Music Source Restoration

Yongyi Zang[1], *Zheqi Dai*[2,3], *Mark D. Plumbley*[3], *Qiuqiang Kong*[2†]

[1]Independent Researcher, Seattle, WA, USA [2]The Chinese University of Hong Kong, Hong Kong SAR, China
[3]CVSSP, University of Surrey, Guildford, UK

***Abstract*—We introduce Music Source Restoration (MSR), a novel task addressing the gap between idealized source separation and real-world music production. Current Music Source Separation (MSS) approaches assume mixtures are simple sums of sources, ignoring signal degradations employed during music production like equalization, compression, and reverb. MSR models mixtures as degraded sums of individually degraded sources, with the goal of recovering original, undegraded signals. Due to the lack of data for MSR, we present RawStems, a dataset annotation of 578 songs with unprocessed source signals organized into 8 primary and 17 secondary instrument groups, totaling 354.13 hours. To the best of our knowledge, RawStems is the first dataset that contains unprocessed music stems with hierarchical categories. We consider spectral filtering, dynamic range compression, harmonic distortion, reverb and lossy codec as possible degradations, and establish U-Former as a baseline method, demonstrating the feasibility of MSR on our dataset. We release the RawStems dataset annotations, degradation simulation pipeline, training code and pre-trained models to be publicly available.**

1. INTRODUCTION

> *"It has been said: The whole is more than the sum of its parts. It is more correct to say that the whole is something else than the sum of its parts."*
>
> *— Kurt Koffka [1]*

Music Source Separation (MSS) [2, 3] aims to decompose musical mixture signals into their constituent instrument signals, assuming that mixtures are simple sums of these components. Recent deep learning approaches have achieved significant progress, with state-of-the-art methods reaching signal-to-distortion ratios (SDR) exceeding 10 dB [4, 5]. Current methods can be broadly categorized into time-domain approaches (such as WaveUNet [6], ConvTasNet [7], and multiple resolution analysis [8]) that directly process audio waveforms, and time-frequency domain approaches that first transform waveforms into spectrograms before applying various neural architectures including fully connected networks [9], RNNs [10], CNNs [11], U-Nets [12, 13], BSRNNs [14], and BS-Roformers [4]. Hybrid approaches [15, 16] have also emerged, combining both domains for improved performance.

However, current MSS evaluation suffers from a fundamental limitation: it assumes an idealized scenario where musical sources are directly summed to form mixtures. This assumption fails to capture the complexity of real-world audio production. Real-world musical recordings undergo numerous transformations throughout the production chain [17, 18], including: room acoustics that introduce reverberation and modal resonances [19, 20]; recording equipment with inherent non-linearities and frequency response characteristics [21, 22]; mixing processes involving dynamic compression, equalization, and harmonic distortion [23]; mastering that further modifies frequency balance and dynamic range [24]; and distribution platforms that apply lossy compression algorithms introducing additional frequency loss and artifacts [25]. In music production, music is created by summing individually degraded instrument signals (See Sec. 2 for details).

To address the gap between idealized source separation and real-world music production, we introduce the **Music Source Restoration (MSR)** task. The MSR task aims to recover the original, undegraded source signals from the mixture. As the degradation processes result in information loss, MSR exhibits an inherent many-to-one mapping: the same degraded signal can be mapped to multiple original signals. For instance, a degraded signal obtained by applying a lowpass filter on the original signal with a cutoff frequency of 2 kHz can be mapped to several original signals with different timbres. To further constraint the solution space, we additionally restrict the recovered source signals to be close to the distribution of unprocessed instruments. We propose two metrics to evaluate for the MSR task: Structural Similarity Index Measure (SSIM) [26] on the mel-spectrogram as a perceptual similarity metric [27], and Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) [28] as a waveform-level metric.

To address the limitations of existing MSS datasets [29, 30] that only contain pre-processed signals, we present RawStems, a new dataset that provides access to the original, unprocessed source signals necessary for training and evaluating MSR systems, annotating 578 songs from the 'Mixing Secrets' multitrack library [23]. With 8 first-level instrument groups, 17 second-level instrument groups, and a total runtime of 354.13 hours. To our knowledge, RawStems is the largest unprocessed hierarchical musical source dataset. We propose possible degradations into five categories: spectral filtering, dynamic range compression, harmonic distortion, reverb and lossy codec, and provide Python implementation to simulate them. By combining U-Net [6] and RoFormer [5], we propose U-Former as a baseline method for our task and dataset, and report its performance. We hope our work marks the first step towards source restoration models that can be more practically used in music production settings. We release RawStems at https://huggingface.co/datasets/yongyizang/RawStems, code at https://github.com/yongyizang/music_source_restoration and model checkpoints at https://huggingface.co/yongyizang/MSR_UFormers.

2. THE MUSIC SOURCE RESTORATION TASK

2.1. Task Definition

The MSR problem models the observed mixture $y \in \mathbb{R}^L$ as a sum of degraded sources, where L is the number of samples of the signal. The goal of MSR is to recover the original, unprocessed forms $s_1, s_2, \ldots, s_n$ similar to direct recordings from a live performance in a professional recording studio from the mixture y. Formally, given a set of degradation functions $\mathcal{F}$, the mixture is expressed as $y = \sum_{i=1}^{n} f_i(s_i)$, where each $f_i \in \mathcal{F}$ is applied to source s_i. As mentioned in Sec. 1, due to the many-to-one mapping inherent in degradation processes, we introduce *neutrality* as a constraint: we require the recovered sources to be consistent with priors p_i representing distributions of original, unprocessed musical stems. MSR is thus framed as conditional generation: given the observed mixture y, priors p_i, and possible degradation functions $\mathcal{F}$, we aim to generate sources s_i such that:

- the source is *plausible*, indicating that there exist degradation functions $\{f_i\}_{i=1}^{n}$ from $\mathcal{F}$ satisfying $\sum_{i=1}^{n} f_i(s_i) = y$
- the source is *neutral*, indicating that each s_i belongs under its respective prior p_i (i.e., $s_i \sim p_i$).

†Please address correspondance to Qiuqiang Kong (qqkong@ee.cuhk.edu.hk)

2.2. The Degradation Set $\mathcal{F}$

Referencing common audio processing techniques used in music production [23, 31], we group possible degradation functions in degradation set $\mathcal{F}$ into five categories: spectral filtering, dynamic range compression, harmonic distortion, reverb and lossy codec. Additionally, we incorporate in $\mathcal{F}$ gain functions $F(x) = ax, a \neq 0$ to account for differences in audio volume across processed samples.

Spectral Filtering. We consider filtering through two approaches: a 16-band graphic equalizer (EQ) [32] utilizing fixed center frequencies (25, 40, 63, 100, 160, 250, 400, 630, 1000, 1600, 2500, 4000, 6300, 10000, 16000, 20000 Hz) with random gains uniformly distributed between -6 and $+6$ dB, and a parametric EQ [33] employing 1-5 randomly selected filter bands. Parametric filters include lowpass (200-8000 Hz), highpass (20-2000 Hz), bandpass/bandstop/peak (100-8000 Hz), lowshelf (50-1000 Hz), and highshelf (1000-10000 Hz), each with Q factors ranging from 0.5-10 and gains from -6 to $+6$ dB. Filter implementations include Butterworth (maximally flat), Chebyshev Type I (0.1-3.0 dB passband ripple), Chebyshev Type II (20-60 dB stopband attenuation), Elliptic (0.1-3.0 dB passband ripple, 20-60 dB stopband attenuation), and Bessel (linear phase) approaches, with even-valued filter orders between 2-8 [34]. For graphic EQ, band-specific Q values are calculated as $Q = f_{\text{center}}/(f_{\text{upper}} - f_{\text{lower}}) \times 2.5$ for peak filters, while parametric filter coefficients are computed directly from center frequency, Q, and gain parameters, with all filters implemented as second-order sections to maintain computational stability across the entire 20 Hz - 20 kHz spectrum. We implement all filters based on the signal processing library of SciPy [35].

Dynamic Range Compression. We consider two types of dynamic range compression algorithms: compressor [36] and limiter [37]. The compressor processes audio signals with a configurable threshold $T_c \in [-20, 0]$ dB, ratio $R_c \in [1.5, 10.0]$, attack time $A_c \in [1, 10]$ ms, and release time $R_{c,\text{rel}} \in [50, 200]$ ms. When the input signal amplitude exceeds T_c, the compressor attenuates the signal according to ratio R_c, where output level change equals input level change divided by R_c. The attack parameter A_c determines how quickly compression engages when threshold is crossed, while $R_{c,\text{rel}}$ controls how rapidly the compressor returns to unity gain when signal falls below threshold. The limiter functions as an extreme compressor with infinite ratio, configured with threshold ($T_l \in [-10, 0]$ dB) and release time ($R_{l,\text{rel}} \in [50, 200]$ ms). Any signal exceeding T_l is immediately attenuated to maintain the threshold ceiling, with fixed instantaneous attack time and configurable $R_{l,\text{rel}}$ governing return-to-unity behavior. We use the implementation provided by Pedalboard [38].

Harmonic Distortion. Diverse sources of harmonic distortion is added throughout the mixing process [23, 31]. For simplicity, we consider only hyperbolic tangent distortion [39], implemented with a configurable drive parameter $D \in [0, 5]$ dB, primarily introducing predominantly odd-order harmonics to the signal. Signal amplitude is first amplified by the drive amount, then passed through the `tanh` non-linearity, creating a compressed waveform with soft clipping behavior. This process simulates analog saturation found in hardware circuits [40]. We use the implementation provided by Pedalboard [38].

Reverb. While many reverb types exist in practice [41], we use the FreeVerb [42], which uses four Schroeder allpasses in series and eight parallel Schroeder-Moorer filtered-feedback combfilters for each audio channel to simulate a reverb tail. The FreeVerb is parameterized by room size $R_s \in [0.1, 1.0]$ controlling decay time of the comb filters, damping $D \in [0.1, 1.0]$ determining high-frequency absorption in the feedback loops, wet level $W_l \in [0.1, 0.5]$ adjusting the ratio of processed to unprocessed signal, and stereo width $S_w \in [0.1, 1.0]$ controlling the phase differences between left and right channels.

Codec. Similar to [43], we consider only the MP3 codec, as it is one of the most popular lossy codec in practice [44]. We use the varying bitrate encoding with a quality parameter $Q_{\text{vbr}} \in [1.0, 9.0]$, where lower values produce higher quality audio with increased bitrates and less perceptual artifacts. We use the implementation provided by Pedalboard [38], which uses the LAME MP3 Encoder [1].

2.3. Evaluation Metrics

The MSR task requires evaluation of generated sources on both *plausibility* and *neutrality* (See Sec. 2.1). We employ SI-SDR and SSIM to assess these aspects respectively.

SI-SDR. We utilize Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) to evaluate the *plausibility* of generated sources. This metric is commonly used in Music Source Separation (MSS) tasks [7] and is particularly appropriate in our context since gain functions are included in our degradation set $\mathcal{F}$, making the scale-invariant property essential for fair evaluation.

SSIM. To assess *neutrality*, we apply the Structural Similarity Index Measure (SSIM), which was originally developed to measure perceived similarity between original and degraded images [26] and has recently been adopted for audio perceptual similarity assessment [27]. We calculate power mel-spectrograms using 256 bins with a Hann window of 4096 samples and a hop size of 1024 samples to capture an appropriate level of detail.

Following standard MSS evaluation practices [45], we compute both SI-SDR and mel spectrogram SSIM on 10-second segments and report their means with 95% confidence intervals.

3. THE RAWSTEMS DATASET

The RawStems Dataset annotates 578 songs, annotated as 8 first-level instrument groups and 18 second-level instrument groups, with a total runtime of 354.13 hours. While the license of audio files belongs to the original creator, we provide the annotation in Apache 2.0 license.

3.1. Instrument Taxonomy

Instrument taxonomy, though well-established in research literature, presents significant challenges when designing a classification system that balances comprehensiveness with usability [30]. Our taxonomy's primary level consists of eight major categories, informed by professional stem mastering practices, where eight stems represent the standard configuration for most summing machines (analog devices used for combining audio signals to achieve optimal analog-to-digital conversion). Each primary category is further divided into specific subcategories, each one with its unique code, yielding 17 subcategories:

- Vocals (Voc): lead vocal (LV), backing vocal (BV), and group vocals (GV)
- Rhythm sections (Rhy): drum kits (DK) and percussions (PERC)
- Guitars (Gtr): acoustic (AG) and electric guitar (EG)
- Keyboards (Kbs): acoustic piano (PN), electric piano (EP), mellotrons (MTR), and organs (OR)
- Orchestra (Orch): strings (STR), brass (BR), and woodwinds (WW)
- Synthesizers (Synth): various electronic synthesizer sounds
- Bass: various bass sounds
- Misc: other signals (bells, effects, etc.)
- MiscRoom: Room microphones (capturing entire room ambience)

A significant challenge in instrument classification is addressing overlaps between categories. For instance, synthesizer strings could

[1] https://lame.sourceforge.io/

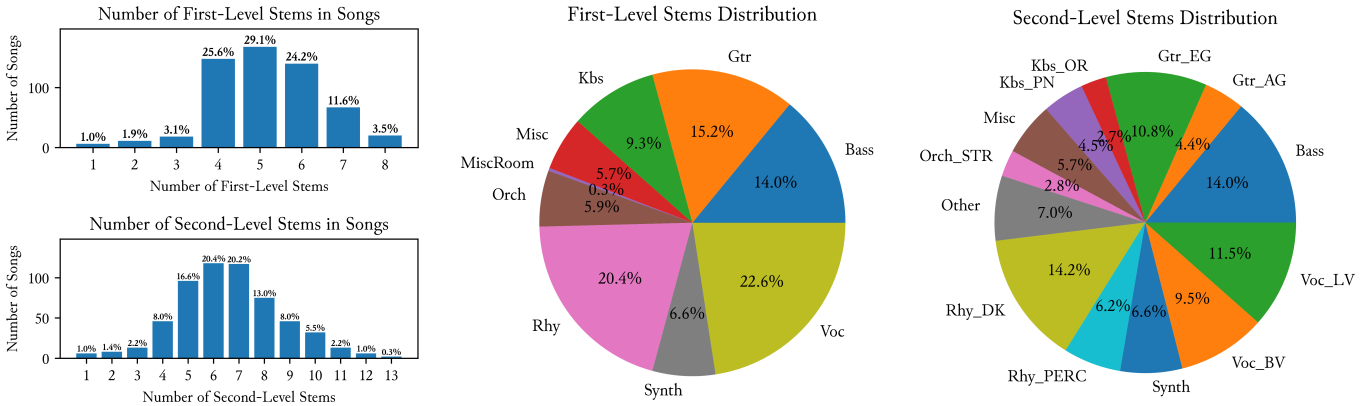

Fig. 1: Analysis on the RawStems dataset at first and second stem level. Stems distribution reported with active playing times.

logically belong to either synthesizers or orchestral strings; similarly, synthesizer bass could be classified as either synthesizer or bass. To address these ambiguities, we established the following rules: all mellotron signals are categorized under keyboards/mellotrons regardless of the simulated instrument, and all synthesizer bass instruments are classified under bass rather than synthesizers. For edge cases not covered by these rules, classification decisions were made through best human subjective judgement, placing each sound in the most appropriate category based on timbral characteristics and musical function. For example, when a signal of a horn section contains both brass and woodwind instruments (such as trumpets and saxophones), we label it a brass in the orchestral class (Orch_BR).

3.2. Data Collection and Annotation

Our annotation process followed three phases: (1) LLM-assisted auto-labeling using Claude 3.7 Sonnet to classify stems based on filenames into 18 subcategories or unknown (UNK) when uncertain; (2) manual verification of all songs containing "UNK" labels through human listening; and (3) manual verification of stems initially classified as "Misc." to ensure proper categorization according to our taxonomy rules.

3.3. Dataset Analysis

An analysis of the stem-level statistics is presented in Figure 1, revealing distinct patterns in instrumental composition across the dataset. The distribution shows a clear concentration, with 78.9% of songs containing between 4-6 first-level stems, while 86.2% feature 4-9 second-level stems. To count active playing time per instrument, we normalize each audio file, then sum the number of 1-second segments surpassing -40 dB RMS as the active playing time in seconds. our findings align with those reported by [30], with vocals, guitars, bass, rhythm section, and keyboards collectively dominating the sonic landscape (81.5% of total active playing time).

To gain a more comprehensive understanding of the hierarchical structure within RawStems, we computed CLAP [46] embeddings at both the individual track and mixture levels. Referencing [47], we project the resulting embeddings into 2D space using t-SNE, as illustrated in Figure 3 for track-level analysis and Figure 2 for mixture-level analysis. At the track level, we observe clustering patterns that align with our expectations: tracks naturally segregate according to their first-level stem classifications, while forming subclusters that correspond to second-level stem categories or even more fine-grained instrument groupings. This hierarchical clustering validates the

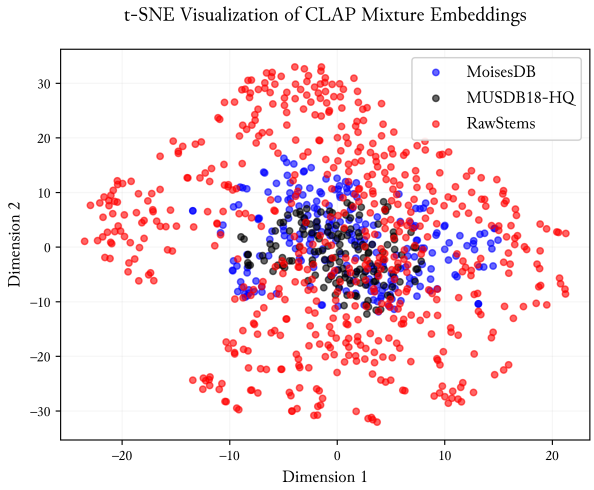

Fig. 2: t-SNE visualization for mixture-level CLAP embeddings of RawStems, MUSDB18-HQ and MoisesDB. Best viewed in color.

taxonomic organization of our dataset. When examining the mixture-level embeddings, RawStems demonstrates broader distribution in the embedding space compared to both MUSDB18-HQ and MoisesDB, indicating a more diverse range of musical compositions and timbral characteristics.

We also analyzed the co-occurrence patterns among different instrument stems by computing the conditional probability $P(B|A)$ (the probability of observing instrument stem B given the presence of instrument stem A) focusing specifically on second-level instrument groupings. The results are visualized in Figure 4. The analysis reveals that certain instruments, like bass, appear across nearly all songs. Additionally, we identified patterns consistent with real-world musical conventions, such as electric guitars frequently co-occur with both bass and acoustic guitars, while acoustic guitars rarely appear alongside the combination of bass and electric guitar.

4. BASELINE EXPERIMENTS

We conduct a series of baseline experiments to demonstrate MSR feasibility and evaluate our proposed dataset. While ideal training would encompass all first- and second-level stems, this requires 25 distinct models—exceeding our compute budget constraints. We therefore train nine baseline models: seven representing first-level

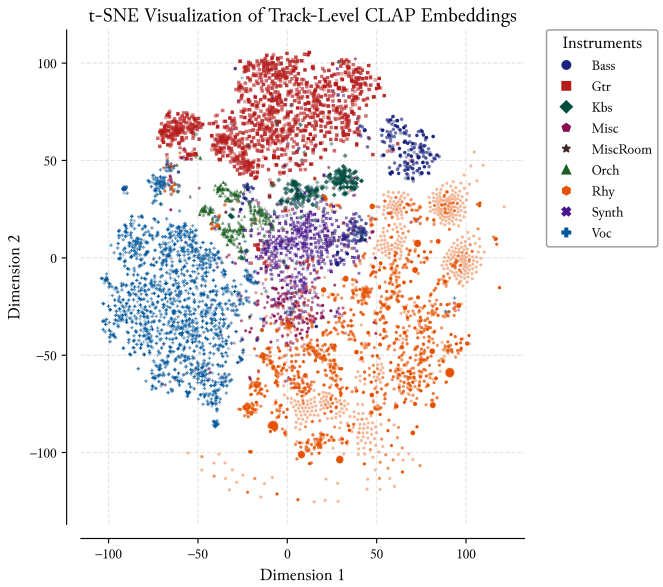

Fig. 3: t-SNE visualization for track-level CLAP embeddings of all individual audio files within RawStems. Best viewed in color.

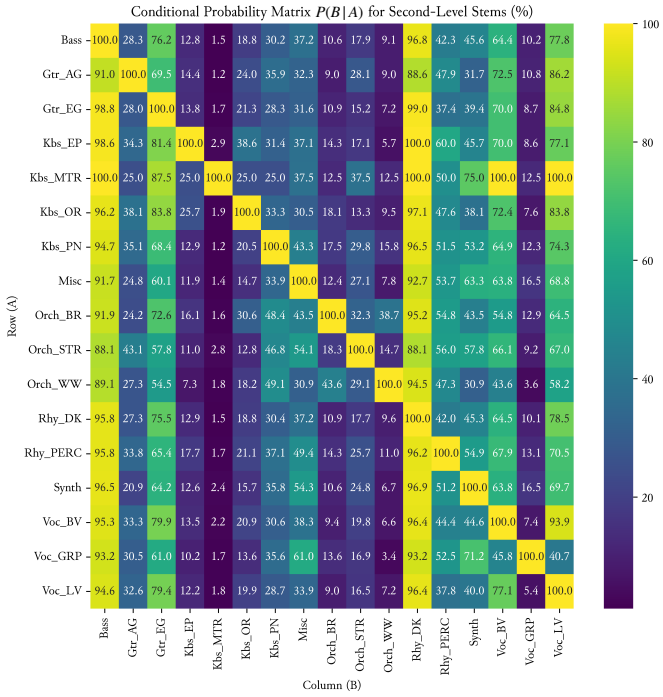

Fig. 4: Bayesian analysis for co-occurrence of instrument stems.

stem groups (excluding "Misc.") and two specifically for acoustic and electric guitar. Empirically, we observe that the U-Net architecture [6, 13] offers training efficiency but exhibits deficiencies in high-frequency detail reconstruction, whereas the BS-Roformer [4] delivers superior quality at significantly higher computational cost. To optimize this quality-efficiency trade-off, we propose U-Former (illustrated in Figure 5), which incorporates RoFormer blocks within U-Net's bottleneck layer, following established hybrid approaches [48–50]. Our implementation utilizes four convolution blocks in both upsampling

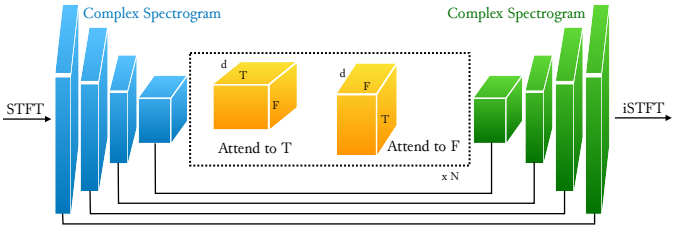

Fig. 5: Model architecture of U-Former.

and downsampling paths with channel dimensions $[16, 64, 256, 256]$; the RoFormer component comprises 6 layers, each with 8 attention heads. Both input and output utilize complex STFT representations computed with 2048-sample FFT length and 441-sample hop size.

4.1. Training Setup

We train our model using a two-stage optimization strategy spanning 85000 steps, which requires approximately 24 hours on a single NVIDIA H20 GPU. The training process is carefully designed to balance reconstruction quality with perceptual fidelity through a progressive loss schedule. During training, we construct each batch by randomly sampling 10-second audio segments from our dataset, a duration that provides sufficient temporal context while maintaining computational efficiency. We employ the Adam optimizer with a linear learning rate warmup over the first 1000 steps to ensure stable initial convergence, and apply gradient clipping with a threshold of 2.0 to prevent training instabilities.

The initial 50000 optimization steps focus exclusively on accurate waveform reconstruction. During this stage, we use a learning rate of 5×10^{-4} with a batch size of 16 samples, optimizing only the waveform-level L_1 reconstruction loss. This reconstruction-only phase allows the model to learn fundamental audio representations without the complexity of adversarial training, establishing a strong foundation for subsequent perceptual refinement. For the final 35000 steps, we introduce adversarial training following the HiFi-GAN framework [51] to enhance perceptual quality. Due to the increased memory requirements from incorporating the discriminator, we reduce the batch size to 4 samples while maintaining the generator learning rate at 5×10^{-4} and setting the discriminator learning rate to 5×10^{-5}, one order of magnitude lower for training stability.

The combined loss function during the adversarial training stage is formulated as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \lambda_g \mathcal{L}_{\text{gen}} + \lambda_f \mathcal{L}_{\text{feat}} \tag{1}$$

where $\mathcal{L}_{\text{recon}}$ is the L_1 reconstruction loss, $\mathcal{L}_{\text{gen}}$ is the generator adversarial loss, and $\mathcal{L}_{\text{feat}}$ is the feature matching loss. We set the loss weights to $\lambda_g = 0.001$ and $\lambda_f = 0.01$, empirically chosen to balance reconstruction accuracy with perceptual quality. The discriminator architecture follows the multi-scale discriminator design, which evaluates audio at different temporal resolutions. The generator, feature-matching and discriminator losses are computed as

$$\mathcal{L}_{\text{gen}} = \mathbb{E}_{x \sim p_{\text{data}}} \left[(1 - D(G(x)))^2 \right] \tag{2}$$

$$\mathcal{L}_{\text{feat}} = \mathbb{E}_{x \sim p_{\text{data}}} \left[\sum_{l=1}^{L} \frac{1}{N_l} \| D^{(l)}(x) - D^{(l)}(G(x)) \|_1 \right] \tag{3}$$

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{\text{data}}} \left[\sum_{l=1}^{L} \left[D^{(l)}(G(x))^2 + (1 - D^{(l)}(x))^2 \right] \right] \tag{4}$$

Table 1: Evaluation results for baseline experiments.

Instrument/Group	SSIM_{Mel} ↑	SI-SDR (dB) ↑
Vocals	0.5058 ± 0.0089	1.86 ± 0.77
Keyboards	0.4941 ± 0.0094	-2.75 ± 0.92
Synth	0.4004 ± 0.0102	-5.65 ± 0.95
Guitars	0.5564 ± 0.0098	-2.40 ± 0.83
Rhythm	0.5927 ± 0.0087	-0.11 ± 0.88
Bass	0.4437 ± 0.0086	-7.90 ± 1.29
Orchestra	0.4722 ± 0.0103	-2.37 ± 0.92
Guitars - Acoustic	0.5543 ± 0.0099	-1.14 ± 0.87
Guitars - Electric	0.5875 ± 0.0097	-1.68 ± 0.81
VoiceFixer [52]	0.2543 ± 0.0052	-40.72 ± 0.87
Guitars - Acoustic (+30k steps)	0.5619 ± 0.0096	-1.31 ± 0.88
Guitars - Electric (+30k steps)	0.6019 ± 0.0093	-1.25 ± 0.76

where $G(\cdot)$ is the generator (the restoration network in our case), $D^{(l)}$ represents the l-th layer features of the discriminator, and N_l is the number of elements in the l-th layer. This two-stage training strategy effectively balances computational efficiency with model performance, allowing the network to first master accurate reconstruction before learning to generate perceptually convincing audio through adversarial training.

4.2. Degradation Setting

To create robust training examples, we independently aggregate randomly sampled audio signals from within the target stem group and from all sources outside the target group. For target stems, we apply a suite of degradation effects (equalizer, downsampling, compression, distortion, and reverb) each with 50% probability of application. Similarly, for the mixture signals, we employ limiter, resampling, and codec artifacts, also each with 50% probability. When any degradation type is selected, we uniformly sample across its parameter space to ensure diverse transformations. To further enhance robustness against noise, we augment 50% of mixtures with Gaussian noise featuring means uniformly distributed between 0.001 and 0.005. The final mixing process combines target stems with background signals at signal-to-noise ratios ranging from -5 to 20 dB.

4.3. Datasets

For each stem group in RawStems, we randomly allocate songs containing the target stem to the testing set by selecting either 10 songs or 10% of the total songs for that stem (whichever is larger). All remaining samples are assigned to the training set. To generate a comprehensive evaluation benchmark, we process the testing set through our degradation pipeline 1000 times, creating a diverse array of degradation scenarios. To facilitate reproducibility, we make this offline-generated testing set publicly available alongside our implementation.

4.4. Evaluation

We show evaluation results in Table 1. As expected, we observe that the SSIM metric and SI-SDR metric only loosely correlate with each other, suggesting they evaluate different aspects of the restored audio samples. Amongst which, we notice that vocal performance is best while bass performance is worst, suggesting for different instrument class the difficulty of the task may vary, as similarly observed for the MSS task [2]. Interestingly, the acoustic and electric guitar model both behaves better than the general guitar model, suggesting benefits for training fine-grained models.

Notably, SI-SDR values are substantially lower than those typically observed in both MSS and audio restoration tasks [53]. We investigated three potential explanations: (1) flaws in our model architecture, training pipeline, or dataset; (2) insufficient training convergence; or (3) inherent task complexity. To test hypothesis (1), we evaluated a pre-trained VoiceFixer [52] model on our vocal test set, as it is trained to recover from both noise, reverb, low frequency rate and clipping, and thus is closest to the MSR task definition. We found it performed substantially worse than our model (SI-SDR near -40 dB). Manual inspection revealed its inability to properly separate vocal content from background instruments and reproduce high-frequency vocal details. For hypothesis (2), we extended training for acoustic and electric guitar models by 30,000 additional steps, yielding minimal improvement, indicating convergence had already occurred. These findings support hypothesis (3)—that MSR is inherently more challenging than related audio tasks—underscoring the need for dedicated research to develop more effective MSR models.

5. DISCUSSION

Our baseline results reveal an unusual gap between waveform-level and perceptual metrics: SI-SDR values remain modest (often negative) despite *mel*-SSIM scores that are on par with—or better than—typical music-source–separation baselines. This confirms that MSR is intrinsically harder than MSS: each degraded mixture stems from an *unknown* sequence of nonlinear, many-to-one transformations, so perfectly reconstructing phase and amplitude is neither feasible nor strictly necessary for musical usefulness. Future work should therefore explore evaluation schemes that combine perceptual similarity with task-oriented objectives (e.g. remixing or effect-re-application), as well as crowd-sourced mean opinion score (MOS) studies, to complement SI-SDR.

A second limitation lies in the *synthetic* nature of our degradation pipeline. Although the chosen effects (EQ, compression, saturation, reverberation, lossy codecs) cover the bulk of practical production steps, their parameter distributions were selected heuristically. Real-world sessions exhibit correlations across effects (e.g. compression followed by EQ boosts) and producer-specific preferences that our independent sampling ignores. Extending RawStems with *profiling* of commercial stems, or weakly supervised learning on user-generated stems scraped from multitrack communities would yield more realistic mixtures and stronger generalization tests.

Finally, our single-stem training regime highlights scalability challenges: computing budgets forced us to train nine separate U-Former models, yet preliminary experiments on narrower sub-classes (acoustic vs. electric guitar) improved both metrics. These findings suggest that (i) *multi-task* or *Mixture-of-Experts* architectures could share low-level representations while retaining class-specific experts, and (ii) hierarchical prompts (e.g. "guitar:acoustic") [54] may let a single conditional model handle the full stem taxonomy. We release our code and data to encourage exploration along these directions.

In all, we call for attention from the research community to tackle this challenge together towards building useful creative tools.

6. CONCLUSIONS

We introduced Music Source Restoration (MSR), a novel task addressing the gap between idealized source separation and real-world audio processing needs by modeling complex degradations in music production. We propose RawStems dataset, the largest-scale dataset for musical sources, annotating 578 songs with hierarchical instrument annotations across 8 primary and 17 secondary categories. We validate the feasibility of task and RawStems through baseline models. To our knowledge, the RawStems dataset is the first publicly available dataset that can be applied to address the MSR problem. In future, we plan to restore large-scale audio recordings from the internet to construct music stems dataset.

REFERENCES

[1] K. Koffka, *Principles of Gestalt psychology*. routledge, 2013.

[2] F.-R. Stöter *et al.*, "Open-Unmix-a reference implementation for music source separation," *JOSS*, vol. 4, no. 41, p. 1667, 2019.

[3] A. Défossez *et al.*, "Music source separation in the waveform domain," *arXiv*, 2019.

[4] W.-T. Lu *et al.*, "Music source separation with band-split RoPE transformer," in *IEEE ICASSP*, 2024, pp. 481–485.

[5] J.-C. Wang *et al.*, "Mel-band roformer for music source separation," *arXiv*, 2023.

[6] D. Stoller *et al.*, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," in *ISMIR*, 2018.

[7] Y. Luo *et al.*, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *TASLP*, vol. 27, no. 8, pp. 1256–1266, 2019.

[8] T. Nakamura *et al.*, "Time-domain audio source separation with neural networks based on multiresolution analysis," *TASLP*, vol. 29, pp. 1687–1701, 2021.

[9] S. Uhlich *et al.*, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *IEEE ICASSP*, 2017, pp. 261–265.

[10] P.-S. Huang *et al.*, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *TASLP*, vol. 23, no. 12, pp. 2136–2147, 2015.

[11] N. Takahashi *et al.*, "MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *IWAENC*, 2018, pp. 106–110.

[12] A. Jansson *et al.*, "Singing voice separation with deep U-Net convolutional networks," in *ISMIR*, 2017.

[13] R. Hennequin *et al.*, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *JOSS*, vol. 5, no. 50, p. 2154, 2020.

[14] Y. Luo *et al.*, "Music source separation with band-split rnn," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1893–1901, 2023.

[15] A. Défossez, "Hybrid spectrogram and waveform source separation," in *ISMIR Workshop*, 2021.

[16] S. Rouard *et al.*, "Hybrid transformers for music source separation," in *ICASSP*, 2023.

[17] J. Watkinson, *Art of Digital Audio*. Routledge, 2013.

[18] U. Zölzer, *Digital Audio Signal Processing*. John Wiley & Sons, 2022.

[19] P. E. Sabine, "The acoustics of sound recording rooms," *Transactions of the Society of Motion Picture Engineers*, vol. 12, no. 35, pp. 809–822, 1928.

[20] P. Bottalico *et al.*, "Reproducibility of voice parameters: The effect of room acoustics and microphones," *Journal of Voice*, vol. 34, no. 3, pp. 320–334, 2020.

[21] I. Corbett, *Mic it!: Microphones, Microphone techniques, and Their Impact on the Final Mix*. Routledge, 2020.

[22] V. S. Fahed *et al.*, "Comparison of acoustic voice features derived from mobile devices and studio microphone recordings," *Journal of Voice*, 2022.

[23] M. Senior, *Mixing Secrets for the Small Studio*. Routledge, 2018.

[24] M. Shelvock, *Audio Mastering as Musical Practice*. The University of Western Ontario (Canada), 2012.

[25] M. R. Islam, "Communication protocols and technologies for multimedia transmission: A comprehensive study," *Journal of Innovative Technology Convergence*, vol. 4, no. 1, 2022.

[26] Z. Wang *et al.*, "Image quality assessment: from Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[27] T. Namgyal *et al.*, "What you hear is what you see: Audio quality metrics from image quality metrics," *arXiv*, 2023.

[28] J. Le Roux *et al.*, "SDR–half-baked or well done?" in *ICASSP*. IEEE, 2019, pp. 626–630.

[29] Z. Rafii *et al.*, "MUSDB18-HQ-an uncompressed version of MUSDB18," 2019.

[30] I. Pereira *et al.*, "MoisesDB: A dataset for source separation beyond 4-stems," *arXiv*, 2023.

[31] R. Izhaki, *Mixing Audio: Concepts, Practices, and Tools*. Routledge, 2017.

[32] R. A. Greiner *et al.*, "Design aspects of graphic equalizers," *JAES*, vol. 31, no. 6, pp. 394–407, 1983.

[33] G. Massenburg, "Parametric equalization," in *AES Convention 42*. AES, 1972.

[34] J. O. Smith, *Introduction to Digital Filters: with Audio Applications*. Julius Smith, 2007, vol. 2.

[35] R. Gommers *et al.*, "scipy/scipy: Scipy 1.15. 0," *Zenodo*, 2024.

[36] D. Giannoulis *et al.*, "Digital dynamic range compressor design—a tutorial and analysis," *JAES*, vol. 60, no. 6, pp. 399–408, 2012.

[37] G. W. McNally, "Dynamic range control of digital audio signals," *JAES*, vol. 32, no. 5, pp. 316–327, 1984.

[38] P. Sobot, "Pedalboard," Jul. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.7817838

[39] A. Schuck Jr *et al.*, "Audio nonlinear modeling through hyperbolic tangent functionals," in *DAFx*, 2016, pp. 5–9.

[40] E. Tarr, "The development of audio software with distortion," in *Distortion in Music Production*. Focal Press, 2023, pp. 13–27.

[41] S. McGuire *et al.*, "Mixing," in *The Art of Digital Orchestration*. Focal Press, 2020, pp. 166–208.

[42] J. O. Smith III, "Physical audio signal processing: For virtual musical instruments and audio effects," 2010.

[43] K. Li *et al.*, "Apollo: Band-sequence modeling for high-quality audio restoration," in *IEEE ICASSP*, 2025.

[44] P. Sripada, "MP3 decoder in theory and practice," 2006.

[45] F.-R. Stöter *et al.*, "The 2018 signal separation evaluation campaign," in *LVA/ICA*. Springer, 2018, pp. 293–305.

[46] Y. Wu *et al.*, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP*, 2023.

[47] A. Vosoughi *et al.*, "Learning audio concepts from counterfactual natural language," in *IEEE ICASSP*, 2024, pp. 366–370.

[48] S. Braun *et al.*, "Towards efficient models for real-time deep noise suppression," in *IEEE ICASSP*, 2021, pp. 656–660.

[49] Z.-B. Chen *et al.*, "DTT-Net: Dual-domain translation transformer for semi-supervised image deraining," in *IEEE ICIP*, 2022, pp. 1621–1625.

[50] A. I. Mezza *et al.*, "Benchmarking music demixing models for deep drum source separation," in *IEEE International Symposium on the Internet of Sounds (IS2)*, 2024, pp. 1–6.

[51] R. Kumar *et al.*, "High-fidelity audio compression with improved rvqgan," *NeurIPS*, vol. 36, pp. 27 980–27 993, 2023.

[52] H. Liu *et al.*, "Voicefixer: Toward general speech restoration with neural vocoder," *arXiv preprint arXiv:2109.13731*, 2021.

[53] S. Godsill *et al.*, *Digital audio restoration*. Springer, 2002.

[54] X. Liu *et al.*, "Separate what you describe: Language-queried audio source separation," *arXiv preprint arXiv:2203.15147*, 2022.