INFO8010: Project Report

Mathias Beguin[1] and **Corentin Jemine**[2]
[1] *mathias.beguin@student.uliege.be (s140309)*
[2] *cjemine@student.uliege.be (s123578)*

I. ABSTRACT

We have experimented with source separation in polyphonic music. Source separation is the task of disentangling combined signals, so as to retrieve individual signals as they were before they were combined. Our method operates directly in the waveform domain.

II. TASK DEFINITION

The task is defined as extracting the signal that makes up for a subset of instruments in the waveform of a polyphonic music. While we intended to work on a single-model approach for any subset of source or target instruments, we did not go through with this goal and instead we prescribe a single model per source and target instrument subsets.

III. INSTRUMENT EMBEDDINGS

We did not carry out the idea of assigning embeddings to instruments, and learning a model that would be able to filter any instrument on demand. Our intention was to adapt [1] to instruments. To do so, we would have synthesized audio for each instrument taken alone, for all existing instruments in the dataset. The instrument encoder model would be trained to differentiate between instruments by creating an embedding per audio segment. Instruments that sound similar would be close in the embedding space, and those that don't would be far apart given a distance metric. This approach derives meaningful embeddings that can be used to condition a generative model.

However, we also noticed that the MIDI standard only references 129 instruments. It wouldn't be unreasonable to condition the model on a one-hot vector for all instruments rather than using an embedding.

IV. CHOICE OF DOMAIN

Most often, audio source separation is performed on a spectrogram of the waveform, rather than directly on the waveform. Waveforms make for a much denser representation of audio than spectrograms and are strictly one-dimensional, whereas spectrograms are two-dimensional, allowing to more easily leverage some spatial connectivity. However, the spectrogram is a lossy representation of the audio waveform that discards the phase. Therefore, models that generate a spectrogram of the audio need another generating function to translate the spectrogram into the waveform domain before it becomes playable audio. This transformation is not trivial. While algorithms such as Griffim-Lim exist, the methods that restore the audio with the best fidelity are based on deep learning models, such as WaveNet [2].

It is however increasingly common in deep learning to directly model the target waveform so as to train in an end-to-end fashion. In applications such as text-to-speech, the waveform is generated entirely from scratch. In our case however, the target signal is fully present in the input signal. Indeed, summing a set of audio waveforms corresponds exactly to a waveform of all signals mixed together, up to a normalizing constant. In light of these facts, we decided to experiment directly in the waveform domain.

Because of the high density of the waveform domain, we believe that a loss based on the difference of the generated and target waveform will not be effective. As a simple example, swapping samples two-by-two consecutively in a waveform will not sound any different than playing the original waveform. However, the loss between these two waveforms could be high. For this reason we operate on a spectrogram-based loss, while still remaining in the waveform domain. The short-time Fourier transform used to derive a spectrogram from an audio waveform is a differentiable operation and can therefore be used in the computation of a loss. Our loss function is thus an L2 loss between the linear spectrogram of the generated waveform and that of the target waveform.

V. DATA GENERATION

Our dataset consists of a large corpus of MIDI files scraped from the web. We generate samples by playing only the requested instruments in each music. Only musics that contain all source instruments are selected. Chunks of 5 seconds of audio are extracted, so as to keep the data batches light in size. Note that only 5 seconds of audio sampled at 44.1kHz already represents 220500 floating-point values. We use heuristics to filter out chunks that would not be effective for training, mainly by ensuring that the source and target waveform differ enough, i.e. that the source waveform contains both instruments that are in the source instruments subset and the target instruments subset. On our dataset, this corresponds to pruning out about 75% of the chunks.

Because the data is very dense while MIDI files are

fairly small in size, we cannot afford to cache the dataset on the disk. Instead, we build a pool of chunks at runtime and allow chunks to be re-used a second or third time for training. By making the pool large enough and keeping it shuffled, two batches are never identical and a same chunk will usually reappear a few steps after having appeared once. We adjust this chunk reuse factor in function of whether the model training or the data generation is the faster process. Note that with the training being done on the GPU, these two tasks are highly (but not quite embarrassingly) parallel.

VI. MODELS

Our baseline model is a simple 4-layer convolutional network with large one-dimensional kernels and ReLU activations. We have also experimented with the WaveNet-based model from [3] and the Wave-U-Net model from [4]. Unfortunately, neither of our implementations of these models are better than our baseline.

Our baseline trains fairly quickly: it reaches its minimum training loss after approximately 15 minutes on a GTX 1080 GPU, regardless of the chosen source and target instruments. Because the training is usually complete before a single epoch of the dataset, we do not need to validate on a separate set (provided that chunks are only used once per epoch). As is often the case in machine learning with digital audio, we do not have a metric that is more meaningful than the loss. One could conceive of a perceptual loss, but we did not explore that possibility.

Informal listening tests show that our model is very good at eliminating the instruments that should not be in the target waveform, but less so at restoring the signal of the other instruments.

[1] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification, 2017.

[2] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[3] Francesc Lluís, Jordi Pons, and Xavier Serra. End-to-end music source separation: is it possible in the waveform domain?, 2018.

[4] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. 2018.