Music source separation in the waveform domain

Corentin Jemine and Mathias Beguin

Source separation

It is the task of disentangling mixed signals, e.g.:

- People talking over each other
- A crossfade between two pictures
- Instruments playing together in a song

A trained model is expected to regenerate the signals as they were before being mixed together.

This is typically a **degenerate** task, where the difficulty varies a lot between domains.

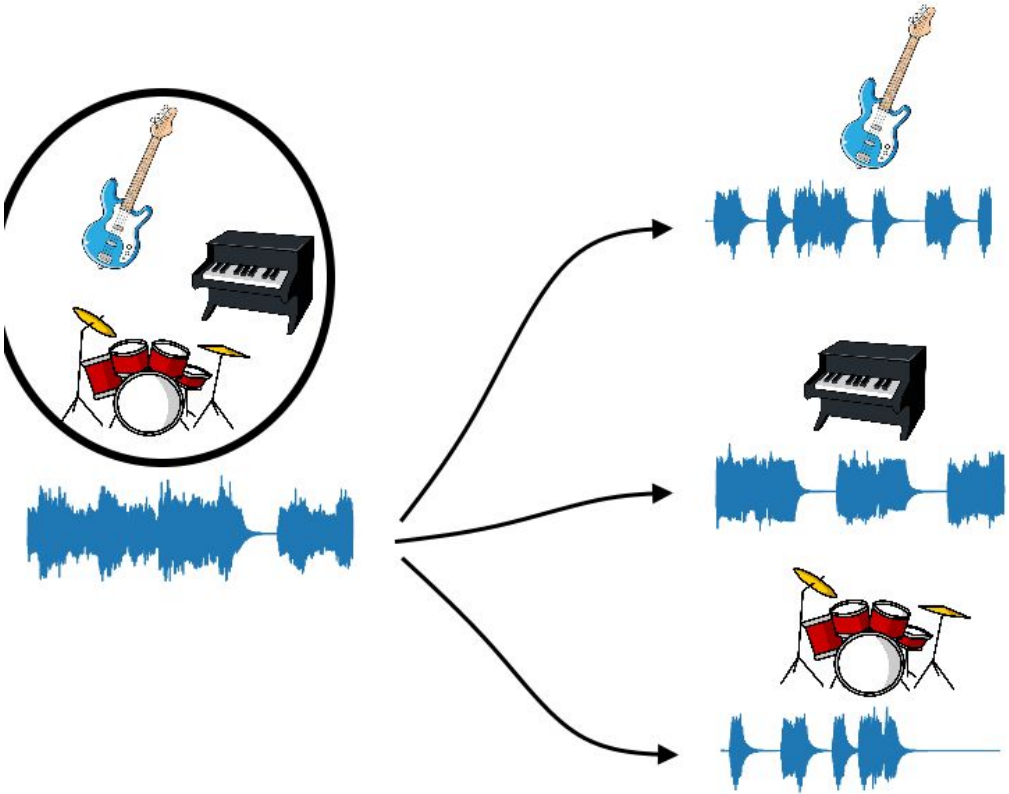
Source separation

The conversation (2018)

- State of the art in speaker disentanglement.
- Impressive results
- The network receives additional information from a clip of the mouth of the speaker.

The Conversation: Deep Audio-Visual Speech Enhancement (1804.04121)

Source separation in music



Given the combined signals, restore the individual signals.

There are variants:

- The model always predicts the same instruments
- The model only predicts instruments that it's made aware of
- The model guesses the instrument present in the audio, then extracts them

Source separation in music

Source separation in music is generally harder, and the state of the art is not as impressive (2018): http://jordipons.me/apps/end-to-end-music-source-separation/

Probable reasons:

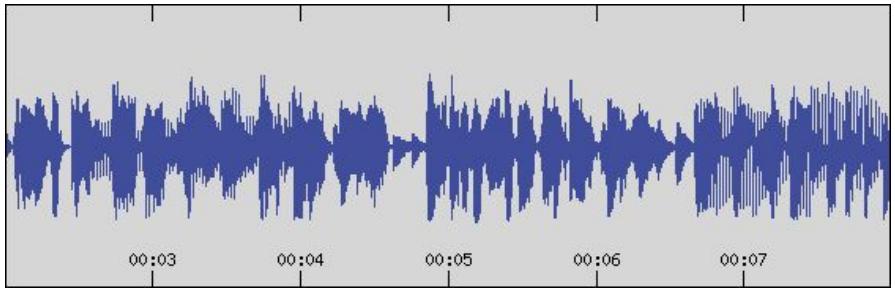
- Very complex domain (more on this later)
- Higher frequency range than the human voice
- Typically higher sampling rate than the human voice (44.1kHz vs 16/22kHz)
- Hundreds of different instruments, with some sounding very similar
- Lack of clean data

End-to-end music source separation: is it possible in the waveform domain? (1810.12187)

Dealing with audio in practice

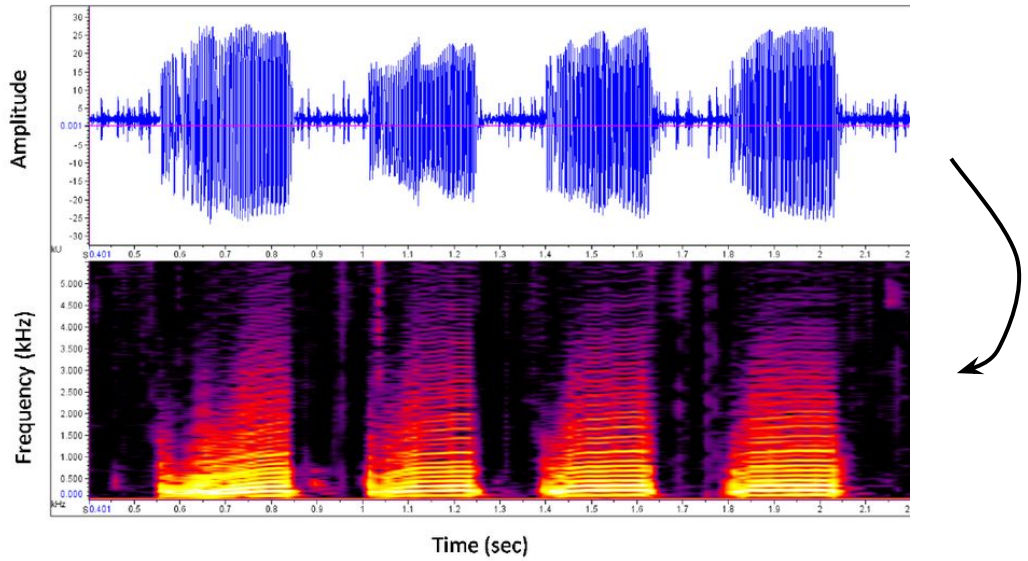
Digital audio is a dense 1-dimensional signal sampled a fixed frequency. In terms of numerical data, an audio waveform of 1 seconds sampled at 44.1kHz is represented by an array of 44 100 floating-point values.

- This is **very dense** data for a **1-dimensional** support
- Corresponds to a 210x210px RGB image worth of information *per second*
- It is also difficult to visualize



Dealing with audio in practice

In machine learning, one is more concerned about the features that can be extracted from the data, rather than the raw data itself. This leads to using time-frequency representations of the audio, i.e. **spectrograms**.



Dealing with audio in practice

Spectrogram representations are the standard in how to deal with audio in machine learning. State of the art text-to-speech and speech recognition methods are based on spectrogram features. Spectrograms are:

- **2-dimensional**, and can thus be treated as an image (2d convolutions, etc...)
- Much **less dense** than their waveform counterpart
- Easier to visualize

However, they are a lossy representation of the waveform! To recover a waveform, one needs a generative function. This leads to a **loss of quality** and generally does not allow for end-to-end training.

Dealing with audio in practice

Recent advances in text-to-speech manage to operate directly on the waveform domain (e.g. WaveNet).

We decided to try our hand at doing the same for source separation for music, trying to see if it is feasible.

Apparently, someone else had the exact same question:

End-to-end music source separation: is it possible in the waveform domain? (1810.12187)

(current state of the art)

Finding data

Finding data for our task is not easy:

- We need actual music for which we can get a clean waveform of each instrument individually.
- Some dataset exist, but the data is rarely clean

Our workaround: MIDI files

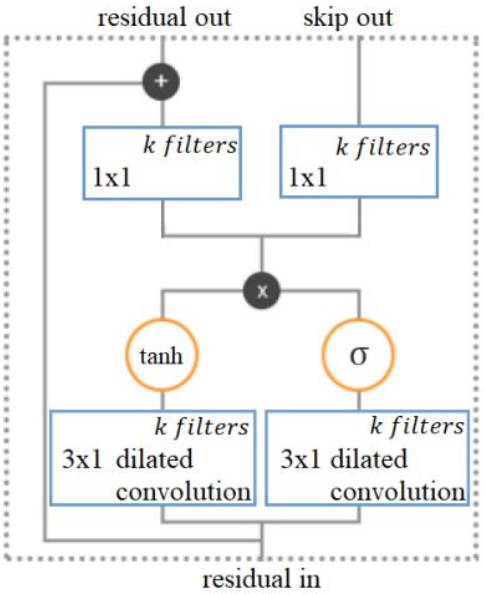
Finding data

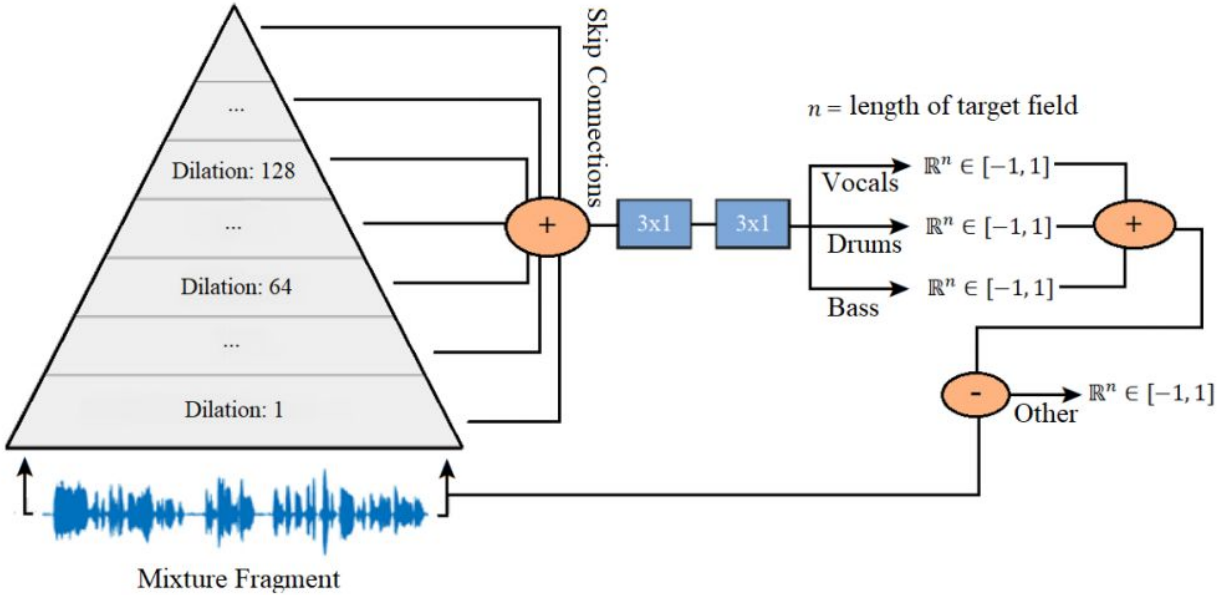
MIDI files are largely available for free on the internet. A MIDI file is simply a list of events that recreate a music when synthesized.

- They’re very small in size: three orders of magnitude smaller than the waveform it synthesizes
- They draw instruments from a standard set of 129 possible instruments.

  -> That makes our domain simpler

Wavenet-based model





Summary of our training findings

- Residual connections are good, especially for this kind of task (a part of the input must be restored at the output)
- Pyramidal dilations are good, especially for this kind of data
- A model that converges quickly might be good, but probably hasn’t learned a very complex function
- A model can only be as good as the loss function you give it is meaningful
- A trivial loss function is probably not enough for source separation, regardless of the complexity of the model

Samples

https://github.com/CorentinJ/SourceSeparation/blob/master/samples.md