PROPOSAL **-** Audio Source Separation using Deep Learning methods

Group Members:
- António Estêvão (fc58203)
- Diogo Venes (fc58216)
- Guilherme Gouveia (fc58176)

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) 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)

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: (basically, what the professor showed me in the Lab class 2 weeks ago)
https://arxiv.org/pdf/2501.11837
https://arxiv.org/pdf/1804.06267
(and more..)

What data will be used? If new data is collected, how will it be gathered?
"MUSDB18" which is a database of more than 100 songs with mixture and solo parts of instruments/vocals.
If for some reason the "MUSDB18" does not work, we also have other datasets such as "Solos".

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 music (for a list of several musics) already transformed/analyzed with FFPs (Fourier Transform), STFTs (Short Time Fourier Transform) and MFCCs (Mel Frequency Cepstral Coefficient), or use Convolutional Layers, but we are still exploring for now as this is a very recent topic for us…
After some more search, we found out that RNN (recursive neural networks) will be a good approach.

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, possibly like spectrograms.

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 think, at the present moment, 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 automatically but also manually performance analysis, by giving a score for example for each final result depending on the quality.

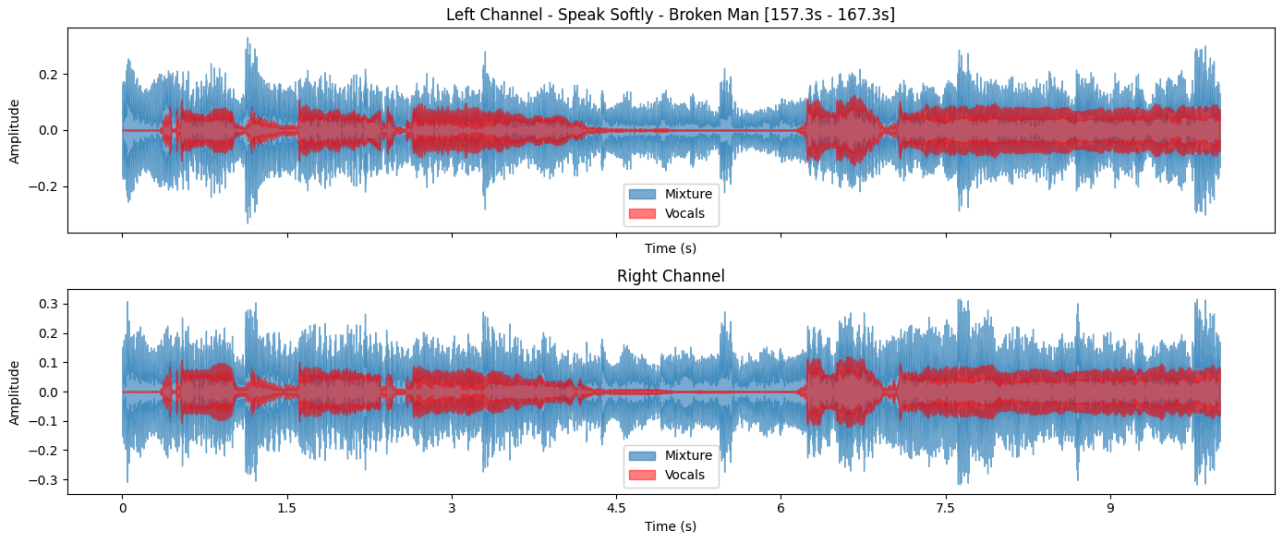
Milestone 1

1. Data Pipeline with Preprocessing

Our project tackles the challenge of separating components of an audio track—such as vocals and instruments—using deep learning models. The chosen dataset is **MUSDB18-HQ**, which contains 100 full-length stereo songs with annotations for separated sources (vocals, drums, bass, etc.).
We implemented a custom PyTorch-compatible dataset loader that:

- Loads mixture and target stems (e.g., vocals) as stereo waveforms at 44.1 kHz.
- Supports optional random cropping (we used 10-second segments = 441,000 samples).
- Handles 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 below 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. Baseline description: model name / architecture depth; input shape; loss; optimiser; training budget (epochs, LR, batch, GPU time)

We implemented, **for now but we plan to change to a RNN in the future**, 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 Guilherme's laptop.

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.

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 Guilherme's discussion 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 above 2), confirming the large performance gap between the two domains.

Milestone 2

Training methods

The architecture chosen for our source separation model is 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.

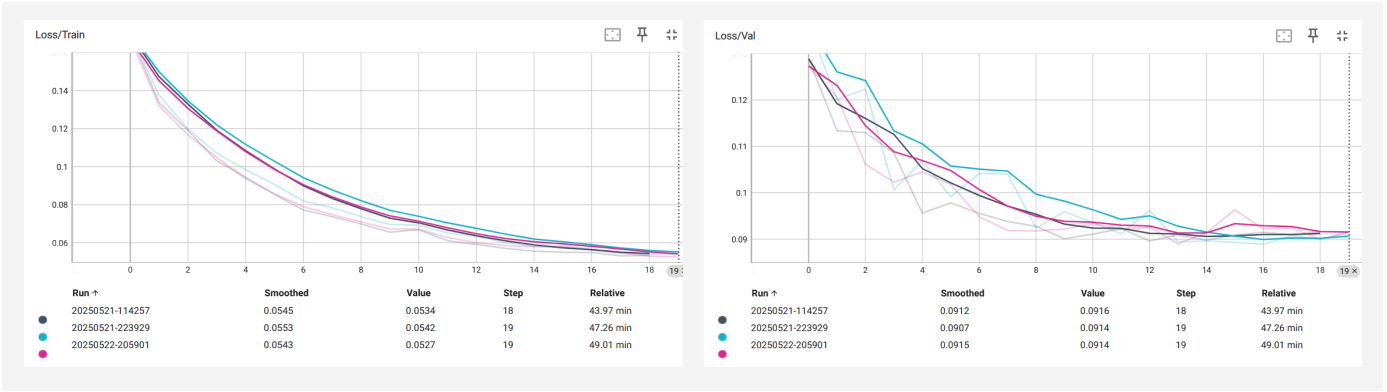
This structure enables more effective modeling of both short- and long-term temporal context compared to a standard BiRNN. Since our setup assumes **Non-Causal Source Separation**, the model has access to **past, present, and future frames** when predicting at a given time step, allowing it to make more 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.

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). In the future, we also might want to try and 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.”
https://stackoverflow.com/questions/72939521/how-to-calculate-metrics-sdr-si-sdr-sir-sar-in-python

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 below reveal that our DPRNN for audio source separation, which is similar to the Open-Unmix, is still not as good as its opponent:

Fonte	Modelo	SDR	SIR	ISR	SAR
Vocals	Open-Unmix	5.259	12.508	14.385	6.607
Accompaniment	Open-Unmix	12.951	19.901	21.255	14.464
Vocals	DPRNN-ass	2.516	5.445	9.482	6.011
Accompaniment	DPRNN-ass	6.094	22.865	6.941	8.449

Either way, we believe that we could still improve it, 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.

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, 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 technical sections when needed. The collaborative effort across all parts of the project ensured a coherent and well-executed outcome.