PROGRESSIVE INFERENCE FOR MUSIC DEMIXING

Giorgio Magalini, Alessandro Manattini, Filippo Marri

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano
Piazza Leonardo Da Vinci 32, 20122 Milano, Italy
[giorgio.magalini,alessandro.manattini,filippo.marri]@mail.polimi.it

ABSTRACT

Isolating individual sources from a musical mixture remains a key challenge in audio engineering, with wide-ranging applications from production to restoration. This work investigates whether a progressive inference strategy, inspired by diffusion models, can enhance the performance of an existing demixing network such as HDemucs. The proposed method iteratively refines the *target stem* by remixing the previous estimate into the original mixture using a scheduled weighting, and reapplying the separation at each step. As a preliminary validation, we demonstrate that increasing the relative gain of a stem in the mixture improves its separation quality—a lemma referred to as the *oracle predictor implication*—thereby justifying the progressive approach. The system is evaluated on the MUSDB18-HQ dataset using scale-invariant signal-to-distortion ratio (SI-SDR) as the metric, and tested under different update strategies and configurations. While early iterations show slight improvements for the target source, the performance plateaus and eventually declines, suggesting that fixed schedules and lack of model adaptation limit the effectiveness of the approach. The findings indicate that simply reusing a pre-trained separator in an iterative loop is insufficient, and that meaningful gains could be achieved by incorporating noise modeling, adaptive blending, and end-to-end retraining within a diffusion-based framework.

Index Terms— Source separation, diffusion models, MUSDB18-HQ, HDemucs

1. INTRODUCTION AND BACKGROUND

Music demixing, also known as source separation, refers to the process of isolating individual instruments (i.e., “drums”, “bass”, “vocals”) from a single, fully mixed audio track. This task holds significant relevance in audio engineering, music production, and scientific research. Over time, numerous methodologies have been explored to address this challenge.

Among traditional approaches is Nonnegative Matrix Factorization (NMF) [1], which decomposes a magnitude spectrogram into a product of basis and activation matrices. Concurrently, repetition-based techniques have gained traction for foreground/background separation in audio signals. These approaches capitalize on the repetitive nature of many musical accompaniments. REPET [2] estimates a periodic background by averaging spectrogram frames spaced at a fixed interval. Its variants, including Adaptive REPET [2], which accommodates slowly changing periods, and REPET-SIM [2], which utilizes a spectral similarity matrix instead of strict periodicity, offer improved adaptability in handling non-stationary or intermittent audio patterns. Another conventional method, Harmonic-Percussive Source Separation (HPSS) [3], distinguishes percussive elements from harmonic content based on temporal characteristics.

Despite the utility of these classical techniques, recent advancements in deep learning have demonstrated superior performance. Traditional digital signal processing (DSP) approaches are grounded in explicit spectral or temporal assumptions, offering interpretability and computational efficiency. However, they often perform poorly when dealing with non-stationary harmonics, complex reverberation, or timbral masking. In contrast, modern machine learning (ML) models—ranging from convolutional neural networks (CNNs) to diffusion models—leverage large-scale datasets to learn non-linear, data-driven representations capable of capturing intricate sonic structures and dynamics [1].

Traditional DSP-based source separation is commonly classified as blind source separation. Machine learning, by contrast, derives substantial advantages from its training phase, enabling the model to extract patterns learned from data rather than relying on pre-defined heuristics. This learning-based paradigm is a key factor behind the improved performance of ML techniques over classical DSP methods.

A seminal ML-based approach to source separation emerged with the introduction of the U-Net architecture, originally developed by Ronneberger et al. [4] in 2015 for biomedical image segmentation. The architecture is named for its characteristic U-shape, composed of a contracting path (encoder) that captures contextual information and a symmetric expanding path (decoder) that facilitates precise localization. A defining feature of U-Net is its skip connections, which link encoder and decoder layers at corresponding resolutions, preserving high-resolution features critical for accurate segmentation and reconstruction. Although originally intended for medical imaging, U-Net’s architecture has proven highly adaptable. In audio processing, music demixing can be analogized to segmenting different frequency components from a composite audio signal, making U-Net a natural fit for this task.

Subsequent advancements led to the development of the Demucs model. The early versions (v1 and v2) adopted a U-Net-based architecture that operated directly on raw waveforms, working entirely in the time domain [5]. These models learned to separate sources by analyzing the audio signal as it is perceived, with the initial layers implicitly performing a time-frequency decomposition via learned convolutions. A major innovation was introduced in Hybrid Demucs (HDemucs), which employs a dual-branch U-Net architecture. One branch processes the waveform in the time domain, while the other operates in the spectral domain via spectrograms [6]. This hybrid configuration enables the model to harness the complementary advantages of both representations. The most recent version, Hybrid Transformer Demucs (HTDemucs) [7], incorporates a cross-domain Transformer encoder into the deepest layers of the U-Net. This design facilitates the fusion of spectral and temporal in-

formation, enabling the model to capture long-range dependencies and further enhancing separation quality.

More recently, generative modeling approaches—particularly diffusion models—have emerged as a powerful framework for source separation [8]. These models are trained to learn data distributions by progressively adding and then removing noise from input signals [8]. The denoising process is mathematically formalized as solving a probability flow ordinary differential equation (ODE) [8], with a neural network estimating the score function that guides the denoising trajectory [8]. In practice, this involves iterative refinement of the extracted sources, analogous to polishing a noisy signal. This concept is readily applicable to music source separation.

Following the framework proposed by Karras et al. in *Elucidating the Design Space of Diffusion-Based Generative Models* [9], where a diffusion model approach is developed for images, we adopt an iterative procedure in which each instrument stem is gradually extracted from the mixture. At each iteration, the neural network refines its output by conditioning on the previous extraction, mirroring the progressive denoising in diffusion models. In this analogy, the mixture corresponds to noise, and the extracted stem is the denoised target. This strategy enables a gradual and accurate reconstruction of individual sources from a complex mix.

2. METHODOLOGY

To conduct the analysis, a jupyter notebook[1] has been created whose code has been structured in a modular fashion in a way that by setting different parameters to a single function, several experiments can be conducted. This function takes as input the stems, it mixes them, it demixes the mixed track by using HDemucs, it computes the SI-SDR for each stem extracted (SI-SDR is the best suited evaluation metric for amplitude-varying signal, that is our case, as shown in paragraph 3.1) and then it iterates the procedure for every song of the dataset. At the end, the results are averaged along the songs dimension obtaining four average value of the SI-SDR (one for each stem) of all the songs in the dataset. This operation is repeated for a certain number of steps where, for every step after the first one, the net is fed with different input according to different mixing techniques. We chose a modular implementation since both the experiments share the majority of the lines of code, the only thing that changes is the mixing strategy that can be:

- *oracle predictor*: the stems used to create the mix are the original stems used in the mixing stage of the song production to create the original mix contained in the dataset we used. In this case, the stems are not extracted using HDemucs. For each iteration the gain of just one stem, that from now on will be called *target stem*, is relatively enhanced;
- *add to mixture*: just the *target stem* comes from the previous extraction made with HDemucs and it is added to the original mixture. At the end, this target stem will be the stem that will be extracted by the algorithm. The gain with which the *target stem* is mixed is expressed by the formula 1.

First of all, we want to demonstrate that if we have four stem tracks ("bass", "drums", "vocals" and "other") and we mix them together relatively increasing the gain of one of them with respect to the others, we obtain a higher value of SDR during demixing with HDemucs for the stem whose gain has been relatively increased. This is a crucial point, since the dissertation exposed here is based on this idea. In fact, if the SI-SDR of a generic stem increases by increasing its gain relatively to the ones of the "other" stems during the mixing, we can infer that the model works better when this relative gain increment occurs. From now on, we will refer to this lemma with *oracle predictor implication*. In order to conduct this demonstration, the mixing technique *oracle predictor* is used. Once the lemma is proved, the same procedure is performed, using the *add to mixture* technique in order to study the effectiveness of the algorithm implemented for every different stem. As hinted before, the schedule used for the *add to mixture* modality can be expressed synthetically by a formula inspired by the diffusion models as in Plaja and Roglans and Karras [10, 9].

Let $\hat{x}_{n-1}$ be the *target stem* extracted at the previous step, y the original mix and β_n the ratio between the present step and the total number of steps of the iteration loop. The current step mix y_n will be expressed by:

$$y_n = (1 - \beta_n) \cdot y + \beta_n \hat{x}_{n-1} \tag{1}$$

In the results section is shown how, if the extraction is perfectly done, at the end of the iteration we expect to obtain only the *target stem* in the mix since everything else is lowered to zero. Let us briefly theoretically demonstrate it considering "vocals" as the *target stem* and the non-target stems as accompaniment $bass + drums + other = acc$, so $y_n = bass + drums + other + vocals_n = acc + x_n$ and $\hat{x}_n = x + err_n$ where x is the original vocal track, $\hat{x}_n$ the extracted vocal track at the step n and err_n the error introduced by the extraction procedure at the step n. By plugging them into 1, we obtain:

$$y_n = (1 - \beta_n)(acc + x) + \beta_n \cdot (x + err_{n-1}) \tag{2}$$

that will become:

$$y_n = (1 - \beta_n) \cdot acc + x + \beta_n \cdot err_{n-1} \tag{3}$$

For the ideal case in which $err_n = 0$, we get:

$$y_n = (1 - \beta_n) \cdot acc + x \tag{4}$$

and, as we can notice, x maintains a constant and unitary gain independently on the step.

Following in the footsteps of Plaja and Roglans [10], a parameter in the code activates or deactivates a Wiener filtering operation that has been implemented to check how the SI-SDR behaves if the signal is Wiener-filtered or not.

3. EVALUATION

3.1. Evaluation metrics

As evaluation metric, scale invariant signal-to-distortion ratio has been used (SI-SDR). It is a variant of the traditional SDR, but it is invariant to the scale (amplitude) of the signal, meaning that it does not penalize the result if the output signal has the right shape but a different energy than the target. Its formula is reported below in the expression 5

$$\text{SI-SDR} = 10 \log_{10} \left(\frac{\|\alpha x\|^2}{\|\alpha x - \hat{x}\|^2} \right) \tag{5}$$

where:

- x is the target (ground truth) signal,

[1] https://github.com/alessandromanattini/DEMIXING

- $\hat{x}$ is the estimated (extracted) signal,
- α is the scaling factor, given by:

$$\alpha = \frac{\langle \hat{x}, x \rangle}{\|x\|^2} \tag{6}$$

The metric is computed using directly the ScaleInvariantSignalDistortionRatio [11] function of the torchmetrics library.

3.2. Dataset preparation

HDemucs and our proposed model are evaluated on the test dataset of songs from MUSDB18-HQ [12], a standard dataset with known ground-truth stems. Only the test dataset has been selected in order to avoid high valued results that are invalid from the analysis standpoint: they would be high just because they are based on the tracks with which the model has been trained.

A new mixed track is created from scratch for every song in which the four stems are simply summed together with homogeneous coefficients in order to ensure that the starting mixed track is linearly mixed as the algebraical sum of the four stems without any kind of non-linear post-processing. This has been done to guarantee that the hypotesis on which the *oracle predictor implication* are based holds true.

Once the dataset with the new mixed track is loaded, each stem and song is cropped in 30 seconds long chunks. This decision has been made in order not to overload the memory of the MPS device on which the experiments are run (more details about the device are reported in section 3.3). We ensured that this decision does not change the results of the extraction by comparing the results of the SI-SDR computation made on the entire length of the songs (run without iterations just once on the CPU) with the results of the SI-SDR computation made on the cropped tracks. They are in fact similar. We highlight that the cropping is done using the trimming function of librosa [13] so that each song and the relative stems start from a point (obviously the same for the stems and the relative mixed track) where all the four stems are not silent. In this way, we avoid the risk of having songs for which, if a stem is completely silent, both the SDR and SI-SDR reach approximately $-\infty$, biasing the average SI-SDR computation. After that, a preliminary analysis is conducted on a single track to check the main structure of the algorithm. There, the mixed track is demixed using the HDemucs model, and the SDR is evaluated as a check: the higher, the better. We computed here the SDR and not the SI-SDR in order to compare it with the results found in the literature.

An important thing that we underline is that, even though we use SI-SDR, normalization is avoided as much as possible. However, in order to have no issues related to domain shift [14], the input of the net must have the loudness of the original track. This is achieved by performing loudness normalization only once at the beginning. Even though, if this is done at the beginning, theoretically we should not bump into any problem neither of clipping nor of domain shift, it could happen that some artifacts make the amplitude overshoots of just a few cents the limit: without any correction, we would get digital distortion, so normalization is introduced just for the tracks for which this phenomenon happens. We highlight that this problem occurs only for 4 songs out of 50 and only for the "drums" stem. Furthermore, the maximum scaling factor applied is equal to 0.9434, which is extremely little.

As last thing, since what is needed by the algorithm is a relative increment of the gain, another precaution we take to avoid clipping is the one of decreasing the gain of the non-target stems instead of directly increasing the gain of the *target stem*.

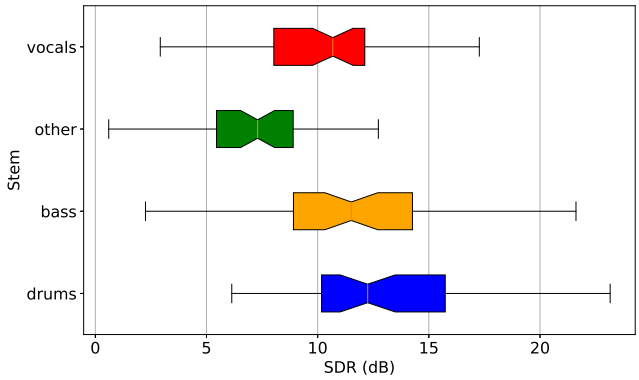

Figure 1: Evaluation of the SDR of the stems demixed by the HDemucs model in the standard usage.

3.3. Performances

The hardware we used for the tests is an "Apple MacBook" with chipset model "Apple M4 Pro", a GPU of 20 cores with "Metal 3" support. The Torch device used to run the test with GPU training acceleration has been the Metal Performance Shaders (MPS) backend. The average inference time has been measured being $0.5130\ s$, while the average time of the SDR computation was $0.0155\ s$ by averaging single results on all the songs. This computation comprises four iterations of the scale-invariant SDR of Torch Metrics: one for each of the four stems. According to our timing measurements, the model is the bottleneck.

4. RESULTS

The literature [6, 15] shows that the actual SDR values for "vocals", "bass", and "drums" are in the range of 8–9 dB with significantly lower SDR for the "other" stem (5–6 dB). This kind of difference is the same that we got by testing the model. As we can see from Fig. 1, the SDR of "vocals", "bass", and "drums" is significantly higher with respect to the one of "other". The former is, in fact, around 7.5–16 dB and the latter around 6–8 dB. This test has been conducted using a specific script written out from the Jupyter notebook to avoid problems of kernel collapse. The script can be found in the same repository.

For what it concerns the *oracle predictor* demonstration, the results are reported in Fig. 2. It can be seen that the average SI-SDR value of the *target stem* ("vocal" in this case) increases, while the others decrease. This is exactly what was expected.

However, the final results of the progressive inference model show that this approach is not optimal to perform this kind of operation, as shown in Fig. 3. In fact, after a small improvement of the SI-SDR of the *target stem* for the first two iterations (≈ 0.01 dB), the trend starts to slightly decrease. Furthermore, even the application of a Wiener filter block after every extraction did not improve the results differently from what happened to Plaja-Roglans [10]. By selecting another stem as *target stem*, the results are similar.

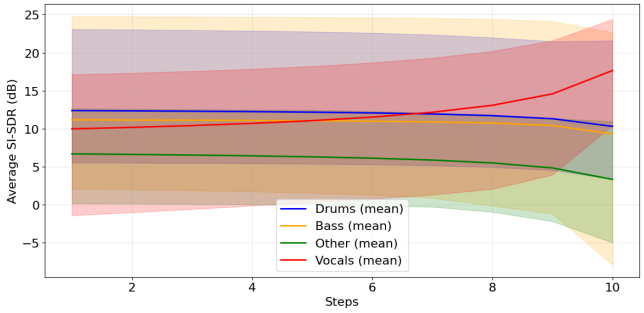

Figure 2: SI-SDR of the four stems when the relative gain of the *target stem* is increased for each epoch with the *oracle predictor* modality. The fill-between represents the span of the values between the minium and maximum of the data distribution.

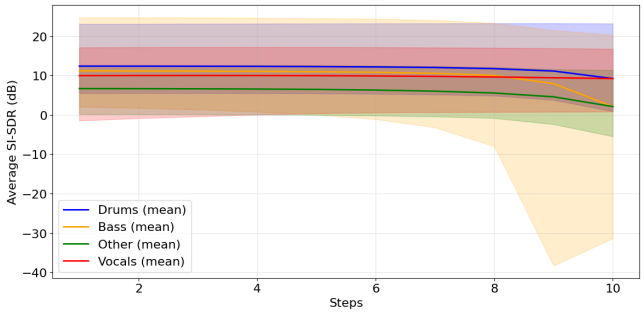

Figure 3: SI-SDR of the *target stem* for each epoch using the *add to mixture* procedure. The fill-between represents the span of the values between the minium and maximum of the data distribution.

5. CONCLUSIONS AND DISCUSSION

Our iterative remixing procedure – where we simply mixed the previous estimate back into the original mixture with a fixed β schedule – did not appreciably improve separation.

In hindsight, our approach lacked several components that appear to be central in recent diffusion-based methods. For instance, Zang et al. [16] suggest that adaptively choosing the mixing parameter β at each step, based on a quality metric, can improve separation performance iteratively. Similarly, Plaja-Roglans et al. [10] train a network end-to-end with a denoising objective inspired by diffusion, progressively transforming a mixture into the target source via a learned Markov chain – achieving strong results on MUSDB18. In contrast, our method employed a fixed β and no model retraining, offering no way for the system to adapt or refine its own outputs across iterations.

Looking forward, we believe it might be fruitful to explore whether introducing noise injection and training the model to denoise over multiple steps – as in denoising diffusion models [10] – could lead to more effective remixing strategies. Instead of deterministically reusing the previous estimate, the system could be trained to interpret the mixture as a noisy version of the target and gradually clean it. Recent work by Karchkhadze et al. [8] also points toward promising directions: they integrate a score-based diffusion model within a source separation pipeline and apply consistency distillation to reduce sampling steps, achieving strong performance gains. Whether elements of this framework – such as consistency objectives or metric-driven guidance – could help our remixing approach converge in fewer steps is an open question worth exploring.

Overall, the recent literature offers several intriguing ideas. Future work could investigate whether adaptively tuning β (as in Zang et al. [16]) or fine-tuning a separator like HDemucs within a denoising framework (as in [10]) helps structure the iterative remixing process more effectively. Likewise, training with consistency constraints (as proposed by Karchkhadze et al. [8]) might enable fast, diffusion-like refinement in just a few steps.

It is also possible that the effects of our proposed remixing strategies – such as fixed-step iteration or simple blending heuristics – might be more noticeable when applied to a simpler baseline model than HDemucs. Given its strong one-shot performance, HDemucs may leave limited room for iterative refinement to visibly improve results. Applying the same remixing ideas to lighter models such as Spleeter [17] (Deezer's open-source separator), Conv-TasNet [18], or a basic U-Net-based architecture could help assess whether the lack of improvement is intrinsic to the remixing procedure or masked by the high capacity and robustness of a state-of-the-art model. While we have not tested these possibilities, they offer intuitions that may guide more effective designs in the future.

6. REFERENCES

[1] L. Xu, Z. Li, L. Yang, Z. Zhang, and D. Wang, "30+ years of source separation research: Achievements and future challenges," *arXiv preprint arXiv:2501.11837*, 2025. [Online]. Available: https://arxiv.org/abs/2501.11837

[2] Z. Rafii, A. Liutkus, and B. Pardo, "REPET for Background/Foreground Separation in Audio," in *Blind Source Separation*, G. R. Naik and W. Wang, Eds. Springer, 2013, pp. 395–411.

[3] FluCoMa learn authors, "Hpss: Harmonic-percussive source separation," https://learn.flucoma.org/reference/hpss/, Fluid Corpus Manipulation (FluCoMa), 2025, accessed: 2025-06-25.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer International Publishing, 2015, pp. 234–241, available on arXiv:1505.04597 [cs.CV]. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a

[5] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019. [Online]. Available: https://arxiv.org/abs/1911.13254

[6] A. Défossez, "Hybrid spectrogram and waveform source separation," CoRR, arXiv:2111.03600, eess.AS, cs.SD, stat.ML 2111.03600, August 2022, revised version of preprint first submitted Nov 5, 2021. [Online]. Available: https://arxiv.org/abs/2111.03600

[7] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5. [Online]. Available: https://arxiv.org/abs/2211.08553

[8] T. Karchkhadze, M. R. Izadi, and S. Zhang, "Improving source extraction with diffusion and consistency models," *arXiv preprint arXiv:2409.12346*, 2024. [Online]. Available: https://arxiv.org/abs/2409.12346

[9] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 26 565–26 577. [Online]. Available: https://arxiv.org/abs/2206.00364

[10] G. Plaja-Roglans, M. Miron, and X. Serra, "A diffusion-inspired training strategy for singing voice extraction in the waveform domain," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, Bengaluru, India, December 2022, pp. 685–693.

[11] PyTorch Metrics Team, *Scale-Invariant Signal-to-Distortion Ratio (SI-SDR)*, Lightning AI, 2025, online documentation, version 1.x. [Online]. Available: https://lightning.ai/docs/torchmetrics/stable/audio/scale_invariant_signal_distortion_ratio.html

[12] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "Musdb18-hq: An uncompressed version of musdb18 (1.0.0) [data set]," *Zenodo*, 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373

[13] librosa development team, *librosa.effects.trim: Trim Leading and Trailing Silence from an Audio Signal*, 0th ed., librosa, https://librosa.org, 2025. [Online]. Available: https://librosa.org/doc/latest/generated/librosa.effects.trim.html

[14] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 28, no. 3. PMLR, 2013, pp. 819–827.

[15] A. Défossez, "Hybrid spectrogram and waveform source separation," *arXiv preprint arXiv:2111.03600*, 2021.

[16] Y. Zang, J. Li, and Q. Kong, "Training-free multi-step audio source separation," *arXiv preprint arXiv:2505.19534*, May 2025, submitted 26 May 2025. [Online]. Available: https://arxiv.org/abs/2505.19534

[17] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[18] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.