Audio Source Separation using Deep Learning methods

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

A. Quick rubric (✓ / △ / ✗)

Criterion	Assessment	Notes
Technical soundness	△	Goal is clear, but architecture choice still vague (CNN-based U-Net, Dual-Path RNN, TDCN++). Need to fix a time-frequency representation and decide between spectrogram masking vs. end-to-end waveform.
Feasible on free-tier GPUs	✓	MUSDB18 (150 min stereo, 44 kHz) fits; U-Net-style models train in < 3 h on a Colab T4 with 90-s crops and mixed precision.
Dataset availability & readiness	✓	MUSDB18 is open; `musdb` Python package streams stems and mixtures.
Starting-code / transfer-learning plan	△	Open-source baselines (Open-Unmix, Demucs v4, Spleeter) exist; choose one to fine-tune rather than training from scratch.
Evaluation metrics	△	Manual listening is subjective; must report **SDR / SI-SDR**, **SAR**, **SIR** (Mir-Eval v0.7) on MUSDB18 test.

(✓ = ready; △ = needs work; ✗ = high risk)

B. Focused suggestions

1. **Fix the processing pipeline now**

```
WAV → STFT (hop 1024, win 2048) → magnitude |S| / phase ∠S
↓                                  ↓
  U-Net mask estimator              copy phase
↓
Ŝ = mask ⊙ |S|        →    ISTFT → separated wav
```

- STFT-masking is simpler than waveform end-to-end and converges faster on free GPUs.
- Use **log-scaled magnitude** (+ epsilon) as input; batch-normalise across time–freq.

2. **Baseline model choices**

Model	Params	Train time*	Open weights
Open-Unmix (3-layer Bi-LSTM encoder–decoder)	3.3 M	2 h	✓
Demucs v4-mini (ConvTasNet 10 M params)	10 M	3 h	✓
Spleeter-5stems (U-Net)	20 M	2 h	✓

*Colab T4, 90-s segments, batch 4, AMP.

Fine-tune a single-target branch (vocals-only or vocals/instruments) first; expand to 2-way later.

3. **Class-imbalance & overfitting**

- MUSDB has ≈ 65 % vocal-free segments. Use **random chunking** with 50 % probability to force inclusion of vocal frames.
- Apply **SpecAugment** (time-mask, freq-mask) to mitigate overfitting short clips.

4. **Objective & loss**

- **Multi-resolution STFT loss** (L1 on |S| at 1024/2048/4096 windows) improves high-frequency detail.
- Add **phase-consistency loss** if you later switch to hybrid T-F + waveform.

5. **Metrics to implement before full training**

```
from museval import eval_mus_track
sdr, sir, sar, _ = eval_mus_track(track_ref, track_est)
```

Report **median SI-SDR** over 50 test tracks and compare to open-unmix baseline to ensure your pipeline is wired correctly.

6. **Compute budget (Colab T4)**

Task	GPU h
Data pre-chunk & cache	0.3
Fine-tune Open-Unmix 100 epochs	1.5
Hyper-param sweep (×2)	1.0
Inference + metrics	0.3
Total	**≈ 3.1 h**

7. **Next steps for Milestone 1**
8. **Clone** `sigsep/open-unmix-pytorch`; modify `--targets vocals drums|` to your 2-class setup.
9. **Write** **`prepare_data.py`** that streams MUSDB, applies STFT and saves 6-s chunks to `.npy`.
10. **Train 10 epochs** to verify GPU memory (should peak ≈ 6 GB) and compute SDR on three validation songs.

With these concrete design fixes your group can spend time tuning models rather than struggling with data or metric code, staying within free-tier GPU limits while producing publishable separation results.