UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

# The Impact of Remixing on Vocal Extraction in Low Data Regimes

*Author:*
Taras SVYSTUN

*Supervisor:*
Taras SEREDA

*A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science*

*in the*

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences

Lviv 2024

# Declaration of Authorship

I, Taras SVYSTUN, declare that this thesis titled, “The Impact of Remixing on Vocal Extraction in Low Data Regimes” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

*“If all art aspires to the condition of music, all the sciences aspire to the condition of mathematics.”*

George Santayana

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

**The Impact of Remixing on Vocal Extraction in Low Data Regimes**

by Taras SVYSTUN

## *Abstract*

The field of music source separation aims to split a musical composition into 4 groups of instruments: *vocals*, *drums*, *bass* and *others*. Increasingly often, researchers use remixing and data augmentation to enlarge the amount of data and its diversity. The purpose of this thesis is to investigate what performance gains can be expected from remixing, as well as what optimal parameters are required for this. Therefore, this study uses the classical remixing method and tests a new method, which consists of mixing human speech with musical accompaniment. After finding the optimal hyperparameters, another data augmentations are used during the final training. The results of this methodology are the list of optimal parameters for remixing, all the steps necessary to reproduce the results, and the improved checkpoint for the SOTA MSS model - Band-split RNN, previously best publicly available. The findings indicate that remixing and data augmentation techniques are indeed powerful and necessary to improve vocal source separation.

# *Acknowledgements*

I am highly grateful to my supervisor, Taras Sereda, for his invaluable advice, constant support, and patience during the course of my thesis. His wide knowledge and diverse experience inspired me throughout my research and everyday life.

# Contents

# List of Figures

To be done...

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **MSS** | **M**usic **S**ource **S**eparation |
| **MAE** | **M**ean **A**bsolute **E**rror |
| **RMSE** | **R**oot **M**ean **S**quared **E**rror |
| **BLSTM** | **B**idirectional **L**ong **S**hort-**T**erm **M**emory |
| **TF** | **T**ime **F**requency |
| **STFT** | **S**hort **T**ime **F**ourier **T**ransform |
| **t-p-l** | **t**orch-**p**ytorch-**l**ighting |

To be done...

*Dedicated to the world*

# Chapter 1

# Introduction

## 1.1 Motivation

69% of people claim **music is important** to their mental health, according to a study of music engagement in 2022[15]. More than that, people spend an average of 20 hours listening to music weekly, compared to 18.4 hours in the previous year of 2021. Modern music consists of many different instruments, and for simplicity, they are usually divided into four categories: *vocals*, *drums*, *bass*, and *others*. In the music industry, artists only upload the final mixed composition, so no one has information about individual instruments.

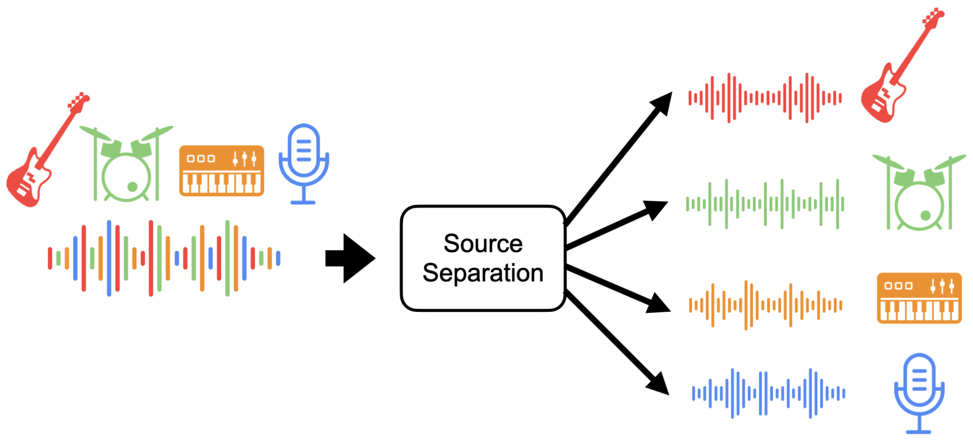


FIGURE 1.1: MSS task. The left part represent "mixed" song, which is the input to MSS models and the right part shows 4 individal sources, each one being the model's target.

The area of music source separation (MSS) is exactly to solve this problem. Usually, the task of separation is to divide a musical composition into the four mentioned groups of categories[27]. Consider Figure 1.1 to better understand the task. Such outputs can be used in **music education**[8], [4], **music remixing**[12], [49], [31], and **music information retrieval**[26], [2], [16], [35], [22]. For example, a teacher can separate a drum part from an audio composition and analyze it in more detail at a music school. A DJ can also extract stems from different songs and get a unique track that didn't exist before.

In artificial intelligence and machine learning, there is a separate area dedicated to audio analysis. Modern models in some audio tasks outperform humans; for instance, this study[13] shows that personalized automatic speech recognition models beat expert human transcribers. At the same time, many recent MSS papers introduced significant improvements each time they were published. This creates a feeling of **important details missed by models** with every publication. Thus, there is promising room for improvement.

To train modern MSS models, each dataset entry is supposed to contain one song (*mix*) and 4 clean sources (*vocals*, *drums*, *bass*, *other*). Public datasets that fit the former description can be counted on the fingers of one hand. It is known that the performance of many MSS models is **limited** by the availability of high-quality clean training **data**. Several modern papers proposed methods to employ unlabeled data during training[10], [47], or to generate pseudo labels from lots of unlabeled data by applying a pre-trained model[44], [43]. In conclusion, lack of **data is a pain**, diverse data is needed hence augmentation comes in handy.

According to the most common MSS model leaderboard[1], Band-split RNN[24] is the model with the highest results as of summer 2022. However, the authors did **not publish** a **trained model**. Instead, the following[2] unofficial pytorch implementation of the BSRNN[24] paper was found. These are the best public results for a BSRNN[24] trained for vocal extraction.

Therefore, the aim of my study was to test a **new remixing method** and use it in combination with other augmentation techniques to improve baseline results (nonofficial implementation).

Unfortunately, due to time and GPU computing power limitations, all remixing experiments were performed **only** with **BSRNN**[24] model. All other SOTA models[19], [36] were used only on the inference level for comparison with my results. Additionally, the BSRNN[24] architecture implies separate training for each stem, so the focus was on the **vocals separation** only. It is motivated by the nature of how vocals are different from the music instruments. The remaining three sound sources are naturally combined into one group, often referred to as the accompaniment. Therefore, the MSS is considered as the vocals vs accompaniment separation.

## 1.2 Contributions

1. Implementation of **new remixing** technique and **its application** to the training dataset
2. **Expanding** the existing code base with other relevant audio **augmentations**[20], [38] and **their application** during training
3. Publishing the **improved checkpoint**[3] for **vocal separation** by the BSRNN[24] model – the final result of all the previous steps.
4. Adjustment of existing BSRNN[24] pytorch[29] implementation to **multi-GPU** acceleration

The reader is invited to listen the vocal extraction and to compare the trained checkpoint to SOTA approaches by this link[4].

[1] `https://paperswithcode.com/sota/music-source-separation-on-musdb18`

[2] `https://github.com/amanteur/BandSplitRNN-PyTorch`

[3] `https://drive.google.com/file/d/19LUERZ8omfZhyAzUVZElSEMwGw34kVF2/view?usp=sharing`

[4] `https://taras-svystun.github.io/Band-Split-RNN/`

## 1.3 Structure Of The Thesis

### 1.3.1 Chapter 2. Related Work

This chapter justifies the choice of model, data, and remixing techniques for this study. The five most recent MSS models are analyzed, and a brief overview of the primary data sources and a list of 5 selected augmentation techniques are provided.

### 1.3.2 Chapter 3. Theoretical Background

In this chapter, the reader is introduced to the basic terminology of the MSS field. It explains the basic notions that are important for understanding the model, the data it works with, the metrics it evaluates, and the techniques for preprocessing speech data for further mixing with music audio tracks.

### 1.3.3 Chapter 4. Proposed Solution

This part of the thesis describes the main steps involved in achieving the results of this study, namely what was changed, how the remixing process was performed, what experiments were conducted, and what statistical methods were used to evaluate the results.

### 1.3.4 Chapter 5. Experiments and Results

In this part of the study, all numbers obtained during the experiments are gathered and analyzed. All necessary steps required to reproduce the results are described in detail, and statistical hypothesis testing is applied to verify that the results are not a coincidence.

### 1.3.5 Chapter 6. Conclusions

In the end, all the results of this study are summarized, it is mentioned whether the goals of the thesis were achieved, and there are some comments regarding the further research.

# Chapter 2

# Related Work

The chapter of related works is highlighting significant models, datasets and notions that have been developed over the recent 2.5 years, from Nov 2021 to Feb 2024.

## 2.1 Models

There are five recent models of main interest: KUIELab-MDX-Net[19] (Nov 2021), Band-split RNN[24] (Sep 2022), Sparse Hybrid Transformer Demucs[36] (Nov 2022), Pac-HuBERT[5] (Apr 2023), and SCNet[42] (Jan 2024). These models represent significant advancements in the MSS field[1], however, neither of them have open source codes available, except for Demucs_v4[9]. In particular, the Sparse Hybrid Transformer[36] requires custom CUDA code that is not released. The source code for SCNet[42] is also not currently accessible. Both SCNet[42] and Sparse Hybrid Transformer[36] demonstrates impressive performance, achieving 9.89 dB and 9.37 dB respectively on the MUSDB18[32] dataset for vocal source. Their mean SDR across the four sources is 9.25 dB and 9.2 dB respectively. The Pac-HuBERT[5] model does not achieve as high results (8.5 dB on vocals and only 6.42 dB average), however some useful pre-training details was introduced by that paper, regarding Pac-HuBERT[5] additional data sources: LibriSpeech[28], LibriLight[18] and FMA[7] datasets for self-supervised pretraining. Like SCNet[42] and Sparse Hybrid Transformer[36], Pac-HuBERT[5] does not have an open source code.

### 2.1.1 KUIELab-MDX-Net

The KUIELab-MDX-Net model[19] (fig. 2.1) is interesting because it consists of 6 models and offers an approach that is not seen in other models.

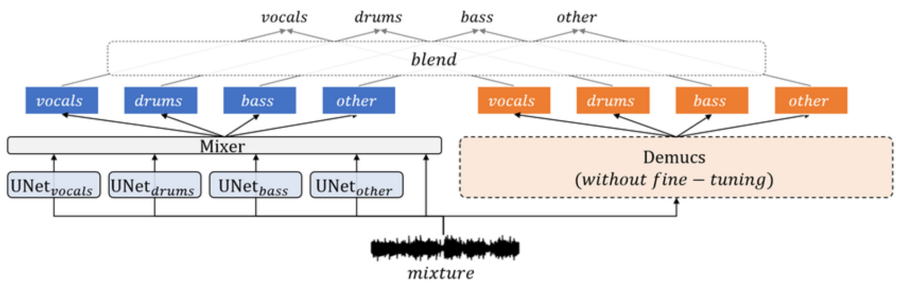


FIGURE 2.1: Architecture of KUIELab-MDX-Net.

[1]https://paperswithcode.com/sota/music-source-separation-on-musdb18

It all started with the release of TFC-TDF-U-Net v1[6] for singing voice separation, which was too computationally heavy for the authors to submit to the MDX challenge[25]. KUIELab-MDX-Net consists of six networks, each of which is trained separately. The four U-Net-based separation models [TFC- TDF-U-Net v2 (fig. 2.2)] first estimate each source independently, then the Mixer model takes these estimated sources (+ mixture) and outputs the improved estimated sources. Unlike the other models considered in this study, MDX's strength is using the fact that the separated 4 sources come from the same mix, for example, MDX having a separated drums source can remove the snare sound that can sometimes leak into the vocal source. In addition, the authors also extract sources using another network, Demucs[9], without fine-tuning. Finally, each estimated source is blended[45].

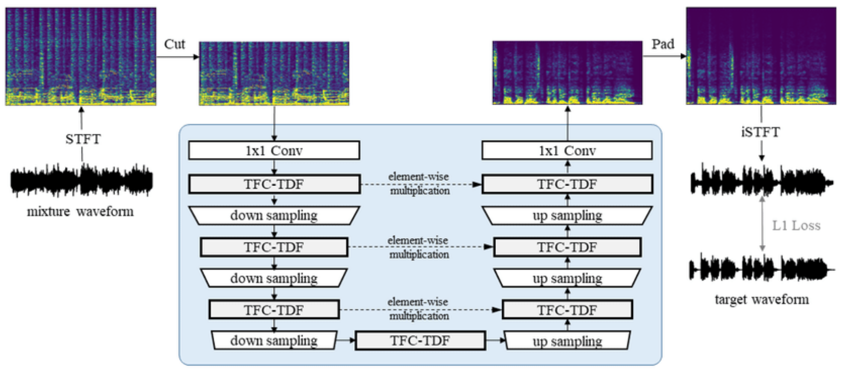


FIGURE 2.2: The architecture of TFC-TDF-U-Net v2.

Just like other models, such as Demucs[36] and Pac-Hubert[5], MDX is trained with L1 loss in the time domain. And like all other models, except HT Demucs[36], MDX[19] is a time-frequency model. The model was fully trained on Musdb18-hq[33], with a standard 86 train, 14 validation split. In 2021 MDX[19] beat all competitors on musdb18[32] and showed 9.00 dB cSDR on vocal and 7.53 dB cSDR average on all sources.

### 2.1.2 Band-split RNN

The Band-split RNN[24] (fig. 2.3) is a time-frequency model. The architecture consists of three large blocks resulting in 32.4 million parameters, trained exclusively on the MUSDB18-HQ[33] dataset. The authors of the paper propose two training scenarios. The first one is the standard supervised training approach. In contrast, the second scenario involves an innovative semi-supervised data sampling technique. This method uses a pre-trained supervised model to produce pseudo-labels and identify clean segments, which aids in the effective fine-tuning of the model on both labeled and unlabeled datasets. Later the final model is trained on a combination of original supervised data and semi-supervised data. The model utilizes L1 composite loss on both time and spectrogram domains.

The architecture consists of 3 main blocks:

1. The Band Split module transforms the input complex spectrogram from STFT into K subband spectrograms, where each subband undergoes layer normalization and a fully-connected layer to generate real-valued subband features.

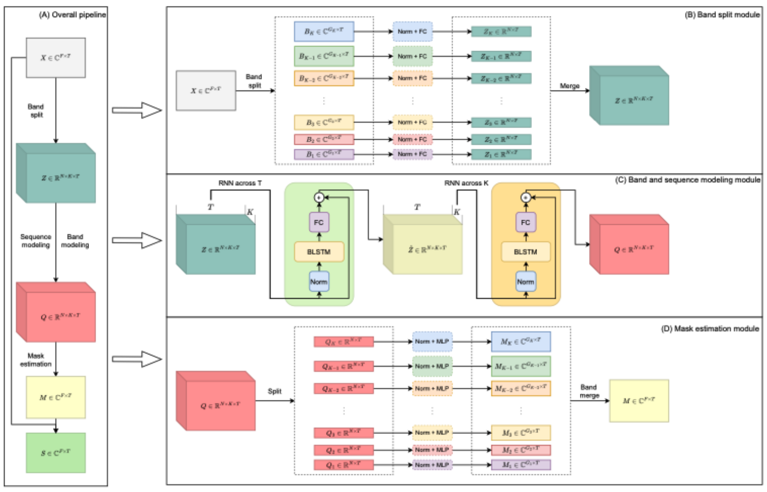

(A) Overall pipeline
$X \in \mathbb{C}^{F\times T}$
Band split
$Z \in \mathbb{R}^{N\times K\times T}$
Sequence modeling
Band modeling
$Q \in \mathbb{R}^{N\times K\times T}$
Mask estimation
$M \in \mathbb{C}^{F\times T}$
$S \in \mathbb{C}^{F\times T}$
(B) Band split module
$X \in \mathbb{C}^{F\times T}$
Band split
$B_K \in \mathbb{C}^{G_K\times T}$
$B_{K-1} \in \mathbb{C}^{G_{K-1}\times T}$
$B_{K-2} \in \mathbb{C}^{G_{K-2}\times T}$
$B_3 \in \mathbb{C}^{G_3\times T}$
$B_2 \in \mathbb{C}^{G_2\times T}$
$B_1 \in \mathbb{C}^{G_1\times T}$
Norm + FC
$Z_K \in \mathbb{R}^{N\times T}$
$Z_{K-1} \in \mathbb{R}^{N\times T}$
$Z_{K-2} \in \mathbb{R}^{N\times T}$
$Z_3 \in \mathbb{R}^{N\times T}$
$Z_2 \in \mathbb{R}^{N\times T}$
$Z_1 \in \mathbb{R}^{N\times T}$
Merge
$Z \in \mathbb{R}^{N\times K\times T}$
(C) Band and sequence modeling module
RNN across T
$T$
$K$
$Z \in \mathbb{R}^{N\times K\times T}$
FC
BLSTM
Norm
$\hat{Z} \in \mathbb{R}^{N\times K\times T}$
RNN across K
$Q \in \mathbb{R}^{N\times K\times T}$
(D) Mask estimation module
$Q \in \mathbb{R}^{N\times K\times T}$
Split
$Q_K \in \mathbb{R}^{N\times T}$
$Q_{K-1} \in \mathbb{R}^{N\times T}$
$Q_{K-2} \in \mathbb{R}^{N\times T}$
$Q_3 \in \mathbb{R}^{N\times T}$
$Q_2 \in \mathbb{R}^{N\times T}$
$Q_1 \in \mathbb{R}^{N\times T}$
Norm + MLP
$M_K \in \mathbb{C}^{G_K\times T}$
$M_{K-1} \in \mathbb{C}^{G_{K-1}\times T}$
$M_{K-2} \in \mathbb{C}^{G_{K-2}\times T}$
$M_3 \in \mathbb{C}^{G_3\times T}$
$M_2 \in \mathbb{C}^{G_2\times T}$
$M_1 \in \mathbb{C}^{G_1\times T}$
Band merge
$M \in \mathbb{C}^{F\times T}$

FIGURE 2.3: The architecture of Band-split RNN model.

2. The Band and Sequence Modeling module utilizes two residual RNN layers for interleaved sequence-level and band-level modeling. The sequence-level RNN operates across the temporal dimension, and the band-level RNN captures intra-band feature dependencies across K subbands, producing an output denoted as Q.

3. The Mask Estimation module calculates a complex-valued time-frequency mask for target source extraction. It processes the output Q from the previous module, splitting it into K features, generating real and imaginary parts of T-F masks for each subband, and merging them into a fullband T-F mask, which is multiplied with the input complex spectrogram to generate the target spectrogram.

The success of this method can be attributed to its unique approach of splitting the input spectrogram into bands. The model is capable of learning useful information about each separate source. The Band-split RNN[24], with the help of BLSTM, learns the band and sequence level interaction for each separate source, which is a crucial factor in its effectiveness. The model's high performance might be attributed to the fact of splitting input information into bands, where some instruments naturally prevail when others occupy their corresponding bands.

### 2.1.3 Sparse Hybrid Transformer Demucs

The Sparse Hybrid Transformer Demucs[36] (fig. 2.4) is the only model that operates in both the time domain and the T-F domain. This model, containing 41.4 million parameters, is trained on the MUSDB18-HQ[33] dataset and an additional private dataset of 800 songs, which unfortunately, is not publicly accessible. Dataset preprocessing involved filtering silent stems, ensuring non-silent periods, and evaluating separation quality using an established criterion. The training process is supervised, utilizing a MAE loss function in the time domain.

The architecture of Sparse Hybrid Transformer Demucs[36] is based on its predecessor[11] and is composed of two U-Nets[34]. Unlike its predecessor the 2 innermost layers in the encoder and the decoder are replaced with a cross-domain Transformer Encoder. Cross-domain encoder is a unique feature, that allows demucs to work

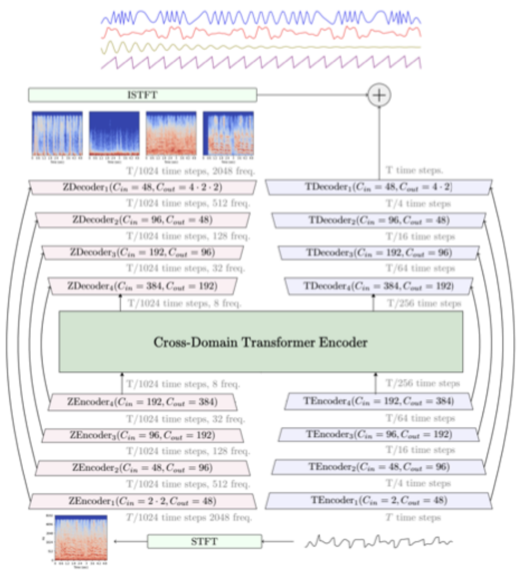

ISTFT
T/1024 time steps, 2048 freq.
T time steps.
ZDecoder$_1$($C_{in}$ = 48, $C_{out}$ = 4 · 2 · 2)
TDecoder$_1$($C_{in}$ = 48, $C_{out}$ = 4 · 2]
T/1024 time steps, 512 freq.
T/4 time steps
ZDecoder$_2$($C_{in}$ = 96, $C_{out}$ = 48)
TDecoder$_2$($C_{in}$ = 96, $C_{out}$ = 48)
T/1024 time steps, 128 freq.
T/16 time steps
ZDecoder$_3$($C_{in}$ = 192, $C_{out}$ = 96)
TDecoder$_3$($C_{in}$ = 192, $C_{out}$ = 96)
T/1024 time steps, 32 freq.
T/64 time steps
ZDecoder$_4$($C_{in}$ = 384, $C_{out}$ = 192)
TDecoder$_4$($C_{in}$ = 384, $C_{out}$ = 192)
T/1024 time steps, 8 freq.
T/256 time steps
Cross-Domain Transformer Encoder
T/1024 time steps, 8 freq.
T/256 time steps
ZEncoder$_4$($C_{in}$ = 192, $C_{out}$ = 384)
TEncoder$_4$($C_{in}$ = 192, $C_{out}$ = 384)
T/1024 time steps, 32 freq.
T/64 time steps
ZEncoder$_3$($C_{in}$ = 96, $C_{out}$ = 192)
TEncoder$_3$($C_{in}$ = 96, $C_{out}$ = 192)
T/1024 time steps, 128 freq.
T/16 time steps
ZEncoder$_2$($C_{in}$ = 48, $C_{out}$ = 96)
TEncoder$_2$($C_{in}$ = 48, $C_{out}$ = 96)
T/1024 time steps, 512 freq.
T/4 time steps
ZEncoder$_1$($C_{in}$ = 2 · 2, $C_{out}$ = 48)
TEncoder$_1$($C_{in}$ = 2, $C_{out}$ = 48)
T/1024 time steps 2048 freq.
T time steps
STFT

FIGURE 2.4: The architecture of Hybrid Transformer Demucs

with heterogeneous data shape both in time and spectral domains, making it a more flexible architecture. It treats in parallel the 2D signal from the spectral branch and the 1D signal from the waveform branch.

In conclusion, the effectiveness of this model can be attributed to several factors. Firstly, it benefits from the utilization of both 1D and 2D features, which come in the form of waveforms and spectrograms, respectively. Secondly, the model employs the transformer, one of the most significant advancements in deep learning, to model sequence data. These elements combined contribute to the model's high performance in musical source separation tasks. In comparison with baselines, Hybrid Transformer Demucs[36] model outperforms the original Hybrid Demucs[9] architecture, achieving a 2.5 dB improvement in SDR. Investigating architectural hyperparameters reveals optimal settings for model performance, emphasizing the significance of data augmentation, particularly in remixing. The recommendations from the paper[9] allign with what is found among other relevant works addressing remixing and data augmentations[17], [20], [38]: pitch change in $\pm$ 3 semi-tones and not more than 15% tempo change.

### 2.1.4 Pac-HuBERT

The Pac-HuBERT[5] (fig. 2.5) model is a spectrogram based approach. Regrettably, the exact size of this model is not specified in the original paper, and no open-source code is available for reference. This model is pretrained using a self-supervised approach, specifically through Masked Unit Prediction. The initial labels for self-supervision are derived from KMeans clustering applied to primitive auditory features, which are computed over spectrogram patches. Three datasets, namely LibriSpeech[18], LibriLight[18], and the FMA[7] dataset, are utilized for unsupervised pretraining. Subsequent fine-tuning is supervised and employs MAE loss in the time domain:

$$MAE = \|s - \hat{s}\|_1,$$

where s and $\hat{s}$ are true and separated signals in the time domain. The fine-tuning process is flexible and it is possible to freeze the pretrained part and fine-tune only the decoder part. Or to unfreeze a few encoder layers, that are the closest to decoder. Or it is possible to fine tune the whole network. The authors train the whole network During this fine-tuning phase, meaning all parameters are subject to training. The authors of the paper present evaluation metrics for scenarios both with and without the self-supervised approach. Fine-tuning the pretrained Pac-HuBERT[5] model on the separation task demonstrates improved SDR for all sources compared to models without pretraining. Additionally, the study explores the impact of limited training data on Pac-HuBERT[5], showing significant performance gains, particularly for vocals and other source classes, with only 25% of supervised data. The conclusion emphasizes the effectiveness of Pac-HuBERT[5] as a solution for leveraging unlabeled music data in MSS tasks.

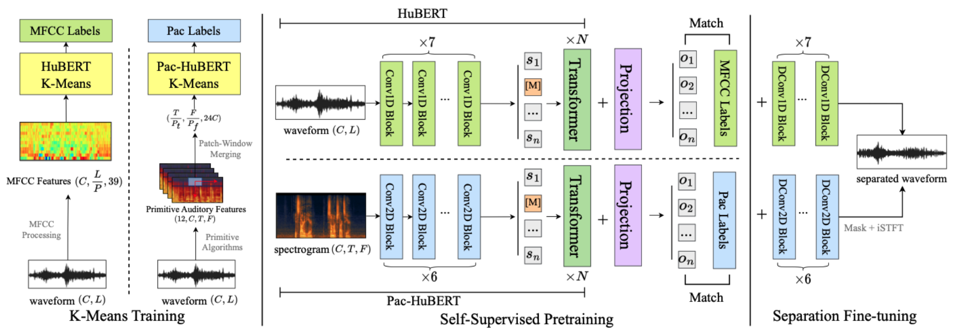


FIGURE 2.5: The architecture of self-supervised Pac-HuBERT.

The architecture consist of 3 main blocks:

1. The encoder model serves as a feature extractor for spectrograms using 2D convolutional blocks to downsample the input data and generate bottleneck features for further processing.
2. The bottleneck model, which consists of transformer encoder blocks containing multi-head self-attention layers and feedforward layers, transforms the encoder's output features into a latent representation suitable for subsequent tasks, such as source separation or self-supervised pretraining.
3. The decoder, which employs deconvolutional (DCNN) layers for both time-domain and TF domain models, reconstructs the separated sources or masks from the bottleneck features. It maintains spatial information through skip-connections between the encoder and decoder blocks, ensuring the output shape matches the input for consistent processing. Skip-connections connect earlier layers in the network directly to later layers. This way, information from earlier layers can "skip" some layers and not get lost. It helps to keep the details from the start of the network until the end.

In conclusion, the effectiveness of the Pac-HuBERT[5] model can be attributed to its innovative use of self-supervised learning, which allows it to leverage a large volume of unlabeled training data. Furthermore, it demonstrates the potency of the BERT concept[2] through Masked Unit Prediction.

[2]BERT, short for Bidirectional Encoder Representations from Transformers, is a method in ML for improving understanding of language context in a model. The power of BERT is shown through Masked Unit Prediction, a technique where some parts of data are hidden during training, forcing the model to predict aka fill the missing parts, thus learning more effectively.

### 2.1.5 SCNet

The SCNet[42] (fig. 2.6) model shares similarities with the BSRNN, as both employ explicit band splitting and use a dual path RNN for separation tasks. SCNet[42] is a Time-Frequency (T-F) model and is the smallest among the discussed models, with only 10.08 million parameters. However, a larger version of the model exists, having 41.2 million parameters. The authors propose two training scenarios: one with additional training data and one without. The additional training data consists of 235 four-stem songs from MoisesDB[30]. Both training procedures are supervised, with the additional data source serving to augment the volume of the training data. Unique to SCNet[42], the model employs the Root Mean Square Error (RMSE) loss of the complex-valued spectrogram:

$$\mathcal{L}_s = \sqrt{(r - \hat{r})^2 + (i - \hat{i})^2},$$

where r and i correspond to real and imaginary part of the source spectrogram; $\hat{r}$ and $\hat{i}$ – real and imaginary parts of the spectrogram of the predicted source.

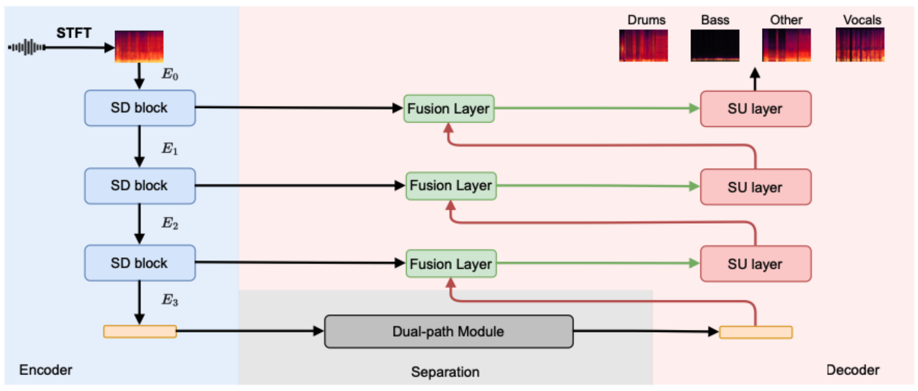


FIGURE 2.6: The architecture of SCNet.

As usual, the architecture consist of 3 main blocks:

1. The encoder module employs sparse down-sampling blocks (SD blocks) to compress the frequency axis of the input spectrogram while enhancing feature dimensions, incorporating convolution modules inspired by the Conformer architecture and prioritizing the preservation of low-frequency details.
2. Separation with Dual-Path Module: Focusing on modeling inter-subband dependencies and global information, the dual-path RNN architecture integrated with feature conversion modules facilitates effective sequence learning by incorporating torch[29] functions *torch.rfft* and *torch.irfft* to manipulate with complex $\mathbb{C}$ tensors between adjacent dual-path layers.
3. The decoder utilizes skip connections to integrate hierarchical features from the encoder, reconstructing the separated spectrogram gradually through a sparse up-sampling layer (SU layer), incorporating a fusion layer with gated linear units (GLU) to enhance information flow and transposed convolution for reconstruction, with output channel dimensions adjusted based on the specified number of sources for separation.

In my understanding, the effectiveness of this method lies in its specific application of explicit band splitting. Unlike BSRNN[24], SCNet's authors use only three bands: high, low, and medium frequencies. The splitting points are determined by the band-split ratios and authors report it as [0.175, 0.392, 433], i.e. splitting points approximately are 3.8 kHz and 12.5 kHz. The authors also apply varying compression rates, recognizing that the low band contains denser information, thus necessitating different treatment. The term "dense information" in this context refers to the fact that low-frequency bands often contain more significant and detailed information about the sound source. This is because low frequencies typically contain more power and carry more energy than high frequencies.

The model subsection could be concluded with the table 2.1 summarizing the key characteristics and performance of the five models.

TABLE 2.1: MSS models summary table

| Model | Size, m | Domain | Loss | Extra? | Data augm | SDR, dB | |
|---|---|---|---|---|---|---|---|
| | | | | | | vocal | mean |
| MDX | — | TF | L1 wave | — | remix + data aug | 9.00 | 7.53 |
| Band-split RNN | 32.4 | TF | L1 wave + freq | — | remix + data aug | 10.01 | 8.24 |
| Band-split RNN | 32.4 | TF | L1 wave + freq | — | remix + semi-sup | 10.47 | 8.97 |
| Sparse HT Demucs | 41.4 | T+TF | L1 wave | 800 | remix + data aug | 9.37 | 9.20 |
| Pac-HuBERT | — | TF | L1 wave | — | remix | 8.07 | 6.09 |
| Pac-HuBERT | — | TF | L1 wave | 890h | remix + self-sup | 8.52 | 6.42 |
| SCNet | 10.08 | TF | L2 freq | — | remix + data aug | 9.89 | 9.00 |
| SCNet | 10.08 | TF | L2 freq | 235 | remix + data aug | 10.17 | 9.25 |
| SCNet-large | 41.2 | TF | L2 freq | — | — | 10.86 | 9.69 |
| SCNet-large | 41.2 | TF | L2 freq | 235 | — | 11.10 | 9.92 |

## 2.2 Datasets

Data is an integral part of any ML research. Andrew Ng often mentions that in deep learning, larger data + bigger models = better performance. The supervised data according to MSS is a collection of mixed songs and 4 corresponding clean sources per each mixture[32]. As mentioned, such data is scarce[17], that's why some studies[5], [24] also adopt unlabeled data – mixtures tracks without clean stems.

### 2.2.1 Supervised

The MUSDB18[32] and its uncompressed version[33] are the most popular datasets for MSS benchmarks according to this leaderboard[3]. It includes 150 unique songs, which is approximately 10 hours of audio data, with separate recordings for every musical instrument. These recordings are classified into four groups - vocals, bass, drums, and an 'other' category for any additional instruments. This classification system helps researchers train and evaluate their models using consistent data, allowing for advancements in music separation technologies.

The dataset was introduced in 2017 and was prominently featured during the Signal Separation Evaluation Campaign (SiSEC) in 2018[40]. Musdb18[32] is composed of other datasets: MedleyDB (Oct 2014) [3] is a dataset of annotated, royalty-free multitrack recordings, which contributed by 46 tracks in the musdb18[32]. Another 100 multiltracks are taken from DSD100[23] dataset. 2 tracks from Native Instruments

[3]https://paperswithcode.com/sota/music-source-separation-on-musdb18

as part of their stems pack[4]. And 2 tracks from The Easton Ellises, the Canadian rock band[5]. This dataset consists of 100 train and 50 test songs. On the main Music Demixing Challenge[25] there are 2 leaderboards: (A) – with constraint of not training on test set and (B) – the one without training constraints, which allowed participants to use test set during training stage.

### 2.2.2 Unsupervised

As mentioned in [5] by utilizing large amounts of unlabeled data, one may improve model training and generalization. In that paper, LibriSpeech[28], LibriLight[18] and FMA[7] are used to pre-train encoder parts of model on large amount of unlabeled data. Most importantly, LibriSpeech[28] consist of dev, test and train versions. As for my research, the dev-clean set is of main interest. Its size is 5.4 hours, which is almost the same as musdb18[32]'s train part of $\approx$ 6.5 hours. Thus, unsupervised data sources analysis was essential in selection of the data source for current study.

## 2.3 Augmentation Techniques

One recent paper[17] highlighted a promising data augmentation technique that requires further research. This technique, called remixing, involves combining random stems from different songs. This method allows researchers to exponentially increase the amount of training data in Musdb18[32], leading to a potential increase of up to 2 dB in the SDR (fig. 2.7). The authors also describe pitch shift and time stretch (fig. 2.8), two very commonly used augmentation techniques, which link to [20] and [47]. The aforementioned augmentatios can be unified in the following configuration: [-2, 2] semi-tone pitch shift and [.9, 1.1] speed change. Finally, the paper about remixing gains[17] describes another useful augmentation, time shift, and their experiments show the optimal shift parameter being 1 sec.

Moreover, the Pac-HUBERT[5] model used the LibriSpeech[28] & Libri-Light[18] datasets to pretrain their models as part of a self-supervised learning process. This thesis work combines ideas from these two papers to develop a new remixing technique that creates random mixes by combining random accompaniment mixed from musdb18-HQ[33] with clean human speech from the LibriSpeech[28] dataset.

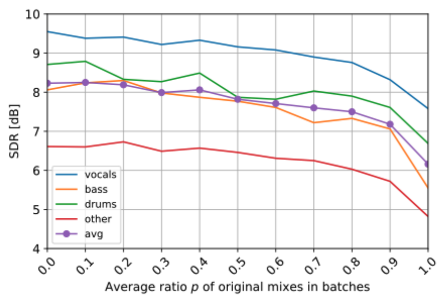


FIGURE 2.7: SDR depends on *remix_prob*

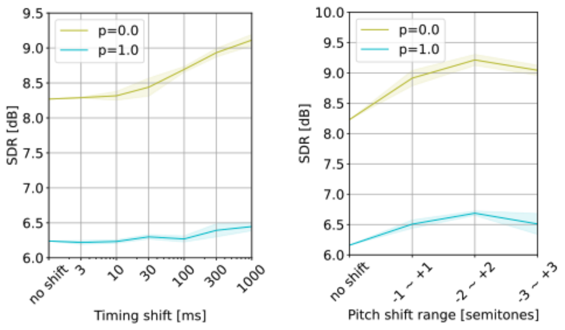


FIGURE 2.8: SDR depends on time & pitch shifts

[4]https://www.native-instruments.com/en/specials/stems-for-all/free-stems-tracks/

[5]https://www.heise.de/ct/artikel/c-t-Remix-Wettbewerb-The-Easton-Ellises-2542427.html#englisch

# Chapter 3

# Theoretical Background

To understand how modern MSS models work, it is required to be familiar with the following concepts:
1. Sampling rate and Nyquist–Shannon theorem
2. Shirt-time Fourier Transform (STFT) and Spectrograms
4. Signal-to-Distortion Ratio (SDR).

## 3.1 Sampling Rate and Nyquist–Shannon Theorem

The sampling rate, or sample rate, is essential in audio signal analysis. It refers to the number of samples of a signal taken per unit of time (usually per second) to represent the signal digitally. The unit of measurement for sampling rate is typically samples per second, $\frac{1}{sec}$ or Hertz (Hz).

The Kotelnikov Nyquist–Shannon theorem[39], also known as the sampling theorem, states that a signal can be perfectly reconstructed from its samples if the sampling frequency is greater than twice the maximum frequency of the signal. In maths language, if a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart. This is often referred to as the Nyquist rate, and it serves as a guideline for selecting the correct sample rate for a given signal[39].

In the context of human speech, the maximum frequency is typically around 8 kHz. Therefore, according to the Nyquist-Shannon theorem, the sampling rate should be at least 16 kHz to accurately represent the speech signal. For example, in this work, the human speech[28] was resampled from 16 kHz to 44.1 kHz to match the sampling rate of the musdb18 dataset[32]. Resampling is a process that changes the original sampling rate of a signal to a different rate. This is achieved by applying a digital filter to prevent aliasing and then changing the sample rate. It is a necessary step when working with signals of different sampling rates.

## 3.2 STFT and Spectrograms

The Short-Time Fourier Transform (STFT) is a technique to analyze the frequency content of signals that vary over time (unlike the original Fourier transform, which describes only the spectral side of the signal). It works by dividing the signal into small, overlapping windows and applying the Fourier Transform to each segment. This approach allows to observe how the frequency components of the signal change over time. The mathematical representation of STFT is given by:

$$STFT\{x(t)\}(f,\tau) = X(f,\tau) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j2\pi ft}dt,$$

where:
- $x(t)$ is the input signal,
- $w(t-\tau)$ is the window function centered around time $\tau$,
- $f$ is the frequency,
- $e^{-j2\pi ft}$ is the complex exponential function, indicating the Fourier Transform,
- $\tau$ is the time around which the window is centered,
- $X(f,\tau)$ is the STFT of $x(t)$, showing the frequency content of the signal at time $\tau$.

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. It is essentially the squared magnitude of the STFT, which can be represented as a 2D image where one axis represents time, another represents frequency, and the intensity of each point represents the amplitude (or energy) of a particular frequency at a particular time. Mathematically, the spectrogram $S(f,\tau)$ is defined as:

$$S(f,\tau) = |X(f,\tau)|^2$$

Hyperparameters of the Spectrogram are:
1. Window size ($N$): This is the size of the window function applied to each segment of the signal. It determines the resolution of the frequency analysis. A larger window size provides better frequency resolution but poorer time resolution, and vice versa.
2. Hop size ($H$): This is the number of samples by which the window is shifted to create the next segment. A smaller hop size results in higher overlap between consecutive windows, leading to a better time resolution in the spectrogram.
3. Window function: This function is applied to each segment of the signal to minimize the edge effects in the analysis, as the Fourier transform requires the function to be around 0 on the edges. Common window functions include Hamming, Hanning, Triangular, and Blackman windows, each with different characteristics affecting the spectral leakage and resolution.
4. Sampling Frequency ($F_s$): Although not a direct parameter of the spectrogram, the sampling frequency of the signal influences the frequency resolution and range of the spectrogram. Higher sampling frequencies allow for a wider range of detectable frequencies but require more computational resources.

Mel spectrograms and log-mel spectrograms are two types of spectrograms that are particularly useful in the field of audio signal processing. Unlike regular spectrograms, which display a signal's frequency content linearly over time, mel spectrograms represent the frequency content in a nonlinear scale, more specifically, the mel scale. This scale is designed to mimic the human ear's response to different frequencies, making it more perceptually relevant.

Log-mel spectrograms take this a step further by applying a logarithmic transformation to the mel spectrograms. This transformation further enhances the perceptual relevance by mimicking the human ear's response to different levels of sound intensity. In essence, the log-mel spectrogram can be thought of as a more human-like representation of the audio signal. For example, this robust model for vocal pitch estimation uses log mel spectrogram[48].

In the context of musical source separation (MSS), the log-mel spectrum can be particularly useful. It provides a more perceptually meaningful representation of the audio signal, which can make the task of separating different sources easier. For instance, it might be easier to distinguish between a high-pitched violin and a low-pitched cello in a log-mel spectrogram than in a regular spectrogram.

Each pixel in a spectrogram represents a specific frequency at a specific time. The intensity of the pixel indicates the presence and strength of that frequency at that time. In the context of MSS, this information can be used to identify and separate different musical sources. For example, if a particular frequency is strong at a specific time, it might indicate the presence of a specific musical instrument.

## 3.3 Signal-to-X Ratios

Initially, there are four metrics commonly used to evaluate MSS performance, which are collectively called Signal-to-X. All four of them were summarized in a recent paper[46] about performance measurement in blind audio source separation.

Distortion Ratio

$$\mathrm{SDR} := 10\log_{10}\frac{\left\|s_{\text{target}}\right\|^2}{\left\|\epsilon_{\text{interf}}+e_{\text{noise}}+e_{\text{artif}}\right\|^2},$$

the Source to Interferences Ratio

$$\mathrm{SIR} := 10\log_{10}\frac{\left\|\varepsilon_{\text{inriget}}\right\|^2}{\left\|e_{\text{interf}}\right\|^2},$$

the Sources to Noise Ratio

$$\mathrm{SNR} := 10\log_{10}\frac{\left\|s_{\text{target}}+e_{\text{interf}}\right\|^2}{\left\|e_{\text{noise}}\right\|^2},$$

and the Sources to Artifacts Ratio

$$\mathrm{SAR} := 10\log_{10}\frac{\left\|s_{\text{target}}+e_{\text{interl}}+e_{\text{noiso}}\right\|^2}{\left\|e_{\text{artir}}\right\|^2}.$$

In Musical Source Separation (MSS), the Signal-to-Distortion Ratio (SDR) serves as a primary metric, which measures the ratio between the power of the desired source and the power of the error (SDR is sometimes equivalent to other metrics from audio analysis: Signal-to-Noise, Source-to-Artifact, Source-to-Interference ratios). This ratio is a reflection of the quality of the separated sources. However, it's important to note that SDR is sensitive to slight alignment errors, which can impact the result. Thanks to the logarithmic scale, SDR is measured in dB and is defined as:

$$SDR = 10 \cdot log_{10}(\frac{||s||^2}{||s-\hat{s}||^2}),$$

where s is the true source waveform and $\hat{s}$ is the estimated waveform.

To better understand SDR, let's consider different values:

1. $||s||^2 = ||s - \hat{s}||^2$ – the energy of a source is the same as the error one. This results in 0 dB SDR.

2. When $||s||^2 > ||s - \hat{s}||^2$, SDR becomes a positive number. For instance, if $\frac{||s||^2}{||s-\hat{s}||^2} = 10$, i.e., the energy of a signal is 10 times more than the energy of an error, SDR would be $10 \cdot log_{10}(10) = 10$ dB.

3. When $||s||^2 < ||s - \hat{s}||^2$, SDR becomes negative. For instance, if $\frac{||s||^2}{||s-\hat{s}||^2} = .5$, i.e., the energy of a signal is 2 times less than the energy of an error, SDR would be $10 \cdot log_{10}(\frac{1}{2}) \approx -3$ dB.

For some visual intuition, please consider fig 3.1. To begin with, consider a mixture $x = s + n$ of a target signal s and an interference signal $n$. Let $\hat{s}$ be an estimate of the target. As can be seen from the figure, prediction $\hat{s}$ is not guaranteed to be orthogonal to the actual source s.

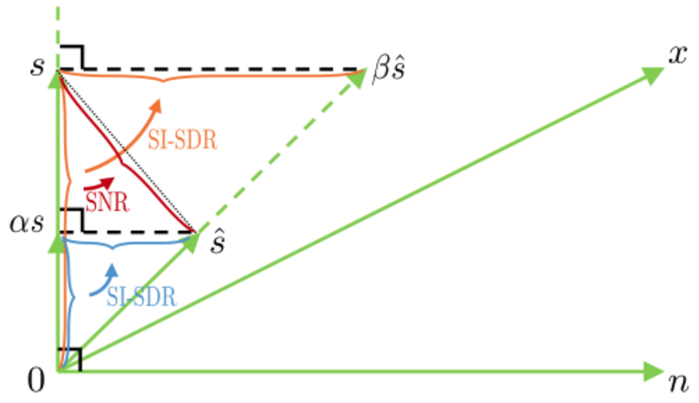


FIGURE 3.1: Difference between SDR (same as SNR is this case) and siSDR.

However, to improve the evaluation of the prediction, the scale-invariant SDR (siSDR) was introduced[37]. This metric ensures that the residual is orthogonal to the target, which can be achieved by either rescaling the target or the estimate. Rescaling the target such that the residual is orthogonal to it corresponds to finding the orthogonal projection of the estimate $\hat{s}$ on the line spanned by the target s, or equivalently finding the closest point to $\hat{s}$ along that line.

The above leads to two equivalent siSDR definitions (refer to fig. 3.1 for geometric visuals):

1. Scaling the prediction $\hat{s}$ to make the distortion orthogonal to the target s:

$$\text{SI-SDR} = \frac{|s|^2}{|s - \beta\hat{s}|^2} \text{ for } \beta \text{ s.t. } s \perp s - \beta\hat{s}$$

2. Scaling the target to make it orthogonal to the error:

$$\text{SI-SDR} = \frac{|\alpha s|^2}{|\alpha s - \hat{s}|^2} \text{ for } \alpha = \text{argmin}_\alpha |\alpha s - \hat{s}|^2.$$

This approach better reflects the prediction evaluation, making siSDR a more convenient metric to use in the studies.

# Chapter 4

# Proposed Solution

## 4.1 Introduction

This chapter describes the main milestones of what happened during this study. First, the choice of the model (Band-split RNN[24]) was made. Then, according to the modern machine learning paradigm, it was necessary to choose the data on which the above model would be trained (Musdb18[33] and LibriSpeech[28]). In the next section, it will be described in detail how the problem of limited data was addressed, namely, 5 remixing techniques used by author during the experiments. After finalizing the previous three sections, it's time for the final training of the selected model on the selected remixed data. After that, the metrics and the strategies used to evaluate the results are described. The final section contains the shortcomings of the proposal.

## 4.2 Model

When it comes to modelling, a few notions are of main interest:
1. Architecture refers to the model blueprint. The analysis of the architecture can give insights about buildings blocks used to define the model
2. Implementation usually means the code base which defines the model based on its architecture information
3. Multi-GPU training is connected to configurations used during training, which covers all optimizers, LR-schedulers, data loading info etc.

### 4.2.1 Architecture

The architecture of the Band-split RNN is described in detail in the Related Work chapter. To briefly refresh, the model consist of 3 modules: band split, sequence modelling and mask estimation. It results in 32.4 million parameter TF model.

### 4.2.2 Implementation

Python BSRNN[24] implementation need to be selected and situation on the the github is as follows:

1. The first repo[1] with BSRNN pytorch & pytorch-lightning contains checkpoints for all 4 sources with 6.26 dB SDR on vocal source, however the average SDR is only 4.99 dB, which is twice as low as theoretical SDR of BSRNN model.

[1] https://github.com/crlandsc/Music-Demixing-with-Band-Split-RNN

2. The second one[2] contains some pytorch & pytorch-lighting modules somehow connected to BSRNN, however time factor doesn't allow to risk and dig into python code with neither comments, nor instructions.

3. The last one[3] was perfect as a starting point: it contains amazing readme file with **train/eval** instructions and as in previous two cases utilized combination of **t-p-l**.

Since the selected pytorch implementation uses floaf_32 precision the model's checkpoint is expected to take $4 \times 32.4 = 129.6$ Mb of disk storage.

### 4.2.3 Multi-GPU training

From the author's experience, modern GPUs can provide up to 1000 times faster than CPUs, for example, this article[14] mentions experiments showing up to 45 times speedup, and this article[41] - up to 39 times.

Another benefit of using GPUs is the possibility of easy horizontal scaling: just add another video card. All of this motivated the author to modify the existing pytorch code to run on multiple GPUs simultaneously.

## 4.3 Dataset

For all experiments, musdb18-hq was used. To speed up the training process, instead of loading whole audio files, the indices of smaller fragments are precomputed. To select these indices, the source activity detector algorithm (SAD)[21] was used, which is introduced in the original BSRNN paper[24]. This algorithm is a simple yet powerful unsupervised energy-based thresholding method and all of its configurations are copied from the BSRNN paper[24]. The aim of the above SAD[21] is to help select meaningful segments from the full track by ignoring quiet areas.

As in original paper, SAD[21] splits each track into 6-second portions with 0.5 overlap. On average, each song in the musdb18[32] dataset creates 55 of such chunks. For example, the musdb18-hq test set of 50 songs is converted into $n = 2772$ 6-second audio samples.

As for human speech, any train part looked unreasonably large compared to the train part of musdb18[32], which is $\approx 6.5$ hours. Therefore, for the experiments, LibriSPeech[28] dev-clean part[4] was used, which consists of 5.4 hours of audio data, 20 male and 20 female speakers. The audio samples were recorded at 16 kHz sampling rate, so the speech was resampled to 44.1 kHz to match the sampling rate of musdb18[32].

## 4.4 Remixing aka Augmentation

In testing and applying remixing and augmentations, the following approach and sequence of actions were used:

1. Check the init checkpoint (provided by *amanteur*) with $mix\_prob = .25$.

[2]https://github.com/yoongi43/music_source_separation
[3]https://github.com/amanteur/BandSplitRNN-PyTorch
[4]https://www.openslr.org/resources/12/dev-clean.tar.gz

2. Adjust the remixing procedure and increase *mix_prob* to .8

3. Try remixing with LibriSpeech, checking[5] 6 values for the variable *speech_remix_prob* from the [0, 1] interval

4. Combine the best *mix_prob* and *speech_remix_prob* parameters from the experiments and adding 3 simple augmentations.

### 4.4.1 Classic Remixing aka Cacophony Remixing

This type of remixing used by the authors of the original BSRNN[24] article was already present in the pytorch release of the model. However, two small details were changed:

1. Firstly, in the *amanteur* implementation the total number of instruments involved in the mixing was also randomized, in addition to the sources random sampling. The high results of the mentioned study[17] were obtained by mixing all 3 instrument stems. That is why it was changed in the code.

2. Secondly, during the training *amanteur* used *mix_prob* 0.25, and according to the mentioned study[17], the optimal *mix_prob* parameter for vocals is in the [0.6, 1] range, which can also be seen in fig 2.7. Therefore, in this study, *mix_prob* was increased to 0.8

This type of remixing is controlled by the main parameter *mix_prob*, which shows the ratio of remixes in the training data. The absence of remixes corresponds to the value $mix\_prob = 0$. And for training on remixes only, *mix_prob* is set to one.

### 4.4.2 Replacing Vocals with Speech from LibriSpeech

Remixing was performed with the online aka on-the-fly mixing strategy. To do this, a binary random variable with probability *speech_remix_prob* is modeled in the *__getitem__()* method of the pytorch dataset so that if true, the speech sampling with replacement is done until the total length reaches 6 seconds. Then the signal is trimmed to match 6 seconds of the input mixture. Next, it is normalized and mixed with an accompaniment with a random Signal-to-Noise ratio uniformly drawn from $[-5, 15]$ dB range. Find the detailed algorithm in 1.

### 4.4.3 3 Simple Augmentations

Finally, there are 3 well-known augmentations that are often mentioned in related MSS works, so there was no doubt about their usefulness. Because pitch shift and speed change require long processing time, these augmentations cannot be implemented like the previous 2 on-the-fly remixing techniques. However, the musdb python api[6] turned out to be quite flexible and allowed processing not only musdb18[32] tracks. All that is needed is to follow the internal structure as in the original dataset. Therefore, it was decided to make all 3 augmentations static, i.e. to save their results on disk with the same file structure as musdb18[32]. By doing so, the author managed to save a considerable amount of time that would have been required to read the aggregated data. As a result, the nominal size of the dataset increased by 7 times.

[5]These values were chosen because it was expected that too large *speech_remix_prob* would lead to worsening of the results. Therefore, the main focus was on numbers in the range [0, 0.3].

[6]`https://github.com/sigsep/sigsep-mus-db`

**Algorithm 1** Remix with Speech

```
procedure REMIX_WITH_SPEECH(accompaniment)
    if uniform(0, 1) < speech_remix_prob then
        speech_total_lengths ← 0
        speech_samples ← []
        while speech_total_lengths < length_of_6_sec_sample_44100 do
            new_speech ← sample_speech()
            normalize(new_speech)
            speech_samples.append(new_speech)
            speech_total_lengths += len(speech_sample)
        end while
        speech ← concat(speech_samples)
        SNR ← uniform(-5, +15)
        remix ← mix(speech, accompaniment, SNR)
        normalize(remix)
        return (remix, speech)
    end if
end procedure
```

To be more specific, 6 new wav files were created for each wav file: 2 with changed pitch, 2 with changed speed, 2 shifted along the time axis.

## 4.5 Evaluation

To be able to measure and compare results statistical hypothesis testing would be applied. All checkpoints are tested on same 50 songs from musdb18-hq test set which results in the sample size of $n = 2772$.

### 4.5.1 Train, Validation, Test splits

By default, musdb18 and musdb18-hq have a partitioning into train (86 songs), valid (14 songs), and test (50 songs) parts. This is done to use the classical ML approach, which can be summarized as follows: training is performed only on the train set, the valid part is used to select hyperparameters and compare different approaches, as well as to evaluate metrics on unseen data during training. Finally, after all the training experiments, the valid set is added to the training set and the final version of the model is tested on the unseen test set. This strategy is preferable to a train/test split because it reduces the chance of overfitting on the test data.

### 4.5.2 SDR Triplet

To compare model checkpoints, there are 3 variations of SDR metric, which are summarized below:

1. cSDR the chunk-level signal-to-distortion ratio calculated by the standard tool bss eval metrics[46] and it is the default evaluation metric in the Signal Separation Evaluation Campaign (SiSEC)[40].

2. uSDR is the modified utterance-level signal-to-distortion ratio suggested in and used as the default evaluation metric in the Music Demixing (MDX) Challenge 2021[25]. The definition of uSDR is identical to the standard SNR.

3. siSDR is Scale-Invariant Source-to-Distortion Ratio[37], which removes SDR's dependency on the amplitude scaling of the signal. In this thesis, the average siSDR is reported across n test segments.

### 4.5.3 Single mean test

One sided t test[1] would be used to check if there is at least 5 times as much signal energy in the predictions as there is distortion energy. Number 5 was subjectively selected by the author of the thesis. Below are the corresponding hypotheses:

$$H_0 : \frac{||s||^2}{||s - \hat{s}||^2} = 5 \tag{4.1}$$

$$H_1 : \frac{||s||^2}{||s - \hat{s}||^2} > 5, \tag{4.2}$$

where s is the ground truth targer and $\hat{s}$ being the MSS model prediction. Let's switch to SDR notion and consequently to dB scale. Using the fact that $log_{10}$ is monotonically increasing function, one can replace (4.2) with the following:

$$log_{10}(\frac{||s||^2}{||s - \hat{s}||^2}) > log_{10}(5), \tag{4.3}$$

which is the same as

$$10 \cdot log_{10}(\frac{||s||^2}{||s - \hat{s}||^2}) > 10 \cdot log_{10}(5), \tag{4.4}$$

where left-hand side of the equation is by definition the SDR metric:

$$SDR > 6.9897, \tag{4.5}$$

Since sample size $n$ is sufficiently large, t-distribution could be replace by standard normal distribution. Additionally standard error estimate $SE = \frac{s}{\sqrt{n}}$ ($s$ is the SDR sample std) is getting tiny, because of the large $\sqrt{n}$ in the denominator. Finally, z test statistic is calculated as $z = \frac{uSDR - 6.9897}{SE}$.

### 4.5.4 Test for Difference in Means

In the most basic case, one is interested in testing $SDR_{init} < SDR_{trained}$ of parameters in the population. For this purpose, the researches consider $uSDR$, the mean SDR on musdb test set. To investigate this, a small change is required and the hypotheses are as follows:

$$H_0 : SDR_{trained} - SDR_{init} = 0, \text{ and } \ H_1 : SDR_{trained} - SDR_{init} > 0$$

To perform difference in means, one should recall that the distribution of differences in sample means has expected value of

$$\mathbb{E}(SDR_{trained} - SDR_{init}) = uSDR_{trained} - uSDR_{init},$$

for sufficient n $\approx >$ 50 sample sizes, where $SDR_{init}$ correspond population SDR of *amanteur* checkpoint and $SDR_{trained}$ – to mine checkpoint. $uSDR_{checkpoint}$ is the average SDR score on 50 musdb test songs.

The standard error SE can be calculated using the formula

$$SE = \sqrt{\frac{s^2_{taras}}{n} + \frac{s^2_{amanteur}}{n}},$$

where $s_{checkpoint}$ is the std SDR on the test dataset.

Finally, we have the standardised test statistic,

$$t = \frac{(uSDR_{trained} - uSDR_{init}) - 0}{SE},$$

and the p-value is calculated using the formula:

$$\text{p-value} = 1 - cdf(t) \approx 1 - cdf(norm)$$

for the right-sided test.

If the p-value is less than 0.05, then the experiment shows enough evidence that my training has significantly improved the init model.

## 4.6 Challenges and limitations

The 2 main factors limiting this study, namely time and computational resources. Due to time constraints, it was not possible to test more granular values for *speech_remix_prob*, as in this study, where the step in increasing *mix_prob* was .1. Also, due to financial constraints, it was not possible to keep the training for several dozen more epochs. No doubt, the results would have been even better, because after examining the training log, the validation uSDR increased steadily, indicating that the model still had room for improvement.

It could also be noticed that this study considered only a fully supervised version of the Band-split RNN[24] model, but the authors of the original article developed and applied a semi-supervised learning approach, which generated pseudo clean targets and residuals (*accompaniment* in case of vocal extraction) using the model trained on musdb18[32].

## 4.7 Summary

The proposed solution can be summarized as follows: this study proposes a novel remixing method by mixing human speech[28] with musical accompaniment to address the problem of data-constrained environment. All experiments are performed with the BSRNN[24] model and the last experiment combines all 5 augmentation techniques to improve the public checkpoint of the BSRNN[24] model.

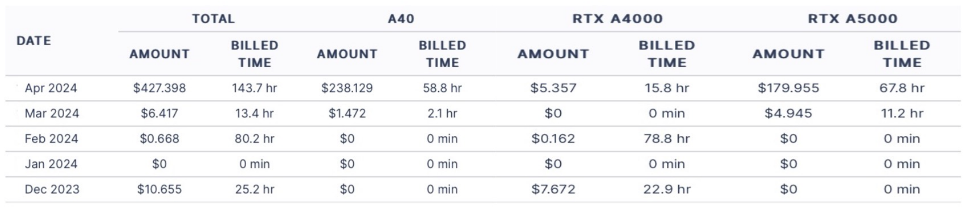
# Chapter 5

# Experiments and Results

## 5.1 Experimantal Setup

From the very beginning, 86 training songs with musdb18-hq[33], after SAD[21] pre-processing, I get 5200 different training samples. 1 epoch on this data on one RTX 3090 takes about 10 minutes. After 1 cycle of augmentation, the data became 29k and one epoch took 1 hour. Finally, when the data became 35k, one epoch took 1 hour and 20 minutes. For comparison, the authors mention 100 epochs of training on 8 GPUs, where one epoch contained about 20k training samples. That's why it was decided to switch to the paid GPU rental.

In the infographic below (5.1), one can see detailed analytics of GPU consumption during training:



| DATE | TOTAL | | A40 | | RTX A4000 | | RTX A5000 | |
|---|---|---|---|---|---|---|---|---|
| | AMOUNT | BILLED TIME | AMOUNT | BILLED TIME | AMOUNT | BILLED TIME | AMOUNT | BILLED TIME |
| Apr 2024 | $427.398 | 143.7 hr | $238.129 | 58.8 hr | $5.357 | 15.8 hr | $179.955 | 67.8 hr |
| Mar 2024 | $6.417 | 13.4 hr | $1.472 | 2.1 hr | $0 | 0 min | $4.945 | 11.2 hr |
| Feb 2024 | $0.668 | 80.2 hr | $0 | 0 min | $0.162 | 78.8 hr | $0 | 0 min |
| Jan 2024 | $0 | 0 min | $0 | 0 min | $0 | 0 min | $0 | 0 min |
| Dec 2023 | $10.655 | 25.2 hr | $0 | 0 min | $7.672 | 22.9 hr | $0 | 0 min |

FIGURE 5.1: GPU analytics.

All experiments took a total of 288 hours ≈ 12 days of continuous gpu computing. 266 hours (11 days) of them were hosted on the RunPod GPU Cloud. On average, $1.7 graphics cards with the computing power equivalent to four Nvidia GeForce RTX 3090s were used every hour ($4 \times 24$Gb $= 96$Gb total Vram).

The remaining 22 hours were spent on the part of the experiments related to the overfit of a 12-second sample of a song from the Musdb18-hq[33] dataset. The mentioned part of the experiments was implemented on a single RTX 3090, which was provided by the supervisor of this thesis.

During studying BSRNN[24] the following list of experiments were conducted, which are listed in chronological order.

## 5.2 Evaluate the starting condition

Before any training, one need to evaluate the starting condition. *amanteur* provided vocal checkpoint for their **t-p-l** BSRNN model and indeed it has the following results:

| SDR | Mean | Std | t | se | p-value |
|---|---|---|---|---|---|
| cSDR | 6.67 dB | 2.22 dB | -7.65 | 0.042 | 0.999 |
| uSDR | 6.88 dB | 2.49 dB | -2.31 | 0.047 | 0.989 |
| siSDR | 4.20 dB | 11.47 dB | -12.8 | 0.218 | 1.0 |

TABLE 5.1: Metrics for starting checkpoint

The mean and std of SDR are the same as in original repo. Also, there is not enough evidence to claim that *amanteur* checkpoint performs better than 7 dB SDR, which is obvious as all SDRs are under 7 dB, so already the p-value is at least 0.5.

## 5.3 12 sec overfit experiment

12 sec song sample produces three 6-sec training examples. During the overfiting, the BSRNN is trained, validated and tested on the same 3 samples, thus there is no sense of performing hypothesis testing.

| SDR | Mean |
|---|---|
| cSDR | 10.94 dB |
| uSDR | 10.06 dB |
| siSDR | 9.61 dB |

TABLE 5.2: Metrics for overfit experiment

Unfortunatelly, the best overfit results achieved by the author are not far from the original BSRNN paper. For instance, overfitted cSDR 10.94 dB is only 9% better than 10.01 dB, which is official BSRNN cSDR obtained after evaluationg the model on musdbhq test set.

## 5.4 Remixing with Speech

This is the section with the most experiments. To test how remixing with speech affects the performance of the MSS model, the following **speech_remix_prob** values were considered **[0, 0.1, 0.2, 0.3, 0.5, 1]**. This choice behind such config is intentional and reflects the beliefs that if there is too much speech in the training data, the init model will be fine-tuned to extract speech only and thus the performance will drop in singing voice separation.

The intuition behind such a non-trivial pattern is that, unlike pre-training the model on LibriSpeech data[5], my experiments are more like fine-tuning for speech and accompaniment separation. Obviously, the more speech mixes are fed during training, the more the initial checkpoint will be reoriented to speech. The reality proved intuition: with increasing speech ratio the following three statements hold true:

1. training gets slower and for speech_remix values larger than 0.5 val loss and uSDR don't even improve
2. all three SDR metrics are getting worse
3. variation of metrics becomes larger, which indicates more instability.

Let's consider the best results, obtained by training with no LibriSpeech and check whether there is enough evidence to claim those are better than theoretical 7 dB SDR.

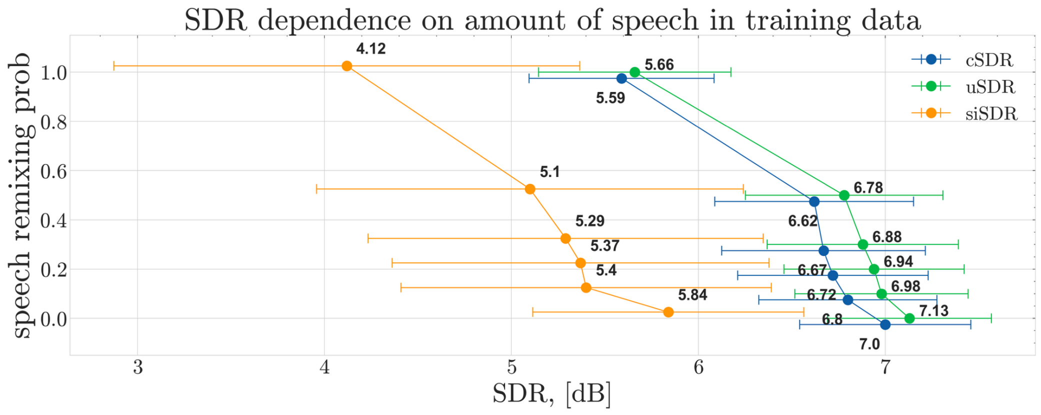


FIGURE 5.2: Training with remixes. More speech (up) – less SDR on music (left).

As cSDR and siSDR do not exceed 7 dB even without a hypothesis test one can immediately claim those two SDR metrics are indistinguishable from 7 dB mark. However, the uSDR of 7.13 dB seems like possible significant result in the test. Whether the best trained checkpoint actually improved or it's just happened by chance, one sided z test is used. Test statistics below.

| **SDR** | **Mean** | **Std** | t | se | p-value |
|---|---|---|---|---|---|
| cSDR | 7.00 dB | 2.75 dB | 0.197 | 0.052 | 0.422 |
| uSDR | 7.13 dB | 2.63 dB | 2.808 | 0.050 | 0.0025 |
| siSDR | 5.84 dB | 4.35 dB | -12.8 | 0.218 | 1.0 |

TABLE 5.3: Metrics for the best model, no speech during training, 400 epochs, 5200 training samples

As expected, cSDR and siSDR didn't allow to reject the null hypothesis, which is supported by high p-values. But indeed 0.14 dB increase in the uSDR compared to 6.9897 dB given by H_0 is significant, thus the (4.1) is rejected in favour of (4.5).

Unlike previous experiments where single checkpoint was considered, now there is a possibility to compare *amanteur* results with mine. To do so, the t test for difference in two means is applied. Test results could be found below in the table 5.4

| **SDR** | **Diff in means** | t | se | p-value |
|---|---|---|---|---|
| cSDR | .33 dB | 4.916 | 0.067 | 0. |
| uSDR | .25 dB | 3.634 | 0.068 | 0. |
| siSDR | 1.64 dB | 7.039 | 0.232 | 0. |

TABLE 5.4: Difference in SDR means between init and trained checkpoints

## 5.5 Remixing + Augmentation

The final training session was set up as follows: $mix\_prob = .8$, no speech was added during training, and 6 different augmentations (2 pitch shifts, 2 speed changes, 2 time shifts). The data volume increase almost 7 times and became 29k samples. As expected, this greatly improved the results, to be more precise, see table 5.5.

| SDR | Mean | Std |
|---|---|---|
| cSDR | 7.28 dB | 2.8 dB |
| uSDR | 7.46 dB | 2.69 dB |
| siSDR | 6.03 dB | 5.18 dB |

TABLE 5.5: Augmented train data, 29k samples, 100 epochs

During the training, $LR$ was set to $8e-4$, with $\gamma = 0.99$ in $lr\_scheduler.LambdaLR$. There were 29k samples (bsrnn[24] had 20k). Initially, 100 epochs were trained.

After analyzing the tensorboard data of the training and validation LR and uSDR, it was clear that the augmentations had greatly diversified the data and the model still had a lot to learn. Therefore, the LR was increased to 1.2e-3, and the validation data was also augmented and added to the training data. And so 30 epochs with LR = 1.2e-3, 30 epochs LR = 1e-3, and 40 epochs LR = 8e-4 were trained.

It is also important to mention that due to static augmentations, there is 7 times more data, namely 35311 samples instead of 5239 initial samples. This resulted in the best trained checkpoint in this study.

| SDR | Mean | % increase | Std |
|---|---|---|---|
| cSDR | 7.94 dB | +19% | 2.96 dB |
| uSDR | 8.07 dB | +19% | 2.68 dB |
| siSDR | 7.19 dB | +71% | 3.39 dB |

TABLE 5.6: Augmented train + val data, 35k samples, 100 epochs

There is no point in running any more statistical tests, as it is clear that all metrics have increased significantly compared to the init checkpoint.

Also, the most satisfying thing was watching the siSDR metric. At the init checkpoint, it was a gigantic 11.47 dB, after training on 0.8 remixes, it decreased to 4.35 dB, and finally became 3.39 dB in the final result. Thus, the siSDR metric increased by 71%, and the standard deviation decreased by 70%.

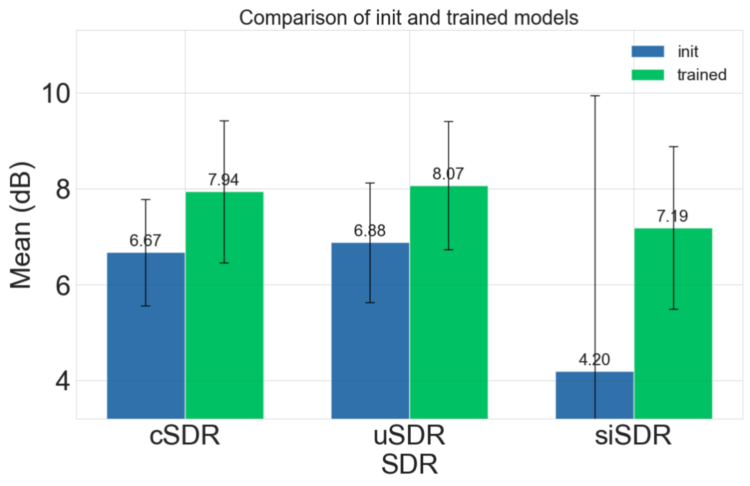


FIGURE 5.3: Comparison between init and trained model checkpoints.

# Chapter 6

# Conclusions

## 6.1 Summary of Results

The main objective of this study was to analyze the impact of remixing as a solution to the problem of limited data. For this purpose, a new remixing method was developed and tested, which consists of mixing human speech together with musical accompaniment. Due to its nature, this method is only suitable for **vocal** vs. **instrumental** separation, so the work focuses only on vocal extraction. In the process, the new remixing scheme was compared with the standard approach, and the optimal hyperparameters were derived to maximize the usefulness of the remixing. In addition, three standard data augmentation techniques were applied and proved to be excellent, which were combined with remixing, leading to an even more impressive improvement in the model results.

The study shows that remixing and data augmentation is an integral part of modern music source separation and ML in general. Not only did they improve the three main metrics, but they also stabilized the training and improved the stability of the Band-split RNN[24] model. These techniques have successfully proven to be not only a method for increasing the amount of training data but also a way to diversify the training data. This is very important for powerful modern models, which, due to their size, can easily overfit and exhaust the training potential of the data.

Therefore, all goals that were set for this thesis were achieved at the same time, providing lots of opportunities for future work.

## 6.2 The Influence of Remixing on Vocal Extraction in Low Data Regimes

This study indicates that remixing is crucial to the performance of the top MSS model, Band-split RNN[24]. The research coincides with the findings of the most recent peer-reviewed paper on remixing[17] and shows that increased remixing can increase the SDR of music source separation models by 1-3 dB. From the experiments, it is clear that remixing is a great way to address the problem of limited data, which is very present in the MSS industry.

## 6.3 Future Research Directions

This topic is very extensive and requires testing numerous models on large datasets to obtain reliable results. Due to the mentioned limitations, the work was very focused, which made it possible to clearly and thoroughly investigate the aspects of remixing on a specific model. Because of this, further research steps include:

1. Increasing the training data using other data sources, in particular, the most recent MoisesDB[30] dataset

2. Investigating the potential benefits that remixing and augmentation could bring to the other models mentioned in the paper

3. Testing augmentation and remixing on the separation of other sources.

4. Utilizing large volumes of unlabeled data to fine-tune existing checkpoints.

# Bibliography

[1] Mowafaq Al-kassab. “The Use of One Sample t-Test in the Real Data”. In: *JOURNAL OF ADVANCES IN MATHEMATICS* 21 (Sept. 2022), pp. 134–138. DOI: 10.24297/jam.v21i.9279.

[2] Mesaros Annamaria and Tuomas Virtanen. “Automatic Recognition of Lyrics in Singing”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2010 (Jan. 2010). DOI: 10.1155/2010/546047.

[3] Rachel Bittner et al. “MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research”. In: Oct. 2014.

[4] Estefanía Cano, Gerald Schuller, and Christian Dittmar. “Pitch-informed solo and accompaniment separation towards its use in music education applications Informed Acoustic Source Separation”. In: *Journal on Advances in Signal Processing* 2014 (Nov. 2014). DOI: 10.1186/1687-6180-2014-23.

[5] Ke Chen et al. *Pac-HuBERT: Self-Supervised Music Source Separation via Primitive Auditory Clustering and Hidden-Unit BERT*. 2023. arXiv: 2304.02160 [cs.SD].

[6] Woosung Choi et al. *Investigating U-nets with Various Intermediate Blocks for Spectrogram-based Singing Voice Separation*. Oct. 2020.

[7] Michaël Defferrard et al. *FMA: A Dataset For Music Analysis*. 2017. arXiv: 1612.01840 [cs.SD].

[8] Christian Dittmar et al. *Music information retrieval meets music education*. 2012. URL: https://publica.fraunhofer.de/handle/publica/231425.

[9] Alexandre Défossez. *Hybrid Spectrogram and Waveform Source Separation*. 2022. arXiv: 2111.03600 [eess.AS].

[10] Alexandre Défossez et al. *Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed*. 2019. arXiv: 1909.01174 [cs.SD].

[11] Alexandre Défossez et al. *Music Source Separation in the Waveform Domain*. 2021. arXiv: 1911.13254 [cs.SD].

[12] O. Gillet and G. Richard. “Extraction and remixing of drum tracks from polyphonic music signals”. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* 2005, pp. 315–318. DOI: 10.1109/ASPAA.2005.1540232.

[13] Jordan R. Green et al. “Automatic Speech Recognition of Disordered Speech: Personalized Models Outperforming Human Listeners on Short Phrases”. In: *Proc. Interspeech 2021*. 2021, pp. 4778–4782. DOI: 10.21437/Interspeech.2021-1384.

[14] Huichao Hong, Lixin Zheng, and Shuwan Pan. “Computation of Gray Level Co-Occurrence Matrix Based on CUDA and Optimization for Medical Computer Vision Application”. In: *IEEE Access* PP (Nov. 2018), pp. 1–1. DOI: 10.1109/ACCESS.2018.2877697.

[15] IFPI. *Engaging With Music 22*. IFPI Submission to the EU Counterfeit and Piracy Watchlist Consultation 3 & nn.5-6 (Feb. 14, 2022). 2022. URL: https://www.ifpi.org/wp-content/uploads/2022/11/Engaging-with-Music-2022_full-report-1.pdf.

[16] Katsutoshi Itoyama et al. "Query-by-Example Music Information Retrieval by Score-Informed Source Separation and Remixing Technologies". In: *Hindawi Publishing Corporation EURASIP Journal on Advances in Signal Processing* 14 (Jan. 2010). DOI: 10.1155/2010/172961.

[17] Chang-Bin Jeon et al. *Why does music source separation benefit from cacophony?* 2024. arXiv: 2402.18407 [eess.AS].

[18] J. Kahn et al. "Libri-Light: A Benchmark for ASR with Limited or No Supervision". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2020. DOI: 10.1109/icassp40776.2020.9052942. URL: http://dx.doi.org/10.1109/ICASSP40776.2020.9052942.

[19] Minseok Kim et al. *KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing*. 2021. arXiv: 2111.12203 [eess.AS].

[20] Tom Ko et al. "Audio augmentation for speech recognition." In: *Interspeech*. Vol. 2015. 2015, p. 3586.

[21] Rajath Kumar, Yi Luo, and Nima Mesgarani. "Music Source Activity Detection and Separation Using Deep Attractor Network". In: Sept. 2018, pp. 347–351. DOI: 10.21437/Interspeech.2018-2326.

[22] Liwei Lin et al. *A Unified Model for Zero-shot Music Source Separation, Transcription and Synthesis*. 2021. arXiv: 2108.03456 [cs.SD].

[23] Antoine Liutkus et al. "The 2016 Signal Separation Evaluation Campaign". In: *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*. Ed. by Petr Tichavský et al. Cham: Springer International Publishing, 2017, pp. 323–332.

[24] Yi Luo and Jianwei Yu. *Music Source Separation with Band-split RNN*. 2022. arXiv: 2209.15174 [eess.AS].

[25] Yuki Mitsufuji et al. "Music Demixing Challenge 2021". In: *Frontiers in Signal Processing* 1 (Jan. 2022). ISSN: 2673-8198. DOI: 10.3389/frsip.2021.808395. URL: http://dx.doi.org/10.3389/frsip.2021.808395.

[26] Nobutaka Ono et al. "Harmonic and Percussive Sound Separation and Its Application to MIR-Related Tasks". In: vol. 274. Oct. 2010, pp. 213–236. ISBN: 978-3-642-11673-5. DOI: 10.1007/978-3-642-11674-2_10.

[27] Nobutaka Ono et al. "The 2015 Signal Separation Evaluation Campaign". In: vol. 9237. Aug. 2015. ISBN: 978-3-319-22481-7. DOI: 10.1007/978-3-319-22482-4_45.

[28] Vassil Panayotov et al. "Librispeech: An ASR corpus based on public domain audio books". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[29] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[30] Igor Pereira et al. *Moisesdb: A dataset for source separation beyond 4-stems*. 2023. arXiv: 2307.15913 [cs.SD].

[31] Jordi Pons et al. "Remixing music using source separation algorithms to improve the musical experience of cochlear implant users." In: *The Journal of the Acoustical Society of America* 140 6 (2016), p. 4338. URL: https://api.semanticscholar.org/CorpusID:206477210.

[32] Zafar Rafii et al. *MUSDB18 - a corpus for music separation*. MUSDB18: a corpus for music source separation. Dec. 2017. DOI: 10.5281/zenodo.1117371. URL: https://inria.hal.science/hal-02190845.

[33] Zafar Rafii et al. *MUSDB18-HQ - an uncompressed version of MUSDB18*. Aug. 2019. DOI: 10.5281/zenodo.3338373. URL: https://doi.org/10.5281/zenodo.3338373.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[35] Aldona Rosner, Björn Schuller, and Bozena Kostek. "Classification of Music Genres Based on Music Separation into Harmonic and Drum Components". In: *Archives of Acoustics* 39 (Mar. 2015). DOI: 10.2478/aoa-2014-0068.

[36] Simon Rouard, Francisco Massa, and Alexandre Défossez. *Hybrid Transformers for Music Source Separation*. 2022. arXiv: 2211.08553 [eess.AS].

[37] Jonathan Le Roux et al. *SDR - half-baked or well done?* 2018. arXiv: 1811.02508 [cs.SD].

[38] Justin Salamon and Juan Pablo Bello. "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification". In: *IEEE Signal Processing Letters* 24.3 (Mar. 2017), 279–283. ISSN: 1558-2361. DOI: 10.1109/lsp.2017.2657381. URL: http://dx.doi.org/10.1109/LSP.2017.2657381.

[39] C.E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: 10.1109/jrproc.1949.232969. URL: https://doi.org/10.1109/jrproc.1949.232969.

[40] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. *The 2018 Signal Separation Evaluation Campaign*. 2018. arXiv: 1804.06267 [eess.AS].

[41] Kalaiselvi Thiruvenkadam and Sriramakrishnan Padmanaban. "GPU based Parallel Computing Approach for Accelerating Image Filters". In: June 2015. DOI: 10.13140/RG.2.2.29241.36964.

[42] Weinan Tong et al. *SCNet: Sparse Compression Network for Music Source Separation*. 2024. arXiv: 2401.13276 [eess.AS].

[43] Efthymios Tzinis et al. "Continual Self-Training With Bootstrapped Remixing For Speech Enhancement". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022. DOI: 10.1109/icassp43922.2022.9747463. URL: http://dx.doi.org/10.1109/ICASSP43922.2022.9747463.

[44] Efthymios Tzinis et al. "RemixIT: Continual Self-Training of Speech Enhancement Models via Bootstrapped Remixing". In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (Oct. 2022), 1329–1341. ISSN: 1941-0484. DOI: 10.1109/jstsp.2022.3200911. URL: http://dx.doi.org/10.1109/JSTSP.2022.3200911.

[45] Stefan Uhlich et al. "Improving music source separation based on deep neural networks through data augmentation and network blending". In: Mar. 2017. DOI: 10.1109/ICASSP.2017.7952158.

[46] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. "Performance measurement in blind audio source separation". In: *IEEE Transactions on Audio, Speech and Language Processing* 14.4 (2006). inria-00544230, pp. 1462–1469.

[47] Zhepei Wang et al. *Semi-Supervised Singing Voice Separation with Noisy Self-Training*. 2021. arXiv: 2102.07961 [eess.AS].

[48] Haojie Wei et al. "RMVPE: A Robust Model for Vocal Pitch Estimation in Polyphonic Music". In: *INTERSPEECH 2023*. interspeech$_2$023. ISCA, Aug. 2023. DOI: 10.21437/interspeech.2023-528. URL: http://dx.doi.org/10.21437/Interspeech.2023-528.

[49] John F. Woodruff, Bryan Pardo, and Roger B. Dannenberg. “Remixing Stereo Music with Score-Informed Source Separation”. In: *International Society for Music Information Retrieval Conference*. 2006. URL: https://api.semanticscholar.org/CorpusID:263699485.