The Journal of Open Source Software

Spleeter: a fast and efficient music source separation tool with pre-trained models

Romain Hennequin[1], **Anis Khlif**[1], **Felix Voituret**[1], **and Manuel Moussallam**[1]

1 Deezer Research, Paris

DOI: 10.21105/joss.02154

Software

- Review
- Repository
- Archive

Editor: Yuan Tang

Reviewers:

- @bmcfee
- @faroit

Submitted: 06 March 2020
Published: 24 June 2020

Summary

We present and release a new tool for music source separation with pre-trained models called Spleeter. Spleeter was designed with ease of use, separation performance, and speed in mind. Spleeter is based on Tensorflow (Abadi, 2015) and makes it possible to:

- split music audio files into several stems with a single command line using pre-trained models. A music audio file can be separated into 2 stems (vocals and accompaniments), 4 stems (vocals, drums, bass, and other) or 5 stems (vocals, drums, bass, piano and other).
- train source separation models or fine-tune pre-trained ones with Tensorflow (provided you have a dataset of isolated sources).

The performance of the pre-trained models are very close to the published state-of-the-art and is one of the best performing 4 stems separation model on the common musdb18 benchmark (Rafii, Liutkus, Stöter, Mimilakis, & Bittner, 2017) to be publicly released. Spleeter is also very fast as it can separate a mix audio file into 4 stems 100 times faster than real-time (we note, though, that the model cannot be applied in real-time as it needs buffering) on a single Graphics Processing Unit (GPU) using the pre-trained 4-stems model.

Purpose

We release Spleeter with pre-trained state-of-the-art models in order to help the Music Information Retrieval (MIR) research community leverage the power of source separation in various MIR tasks, such as vocal lyrics analysis from audio (audio/lyrics alignment, lyrics transcription...), music transcription (chord transcription, drums transcription, bass transcription, chord estimation, beat tracking), singer identification, any type of multilabel classification (mood/genre...), vocal melody extraction or cover detection. We believe that source separation has reached a level of maturity that makes it worth considering for these tasks and that specific features computed from isolated vocals, drums or bass may help increase performances, especially in low data availability scenarios (small datasets, limited annotation availability) for which supervised learning might be difficult. Spleeter also makes it possible to fine-tune the provided state-of-the-art models in order to adapt the system to a specific use-case. Finally, having an available source separation tool such as Spleeter will allow researchers to compare performances of their new models to a state-of-the-art one on their private datasets instead of musdb18, which is usually the only used dataset for reporting separation performances for unreleased models. Note that we cannot release the training data for copyright reasons, and thus, sharing pre-trained models were the only way to make these results available to the community.

Implementation details

Spleeter contains pre-trained models for:

- vocals/accompaniment separation.
- 4 stems separation as in SiSec (Stöter, Liutkus, & Ito, 2018) (vocals, bass, drums and other).
- 5 stems separation with an extra piano stem (vocals, bass, drums, piano, and other). It is, to the authors' knowledge, the first released model to perform such a separation.

The pre-trained models are U-nets (Jansson et al., 2017) and follow similar specifications as in (Prétet, Hennequin, Royo-Letelier, & Vaglio, 2019). The U-net is an encoder/decoder Convolutional Neural Network (CNN) architecture with skip connections. We used 12-layer U-nets (6 layers for the encoder and 6 for the decoder). A U-net is used for estimating a soft mask for each source (stem). Training loss is a L_1-norm between masked input mix spectrograms and source-target spectrograms. The models were trained on Deezer's internal datasets (noteworthily the Bean dataset that was used in (Prétet et al., 2019)) using Adam (Kingma & Ba, 2014). Training time took approximately a full week on a single GPU. Separation is then done from estimated source spectrograms using soft masking or multi-channel Wiener filtering.

Training and inference are implemented in Tensorflow which makes it possible to run the code on Central Processing Unit (CPU) or GPU.

Speed

As the whole separation pipeline can be run on a GPU and the model is based on a CNN, computations are efficiently parallelized and model inference is very fast. For instance, Spleeter is able to separate the whole musdb18 test dataset (about 3 hours and 27 minutes of audio) into 4 stems in less than 2 minutes, including model loading time (about 15 seconds), and audio wav files export, using a single GeForce RTX 2080 GPU, and a double Intel Xeon Gold 6134 CPU @ 3.20GHz (CPU is used for mix files loading and stem files export only). In this setup, Spleeter is able to process 100 seconds of stereo audio in less than 1 second, which makes it very useful for efficiently processing large datasets.

Separation performances

The models compete with the state-of-the-art on the standard musdb18 dataset (Rafii et al., 2017) while it was not trained, validated or optimized in any way with musdb18 data. We report results in terms of standard source separation metrics (Vincent, Gribonval, & Fevotte, 2006), namely Signal to Distortion Ratio (SDR), Signal to Artifacts Ratio (SAR), Signal to Interference Ratio (SIR) and source Image to Spatial distortion Ratio (ISR), are presented in the following table compared to Open-Unmix (Stöter, Uhlich, Liutkus, & Mitsufuji, 2019) and Demucs (Défossez, Usunier, Bottou, & Bach, 2019) (only SDR are reported for Demucs since other metrics are not available in the paper) which are, to the authors' knowledge, the only released system that performs near state-of-the-art performances. We present results for soft masking and for multi-channel Wiener filtering (applied using Norbert (Liutkus & Stöter, 2019)). As can be seen, for most metrics Spleeter is competitive with Open-Unmix and especially on SDR for all instruments, and is almost on par with Demucs.

	Spleeter Mask	Spleeter MWF	Open-Unmix	Demucs
Vocals SDR	6.55	6.86	6.32	7.05
Vocals SIR	15.19	15.86	13.33	13.94

	Spleeter Mask	Spleeter MWF	Open-Unmix	Demucs
Vocals SAR	6.44	6.99	6.52	7.00
Vocals ISR	12.01	11.95	11.93	12.04
Bass SDR	5.10	5.51	5.23	6.70
Bass SIR	10.01	10.30	10.93	13.03
Bass SAR	5.15	5.96	6.34	6.68
Bass ISR	9.18	9.61	9.23	9.99
Drums SDR	5.93	6.71	5.73	7.08
Drums SIR	12.24	13.67	11.12	13.74
Drums SAR	5.78	6.54	6.02	7.04
Drums ISR	10.50	10.69	10.51	11.96
Other SDR	4.24	4.55	4.02	4.47
Other SIR	7.86	8.16	6.59	7.11
Other SAR	4.63	4.88	4.74	5.26
Other ISR	9.83	9.87	9.31	10.86

Spleeter (Hennequin, Khlif, Voituret, & Moussallam, 2019) source code and pre-trained models are available on github and distributed under a MIT license. This repository will eventually be used for releasing other models with improved performances or models separating into more than 5 stems in the future.

Distribution

Spleeter is available as a standalone Python package, and also provided as a conda recipe and self-contained Dockers which makes it usable as-is on various platforms.

Acknowledgements

We acknowledge contributions from Laure Pretet who trained first models and wrote the first piece of code that lead to Spleeter.

References

Abadi, M. et a. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

Défossez, A., Usunier, N., Bottou, L., & Bach, F. (2019). Music source separation in the waveform domain. Retrieved from http://arxiv.org/abs/1911.13254

Hennequin, R., Khlif, A., Voituret, F., & Moussallam, M. (2019). Spleeter. Retrieved from https://www.github.com/deezer/spleeter

Jansson, A., Humphrey, E. J., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep u-net convolutional networks. In *Proceedings of the international society for music information retrieval conference (ismir)* (pp. 323–332).

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, arXiv:1412.6980. Retrieved from http://arxiv.org/abs/1412.6980

Liutkus, A., & Stöter, F.-R. (2019, July). Sigsep/norbert: First official norbert release. doi:10.5281/zenodo.3269749

Prétet, L., Hennequin, R., Royo-Letelier, J., & Vaglio, A. (2019). Singing voice separation: A study on training data. In *ICASSP 2019 - 2019 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 506–510). doi:10.1109/ICASSP.2019.8683555

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2017, December). The MUSDB18 corpus for music separation. doi:10.5281/zenodo.1117372

Stöter, F.-R., Liutkus, A., & Ito, N. (2018). The 2018 Signal Separation Evaluation Campaign. *arXiv e-prints*, arXiv:1804.06267. Retrieved from http://arxiv.org/abs/1804.06267

Stöter, F.-R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*. doi:10.21105/joss.01667

Vincent, E., Gribonval, R., & Fevotte, C. (2006). Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(4), 1462–1469. doi:10.1109/TSA.2005.858005