Demucs: Deep Extractor for Music Sources

Yuval Lifshitz 207639543

Amit Stein 207195280

Abstract

In this project, we attempted to recreate the results of the original HDemucs paper [1,2], which focuses on music source separation of vocals, drums, bass, and "other" from a mixed audio track with a semi-supervised learning approach. Since not much labelled data exists, the paper presents an auxiliary model in addition to the source separation model, tasked with labelling unlabelled data. This newly labelled data is then subsequently used in training the main source separation model. The exact implementation of the auxiliary model was not discussed in the paper nor provided in the official repo, so we decided to recreate only the main model using the original dataset, achieving ~1.2dB lower SDR compared to HDemucs when evaluating on the MUSDB dataset. Additionally, instead of the auxiliary model, we attempted to find a heuristic that would allow us to quantify the quality of a separated source without the use of a ground truth. We hoped this would allow us to create new samples of data by distilling the results of the pre-trained model provided by the paper, however, we were not able to find such a heuristic.

Introduction

Music source separation is the task of decomposing a mixed audio track into its individual components, such as vocals, drums, bass, and accompaniment. This has applications in music remixing, karaoke, and audio analysis. Traditional approaches often relied on signal processing techniques, but advances in deep learning have led to significant improvements in this area.

The Demucs paper [1], published by Facebook AI Research in 2019, introduced a novel architecture based on a convolutional encoder-decoder with bidirectional LSTMs and a U-Net structure, trained end-to-end on raw waveforms. HDemucs further improved on its predecessor's success, achieving state of the art performance by utilizing both temporal and spectral information.

Related Work

Music Source Separation is a well researched task in audio processing. There exist many solutions to this problem with varying efficacy, ranging from the purely signal processing based methods, to the newer machine learning methods.

Some of the better known classic solutions include non-negative matrix factorization [3], principal component analysis (PCA), independent component analysis (ICA) [4], and HMM-based prediction or segmentation ([5,6]), however each of these comes with a myriad of limitations and assumptions, such as some measure of independence between each of the signal sources, which - especially in the case of music - is difficult to maintain.

With deep learning's rise in popularity, many familiar fully supervised model architectures were used to attempt to solve the problem: fully connected networks [7], LSTMs [8], convolutional networks and RNNs were trained [9], using both raw waveforms and spectrograms. Prior to the Demucs paper, these models achieved a higher efficacy when training on spectrograms,

however, a U-Net structure named Wave-U-Net was also able to achieve considerable performance when training on raw waveforms [10].

The Innovation in the Demucs paper [1] is the semi-supervised approach presented, using an auxiliary model. The researchers trained the model to classify segments of audio and detect the absence or presence of each source from each segment. Now knowing if a source is absent in a segment, this allowed the researchers to splice in (known) audio signals of the absent source, providing a "ground truth" for that source on the segment, and using a weakened reconstruction loss on the remaining sources. This allowed the researchers to utilize unlabelled data that is much easier to obtain, in order to enhance and generalize the training procedure.

Since the original release of Demucs, many advancements have been made to the Demucs architecture, including HDemucs [2] - which we trained in this project – that incorporates simultaneous training with waveforms and spectrograms and HTDemucs [11] which replaced the LSTMs in the original architecture with transformers. Following these releases, other methods utilizing other architectures were further developed, such as Band Split RNN (BSRNN) [12], diffusion based models [13] and ResUnet [14].

Method

Before we present the HDemucs architecture, we first discuss the original Demucs architecture.

Demucs:
Demucs is a U-Net encoder/decoder model, with a biLSTM between the encoder and decoder layers which provides a longer context. The encoders and decoders are built symmetrically, such that for each encoder we have a corresponding decoder, and we have a total of 6 layers. Each encoder, other than the input encoder, receives the output of the previous encoder as input. Each decoder, other than the first one, receives the output of the previous decoder as well as the output of the corresponding encoder as its input. The first decoder is connected to the LSTM.

Each encoder layer applies a convolution to the input, followed by a RELU activation layer, and lastly a GLU activation, such that in total we end up doubling the number of channels after each encoder layer, while halving the length of the signal. The exception to this is the first encoder layer which sets the number of channels to 64, to match the input of the next encoder layer.

Each decoder layer begins with a convolution on the output of the previous decoder layer, followed by concatenation with the output of the corresponding encoder layer, a GLU layer, and finally a transposed convolution and a RELU layer. Although concatenation with the encoder output doubles the number of channels, the GLU collapses them back to the original length and allows the transposed convolution to create the desired number of channels.

A complete visualization can be seen in appendix 1.

Hybrid Demucs:
The improvement from Demucs to HDemucs lies in the addition of spectral information to the architecture. The two structures are very similar, where HDemucs adds another U-Net branch for handling the spectral information, and some intermediate mixed layers which take both domains into account. Other than the shared layers, the temporal branch of the complete structure is identical to the previous architecture. On the other hand, the spectral branch operates with a slightly different method.

The input of the spectral branch is an STFT of the mixture signal with a hop length of 1024. Each encoder layer in the spectral branch operates nearly identical to the temporal branch, the main difference being the domain in which it operates. While the encoders in the temporal branch compress the data on the time axis, the convolution in the spectral encoders compresses the number of frequencies in each bin, arriving at one "frequency" after the final encoder layer. This makes the exact hop length of the STFT critical, as following the final spectral encoder, we want the length of the output – in $Time \times Channels$ – to be equal to the output of the final temporal encoder.

The outputs of the spectral and temporal encoder layers are summed and inputted into the shared layers. These are comprised of one encoder and one decoder, each operating in the time domain, with the biLSTM connecting them. The output of the shared decoder is inputted into both the spectral and temporal decoder branch. This allows the model to share information between both domains and is the main advantage of the hybrid architecture.

Similarly to the original architecture and the encoder layers, the spectral decoder layers expand the number of frequencies after each layer. Each decoder receives the previous decoder's output and the output of the corresponding spectral encoder. The final output of the spectral branch is inversed using ISTFT and summed with the output of the temporal branch. This is the final prediction of the model.

A complete visualization can be seen in appendix 2.

Separation Quality Heuristics

We present 3 unsupervised heuristics we researched in attempt to quantify separation quality.

HNR

The Harmonic-to-Noise Ratio measures the ratio between periodic and non-periodic components of sound. Originally developed for analysis of speech signals [15,16], we believed it may provide a good measure for our separation quality. Given some mixture signal X, comprised as the sum of a periodic signal H and a noise (non-periodic) signal N, we define the HNR of X as

$$HNR \triangleq 10 \log_{10} \frac{\int |H|^2 \, d\omega}{\int |N|^2 \, d\omega} = 10 \log_{10} \frac{E_h}{E_n} \; [dB],$$

that is, the total energy of H divided by the total energy of N. Since we do not know H and N, we can estimate the HNR of the signal using its autocorrelation, as is further expanded upon in appendix 3.

HPR

The Harmonic-to-Percussive Ratio measures the ratio between the periodic and instantaneous (percussive) components of a signal. We believed we could use this heuristic, since we would always expect the 'vocal' and 'bass' separations to have a high HPR, and the 'drums' to have a low, possibly negative, HPR. Similarly to HNR, HPR is defined as

$$HPR = 10 \log_{10} \frac{\int |H|^2 \, d\omega}{\int |P|^2 \, d\omega} = 10 \log_{10} \frac{E_h}{E_p} \; [dB].$$

Where we swapped the noise signal N to the percussive component P of the original signal X. As opposed to HNR, when calculating HPR we first estimate H and P, and then explicitly calculate the HPR value using our estimates. We do this using an algorithm called Harmonic-Percussive Source Separation (HPSS) [17,18] which applies a median filter over the spectrogram of the signal. When applying the filter horizontally (taking the median on the time axis) we obtain the

harmonic component estimate, and when applying it vertically we obtain the Percussive component estimate.

Reconstruction Loss

We also attempted to use a simple l_2 norm on the difference between the input mixture and the sum of the output separations which we normalized by the length of the signal. This was suggested mainly in order to classify the separation of 'other' which could theoretically obtain any value of HNR or HPR even with a good separation.

Evaluation

Evaluation of the model was done on the MusDB test set by calculating the SDR of the separated signals and the ground truth stems, as is standard for this benchmark.

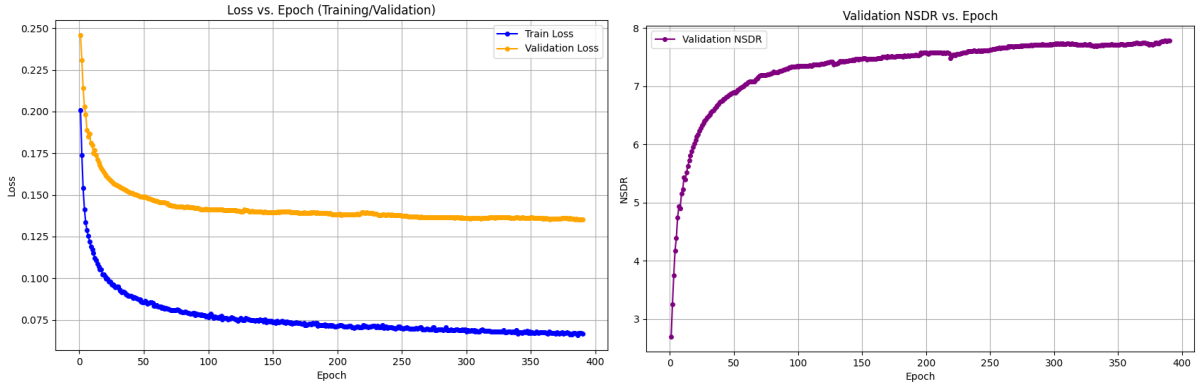
To evaluate the heuristics, we took 10 tracks, separated them using the pretrained HTDemucs and ranked the quality of separation of each track manually (subjectively). We tried to find some correlation between the rankings and the calculated metrics. A full list of the songs we used can be found in appendix 4.

Experimental Setup

Training as well as evaluation were run on the university's Slurm cluster, run in parallel on 8 GPUs. The GPUs we used were from the following: Titan XP, RTX 3090 and RTX a6000, whichever was available at the time. We trained only using the MusDB train set with a batch size of 32 (4 examples per GPU). We ran for 390 epochs, keeping intermediate results when the Slurm system stopped us. All other hyperparameters were set as the defaults of the script from the original repo.

Results

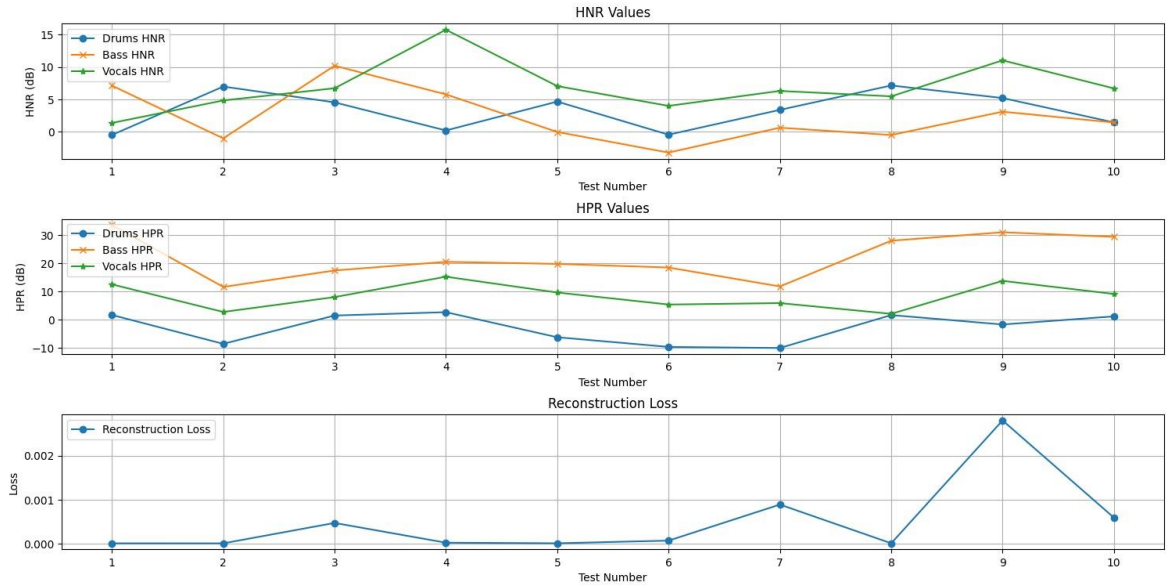
We present the convergence graph of the model training and the NSDR of the valid set on training:



We compare the SDR we obtained at various checkpoints to the SDR of the original models. Interestingly, we quickly achieved better results than the original Demucs model, suggesting the improvement to the architecture had a better effect than the larger training set:

Model	Train Set Size	SDR [dB]
$\boldsymbol{Original\ Demucs}$	100 + unsupervised set	5.67
$\boldsymbol{Original\ HDemucs}$	250	7.68
$\boldsymbol{Our\ HDemucs_{127}}$	100	6.09
$\boldsymbol{Our\ HDemucs_{218}}$	100	6.3
$\boldsymbol{Our\ HDemucs_{282}}$	100	6.37
$\boldsymbol{Our\ HDemucs_{390}}$	100	6.48

Additionally, We present the calculated heuristics in order from best separation to worst separation:



As we can see, there is no correlation between the observed (subjective) quality of each separation and the values of each metric, and so the proposed heuristics were ineffective in classifying the quality of separations.

Future Work

Although Demucs produces a remarkable result, it is still limited in that it separates the mixture into three tracks + others. In 2023 a new dataset named MOISESDB [19] was released and includes more detailed stem tracks, including piano, guitar, bowed instruments and more. The next iteration of Demucs should be able to separate these as well and could even maintain the same architecture, except the final output decoder which would split into more than 4 tracks.

Limitations and Broader Impact

The main risks of this model are that it may allow some form of copyright infringement. Most song releases only contain the mixture, and not each stem separately, so although you may own a copy of a song, you were most likely not intended to own, modify, redistribute, etc. any individual stem.

Bibliography

[1] Défossez, A., Usunier, N., Bottou, L., & Bach, F. (2019). Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*.

[2] Défossez, A. (2021). Hybrid spectrogram and waveform source separation. *arXiv preprint arXiv:2111.03600*.

[3] P. Smaragdis, C. Fevotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman. Static and dynamic source separation using nonnegative factorizations: A unified view. IEEE Signal Processing Magazine, 31(3), 2014.

[4] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. Independent component analysis. John Wiley &Sons, 2004.

[5] Sam T. Roweis. One microphone source separation. In Advances in Neural Information Processing Systems, 2001.

[6] Francis Bach and Michael I. Jordan. Blind one-microphone speech separation: A spectral learning approach. In Advances in neural information processing systems, 2005.

[7] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[8] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[9] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. Mmdenselstm: An efficient combi nation of convolutional and recurrent neural networks for audio source separation. Technical Report 1805.02410, arXiv, 2018.

[10] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. Technical Report 1806.03185, arXiv, 2018.

[11] Rouard, S., Massa, F., & Défossez, A. (2023, June). Hybrid transformers for music source separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1-5). IEEE.

[12] Luo, Y., & Yu, J. (2023). Music source separation with band-split RNN. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *31*, 1893-1901.

[13] Mariani, G., Tallini, I., Postolache, E., Mancusi, M., Cosmo, L., & Rodolà, E. (2023). Multi-source diffusion models for simultaneous music generation and separation. *arXiv preprint arXiv:2302.02257*.

[14] Kong, Q., Cao, Y., Liu, H., Choi, K., & Wang, Y. (2021). Decoupling magnitude and phase estimation with deep resunet for music source separation. *arXiv preprint arXiv:2109.05418*.

[15] Ferrand CT. Harmonics-to-noise ratio: an index of vocal aging. J Voice. 2002 Dec;16(4):480-7. doi: 10.1016/s0892-1997(02)00123-6. PMID: 12512635.

[16] Fernandes, Joana & Teixeira, Felipe & Guedes, Vitor & Junior, Arnaldo & Teixeira, João. (2018). Harmonic to Noise Ratio Measurement - Selection of Window and Length. Procedia Computer Science. 138. 280-285. 10.1016/j.procs.2018.10.040.

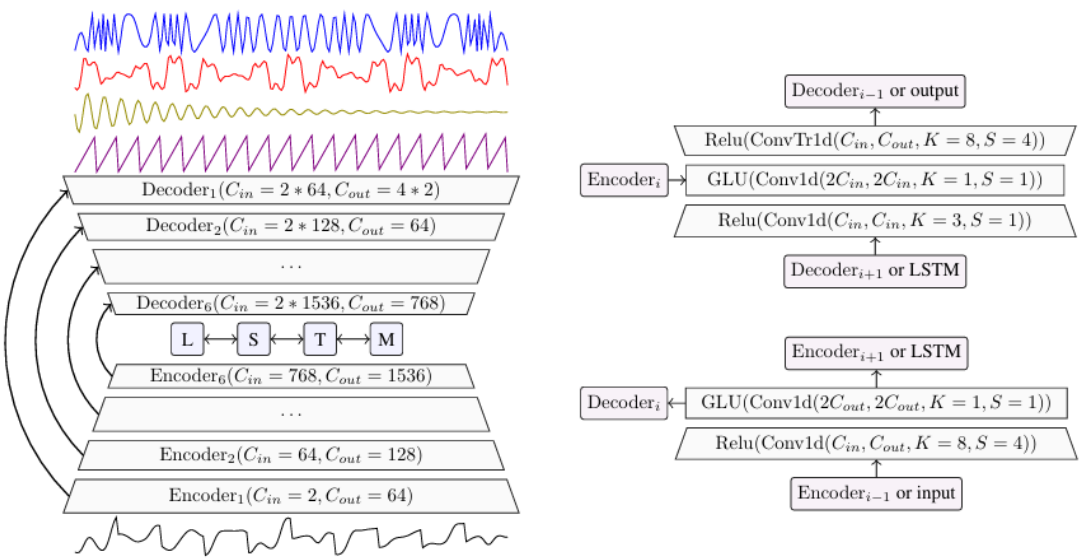
[17] Fitzgerald, Derry. (2010). Harmonic/Percussive Separation using Median Filtering. 13th International Conference on Digital Audio Effects (DAFx-10).

[18] Jonathan Driedger, Meinard Müller, & Sascha Disch. (2014). Extending Harmonic-Percussive Separation of Audio Signals. Proceedings of the 15th International Society for Music Information Retrieval Conference, 611–616. https://doi.org/10.5281/zenodo.1415226

[19] Pereira, I., Araújo, F., Korzeniowski, F., & Vogl, R. (2023). MoisesDB: A dataset for source separation beyond 4-stems. *arXiv preprint arXiv:2307.15913*.

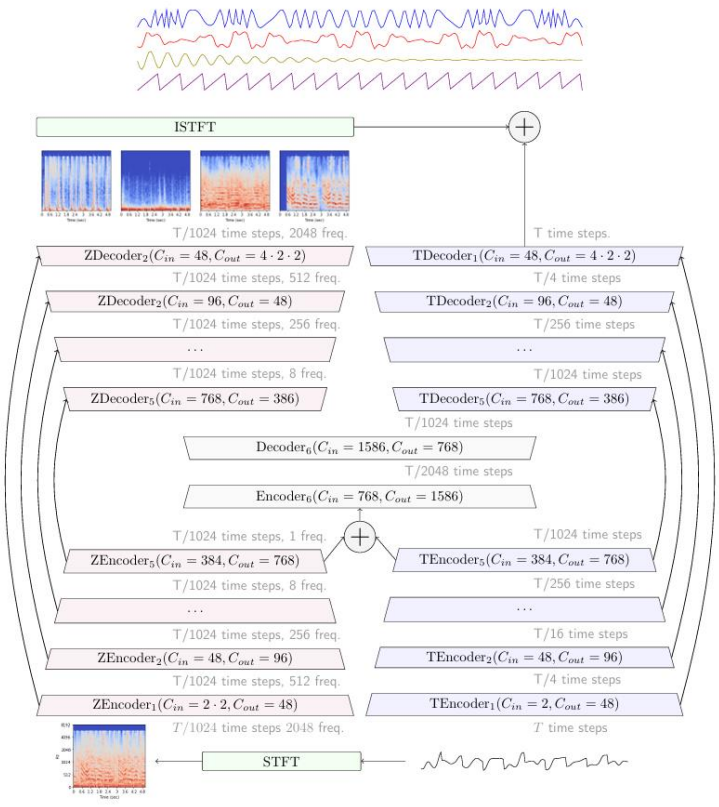
Appendix

Appendix 1 – Demucs architecture visualization:



On the left is the larger structure of the architecture, and on the right we zoom in to see the structure of each encoder and decoder. Figure from [1].

Appendix 2 – HDemucs architecture visualization:



On the left we see the spectral branch and on the right the temporal branch. Figure from [2].

Appendix 3 – Estimating the HNR of a signal through its autocorrelation:
We define the autocorrelation (and the normalized autocorrelation) of X as

$$r_x(\tau) \triangleq \int_{-\infty}^{\infty} x(t)x(t+\tau)dt\,, \qquad r_x'(\tau) \triangleq \frac{r_x(\tau)}{r_x(0)}.$$

We notice the autocorrelation receives a global maximum at $\tau = 0$ by definition, and that this value is also the total energy of the signal X. Additionally, were it a purely periodic and infinite signal with period T, the autocorrelation would also be periodic with period T, so we would get global maxima at every mT for a whole m. Since it is not purely periodic nor infinite, we define τ_{max} as the $\tau \neq 0$ which achieves the largest local maximum. τ_{max} serves as our estimate for the period T. Since X is the sum of H and N, the same is true for their corresponding autocorrelation functions. Therefore we get

$$r_x'(\tau_{max}) = \frac{r_h(\tau_{max}) + r_n(\tau_{max})}{r_x(0)} \approx \frac{E_h + r_n(\tau_{max})}{r_x(0)},$$

$$1 - r_x'(\tau_{max}) = \frac{r_h(0) - r_h(\tau_{max}) + r_n(0) - r_n(\tau_{max})}{r_x(0)} \approx \frac{E_n - r_n(\tau_{max})}{r_x(0)}.$$

Finally, we define our estimate for HNR as:

$$\widehat{HNR} = 10\log_{10}\frac{r_x{}'(\tau_{max})}{1 - r_x{}'(\tau_{max})} = 10\log_{10}\frac{E_h + r_n(\tau_{max})}{E_n - r_n(\tau_{max})}\ [dB].$$

Appendix 4 – List of songs for Heuristic Testing
The list is sorted by (subjective) quality of separation achieved using the pretrained HTDemucs model.

Song #	Title	Artist	Link
1	Honolulu March	George Kulokahai	Link
2	Another One Bites the Dust	Queen	Link
3	Die House	Kristofer Maddigan	Link
4	La Vie en Rose	Louis Armstrong	Link
5	Let's Groove	Earth, Wind & Fire	Link
6	Back in Black	AC/DC	Link
7	Uptown Funk	Mark Ronson ft. Bruno Mars	Link
8	Lose Yourself	Eminem	Link
9	שני משוגעים	עומר אדם	Link
10	Shake It Off	Taylor Swift	Link