Dual-Path Recurrent Neural Network for Audio Source Separation

Faculdade de Ciências da Universidade de Lisboa

António Estêvão fc58203@alunos.fc.ul.pt

Diogo Venes fc58216@alunos.fc.ul.pt

Guilherme Gouveia fc58176@alunos.fc.ul.pt

Abstract

Audio source separation is the task of isolating individual sound sources, such as vocals or instruments, from a mixed audio track. It's a challenging and active research area that has real world uses, for example in music production. Recent progress in deep learning, particularly with convolutional and recurrent neural network architectures, has significantly improved source separation performance. Notably, systems like Open-Unmix leverage time-frequency representations and achieve strong results on benchmark datasets like MUSDB18-HQ. Despite these advances, understanding how to best model both local and long-range spectral-temporal dependencies, especially in stereo audio, remains a challenge. Additionally, effectively incorporating phase information and minimizing artifacts is still a major hurdle. We implemented a dual-path recurrent neural network (DPRNN) trained on log-magnitude spectrograms of stereo audio from the MUSDB18-HQ dataset. Our model incorporates phase-aware loss and uses SpecAugment during training for improved robustness. When benchmarked against the Open-Unmix umxhq baseline, our DPRNN-ass achieved an average SI-SDR of 6.46 dB for vocal separation, while Open-Unmix reached 9.63 dB. Traditional SDR, SIR, SAR, and ISR metrics also showed our model to be competitive (even achieving a higher score in one case), though still generally outperformed by the more mature baseline. While our DPRNN-ass did not surpass the state-of-the-art, it demonstrated consistent learning behavior and highlighted the importance of design choices like loss function selection/composition and data augmentation. The architecture shows promise for future improvements and helps illuminate critical factors for effective audio source separation.

1. Proposal

1.1. What problem will be investigated and why is it interesting?

The problem we are trying to investigate is how we are able to separate specific parts of an audio track (e.g. Voice and Instruments/Accompaniment) using DL **Convolutional/Recursive** Neural Networks.
We find it interesting as a way to better understand how to apply this type of network to an audio source (which in theory should be harder than doing it on images but easier than doing it on video files) and because it is a great tool for sampling new tracks only with a specific combination of instruments/voice (*e.g.* we could try to generate a karaoke track without the voice of the singers or separate the bass line from a song for a bassist to learn it).

1.2. What sources will be reviewed to provide context and background?

IEEE ICASSP conferences, *Nobutaka Ito's* papers and others related with this specific problem, such as:
https://arxiv.org/pdf/2501.11837
https://arxiv.org/pdf/1804.06267
among others.

1.3. What data will be used? If new data is collected, how will it be gathered?

MUSDB18-HQ, which is an industry standard dataset of 150 full-length **stereo** tracks (~10 hours) of different genres split into 86 train tracks, 50 test tracks and 14 validation tracks. Each track comes pre-split into **mixture**, **bass**, **drums**, **vocal** and **other** *.wav* files, of which we use the **vocal** and **mixture** files.

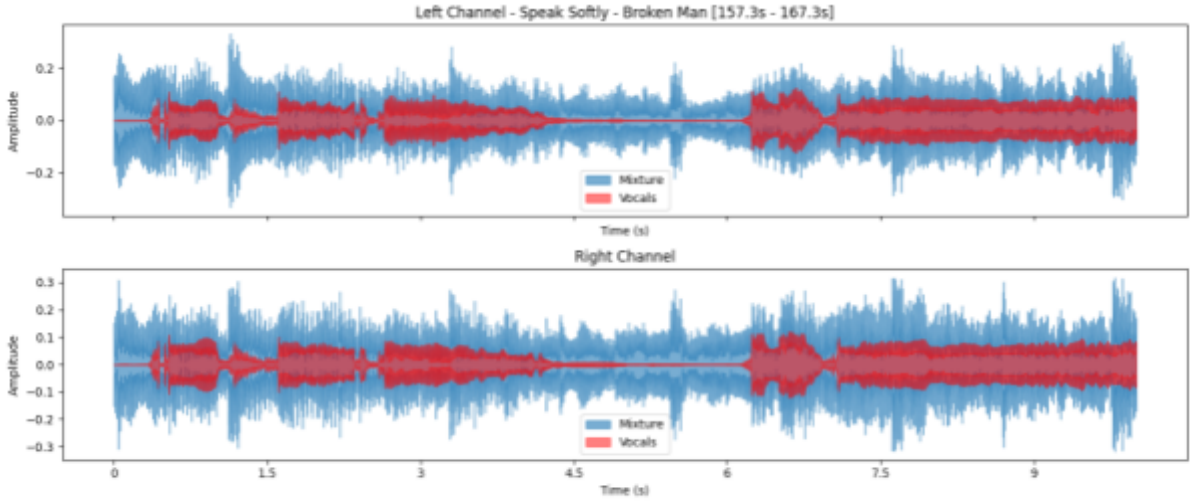


Figure 1: Waveform plot of one of the tracks

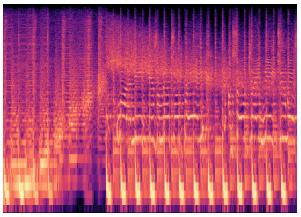
1.4. What method or algorithm will be used? If existing implementations are available, will they be utilized, and how? While an exact answer is not required at this stage, a general approach to the problem should be considered.

We could follow a more traditional approach without Convolutional Layers, where we just feed our Deep Neural Network model various samples of mixture and independent audio tracks of the same song (for a list of several songs) already transformed/analyzed with FFPs (Fourier Transform), STFTs (Short Time Fourier Transform) and MFCCs (Mel Frequency Cepstral Coefficient), or use Convolutional Layers. At this point, the problem was new to us and we were still understanding it.

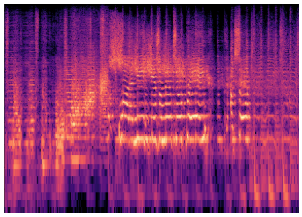
After some more research, we found out that RNN (recurrent neural networks) would be a good approach.

1.5. How will the results be evaluated? Qualitatively, what kinds of results are expected (e.g., plots or figures)?

We expect to obtain audio tracks as an output to be compared to the originally fed audio track, as well as some audio-related graphs, like spectrograms and waveform plots.



Mixture Spectrogram



Target (Vocals) Spectrogram

1.6. Quantitatively, what kind of analysis will be used to evaluate and/or compare the results (e.g., performance metrics or statistical tests)?

To analyse the performance of our model, we thought of comparing the clarity of each final separation because not every instrument is easy to "digest" and we thought, initially, that if a track has a lot of instruments, the harder it will be to separate them with success. So, to conclude, there could be automatic but also manual performance analysis, by giving a score for example for each final result depending on the quality. This was later achieved using industry standard performance evaluation metrics, like SDR, SAR, SIR and ISR. Afterwards, we also implemented the updated metric SI-SDR.

2. Milestone 1.

2.1. Data Pipeline with Preprocessing

We implemented a custom PyTorch-compatible dataset loader that:

- Loaded mixture and target stems (e.g., vocals) as stereo waveforms at 44.1 kHz.
- Supported optional random cropping (we used 10-second segments = 441,000 samples).
- Handled waveform loading using librosa.

However, we then discovered that we could simply use the *musdb* import to function as a pipeliner, which would make the process simpler and more efficient. Therefore, we settled on that approach.

Example visualization in **Figure 1** shows waveforms of mixture and isolated vocals.

Preprocessing also includes computing STFT to inspect **magnitude** and **phase** components. We confirmed that magnitude is easier to interpret, but phase plays a crucial role in reconstruction quality. Initially, we ignore phase during training and reuse the mixture's phase when doing inverse STFT for reconstruction—a common simplification in early-stage models.
Since we're benchmarking against **Open-Unmix** (which uses **Time-Frequency Domain**) and using **MUSDB18-HQ**, we will use **spectrogram masking** for a strong and interpretable baseline.

2.2. Baseline description: model name / architecture depth; input shape; loss; optimiser; training budget (epochs, LR, batch, GPU time)

We first implemented, **before changing to a RNN**, a 1D Convolutional Neural Network (CNN) to predict the vocal track from the mixture. The model architecture is as follows:

Baseline Model (Conv1DSeparator)
*Input shape: (batch_size, 2, 44100 * segment_time) [mono channel, 6-second segments]*
Architecture:
Conv1D(2 → 32, kernel=15, ReLU)
Conv1D(32 → 64, kernel=15, ReLU)
Conv1D(64 → 32, kernel=15, ReLU)
Conv1D(32 → 2, kernel=15, linear)
Loss: Mean Squared Error (MSE)
Optimizer: Adam
Learning Rate: 1e-3
Batch Size: 2
Epochs: 10
Hardware: NVIDIA GeForce RTX 2060 (via CUDA 11.8, PyTorch 2.7)

This baseline was chosen for simplicity and fast experimentation. The output is a time-domain waveform matching the target vocals.
It took about **14 minutes** to fully train this model in a GeForce RTX 2060-equipped laptop.

2.3. Preliminary results and training curves

After training for 10 epochs, both training and validation losses showed a consistent decreasing trend (mostly on previous runs):

Epoch	Train Loss	Val Loss
0	0.0054	0.0057
4	0.0023	0.0022
7	0.0022	0.0024
10	0.0023	0.0023

TensorBoard was used to monitor loss curves, and performance stabilized after ~5 epochs, having a **final test loss of 0.0020**.
We also used the Open-Unmix pre-trained model (`umxhq`) as a qualitative benchmark. It successfully separated vocals and bass from stereo input. The results were saved to WAV files and compared to our custom model. Open-Unmix's output was higher quality due to its more sophisticated architecture and large-scale training.

2.4. Known issues and next steps, including a detailed plan for the rest of the experiments

Using a traditional approach revealed key issues, notably the complexity of **phase spectrum prediction**. While source separation often uses only the magnitude for masking, reuses the mixture's phase for iSTFT, and accepts some quality loss (unless working in the time domain), phase cannot be ignored for high-quality waveform reconstruction.
We also **faced compatibility problems with libraries** like *museval*, *musdb*, *stempeg*, and *ffmpeg*, struggling locally and on Colab, ultimately resolving this by using **Miniconda**.
We opted for **Stereo over Mono** audio to preserve spatial fidelity, despite the higher computational cost, as Stereo captures potential differences between channels that reflect 3D spatial cues.
With just 10 samples and a small batch size, **overfitting is a significant concern**. We're still finalizing our architecture but have chosen **Open-Unmix** as the open-source baseline. Evaluation metrics require further understanding, and we must address class imbalance (~65% of segments are vocal-free).
After discussing with the professor, we learned our initially impressive loss scores stemmed from working in the **time domain**. Switching to the **time-frequency domain** resulted in much higher initial losses (starting

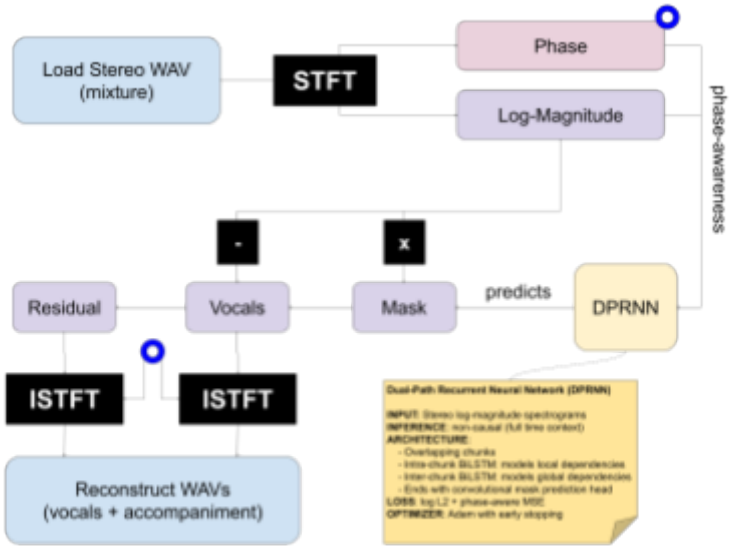


Figure 2: Methodology diagram

above 2), confirming the large performance gap between the two domains.

3. Milestone 2 and Poster Session

3.1. Training methods

The architecture eventually chosen for our source separation model was a **Dual-Path Recurrent Neural Network (DPRNN)**, which operates in the **time-frequency domain** using **stereo log-magnitude spectrograms**. The DPRNN is a powerful variant of RNN-based models that combines two types of bidirectional LSTMs:

- **Intra-chunk RNNs**, which process local patterns **within** small chunks of the spectrogram.
- **Inter-chunk RNNs**, which capture **long-range dependencies across** chunks.

This structure enables more effective modeling of both short- and long-term temporal context compared to a standard BiRNN. Since our setup assumes **Non-Causal Source Separation**, the model has access to **past, present, and future frames** when predicting at a given time step, allowing it to make more informed decisions. This is analogous to listening to a full track with the ability to scrub through time, rather than processing it in real-time.
Following the recurrent layers, a **convolutional mask prediction head** is used to produce the final separation output.
Training was performed on **6-second stereo spectrogram chunks** extracted from the **MUSDB18-HQ** dataset. To enhance robustness to noise and occlusions, **SpecAugment** was applied during training. The model was optimized using the **Adam optimizer**, with **early stopping** based on validation loss to prevent overfitting. The loss function combined **log L2 loss** with a **phase-aware MSE loss**, as recommended by the teacher in his feedback, to help reduce artifacts and preserve audio quality. **TensorBoard** was used to log training and validation losses, as well as loss curves for visualization. The methodology can be viewed in full in **Figure 2**.

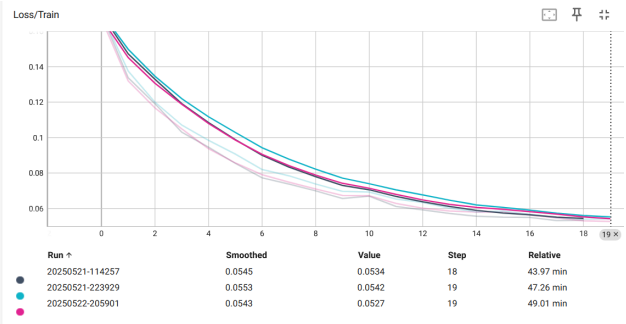
3.2. Experiments

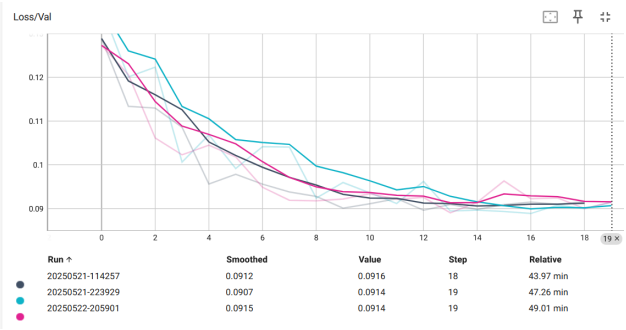
We used the Open-Unmix **umxhq** as a baseline model, a well established time-frequency domain pre-trained model, also trained on the MUSDB18-HQ dataset. Our DPRNN was pipped against the baseline model in single-track and multi-track (album) median scores. The metrics used included **SDR** (which is an overall measure of separation quality) **SAR** (which measures how clean the generated separated audio is from unwanted artefacts), **SIR** (which measures how well other unwanted sources were removed from the original audio) and **ISR** (which measures how well spatial characteristics were preserved in the generated audio). After the poster session, we also decided to use **SI-SDR** (Scale-Invariant SDR) as it “aims to remedy the issues with SDR, by removing SDR’s dependency on the amplitude scaling of the signal.”[8].

3.3. Results

By using Tensorboard from the beginning of this project, we were able to constantly understand and follow the learning curves of the losses of train and validation sets. As we can see, with the parameters we use by the time being, it seems that we are training the model to it’s limit,

which is good, as it does not overfit to the data, but also reveals that some changes (like hyperparameters) need to be made in the future if we really want to improve the scores of our model with the current data that we have.





We tested the 2 models for a random set of 25 test tracks (from the musdb test dataset).
The scores (average of the metrics across the set of 25 random tracks) in the table below reveal that our DPRNN for audio source separation, which is similar to the Open-Unmix, is still not as good as its opponent:

Target	Model	SDR (dB)	SIR (dB)	ISR (dB)	SAR (dB)
Vocals	Open-Unmix	5.259	12.508	14.385	6.607
Acc.	Open-Unmix	**12.951**	19.901	**21.255**	**14.464**
Vocals	DPRNN-ass	2.516	5.445	9.482	6.011
Acc.	DPRNN-ass	6.094	**22.865**	6.941	8.449

As mentioned previously, we then implemented the average SI-SDR metric across a different set of 25 random tracks, so it can't be inserted in the above table. Instead, we present it next:

Target	Model	SI-SDR (dB)
Vocals	Open-Unmix	**9.63**
Vocals	DPRNN-ass	6.46

Notice that for both models, when compared with the normal SDR score, the SI-SDR score is more or less the double, having a slightly bigger increase (percentaly) for our model.

Either way, we believe that we could still improve the system by doing more augmentations, using methods such as GridSearch exploration to find the best combinations of parameters and keep exploring already discussed ways (in scientific relevant papers) to improve the model performance, like we did by applying phase awareness, adding a second component to the loss which was very fundamental to get a somehow decent model for comparison with a state of the art model for this type of task, such as Open-Unmix.

3.4. Contributions

All group members contributed actively to the success of the project. Guilherme took the lead on the development side, with Diogo and Antonio closely involved in understanding the implementation, helping with testing and evaluation, understanding and analyzing the results and metrics, trying to improve them and assisting in resolving issues as they came up. Diogo and Antonio were primarily responsible for writing the report, organizing the content, and clearly presenting the results, with Guilherme providing input and clarifications on specific details when needed. Diogo and Guilherme took the lead with the poster, but António helped wherever necessary. The collaborative effort across all parts of the project ensured a coherent and well-executed outcome.

References

[1] Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2019). MUSDB18-HQ - An uncompressed version of MUSDB18. *Zenodo*. https://doi.org/10.5281/zenodo.3338373

[2] Stöter, F.-R., Liutkus, A., & Ito, N. (2018). The 2018 Signal Separation Evaluation Campaign. *arXiv preprint arXiv:1804.06267*. https://arxiv.org/pdf/1804.06267

[3] Araki, S., Haeb-Umbach, R., Ito, N., Wichern, G., Wang, Z-Q. & Mitsufuji, Y. (2025). 30+ Years of Source Separation Research: Achievements and Future Challenges.

arXiv preprint arXiv:2501.11837. https://arxiv.org/pdf/2501.11837

[4] Solovyev, R., Stempkovskiy, A.& Habruseva, T. (2024). Benchmarks and Leaderboards for Sound Demixing Tasks. *arXiv preprint arXiv:2305.07489*. https://arxiv.org/pdf/2305.07489

[5] Stöter, F.-R., Liutkus, A. et al. (2019). Open-Unmix - A Reference Implementation for Source Separation. *GitHub repository*. https://github.com/sigsep/open-unmix-pytorch

[6] Butler, S.. (2024). Mono vs Stereo: What's the Difference and When Does It Matter? *How-To Geek*. https://www.howtogeek.com/mono-vs-stereo-whats-the-difference-and-when-does-it-matter/

[7] StackExchange User. (2015). What is the Importance of Phase Spectrum in Fourier Transform? *Mathematics Stack Exchange*. https://math.stackexchange.com/questions/1290620/what-is-the-importance-of-phase-spectrum-in-fourier-transform

[8] StackOverflow User. (2022). How to Calculate Metrics SDR, SI-SDR, SIR and SAR in Python. *Stack Overflow*. https://stackoverflow.com/questions/72939521/how-to-calculate-metrics-sdr-si-sdr-sir-sar-in-python

[9] Manilow, E., Seetharaman, P. & Salamon, J. (2020). Why Use Evaluation Metrics Instead of Human Reviews? *Source Separation Tutorial*. https://source-separation.github.io/tutorial/basics/evaluation.html

[10] Le Roux, J., Wisdom, S., Erdogan, H., & Hershey, J. R. (2019). Scale-Invariant Source-to-Distortion Ratio: Optimization and Application to Speech Enhancement. *Mitsubishi Electric Research Labs Technical Report TR2019-013*. https://www.merl.com/publications/docs/TR2019-013.pdf

[11] Gusó, E., Pons, J., Pascual, S. & Serrà, J. (2022). Improving Music Source Separation Based on Deep Neural Networks through Data Augmentation and Network Blending. *arXiv preprint arXiv:2202.07968*. https://arxiv.org/pdf/2202.07968

[12] Luo, Y., Chen, Z. & Yoshioka, T. (2020). Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation. *arXiv preprint arXiv:1910.06379*. https://arxiv.org/pdf/1910.06379

[13] Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T. & Takahashi, N. (2017). Improving Music Source Separation Based on Deep Neural Networks through Data Augmentation and Network Blending. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 261–265). https://ieeexplore.ieee.org/document/7952158