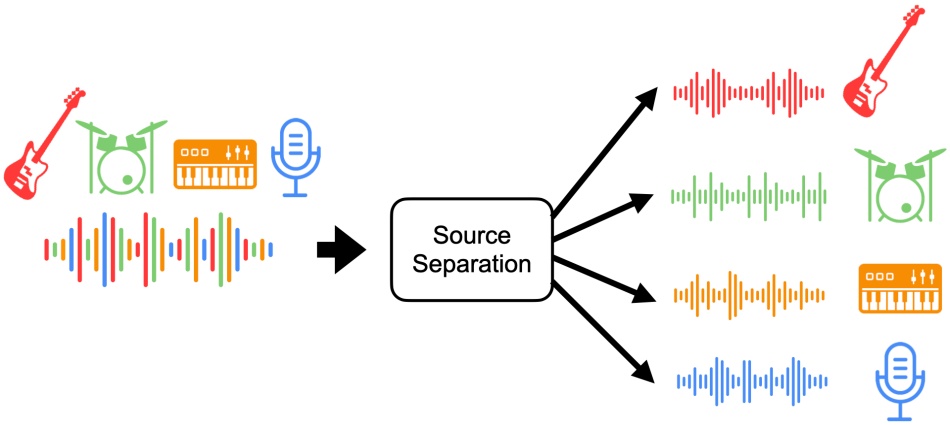
Genevieve LaLonde
CS510-Computers, Sound and Music
Prof. Dr. Bart Massey
6 June 2022

Music and Sound Separation

In this machine learning research project I have reproduced the work of Kong et al in their paper "Decoupling Magnitude and phase estimation with deep ResUNet for music source separation" (2021). Music source separation involves the separation of a music track into its constituent instruments, including audio. This topic interests me because I am pursuing Natural Language Processing. Music separation lends itself to audio separation, which can be applied in areas such as speech enhancement, and audio clarification. In particular I was interested in determining how effective a model trained on picking vocals out of songs would be at picking audio out from background noise. This required setting up their algorithm bytesep, then running the pretrained model on a different dataset and evaluating the output. Due to incompatibilities with the museval evaluation algorithm I was not able to statistically measure results. However, I was able to run bytesep and found some interesting results along the way.



As an intermediary step after implementing bytesep, I ran it on a variety of types of music, and found that it has an inherent weakness for classical jazz. While I might also say that I have a weakness for classical jazz, this algorithm's weakness is not a form of appreciation, but rather the inability to differentiate horn based accompaniment such as trumpets or saxophones from the vocals. In all the jazz tracks that I ran bytesep on, it included horns in the vocal track that it separated out. This was particularly noticeable on the classical jazz tracks. However it was also noticeable on the modern jazz track "Cry me a river" sung by Natalie Cole. At time 4:15 into the song there is a saxophone solo which is clearly heard in the separated vocals. I tested this further on other modern tracks with prominent horn solos, such as the ska group Less than Jake. While most of the horn parts were able to be removed from the vocals, they were more prominent when they more closely resembled a melody line. This can be noticed in particular in their track "Gainesville Rock City" which has a horn introduction that gets included in the vocal track, as well as only partially deactivating the horns later at about 4:18 so they almost end up sounding like a backup singer. These tracks can be found in my github account[1].

[1] https://github.com/coding-gen/cs541/tree/main/project/separated_vocals


Left: Duke Ellington with his band, including the horn section on the left.
Right: Less Than Jake, with horn section playing the show.

I was also able to run bytesep on several noisy audio tracks from the LibriMix dataset. I found that it was able to remove these simple background noises. However, bytesep typically preserves all audio in the vocal track it produces. Therefore it is not currently suitable for separating speech out from noisy background that includes speech, such as you would encounter in a typical crowded cafe. Therefore I would recommend holding all clandestine meetings at cafes, as opposed to a noisy rock show. I suspect that with further training on data with labeled horn tracks, bytesep would be able to learn to separate these out just as it has the other non-melodic instruments. However I suspect it will take much more work before it will be able to separate audio from different speakers.

Related Work

As an initial step in this project, I reproduced the work of Kong et al who created the bytesep algorithm which implements ResUNet for music source separation. The chief highlight of their work was addressing greater variety in magnitudes, separately estimating phase information, and using a deep residential UNet hardware. They used the MUSDB18 dataset, which is a collection of 150 songs. These are separated into tracks for vocals, drum, bass, and other. Together the latter three are considered the accompaniment. This dataset is commonly used in music separation because it is open, relatively large, and already separated which lends itself to supervised learning. They used the museval algorithm to evaluate their results. Museval measures the signal to distortion ratio, which is the log ratio differing the estimated projection on the ground truth, against the volume of everything else.

A lot of research papers in the area of music and audio source separation use the same dataset, and the evaluation tool. For example Défossez et al in “Music Source Separation in the Waveform Domain” (2021) similarly used MUSDB18 and museval. They proposed their algorithm called Demucs, which implements a waveform to waveform model with U-Net structure and bidirectional long short term memory (LSTM). In particular they were comparing the typical spectrogram model with their waveform domain model. Their results chiefly helped reduce artifacts in the results. This waveform model was most effective at separating bass and drums, but suffered on separating vocals and other sources.

VM Setup

I hosted my work on a virtual machine as a compute instance in the Google Cloud Platform (GCP). At first I created a very small host with the default size of e2-medium which is 2 vCPU and 4 GB of ram. This enabled me to save costs as I set up bytesep and its requirements, which took an unexpectedly long time. The full detailed instructions are available on my github as sound_project_vm_setup.md. Essentially it requires setting up python 3.7 in a virtual environment, fixing some typos in the repo's documentation and files, and installing several system level audio libraries that are commonly used with python, but not included in the requirements for bytesep. Determining exactly what was required took a lot of trial and error and not a little luck. Finally I was able to run bytesep to separate the vocals from some songs. However it was taking 10-20 minutes to process a single song, so I stopped the instance and reprovisioned it as a compute optimized c2-standard-8 model with 8vCPU and 32 GB of memory. This brought the runtime for "Magnetic North", by Less than Jake from 1160.948 s to 105.891 s. An example usage for processing this single file is shown below. As can be seen, it is necessary to use the --cpu param if your system does not have any GPUs like mine. This disables the default CUDA option. It is worth noting that if you deploy a GCP compute instance without any GPU, it is not possible to adjust the VM to add them later. Therefore be sure to initiate your instance with at least one GPU if you intend on using them later. Additionally it can be seen that the default PySoundFile fails, though I have not been able to find details as to why. The backup option audioread works though.

```
$ python -m bytesep separate \
>   --cpu  \
>   --source_type="vocals" \
>   --audio_path="../data/own/lessThanJake/magnetic-north.mp3" \
>   --output_path="separated_results/test_output.mp3"
Using cpu for separating ..
/home/gen/venv/lib/python3.7/site-packages/librosa/core/audio.py:162:
UserWarning: PySoundFile failed. Trying audioread instead.
  warnings.warn("PySoundFile failed. Trying audioread instead.")
Separate time: 105.891 s
Write out to separated_results/test_output.mp3
```

Later on, I did need to expand the disk capacity. Originally I set the VM very small, but ended up expanding it larger and ever larger as I pulled down more data models and determined they were unsuitable. Eventually after settling on the LibriMix dataset I set it to run and generate track mixes. Considering the two source datasets were only 30 GB and 49 GB I was shocked when my 400 GB system ran out of disk space. Upon further reading of their github documentation[2] I found that the generated data is expected to consume 430 GB in addition to the size of the datasets it is generated from. Unfortunately I could not size my instance above a disk size of 450 GB as GCP raised an error that there was not enough disk space available in my region. I determined I could simply use the files I had already generated. However in the future I would recommend to both create a compute instance with the intended disk size if you need a large disk, and to use the generator options to produce only certain types of audio mixtures as indicated in their repo. It is possible to edit the generator script to modify the number of sources, sample rate, mixture mode, and type of mixture. Were I to regenerate the data, I would have only generated at the highest sample rate of 16 kHz, only the min mixture that ends the resulting mixture when either of the audio samples ends, and not generated the mixture of only speech since that is not suitable for my purposes.

[2] https://github.com/JorisCos/LibriMix

Data

As mentioned above, the models for bytesep and demucs were trained on the MUSDB18 dataset, which is composed of 150 full length music tracks separated into the isolated tracks for drums, bass, vocals, and others. It also provides a mixed track which is composed of all of these as a full song. It is split into 100 training songs, and 50 testing songs. Each of these songs themselves come from a variety of open sources, as detailed on their site[3]. As they are essentially labeled by already being separated into the target isolated tracks, this data is suitable for supervised learning. The tracks are recorded in stereo format and at 44.1kHz.

However I was more interested in applying the bytesep pretrained model to the task of audio source separation. There are also several widely known audio source separation datasets available, however I had difficulty locating one which is open source and already composed of both audio, and a noisy background. Originally I worked with the VCTK dataset. This is a very large audio dataset from 110 English speakers with different accents. They are recorded reading various things from newspaper selections, to the rainbow passage. This is a very large dataset, however it does not have any background noise.

Thus I looked into the WHAM! dataset of background noises from Whisper[4].This is a collection of noises recorded in restaurants, cafes, bars, and parks, with attention paid to removing any intelligible speech. There are also several varieties of this dataset offered. WHAMR! Is a version which also seeks to include reverberating types of noise. These add an additional layer of complexity that is not often found in simulated speech over audio, but is often found in the real world. Additionally Whisper offers a pre-processed form of the dataset for combination with the WSJ0-2mix dataset. This includes clipping the noise samples to match the time of the WSJ0-2mix samples. The latter is a dataset composed of readings from the Wall Street Journal. Unfortunately these datasets are not premixed, as WSJ0-2mix is not open source. I was able to locate it for sale to non-members at $1500 from the Linguistic Data Consortium[5]. Unfortunately this was out of my price range.

Initially I attempted to mix the VCTK and the WHAM! datasets myself. WHAM! does include data generation scripts and documentation. However in the process, I located a dataset which is already premixed. LibriMix was composed as an alternative to mixing WHAM! And WSJ0-2mix, instead mixing the former with the LibriSpeech dataset. Not only is it open source, but its creation in 2020 sought to overcome certain weaknesses of the WSJ0-2Mix dataset. In particular, generalizability to audio not derived from the same dataset.

Evaluation

For models trained on the MUSDB18 dataset, it is common to use the associated museval SDR evaluation metric. Museval calculates the signal to distortion ratio, by the formula and loss function below. A higher result represents a better score, and in theory a perfect score would be infinity.

[3] https://sigsep.github.io/datasets/musdb.html#musdb18-hq-uncompressed-wav
[4] http://wham.whisper.ai/
[5] https://catalog.ldc.upenn.edu/LDC93S6A

$$SDR(s, \hat{s}) = 10\log_{10} \frac{||s||^2}{||\hat{s} - s||^2}.$$

Museval is an open source python library made available on github[6]. It is built on top of several other tools by the same developers, including MusDB for processing audio tracks[7] and MusIO[8] for generating multitrack step files that museval requires. I spent a lot of time trying to get results into the format required by museval before determining that it is only applicable to the MUSDB18 dataset. I had misread a line in the repository's readme which states "museval can also be used without musdb". I took this to mean it could be used on audio separated from datasets other than the MUSDB18 dataset. However it actually means museval can be used without the musdb tools. In the process I did learn how to convert between mp3, wav, and mp4 files, and did get a partial evaluation of the vocals from Louis Armstrong's "A Kiss to Build a Dream On" as seen below.

```
 estimates {'vocals': array([[-0.00064087, -0.00024414],
        [-0.00082397, -0.00042725],
        [-0.00061035, -0.00027466],
        ...,
        [ 0.        ,  0.        ],
        [ 0.        ,  0.        ],
        [ 0.        ,  0.        ]]), 'accompaniment': array([[-0.00064087,
        -0.00024414],
        [-0.00082397, -0.00042725],
        [-0.00061035, -0.00027466],
        ...,
        [ 0.        ,  0.        ],
        [ 0.        ,  0.        ],
        [ 0.        ,  0.        ]])}
vocals
accompaniment
/home/gen/venv/lib/python3.7/site-packages/numpy/lib/nanfunctions.py:1114:
RuntimeWarning: All-NaN slice encountered
  overwrite_input=overwrite_input)
vocals          ==> SDR:     nan  SIR: 140.228  ISR: 184.388  SAR: 140.228
accompaniment   ==> SDR:   3.522  SIR: 140.229  ISR:   3.522  SAR: 140.228
```

[6] https://github.com/sigsep/sigsep-mus-eval
[7] https://github.com/sigsep/sigsep-mus-db
[8] https://github.com/sigsep/sigsep-mus-io

As you can see here, I modified the script to print out the estimates as well as the scores below. Here the SDR score was not able to be calculated for the vocals due to incomplete estimates. I determined this is because museval expects a .stem.mp4 file which appears to be generated from all for parts of the music tracks by the MusIO library. My data would be composed of only three tracks: one or two audio, and one background noise. Therefore my data is not suitable for being combined by sigsep MusIO and therefore will always return incomplete results when evaluated with MusEval. You can see below an example of the expected output when MusEval is run on one of the .stem.mp4 tracks from the MUSDB18 dataset:

```
 estimates {'vocals': array([[3.05175781e-05, 3.05175781e-05],
        [9.15527344e-05, 9.15527344e-05],
        [3.05175781e-05, 3.05175781e-05],
        ...,
        [0.00000000e+00, 0.00000000e+00],
        [0.00000000e+00, 0.00000000e+00],
        [0.00000000e+00, 0.00000000e+00]]), 'accompaniment':
       array([[3.05175781e-05, 3.05175781e-05],
        [9.15527344e-05, 9.15527344e-05],
        [3.05175781e-05, 3.05175781e-05],
        ...,
        [0.00000000e+00, 0.00000000e+00],
        [0.00000000e+00, 0.00000000e+00],
        [0.00000000e+00, 0.00000000e+00]])}
vocals
accompaniment
vocals          ==> SDR:  -1.775  SIR:  -1.772  ISR:  32.834  SAR:  27.927
accompaniment   ==> SDR:   1.781  SIR:   1.784  ISR:  35.266  SAR:  27.927
```

Results

I had a lot of difficulty in implementing the pre-trained model from Kong et al on my VM as described above. Once it was operational, I was astounded with the results. I made the mistake of first separating the vocals out of the Foo Fighters' "My Hero". This track starts out mostly instrumental, though since it is a live recording there is some cheering. This cheering is considered human speech and is included in the vocal track. This made it very confusing to listen to, and at first I thought I had gotten my tracks mixed up and this was a sound file for examples of cheers. Eventually Dave Grohl comes in with the vocals and it all makes sense. I am particularly impressed that the algorithm was able to pick up that the cheers were human audio. However this does not spell good signs for picking primary audio out of other audio, as detailed below.

While I was manipulating datasets and learning museval, I set bytesep to separate the vocals out of several music files from my own collection, in a variety of genres: rock, classic rock, classic jazz, modern jazz, male and female singers, operatic vocals over metal back, acapella. While the results are astounding in general, I found that bytesep did very poorly on classical jazz. It included a lot of instrumental sounds in the separated vocals. In all of the separated vocals, most include hints of cymbals or tiny bits of swelling music, or high and fast snippets in the vocals track. The presence of the cymbals can be explained, since they are a pure noise. They have been engineered to produce all frequencies and none at all. This makes them

sound to humans like random noise or “snow”. To an algorithm they are very difficult to distinguish. The same is true for fricatives "f" and "s" which are common in human audio. Even advanced speech recognition software has difficulty distinguishing between fricatives from spectrograms. However, in Louis Armstrong's "A Kiss to Build a Dream On", not only the cymbals, but the entire rhythm section and trumpet are also featured heavily in the audio. I had to go back and listen to the original before I could identify that the piano and most of the clarinets at least were filtered out. I dug into this further to attempt to determine what it is about horns in classical jazz that bytesep is having trouble with. I ran other similar types of music through the separator, including more classical jazz, big band swing - which prominently features trumpet, trombone, saxophone and clarinet - modern jazz, ska, and acapella. Bytesep appears to consider horns to be a part of the vocals when they carry the melody line, especially if they are old recordings. Thus, the saxophone at 4:40 in the modern Jazz piece "Cry me a river" performed by Natalie Cole was included in the vocals. However the prominent horns in the ska album "Borders & Boundaries" from Less than Jake were nearly always removed. The closer they were to following the melody line, the more likely they were to be included in the vocals, and the louder they were preserved. Thus in many tracks on the album, they almost sound like a backup singer, reflecting the actual vocals. This can be heard in the intro to "Gainesville Rock City" where the horn section plays the melody to start the song off, and is almost fully preserved in the byte separated vocals.

I also included the acapella album Joh Eh Ba Dop by the Lewis & Clark College acapella group Momo and the Coop, in order to determine how bytesep would handle vocalized approximations of a rhythm section and supporting instruments. It performed quite well in the first track, which was a disordered collection of sounds and speech as the group prepared to record. Bytesep removed background noises like the piano keys, tuner, and claps, while preserving the chaotic collection of voices. In "Billie Jean / Smooth Criminal" most of the beatboxing was removed, especially the bass parts, while fricatives in the beatboxing like "sh", "k" and "ts" were preserved in the vocals. These fricatives are actually intended to approximate the sound of cymbals. Since the bytesep algorithm had difficulty separating cymbals out from the vocals in all the other tracks I tested, I suspect that is why these parts of the beatboxing were preserved as well. See tracks 01 Joh Eh Ba Dop and 07 The District Sleeps Alone Tonight which contain a lot of beatboxing[9]. This pattern was repeated in the other songs. The bass parts of the beatboxing, as well as any base lines were nearly fully removed. Remaining vocal lines and vocal sounds were preserved.

I have also run bytesep on several of the mixed tracks from the LibriMix dataset, and have found no appreciable separation of audio from background speech-like noises. For example see all of the tracks in the 8425_audio directory. These are all segments of a reading, positioned over background speech-like noise. In the resulting “vocal” track, both the foreground and background are preserved. For other audio tracks where more than one audio track was combined with a background noise like a loud hum, bytesep was able to reduce the background hum, but again fully preserved both audio tracks in the vocals. As we can see then, training a model to pick vocals out of a music source is promising in terms of removing some background noise. However it is completely inadequate at the task of removing typical cafe or bar type noise which also features speech.

[9] https://github.com/coding-gen/cs541/tree/main/project/separated_vocals/acapella/vocals

Future work

My initial results after working with bytesep indicate it would perform quite poorly on the task of audio separation from a noisy background and would require further fine tuning. For example it still has difficulty with pure noises like cymbals, brushes on a snare, and fricatives. These were often preserved in the vocals track that bytesep produces. Additionally most background noises like claps were often still at least partially present. Also, a lot of background noise recordings represent cafes or bars, where the noise is other voices. In the acapella recordings as well as recordings with more than one vocalist, all the vocal parts were usually maintained. This includes backup singers, cheers, overlapping voices, and even horns that fulfilled the role of backup singers when they also followed the melody. The only vocal parts that were removed from the acapella recordings were base lines, which would have qualified as bass or accompaniment. The model was only trained to detect the four tracks: vocals, bass, accompaniment, and other. Therefore it does not appear to be able to distinguish between the primary vocals, and supporting vocals. While it would feed quite well into the downstream task of melody detection, I believe it would need to be trained to detect horns like trumpets and saxophones as another track, and primary versus secondary audio, before it could be used in separating audio tracks from backgrounds which contain voices.

In the future I would also like to continue to look into why horns were almost completely preserved in the old classic jazz recordings, but at least partially removed from more modern recordings. I originally assumed this was due to differences in audio equipment. On the one hand, horns were mostly able to be removed from the ska tracks, which supports the idea that older recording techniques lead to the confusion. However, when the horns more closely followed the melody line for a longer period of time, they were more often included. Additionally, the saxophone in the modern rendition of "I'm beginning to see the light" was also included in the vocals, which seems to rule out older recording techniques as the culprit. Therefore I suspect it is a fault of training. Since bytesep was not fed data labeled for horns, they ended up getting lumped in with the audio. The only other section they could have fallen into would be accompaniment or other. It is more often that accompaniment is performed by the rhythm section like piano, guitar, or bass guitar. I would like to investigate the training set further to determine what kinds of accompaniment it was trained on, and whether simply adding more data with horn sections would resolve the issue. Likely the original MUSDB18 dataset is a good starting point for training, and further fine tuning on more specific data would improve performance in both of the areas discussed.

Bibliography

Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge and Emmanuel Vincent. “LibriMix: An Open-Source Dataset for Generalizable Speech Separation”. 2005.11262, arXiv, 2020.

Alexandre Défossez, Nicolas Usunier, Léon Bottou, Francis. Bach. Facebook AI Research, École Normale Supérieure, PSL Research University. 2021. “Music Source Separation in the Waveform Domain”. 1911.13254v2, arXiv, 2021

Qiuqiang Kong, Yin Cao, Haohe Liu, Keunwoo Choi, Yuxuan Wang. ByteDance, University of Surrey. 2021. “DECOUPLING MAGNITUDE AND PHASE ESTIMATION WITH DEEP ResUNet FOR MUSIC SOURCE SEPARATION”. 2109.05418v1, arXiv, 2021.

Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, “MUSDB18 - a corpus for music separation,” 2017. 10.5281/zenodo.1117372 https://doi.org/10.5281/zenodo.1117372 2017.

Fabian-Robert Stöter, & Antoine Liutkus. (2019). museval 0.3.0 (v0.3.0). Zenodo. https://doi.org/10.5281/zenodo.3376621

Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, Jonathan Le Roux. Mitsubishi Electric Research Laboratories (MERL), Whisper.ai. “WHAM!: Extending Speech Separation to Noisy Environments”. 1907.01160v1, arXiv, 2019.