction

For my deep learning project I decided to combine source separation and automatic music transcription. Source separation takes a sound file that's a "mix" of several sound sources and separates them into "stems": sound files that only contain the audio from individual instruments. Automatic music transcription takes an audio file and seeks to transcribe it, typically in the form of its sheet music or midi representation. Since many automatic transcription algorithms are meant for transcribing a single instrument, my goal was to run source separation on an audio file and then run a transcription algorithm on one of the stems. I used open unmix's pretrained model for source separation and worked on coding my own network for automatic transcription, based on the work of others.

2 Source Separation

One of the most successful source separation algorithms is the open-unmix model published in 2019 (2). The open-unmix network was trained on the MUSDB18 dataset for music transcription. As with many other audio processing networks, open-unmix's output is a mask applied to the original mix's spectrogram. It is capable of separating stems for vocals, bass, and drums. Since the network functions using spectral masks, it consequently can also separate any other sources grouped into an "other" stem.

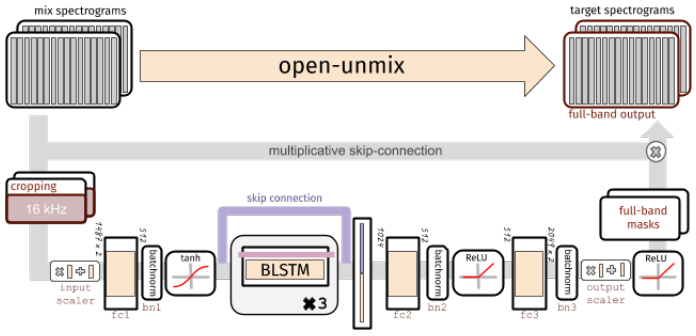

Figure 1: Diagram of open-unmix model

The full model of the open-unmix algorithm is shown in figure 1. At the core of the open-unmix algorithm are 3 layers of bi-directional LSTMs. LSTM stands for Long-Short-Term-Memory. These are layers specifically designed to retain informational context on a long term scale. In theory, such a feature is achieved by the much simpler RNN (residual neural network) layers. However in practice, the much more deliberate LSTMs, with multiple gates for determining when to remember and forget new and old information, are much more effective for applications that require such temporal context. The model also features fully connected layers for encoding and decoding the data, and batch normalizations for stabilization. The creators of open-unmix, in addition to supplying all their code on github, also provided a pre-trained, ready to use model, which I elected to use.

3 Automatic Music Transcription

As with many problems deep learning has attempted to solve, automatic transcription suffers from a lack of suitable training data to train and test an effective model. Most of the existing models I was able to find are designed to transcribe piano audio, all trained with the same "MAPS" dataset: MIDI Aligned Piano Sounds (1). This dataset comprises of 31GB of piano recordings in three forms: wav files, midi files, and text files containing note events.

3.1 Implementation

My model and code are heavily based on two other models trained on the same MAPS dataset, created by Jon Sleep (4) and Qiu Qiang Kong (5), respectively. At the highest level, the network will convert wav filesof piano recordings into midi files. First the mel spectrograms of the wav files are calculated to be used as inputs. I chose mel spectrograms to better represent the logarithmic structure of harmonics present in tonal music. Though my ultimate output will be midi files, my network uses MAP's text files to train. Each line of the text files contained the midi pitch, onset time, and offset time of a note event. These are parsed and converted into a piano roll representation with the same number of timestamps as the spectrogram of the audio file it corresponds to. To feed data into the network, the spectrograms and piano rolls of all files are combined into a single matrix, appended along the time axis so that the dimension of the frequency and midi bins stays the same.

From a naive standpoint, the frequency values from a single slice of time from the spectrogram should be enough to figure out the notes being played. However in reality, the spectral content of notes often changes over its duration. To account for this, the input to the network is instead a window of frames, matched to the ground truth of the piano roll at the timestamp of the window's center. To make full use of all information within the timestamp, I decided to follow a design similar to Qiu Qiang Kong's, using four fully connected layers with a hidden layer size of 500 weights. However, instead of Kong's 3-frame window, I decided to use a 7-frame window, similar to Sleep's algorithm. I reasoned that this would provide more temporal context for the algorithm to use. With a spectrogram hop size of 512 and a sampling rate of 22050 Hz, this window corresponds to about 0.16 of a second.

4 Combining the Models

While open-unmix's results were formidable, the model was still limited to separating vocals, bass guitar, and drums from the rest of the mix. So, with a automatic transcription algorithm targeting pianos, I primarily sought to run my models on a piece that only contained piano, vocals, bass, and drums, so that the "other" category of separation would theoretically be piano-only.

For the full two-model pipeline, an audio file is provided as the input and a midi file is the output. To connect the two models, the waveform vector output of the source separation can be passed in as the input for the automatic transcription network (after converting to mono).

4.1 Test-case file

My test case file is My Friends, a song from the group Oh Wonder. It features two vocalists, a piano, and strings. I chose this song as several different sections in the songs feature different combinations of instruments: piano only, piano and vocals, and piano, vocals, and strings. Although the model was trained specifically for pianos, I was curious to see what the effect of the strings would be in the final output.

4.2 Raw Results

For the open-unmix output of the test case file, non-target sounds were impressively attenuated, but not perfect. Listening to the stems, the filtered out sounds could still be heard albeit much quieter and with a somewhat garbled tone. The distortion was fittingly reminiscent of audio effects applied in the frequency

Model	F-measure
ConvNet (3)	74.45
Qiu Qiang Kong (5)	75
AMTNet	54.9

Table 1: F-measure results in comparison to other piano transcription networks

domain such as whisperization or robotization. However, the target instrument or instruments being separated were very clearly present with almost no perceivable distortion to them. For the test file, the vocals were attenuated but still noticeabl in the background, though the piano was remarkably clear.

In contrast, my own network for automatic music transcription was much less successful. My best evaluation metrics were around 76% precision and 43% recall, with an F-measure of about 55%. In comparison to other works on automatic music transcription, these numbers are decent but by no means the best, as seen in table 1. But while these are poor numbers, they should be contextualized by the form of the ground truth vector, in this case the piano midi roll. To obtain a perfect precision, not only can there not be any wrong notes, but the correct notes must start and end at exactly the right time. So, to get a better understanding of algorithm's performance, I more closely examined the final midi output. The output had 2 primary failings, firstly that the timing of the transcribed notes was inconsistent and sporadic, and secondly that there were some added notes that were audibly wrong. In particular, the transcribed notes had very brief durations, which was a major factor in the low recall value. In addition, simultaneous notes were rarely transcribed as starting at the exact same time, typically being offset by some fractions of a second. These temporal irregularities, in addition to handfuls of erroneous notes, added up to outputs that were difficult to compare to the original input. However, correct notes were identified, as evidenced by the evaluation metrics and listening to the playback.

Both models' weaknesses show up in the final output. While the test case file's final output is more targeted to the piano transcription, the poor temporal behavior and wrong notes are still present from the transcription algorithm. In addition, though heavily attenuated, the vocals that leaked past the open-unmix algorithm show up in the final transcription. In particular, areas with the leaked vocals seem to increase the likelihood of off-key notes. The sections with string instruments similarly seemed to be off-key. I would hypothesize this is due to the introduction of unfamiliar harmonic structures (timbres) even if the other instruments are in the same musical scale. Lastly the final midi output has no way of judging the correct volume for each of the notes and the algorithm leaves them all at the same constant level.

4.3 Post processing and other future steps

Considering the main two weaknesses of the output I believe these can be alleviated in a few ways, particularly in a digital audio workstation (DAW). Firstly quantizing the is a feature available to most DAWs. Quantizing automatically aligns notes to the closest note subdivison, with the smallest allowed subdivision set by the user. This will help especially on chords where multiple notes are played usually simultaneously. Furthermore, to make up for the extremely short note duration, most DAWs allow the user to select all midi notes and extend their duration. Taking these actions converted the outputs into much more recognizable forms. Outside of DAWs, a preprocessing action that should work for most mainstream music is removing all notes that do not fit the musical scale of the piece. The scale can be retrieved by using tools such as Krumhansl and Kessler's 1982 tone profiles, which use the probability distribution of occurring notes and matches them a scale's profile.

In terms of improving on the model itself I did not have the time to reload and train my data using different window sizes. I would have liked to have seen whether a window size of 7 may have been providing too much or too little temporal context to the model. Furthermore, I would be interested in taking a closer look at the training dataset. The few files I checked were fast paced and had complicated melodies, and I suspect it maye have influenced the temporal irregularities found in my output. I only ended up using around half the MAPS database as well, so its possible using more data would have built a more robust network.

References

[1] Emiya, Valentin, Roland Badeau, and Bertrand David. "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle." IEEE Transactions on Audio, Speech, and Language Processing 18.6 (2009): 1643-1654.

[2] Stöter, Fabian-Robert, et al. "Open-unmix-a reference implementation for music source separation." (2019).

[3] Sigtia, Siddharth, Emmanouil Benetos, and Simon Dixon. "An end-to-end neural network for polyphonic piano music transcription." IEEE/ACM Transactions on Audio, Speech, and Language Processing 24.5 (2016): 927-939.

[4] Sleep, J. "wav2mid: Polyphonic Piano Music Transcription with Deep Neural Networks." (2017), GitHub repository, https://github.com/jsleep/wav2mid/

[5] Kong, Qiu Qiang. "Automatic music transcription (AMT) of polyphonic piano using deep neural network." (2018), Github repository, https://github.com/qiuqiangkong/music_transcription_MAPS/

5 Appendix

5.1 Learned Lessons and Conclusions

While the full systems does not work quite as well as I would have hoped, I certainly learned a lot about the capabilities and limitations of current neural networks in source separation and automatic transcription. This was also my first time trying to run a full network from start to finish working with a large database. My main challenges were keeping track of the dimensions of all the data and learning better practices for dealing with large datasets. It was particularly helpful to learn from Sleep and Wang's codebases, in particular the segmentation of loading and storing processed data before training. It gave me a greater appreciation of the time involved in each step of the process and the importance of checkpointing your work during development so that I wouldn't need to complete re-do hours of code being run.