Master Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Quality Enhancement of Overdub Singing Voice Recordings

Benedikt Wimmer

Co-Supervisor: Jordi Janer

Co-Supervisor: Merlijn Blaauw

August 2021

Master Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Quality Enhancement of Overdub Singing Voice Recordings

Benedikt Wimmer

Co-Supervisor: Jordi Janer

Co-Supervisor: Merlijn Blaauw

August 2021

Contents

1 Introduction **1**

1.1 Use Case: Virtual Choir Rehearsal Software 1

1.2 Research Goals 2

1.3 Objectives and Structure of the Report 3

2 Related Work **4**

2.1 Denoising and Source Separation 4

2.2 Acoustic Echo Cancellation and Leakage Removal 6

2.3 Network Architectures 8

2.3.1 Wave-U-Net 9

2.3.2 FullSubNet 10

3 Datasets and Preprocessing **13**

3.1 Singing/Speech 14

3.2 Choir Singing 16

3.3 Impulse Responses 17

3.4 Noise profiles 19

4 Methods for Denoising **23**

4.1 Data Augmentation 24

4.2 Training and Experimental setup 25

4.3 Wave-U-Net: Configuration 26

4.4 FullSubNet: Configuration 26

4.5 Evaluation Methods 27

5 Methods for Leakage Removal **30**

5.1 Data Augmentation 31

5.2 Architecture 36

5.3 Training and Experimental Setup 39

5.4 Model Configuration 40

5.5 Baseline 40

5.6 Evaluation 40

6 Results & Discussion **42**

6.1 Denoising 43

6.2 Leakage Removal 44

7 Conclusions **47**

7.1 Conclusions 47

7.2 Contributions 47

7.3 Future Work 48

7.4 Reproducibility 49

List of Figures **50**

List of Tables **52**

Bibliography **53**

Abstract

Singing enhancement aims to improve the perceived quality of a singing voice recording in various aspects. Focusing on the aspect of removing degradation such as background noise or room reverberation, singing enhancement is related to the topic of speech enhancement. In this work, two neural network architectures for speech denoising – namely FullSubNet and Wave-U-Net – were trained and evaluated specifically on denoising of user singing voice recordings. While both models show similar performance as for speech denoising, FullSubNet outperforms Wave-U-Net on this task. Furthermore, the removal of sound leakage (i.e. reference signal/accompaniment for overdubbing that becomes audible in the background of a recording) was performed with a novel modification of FullSubNet. The proposed architecture performs *leakage removal* by taking the signal leading to aforementioned leakage as an additional input. For the case of choir music and for leakage removal, this modified FullSubNet architecture was compared to the original FullSubNet architecture. Evaluation results show its overall efficacy on leakage removal as well as significant benefits introduced by usage of the additional input.

Keywords: Singing Enhancement · Speech Enhancement · Denoising · AEC

Chapter 1

Introduction

Over the last decades, Web 2.0 technologies, increasing availability of internet and mobile devices have revolutionized human interaction. Social networks and (real-time) communication technologies are omnipresent, but also video games reach a broad audience. While music technology has been constantly evolving along with software and hardware capabilities, remote collaboration on music has long been limited to music production tasks. During the Covid-19 pandemic, this audience has radically increased as larger ensembles could not meet for rehearsals physically. Especially in the choir community, rehearsals would therefore commonly be conducted via videotelephony. While social interaction and basic singing exercises might be possible with such a setup, rhythmically aligned singing and practicing in the context of polyphony is often impeded by latency and connectivity issues.

1.1 Use Case: Virtual Choir Rehearsal Software

Developed specifically as a platform for rehearsal of choir repertoire, the Cantāmus app[1] represents an alternative rehearsal scenario for choirs. Here, an individual singer's repertoire learning process is enhanced by providing synthesised voices as auditory guidance/reference. As a collaborative effort, individually rehearsed and recorded parts can be mixed with other choir members' recordings. The outcome

[1] Voctro Labs: Cantāmus. `https://cantamus.app/`

would then be a full choir recording of a given piece. Cantāmus was developed as part of the TROMPA (Towards Richer Online Music Public-domain Archives) project[2], which is an effort to improve accessibility of public-domain digital music resources through music technology.

Customarily, Cantāmus users practice and record their parts at home. Suboptimal room environment and recording equipment can lead to audio quality degradations such as background noise, room reverberation, microphone coloration and distortion. A special kind of background noise in overdub recordings is what, in this work, we call *leakage*. This is also commonly referred to as *spill*, *bleed* or *cross-talk* and describes the case when, in an overdub recording scenario, the reference audio listened to by a user becomes audible in the recording. In our use case, reference audio would mean a mix of choir voices and/or accompanying instruments. Regardless of the degradation type, mixing of multiple individual degraded recordings can worsen the perceived disturbance. This can be explained with occurrence frequency of degradations or with the perceptual concept of critical bandwidth [1]. Given this situation, quality enhancement on individual recordings is needed to ensure a pleasant user experience.

1.2 Research Goals

This work focuses specifically on removal of background noise and leakage in the context of the given use case. Please note that, hereafter, the terms *denoising* and *leakage removal* will be used for those respective tasks.

As a first goal, we want to explore the adaptation of two speech denoising architectures for denoising of singing voice recordings. As explained in Section 2.1, denoising of speech has been subject to extensive research and the best results were achieved with deep learning approaches.

Inspired by the work in [2], our second goal is to modify a speech denoising architecture to perform leakage removal. In leakage removal, there exists knowledge

[2]TROMPA: Towards Richer Online Music Public-domain Archives. `https://trompamusic.eu/`

about the signal causing the leakage (see Section 2.2). This differentiates leakage removal from denoising and, accordingly, the modified model must be able to handle a second input.

1.3 Objectives and Structure of the Report

The report is structured so that a general context is given before the actual work is presented and evaluated. Therefore, Chapter 1 gives an idea on the motivation, use case scenario and targeted subproblems. Chapter 2 contains an overview of related research with an emphasis on deep-learning-based approaches to speech enhancement. In Chapter 3, all audio data resources utilized are portrayed and information on their accessibility is given. Chapters 4 and 5 describe the methodology for denoising and leakage removal tasks. They both specify how computational data augmentation was performed to approximate their respective problem scenarios. Specifically, Chapter 4 describes how two speech denoising architectures were trained and evaluated on denoising of singing voice recordings. Chapter 5 proposes a modified speech denoising architecture with an additional input for leakage removal and reports on its training and evaluation process. Results on both tasks are presented and discussed in Chapter 6. Finally, conclusions and future work are pointed out in Chapter 7.

Chapter 2

Related Work

This chapter gives an overview of research related to our work on singing (quality) enhancement. This includes mainly speech enhancement topics, as speech enhancement is naturally related with singing enhancement. In both fields, the general goal is to optimize the subjective and objective quality of a recording of human voice. For our work on singing voice, this means that we can draw extensive knowledge from a recent, well-founded and vivid field of scientific research.

The speech enhancement subproblems of denoising and acoustic echo cancellation (AEC) are described and similarities are elaborated. Recently published neural network architectures for denoising are presented.

2.1 Denoising and Source Separation

Speech enhancement subproblems are the removal of degradations such as ambient noise, room reverberation, microphone coloration, distortion or sound leakage, to name a few. For the removal of background noise – also commonly referred to as *denoising* or *noise suppression* – the field of audio source separation comes into play. Source separation is the task of deriving individual sound sources from a mixture of sounds. The problem here can be mathematically described as follows:

$$y(t) = \sum_{i=1}^{N} x_i(t) \tag{2.1}$$

where $y(t)$ is a musical mixture, $x_i(t)$ is the ith of N audio sources and t denotes time. The musical mixture is assumed to be a summation of multiple sources and the goal is to derive, given y, the individual sources x_i. In denoising, the given Equation 2.1 can be concretized to:

$$y(t) = x(t) + n(t) \tag{2.2}$$

where $y(t)$, $x(t)$ and $n(t)$ denote mixture, clean speech and noise, respectively. In denoising, the goal is to find a function/model $f(y)$ to derive a best-possible estimate $\hat{x}$ of x from knowledge about y.

$$f(y) = \hat{x} \tag{2.3}$$

Mathematically, source separation is an abstraction of denoising and builds the foundation for it. Both these problems are mathematically underdetermined and therefore cannot be solved in a conventional way.

Since the year of 2013 [3], neural networks have taken over the field of speech enhancement, especially denoising. Normally, the proposed approaches would either work on 2D Short-time-Fourier-Transform (STFT) spectral representations of audio like in [4] or on 1D time-domain representations like in [5]. The same holds true for source separation systems. A majority of 2D architectures do only take into account the magnitude of a signal, such not being *phase-aware* and literally reducing the complexity of the time-frequency audio representations at hand. This property is – despite their success – commonly regarded as a disadvantage of such networks [6]. On the other hand, there has been research comparing both representation approaches that led to similar results on source separation [7]. In recent publications, discrete cosine transform has been explored. With cosine transform,

it is possible to achieve an STFT-like audio representation which is real-valued and implicitly includes phase information [8].

Customarily, networks for speech enhancement use supervised learning: The input would be speech audio augmented with artifacts such as general background ambience (for denoising) [3] or convoluted room reverberation (for dereverberation [9]). The learning target would be the same speech without these artifacts. Resources like the LibriSpeech corpus [10] provide clean speech and can be augmented with audio from e.g. Diverse Environments Multichannel Acoustic Noise Database (DEMAND) [11] or Aachen Impulse Response (AIR) database [12] to retrieve an input-target pair for training. Since access to clean speech data is limited, recent publications suggest including noisy data as an augmentation start point. Input-target pairs can be derived by using noisy data as target and augmenting it further (e.g. with more noise) to retrieve a training input [13]. Another approach lies in denoising the given noisy data with a pre-existing speech enhancement model to retrieve clean data as a target [4]; using the noisy data as input.

Next to such synthetic augmentation approaches, there exist datasets such as DAPS or DR-VCTK [14, 15]. They provide samples of clean speech next to samples of the same speech re-recorded on cellphones within various real-world environments. While being harder to scale, such data is more realistic and can be valuable for combinatorial approaches of not only denoising, but also dereverberation and quality enhancement at the same time.

2.2 Acoustic Echo Cancellation and Leakage Removal

A problem related to denoising is echo cancellation. Mainly problematic in teleconferencing, this describes the situation when a *far-end* speaker's voice is played back at another *near-end* location and fed back into the microphone of the same near-end location. This can then again become audible as echo on the far-end speaker's side. In such a case, general annoyance and intellegibility can become issues. Acoustic

echo cancellation (AEC) describes efforts to solve this problem and has been subject to research since the beginning of teleconferencing technologies [16]. Currently – conditioned by the public need for remote solutions and the rise of artificial intelligence in speech processing – the field of AEC is undergoing a renaissance [17, 18, 19]. In this, conventional approaches to AEC like adaptive filtering are being replaced by deep neural networks or improved with the use of artificial intelligence [17].

Preventing far-end signal from being fed back as echo is customarily tackled at the near-end side; the microphone signal $y(t)$ can be described as:

$$y(t) = s(t) + d(t) = s(t) + h(x(t)) \tag{2.4}$$

where $s(t)$ is the near-end speech; $d(t)$ is the acoustic echo, which is assumed to be the output of a room/degradation filter h applied to the far-end speech $x(t)$. This might appear as being identical to the problem of denoising (see Equation 2.2), but the complexity here is that the acoustic echo $d(t)$ has characteristics of human voice. Moreover, there is knowledge about the far-end speech signal $x(t)$. Since there is no knowledge about the degradations introduced through h, this problem is still underdefined. AEC is searching for a function or model $f(x, y)$, which, given the near-end microphone signal y and far-end signal x, produces a best-possible estimate $\hat{s}$ for s.

$$f(x, y) = \hat{s} \tag{2.5}$$

The additional knowledge about x facilitates this source separation task [18] as compared to source separation from just a single mixture (like in Equation 2.1), which is also called *blind* source separation. Despite described differences, Westhausen et al. have exemplarily shown the efficacy of adapting a denoising system to AEC [2]. In their work, the additional input x was incorporated with an approach based on concatenation of features derived from x and y.

In Section 1.1, we have addressed the problem of sound leakage during overdub

recording in our use case. Users are listening to reference audio while singing and this reference audio can become audible in the recording. This problem is very similar to AEC, differing only in the targeted use case. It can also be described with Equation 2.3: $y(t)$ would be the user recording (mixture); $s(t)$ the user's singing voice; $d(t)$ would be the leaked signal, which is assumed to be the output of a room/degradation filter h applied to the reference audio $x(t)$. In our case, the user is listening to synthesized voices and/or orchestra instruments as reference. As is the case for AEC, this implies voice-alike leakage, but, given the musical nature of choir music, the reference audio $x(t)$ is highly correlated with $s(t)$ in terms of rhythm, dynamics and harmonic content; technically spoken: $x(t)$ is highly correlated with $s(t)$ in the time- and frequency domain. Leakage removal could also be understood as a problem of vocal extraction, but in our case we need to deal with voice signals in both singing (foreground) and background music, while, in vocal extraction, voice is only expected in the foreground signal.

2.3 Network Architectures

In this section, two neural network architectures with shown efficacy in speech enhancement, specifically speech denoising, are presented. These are FullSubNet [20] and Wave-U-Net [21] and were used in our experiments (see Sections 4 and 5). While, over the years, various works on the Wave-U-Net architecture have produced competitive results in source separation, denoising and AEC [21, 22, 23, 24, 25], FullSubNet was recently published, reporting state-of-the-art results on speech denoising [20], but not further explored at the time of writing. FullSubNet and Wave-U-Net are representant members of two families of neural network architectures. Wave-U-Net is a convolutional U-Net architecture with encoder and decoder blocks while FullSubNet represents such approaches that utilize recurrent neural networks.

Most recent publications in speech denoising can be assigned to one of those two families and can be distinguished further based on architectural details, audio feature representations and loss functions used. Audio feature representations include various time-domain and frequency-domain representations, which was elaborated

further in Section 2.1. Loss functions include standard metrics like L1 and L2, but also signal-to-noise ratio (SNR) or perceptually motivated objective metrics have been used [26]. In [4], a perceptually-motivated frequency weighting has been applied to an L1 metric. In [27], a loss function was learned from subjective human judgements on speech quality. Contrasting to the previously described methods, adversarial networks have also shown to be effective on speech enhancement tasks [28]. Here, loss functions would be replaced or supported by an adversarial discriminator.

2.3.1 Wave-U-Net

As a derivative of 2D U-Net source separation architectures as in [29], Wave-U-Net [21] adapts the concept of deep convolutional networks for time-domain processing of audio signals. The architecture can be studied in Figure 1, it consists of a number of convolutional layers and can be adapted to speech enhancement by interpreting the noisy speech signal as a mixture of clean speech and noise, where clean speech and noise are considered to be separated source signals (as explained above in Section 2.1). This adaption has been further elaborated and explored in [23], [22] and [24].

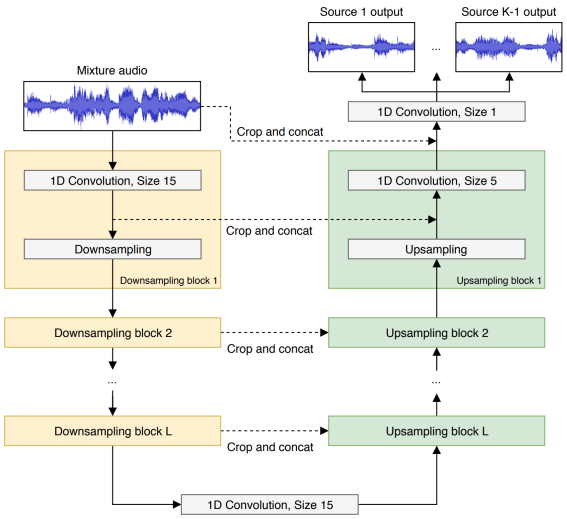


Figure 1: Source separation as performed by Wave-U-Net [21]. Dashed arrows depict so-called *skip connections*.

The architecture resembles an encoder-decoder architecture where the downsampling

blocks form an encoder and the upsampling blocks form a decoder. For each of the convolutional downsampling blocks, the respective input data is subsampled into gradually smaller (decimation by factor 2) chunks of a linearly increasing number of features. This process resembles investigating on the audio with decreasing attention for temporal detail and increasing interest in meta information. In contrast to conventional Auto-Encoder architectures, Wave-U-Net – and U-Net architectures in general – implement the concept of *inter-connected layers* (or self-attention) through so-called *skip connections*. At each stage of depth in the downsampling process, the output of a block is not only passed on to the block beneath, but also to the corresponding decoding block beside. This implies that each upsampling block contains data from two respective adjacent layers and concatenates both before further convolutional processing. This provides the decoding block with information about the encoding process that would otherwise be lost and aids the signal reconstruction process [7].

2.3.2 FullSubNet

In contrast to U-Net architectures, FullSubNet [20] achieves temporal context awareness and depth through LSTM layers and fully connected layers. It manages to effectively combine processing of standard spectral audio representation with a more intricate (frequency-domain-)context-informed spectral representation. The authors decompose FullSubNet into two sub-models: A *full-band* model G_{full} and a *sub-band* model G_{sub}. G_{sub} resembles the author's previous work in [30].

At a given time t, so-called full-band output is achieved by processing the normalized magnitude spectral feature vector X_t ($X_t \in \mathbb{R}^F_{\geq 0}$ where F is the number of frequency bins) through G_{full}. G_{full} utilizes two LSTM layers and one fully connected layer before applying a ReLU activation function. G_{sub} behaves accordingly, but, in this case, a sub-band spectral feature matrix is used as input. This matrix consists of F rows corresponding to F frequency bins. Each of these rows denotes a sub-band $\widetilde{x}_t(f)$:

$$\widetilde{x}_t(f) = [X_t(f-N), \ldots, X_t(f), \ldots, X_t(f+N)] \tag{2.6}$$

The fth sub-band $\widetilde{x}_t(f)$ is defined as a vector including the $2N$ frequency bins neighbouring the fth frequency bin and the fth frequency bin itself. It follows that the sub-band model input at a time t, $\widetilde{x}_t$ is a matrix of dimensions $F \times (2N + 1)$. This representation is meant to facilitate the discrimination between speech and noise [30].

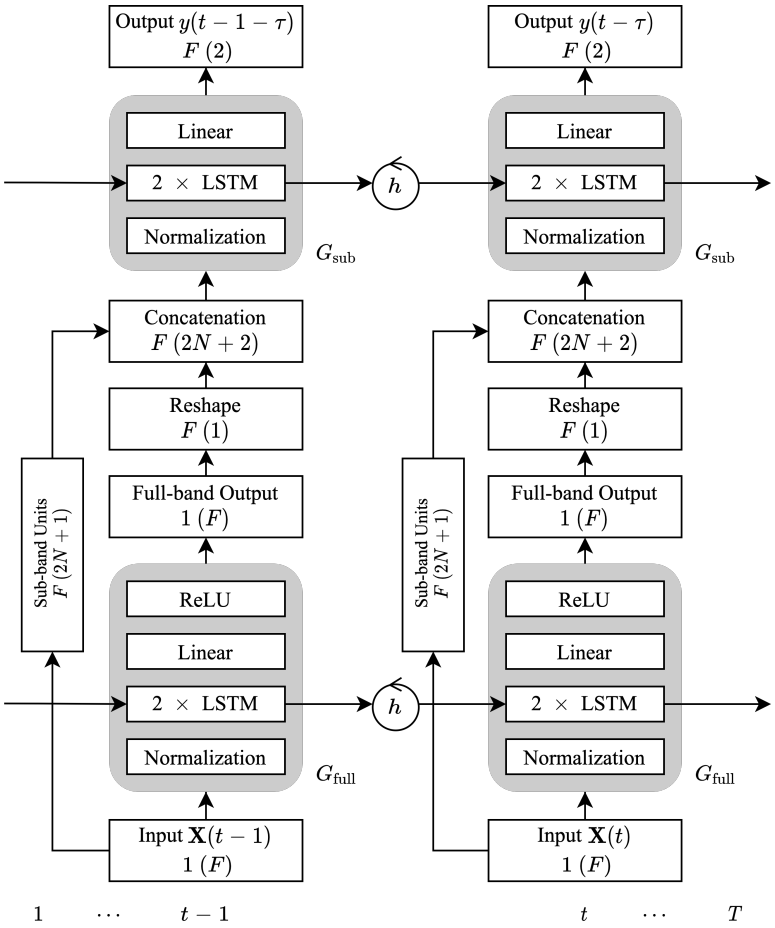


Figure 2: The FullSubNet architecture for denoising of speech signals [20]. It represents a fusion of a full-band model G_{full} and a sub-band model G_{sub}, which both rely on LSTM layers and dense layers.

Both full-band and sub-band output form a (complex) frequency spectrum and can be interpreted as audio data just like the input spectrum X_t. Figure 2 depicts how full-band model G_{full} and sub-band model G_{sub} are brought together to build the final FullSubNet model. In the final model, the sub-band model G_{sub} is computed exactly like before, but the sub-band input matrix is concatenated frequency-wise with the full-band output $G_{full}(X_t)$. The rows $\widetilde{x}'_t(f)$ of this new matrix are similar

to the previous definition of $\widetilde{x}_t(f)$ (Equation 2.6):

$$\widetilde{x}'_t(f) = [X_t(f-N), \ldots, X_t(f), \ldots, X_t(f+N), G_{full}(X_t)(f)] \tag{2.7}$$

The full-band output for a specific frequency, $G_{full}(X_t)(f)$ can be considered a scalar value. It follows that the sub-band model input in the final model, $\widetilde{x}'_t$ is a matrix of dimensions $F \times (2N+2)$.

While the input is derived solely from the magnitude spectral feature X_t, the training target is a complex ideal ratio mask (cIRM) as described in [31]. This method implies that, in order to achieve the actual estimate of the original speech, the complex-valued mask will be multiplied with both real and imaginary parts of the transformed noisy signal. To estimate such a complex ratio mask, the sub-band model outputs separate values for its real and imaginary parts, thus forming a matrix of dimensions $F \times 2$. For further details on complex ideal ratio masking, please refer to [31].

At the time of writing, the paper for FullSubNet has just been presented in the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). It reports state-of-the-art results [20] in speech denoising on the DNS Challenge dataset [32], but has not yet been subject to further research.

Chapter 3

Datasets and Preprocessing

Disclaimer: *In order to facilitate data retrieval for future efforts, all datasets are described with a certain amount of detail and a link is provided where possible. Some datasets are proprietary and cannot be shared, but are described nonetheless.*

In this chapter, the datasets used for our experiments in denoising and leakage removal are described. Next to singing and speech data, this includes impulse responses and noise recordings used for data augmentation. Moreover, information on pre-processing and balancing of data is provided. Any preprocessing steps performed on the described data will be emphasized with bold text and curly brackets. Preprocessing steps did include (all were optional depending on the nature of each dataset):

[$\mathcal{M}$] Exploding multi-channel recordings of n channels to n separate monoaural tracks. This does not only speed up the training process by reducing data loading times, but also helps in exploiting the variety in measurements originally provided for each dataset.

[$\mathcal{S}$] Manual selection of a subset of recordings from the given dataset. This was done to approximate our use case of (overdubbed) singing home recordings in the best possible way (see Section 1.1).

[$\mathcal{T}$] Trimming all audio files from a dataset to 60 seconds of length. This

helps speed up the training process by reducing data loading times.

Data derived from preprocessing was further segmented and balanced for the tasks of training, validation and testing. Note that, for each dataset, data was segmented in such a way that the underlying categories (e.g. speaker identity, room type) would be kept distinct between the segments for training, validation and testing. This ensures that data from the test/validation set would be unseen by a given model at each point of the training process. Please refer to tables 1, 2, 3 and 4 for exact numbers. Note that, in those tables, the *bal* value indicates which relative proportion of the preprocessed data was actually used after segmentation. An example: A number of 800 samples (after preprocessing) that would be segmented into segments including only 600 samples or even 1000 samples would have a *bal* value of $\frac{600}{800} = 0.75$ or $\frac{1000}{800} = 1.25$, respectively. This refers to the known concept of over/undersampling. It implies either selecting a subset of given data (undersampling, $bal < 1$) or copying individual pieces of data to increase its numbers (oversampling, $bal > 1$). Undersampling means reduced variety of data, but oversampling does not increase variety.

3.1 Singing/Speech

Not all data described in this section is available to the public, which is discussed in Section 7.4; in this work, it was used for training and testing our denoising systems (see Section 4).

- **Singing data from VocalSet[1]:** One publicly available resource for isolated/a-capella singing voice recordings is VocalSet. It has been published in 2018 and consists of about ten hours (3615 individual recordings of about 10 seconds in average length) of monoaural singing voice recordings from 20 singers of mixed gender (11 male, 9 female) and voice type [33]. It is to mention that an *Alto* voice type is not included in VocalSet. Recordings were conducted in a professional recording booth and captured at a sample rate of 44.1 kHz. Singers were asked to

[1] VocalSet: A Singing Voice Dataset: `https://zenodo.org/record/1193957`

perform excercises including arpeggio, longtone and scale exercises in 16 different styles (e.g. *belt, trill, forte*), each on five different vowels (i.e. *a, e, i, o, u*). Furthermore, excerpts of popular songs (including English and Italian language) were recorded in three different styles (*sung, spoken, vibrato*). For exact details and exemplary classification results, please refer to the referenced paper [33]. **[$\mathcal{S}$]**: In our work, this dataset was used solely for testing. To best approximate real-world singing scenarios, only the subset with excerpts (139 recordings, 0.8 hours) was used.

- **Singing data from Voctro Labs[2]:** More recordings were retrieved from a dataset that is property of the company Voctro Labs S.L. Please refer to Section 7.4 on how the unavailability of this dataset could be compensated for by using VocalSet in future efforts. It consists of about 7.5 hours (5220 recordings averaging about five seconds in length) of multilingual singing voice recordings. The recording process is comparable to that described in [33] and also leads to monoaural recordings of 44.1 kHz sample rate. Here, singers of all major voice categories (i.e. including *Soprano, Alto, Tenor, Bass*) were asked to vocalize written text excerpts with melodies of their choice. The text excerpts are in the languages of English, German, Mandarin and Spanish.

- **User recordings from Cantāmus:** A resource for subjective testing of our denoising systems was found in user recordings conducted by real users with the Cantāmus app. **[$\mathcal{S}$]**: At the time of writing, it provided 493 monoaural recordings (8.1 hours), of which an automatic subselection (six recordings) was used for user testing (Section 4.5). This data is again property of Voctro Labs S.L.

- **Speech from VCTK[3]:** The pool of human voice recordings for our experiments was expanded by including speech data from Voice Cloning Toolkit (VCTK) corpus [34]. It was published in 2017 by members of the University of Edinburgh and the latest version known at the time of writing is available online. Originally aimed for text-to-speech synthesis systems and later published for voice

[2]Voctro Labs S.L.: `https://www.voctrolabs.com/`
[3]CSTR VCTK Corpus: `https://datashare.ed.ac.uk/handle/10283/3443`

Source	Pre-process	Training		Validation		Testing		**Total**		
		#	len	#	len	#	len	#	len	bal
VCTK	$\mathcal{M}$	16000	15.0	4000	3.8	0	0	20000	18.8	0.23
Voctro Labs		16000	24.0	4000	4.8	0	0	20000	28.8	3.83
VocalSet	$\mathcal{S}$	0	0	0	0	139	0.8	139	0.8	1.0
Cantāmus	$\mathcal{S}$	0	0	0	0	6	0.0	6	0.0	1.0
Total		32000	39.0	8000	8.7	145	0.8	40145	48.5	0.43

Table 1: Segmentation of singing and speech data; raw data for experiments on denoising. Source labels according to Section 3.1. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$.

cloning, this dataset provides about 41 hours (about 44000 recordings averaging about three seconds in length) of text-annotated English speech with about 400 sentences per 109 speakers. The recordings were conducted with two different microphones and in a hemi-anechoic chamber [34]. VCTK samples are published with a sample rate of 48 kHz and in stereo format. **[$\mathcal{M}$]**: Treating each of the microphone signals as an individual recording, this resource provides approximately 88000 individual recordings of about 82 hours in total length.

Segmentation was performed with an 80/20 split, based on singer/speaker identity. Training data was balanced to equal numbers of speech and singing. Test data consists solely of singing datasets (i.e. VocalSet) (see Table 1).

3.2 Choir Singing

Data described in this section is not available to the public, which is discussed in Section 7.4; in this work, it was used for training and testing our proposed leakage removal system (see Section 5).

For the task of leakage removal, a good approximation of the real-world use case – individual singers doing overdub recordings while listening to a synthesised reference choir and/or instruments – demands the use of polyphonic choir recordings. This cannot be easily approximated with e.g. randomly combined singing voice recordings because such an approach would likely imply losing the property of harmonic and

rhythmic correlation between foreground singing and background music (as mentioned in Section 2.2). Therefore, we have gathered multi-track audio resources specifically targeting choir music.

- **Choir recordings from Cantoría:** In the scope of the TROMPA project, three accompanied pieces (i.e. *Soprano, Countertenor, Tenor, Bass, Organ*) had been recorded for a workshop together with the vocal quartet Cantoría [35]. The outcome were about 23 minutes of monoaural audio, containing 15 individual tracks with an average length of about 90 seconds. Although not being extensive for training a neural network, this data was used for evaluation of our leakage removal system (see Section 5.6).

- **Synthesised choir recordings from Cantāmus:** At the time of writing, 148 choir pieces with diverse instrumentations had already been synthesised for the Cantāmus app, thus being property of Voctro Labs S.L. This is a resource of 612 monoaural synthesised voice tracks and 137 instrumental accompaniment tracks, all averaging about 160 seconds in length. It yields about 33 hours of audio, which was sufficient as a starting point for data augmentation applied in training and testing of the proposed leakage removal system (Section 5.1). The voice tracks are synthesised with the voice types *Soprano, Countertenor, Tenor* and *Bass*; the accompaniment tracks include sounds of various non-percussive orchestra instruments. **[$\mathcal{S}$]**: Seven files were duplicates and therefore a total of 742 files from this resource have been used.

Segmentation was performed with an 80/20 split, based on piece identity. Singer identities overlap between training and validation. Test data consists of real recordings from Cantoría, while training data includes only synthesised voices. No balancing was applied (see Table 2).

3.3 Impulse Responses

*IR*s (impulse responses) can be used to simulate certain recording scenarios such as room reverberation or microphone coloration. For our experiments, we have

Source	Pre-process	Training		Validation		Testing		**Total**	
		#	len	#	len	#	len	#	len
Cantoría		0	0	0	0	15	0.4	15	0.4
Cantāmus Syn	$\mathcal{S}$	592	26.3	150	6.7	0	0	742	33.0
Total		592	26.3	150	6.7	15	0.4	759	33.4

Table 2: Segmentation of choir singing data; raw data for experiments on leakage removal. Source labels according to Section 3.2. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours.

retrieved *RIR*s (room IRs) and *MIR*s (microphone IRs) from various sources. The RIRs have been individually curated to fit our use case, so RIRs measured in e.g. very large rooms have been neglected. This data was used for experiments in both denoising and leakage removal.

- **RIRs from the ACE challenge[4]:** The Acoustic Characterization of Environments (ACE) challenge 2015 was an effort to tackle the automatic characterization of acoustic environments. For this, seven real-world room environments (e.g. office, building lobby) have been measured with six different microphone setups (e.g. phone, microphone array; including consumer devices) in two different spatial positions, each measurement sampled at 48 kHz [36]. **[$\mathcal{M}, \mathcal{S}$]**: ACE provides 84 RIRs, of which 48 were selected for our purposes. Treating each microphone channel (recordings were made with microphone arrays of up to 32 channels) separately, we can derive 408 monoaural RIRs.

- **Aachen Impulse Response (AIR)[5]:** First published in 2009 and later enlarged, this database provides binaural RIRs from eleven real-world acoustic environments (e.g. office, lecture room) [12, 37]. To capture binaural sound, IRs were recorded with multiple microphones at various distances to the sound source. As waveform audio, AIR includes a total of 344 individual IR recordings corresponding to eight environments, each sampled at 48 kHz. **[$\mathcal{S}$]**: For our purposes, we have selected a subset of 68 RIRs from AIR.

[4] Acoustic Characterisation of Environments: THE ACE CHALLENGE: `http://www.ee.ic.ac.uk/naylor/ACEweb/`

[5] Aachen Impulse Response Database: `https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/`

- **RIRs from the REVERB challenge[6]:** Measured RIRs have been shared as part of the REVERB challenge 2014, which was an effort to evaluate then state-of-the-art dereverberation and automatic speech recognition methods [38]. A number of 24 RIRs were measured at 16 kHz, in six different room conditions, with four different microphone positions and with an eight-channel microphone array [39]. **[$\mathcal{M}$,$\mathcal{S}$]**: Treating each microphone channel as a separate source, this collection provides 192 RIRs, 64 of which were elicited for our work.

- **OpenSLR Simulated Room Impulse Response Database[7]:** These simulated RIRs have been shared as part of work done by Ko et al. in 2017 [40]. The RIRs were created with an RIR generator tool provided by AudioLabs Erlangen [41]. **[$\mathcal{S}$]**: The collection provides 60000 monoaural IRs sampled at 16 kHz, of which 20000 were selected for our experiments.

- **Vintage Mics[8]:** This collection includes 65 vintage (e.g. AKG D12) MIRs from 30 manufacturers, all of them originally shared by the Microphone Impulse Response Project (MicIRP) [42]. The MIRs were captured as monoaural audio at 44.1 kHz and 48 kHz; recordings were conducted in a small, acoustically treated booth (surrounding the microphone at about 30 cm distance) [42].

RIR segmentation was based on room identity, MIR segmentation was based on microphone identity. For each dataset, individual splits were performed. Test data originates only from one dataset for both MIR and RIR. Balancing was applied to provide even numbers between datasets in the case of training (see Table 3).

3.4 Noise profiles

Noise audio can be used to degrade signals with additive noise. We retrieve noise recordings from various sources. The noise audio files have been individually curated to fit our use case. Therefore, noise recorded in e.g. a subway environment or very reverberant environments has been neglected.

[6]TrainData from the REVERB challenge: `http://reverb2014.dereverberation.com/tools/reverb_tools_for_Generate_mcTrainData.tgz`
[7]OpenSLR: Simulated Room Impulse Response Database: `https://www.openslr.org/26/`
[8]AudioThing: Vintage Mics: `https://www.audiothing.net/impulses/vintage-mics/`

Source	Pre-process	Training	Validation	Testing	**Total**	
		#	#	#	#	bal
ACE	$\mathcal{M}, \mathcal{S}$	400	100	0	500	1.23
AIR	$\mathcal{S}$	400	0	0	400	5.88
REVERB	$\mathcal{M}, \mathcal{S}$	400	0	0	400	6.25
OpenSLR	$\mathcal{S}$	400	100	100	600	0.03
RIR-Total		1600	200	100	1900	0.09
Vintage Mics		45	10	10	65	1.00
MIR-Total		45	10	10	65	1.00

Table 3: Segmentation of RIR and MIR data; raw data for experiments in denoising (RIR) and leakage removal (RIR+MIR). Source labels according to Section 3.1. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$.

- **Noise from the ACE challenge:** These noise profiles can be retrieved following the same link as described in Section 3.3 and were shared as material for the ACE challenge 2015, utilizing the recording setup described above. For each room, microphone and microphone position, three noise types (i.e. *ambient, fan, babble*) were recorded, thus leading to 252 individual noise profiles of 13.6 hours in length [36]. $[\mathcal{M}, \mathcal{S}, \mathcal{T}]$: For our purposes, only the 84 *fan* profile recordings were selected, each trimmed to 60 seconds (original: 171 seconds on average), totalling 1.4 hours in length; this yields 710 individual monoaural channels with a total length of about 11.8 hours.

- **DEMAND[9]:** Diverse Environments Multichannel Acoustic Noise Database (DEMAND) provides about 24 hours of noise recordings in 18 different real-world environments; environments including indoors (e.g. a cafeteria) and outdoors (e.g. a park). Recordings were conducted with a 16-channel microphone-array and sampled at 48 kHz [11]. $[\mathcal{S}, \mathcal{T}]$: For this work, out of the 18 scenarios, 5 were selected, yielding 80 monoaural tracks. All those tracks were trimmed to 60 seconds (original: 300 seconds), so a subset of about 1.7 hours was used in our work.

- **Noise from the REVERB challenge:** Next to RIRs, noise profiles have been shared as part of the aforementioned REVERB challenge (see Section 3.3). Un-

[9]DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments: `https://zenodo.org/record/1227121`

der similar circumstances as described above (here, 10 instead of 4 microphone positions were used), 60 samples (30 minutes) of static room background noise, mainly originating from air conditioning systems, was recorded [39]. [$\mathcal{M}$]: Treating each microphone channel as a separate source, this collection provides 480 noise profiles with a total length of 4 hours.

- **Laptop Noises from Freesound[10]:** Freesound [43] is a community-based platform for sharing audio samples of all sorts. It has an API to query and download audio files, which was used to search for noise recordings provided under the CC0 license. With the query terms 'laptop mic', 'mic hit', 'keyboard click', 'computer fan' and 'laptop fan' and manual selection we were able to retrieve 55 audio files of 31 minutes in length. These are assumed to be beneficial for simulating possible noise degradations of user recordings; especially static fan noise and hits to the case/microphone of the end user device. These recordings have sample rates between 16 kHz and 48 kHz, but more detailed recording conditions are not documented. [$\mathcal{S}, \mathcal{T}$]: 44 of them are stereophonic, so there are a total of 99 monoaural channels with a length of 56 minutes available.

Segmentation was performed based on room identity/environment. Data retrieved from Freesound was just split based on recording identities (Freesound ID). For each dataset, individual splits were performed. Test data originates from multiple datasets to provide a variety of noise profiles. Balancing was applied such that the case-specific data retrieved from Freesound would be overrepresented by a factor of two (see Table 4).

[10] wimmerb: Laptop noises from Freesound `https://github.com/wimmerb/laptop_noise_freesound/`

Source	Pre-process	Training		Validation		Testing		**Total**		
		#	len	#	len	#	len	#	len	bal
ACE	$\mathcal{M}, \mathcal{S}, \mathcal{T}$	280	4.7	60	1.0	60	1.0	400	6.7	0.56
DEMAND	$\mathcal{S}, \mathcal{T}$	280	4.7	60	1.0	60	1.0	400	6.7	5.00
Freesound	$\mathcal{M}$	560	6.1	120	1.1	120	0.8	800	7.9	8.01
REVERB	$\mathcal{M}$	280	2.3	60	0.6	0	0	340	2.8	0.71
Total		1400	17.8	300	3.6	240	2.8	1940	24.1	0.71

Table 4: Segmentation of noise data; raw data for experiments on denoising. Source labels according to Section 3.4. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$.

Chapter 4

Methods for Denoising

Two deep learning architectures with demonstrated efficacy in denoising of speech, FullSubNet [20] and Wave-U-Net [22, 23], were compared on denoising of singing voice recordings. To achieve comparable results, they were each specifically trained and tested on singing audio of the same origin and nature, augmented with the same techniques.

Below, we will be using the following notations:

$\boldsymbol{x}$	Clean (reverberant) singing/speech signal, also referred to as *target*
$\boldsymbol{y}$	Mixture of target signal x and noise n, also referred to as *input*
$\boldsymbol{n}$	Noise profile
$\boldsymbol{h}$	Room impulse response (RIR)

In general, the models were trained with input-target pairs of noisy data y and clean data x, where y is derived by adding a noise n to x (see Equation 2.2). As described below, 75% of the clean target data x was reverberant.

All audio processing in this chapter was done on monoaural audio with a sample rate of 16 kHz (Wideband audio). Audio data after preprocessing (see Chapter 3) will be referred to as *raw data*.

4.1 Data Augmentation

For each data point tuple (x, y) of training data, the following augmentation/processing stack was used and dynamically applied on x to compute y. For each augmentation technique, the percentage of samples it was applied to is given next to a more detailed description. For all random numbers mentioned, discrete numbers would be drawn from a uniform distribution within the given range. Note that some augmentations affect not only input data y, but also target data x.

- **Silent target. 1%:** To simulate the scenario of unvoiced excerpts, the target x is silent. Through $x = 0$, it follows that $y = n$. In this case, the rest of the augmentation stack is ignored.

- **Pass. 5%:** Signals that are already of ideal quality should be preserved. We seek to minimize $f(x) - x$ by not applying any noise to the target signal. In this case, the rest of the augmentation stack is ignored (but this case is overshadowed by the above case of 'silent target').

- **Pitch shift. 25%:** Random pitch shift between -800 cent and 800 cent (corresponding to 17 semitones) is applied to the noise n. The purpose of this is to enhance the general variety in our noise profiles, covering real-world cases where noises are resonant in very low/high frequencies.

- **Reverberation. 75%:** Using a reverberant target. Given a clean signal x' we derive the target $x = h * x'$ through convolution with an RIR h. This is done accordingly in [20] and targets robustness to non-optimal recording conditions. As described in Section 3.3, some of the given RIRs implicitly target microphone coloration from consumer devices.

- **Levelling. 100%:** Between target x and noise n, a random *SNR* (signal-to-noise ratio) between -3 dB and 25 dB was established to form y. Note that the SNR range includes negative values, opening the possibility for very noisy scenarios. Thereafter, y was levelled to a random level between -35 dBFS and -15 dBFS; of course, x was leveled proportional to the levelling of y to ensure

correctness of Equation 2.2. Here, dBFS is the amplitude-oriented measure of decibels relative to full scale.

- **Subsampling. 100%:**. For each audio (voice, noise) sample from the raw data, a randomly positioned snippet of 16384 (1.072 seconds; for Wave-U-Net) or 41952 samples (3.072 seconds; for FullSubNet) in length was taken to create an individual data point. Note that the lengths are multiples of the Wave-U-Net input size, which facilitates evaluation.

4.2 Training and Experimental setup

The process described here was applied on both Wave-U-Net and FullSubNet. Raw data segments for training and validation were derived from singing and speech data (see balancing in Table 1); for the origins of noise and RIR data, please refer to tables 3 and 4. During training, the described augmentations were applied to the raw data dynamically, coupling random files from the mentioned voice, noise and RIR sources. The convolution processing was sped up by convolving 10 raw singing/speech samples and 10 RIRs simultaneously instead of doing convolutions piece by piece. One such batch convolution would produce $10 \times 10 = 100$ reverberant samples that could then be used for the case of reverberant target data described above (Section 4.1).

After each training epoch, objective validation scores were computed on a validation set and closely monitored for manual selection of a best model. For the scores, we used the metrics of SI-SDR, STOI and PESQ, which are described in section 6. To ensure comparability between epochs and in contrast to the training process, the validation set, consisting of 400 parallel data point tuples (x, y) was pre-computed. Therefore, random files from raw data segments (voice, noise and RIR; see tables 1, 3 and 4) for validation were coupled and augmented. Augmentations were conducted in the same way as training data, but disregarding the 'silent target' technique (see Section 4.1), which would otherwise inhibit the correct computation of the SI-SDR metric.

4.3 Wave-U-Net: Configuration

The configuration used is identical with the configuration in the original Wave-U-Net paper [20]. The Wave-U-Net was configured to have a depth of 12 layers with kernel sizes of 15 for the downsampling blocks and 5 for the upsampling blocks. Hence, a signal providing 1 feature of size 16384 is decomposed into 288 features of size 4 and reconstructed to the original shape. The described configuration yields about 10.13 million parameters. Training was performed with a batch size of 16. The Adam optimizer with a learning rate of 0.0001 and decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was applied. After early stopping, its learning rate was adjusted to 0.00001 for fine-tuning, which is accordance with the original paper [21]. Training objective was the mean squared error (MSE) on each batch as waveform data ($16 \times 1 \times 1684$). A Pytorch implementation [44] targeting the adaption of Wave-U-Net for Speech Enhancement (according to [22]) was used.

4.4 FullSubNet: Configuration

The configuration used is identical with the configuration in the original FullSubNet paper [20]. For the two LSTM layers of full-band and sub-band separation blocks, a unit size of 384 and 512 was used, respectively. This configuration yields about 5.64 million parameters. Audio was transformed to an STFT representation, utilizing Hanning windows of FFT size 512 samples and a hop size of 256 samples. Together with the aforementioned sample lengths, this would produce 192 frames per sample. A look-ahead of 2 frames was chosen to predict the current frame, which therefore describes a semi-causal approach. Training was performed with a batch size of 16. The Adam optimizer with a learning rate of 0.001, decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and gradient norm clipping of 10 was applied, which is in accordance with the original paper [20]. Training objective was the mean squared error (MSE) measured on the predicted complex ideal ratio mask for each STFT frame and per batch ($16 \times 192 \times 512 \times 2$; batch size $\times$ frames $\times$ FFT size $\times$ complex axis). The official Pytorch implementation of FullSubNet was used [45].

4.5 Evaluation Methods

Objective and subjective evaluations have been conducted on one synthetic test set (objective and subjective evaluations) and one real-world test set of noisy singing voice recordings (only subjective evaluations).

For objective evaluation, the metrics of SI-SDR, STOI and PESQ were applied by computing their averages over a test set. As opposed to the mathematically motivated SI-SDR measure, PESQ and STOI (described in Chapter 6) have been found to correlate with human perception on quality and intellegibility of speech [46, 47], thus giving us an impression on general perceived voice and audio quality. Singing-specific metrics like PESnQ [48] were disregarded, as they partly reflect pitch correctness, whereas pitch is not subject to any modifications in our experiments. All objective evaluations would be conducted on augmented (noisy) and enhanced data. The test set (200 instances of parallel data clean/noisy) was pre-computed with the same data augmentation techniques as for training, but disregarding the 'silent target' technique (see Section 4.1). Raw data for the test set was derived from the unseen VocalSet and an unseen split of noise and RIR files (see tables 1, 3 and 4). Those raw files were randomly coupled for augmentation.

For subjective evaluation, two tests were conducted with a consistent set of 20 volunteers. The testing environment has been created with an adaption of the listening test framework BeaqleJS [49, 50].

As a test on real world data, user recordings from the Cantāmus app (see Section 3.1) have been judged by the volunteers. The expenses for mean opinion score testing as specified in ITU-T P.808 [51] would unfortunately not have been feasible for this work, so we have decided to ask the volunteers for their *ACR* (absolute category rating) on noisy and enhanced versions of six user recordings. ACR is part of the test ensemble recommended by ITU-T P.808 and measures subjective opinions on speech quality within the categories *Bad, Poor, Fair, Good and Excellent* corresponding to discrete scores from 1 to 5. For the test's visual appearance, please refer to Figure 3.

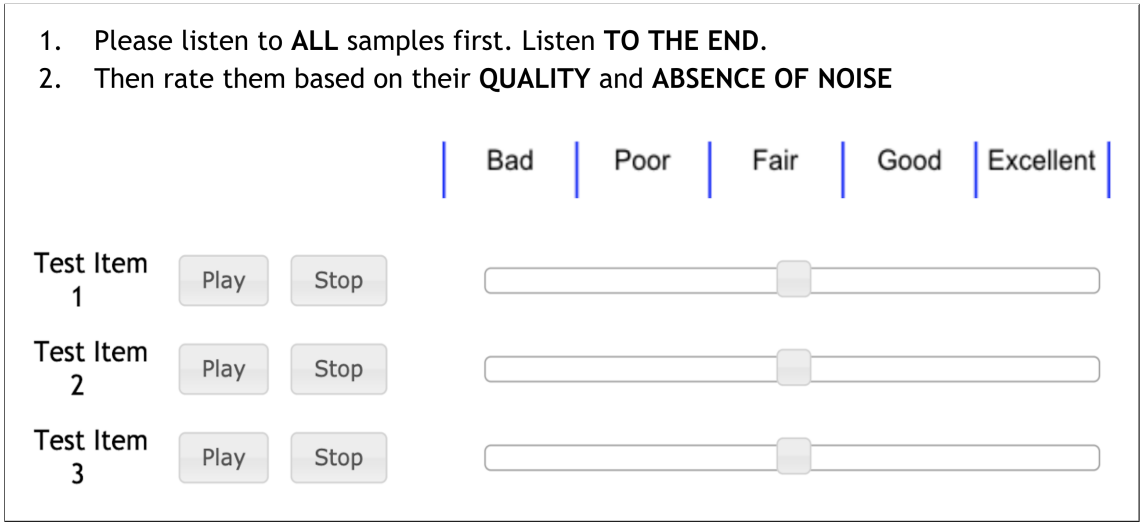


Figure 3: ACR testing: Participants were asked to listen through all **Test Items** and then rate them based on perceived quality and absence of noise. Note that the slider 'snaps' to discrete positions according to the given categories.

Furthermore, a small set of six random audio subsamples from the described synthetic test set have been evaluated with a test similar to *MUSHRA* (MUlti Stimulus test with Hidden Reference and Anchor) [52]. In MUSHRA, participants are asked to judge any and all differences between specific audio samples (*conditions*) and a (clean) reference sample, on scale from 0 to 100. One of the conditions is the reference itself, which is called *hidden reference*. MUSHRA also introduces the concept of *hidden anchor*, which means adding deliberately degraded versions of the clean reference to the set of conditions. Reporting results on these hidden reference and anchor conditions should facilitate interpretation/comparability of other test results. Originally proposed to evaluate quality of e.g. transmission systems or audio codecs, MUSHRA could not be adapted in its entirety to the evaluation of denoising systems. We propose the following test (see Figure 4): From our parallel test data, we have used the clean audio as a reference and as a condition (hidden reference). However, a purposefully degraded hidden anchor derived from the reference signal would not have been expedient. We have not used such a hidden anchor per se, but instead have added the noisy signal as a condition, which was believed to fulfill the anchor's function as a lower reference point. The other conditions were enhanced signals from Wave-U-Net and FullSubNet. Moreover, the original MUSHRA test divides its 101-point judgment scale into the qualities *Bad, Poor, Fair, Good and*

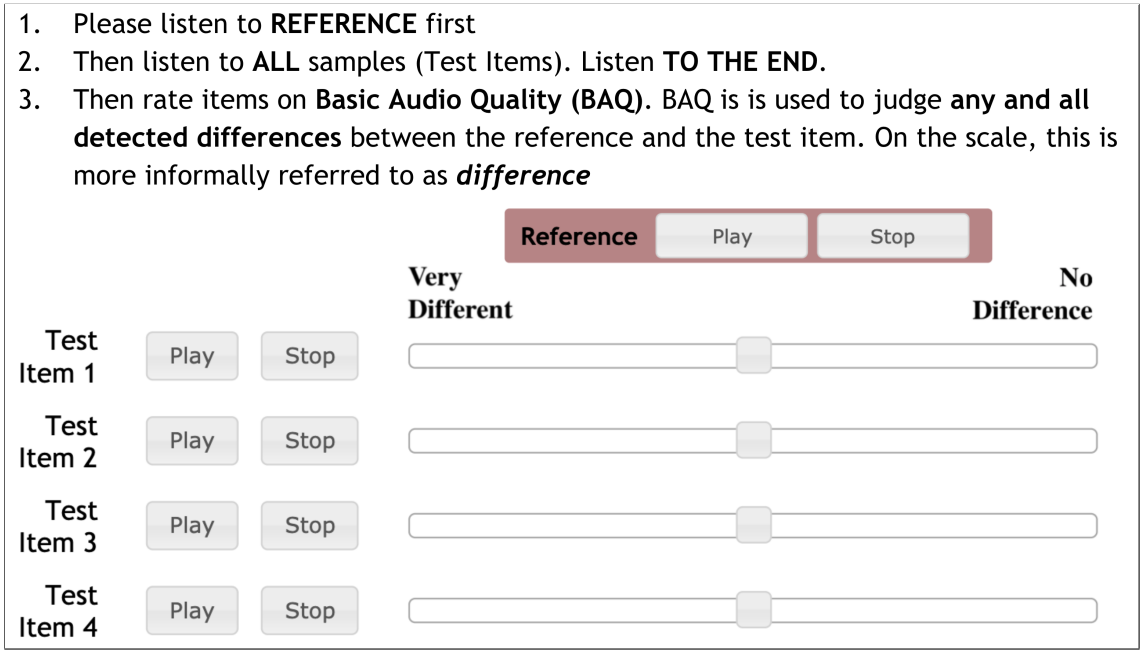


Figure 4: MUSHRA-like testing: After listening through the **Reference** and all **Test Items**, participants were asked to rate any and all detected differences between the reference and the test item on a scale from *Very Different* (0) to *No Difference* (100).

Excellent, a practice which has been subject to criticism due to potential biases [53]. As a reaction to that, we have chosen to use only two labels – *Very Different* and *No Difference* – for respective scale ends.

Chapter 5

Methods for Leakage Removal

As explained in Section 1.1, leakage removal and AEC describe the removal of artifacts that originate from listening to other audio (i.e. background music/accompaniment or far-end speech) while recording. This resembles the problem of denoising, but with the difference that, here, we have knowledge about the signal leading to the degradation. In [2], a novel approach in AEC has been presented by exploiting the same relationship between denoising and AEC. There, a pre-existent denoising architecture was modified to retrieve and include information from a second input.

Inspired by this, we have adapted a FullSubNet denoising architecture to leakage removal. The proposed architecture represents a FullSubNet architecture with two parallel inputs. It was trained specifically for the application of leakage removal on our use case. On this task, it was evaluated against a conventional FullSubNet baseline model.

Hereafter, we will be using the following notations; please refer to Section 2.2 with equations 2.4 and 2.5 for a thorough explanation on their particular relationships:

$\boldsymbol{y}$	A leakage-degraded user recording; referred to as *noisy* signal
$\boldsymbol{x}$	Original *reference* audio causing leakage audible in y
$\boldsymbol{s}$	The user recording without leakage degradation; referred to as *target*

or *clean* signal

$\boldsymbol{h}$ A room/degradation filter (applied to x)

$\boldsymbol{f(x, y)}$ The proposed neural model f, that, given x and y (same as above) should predict $\hat{s}$, a best-possible estimate on s

$\boldsymbol{\hat{s}}$ The prediction produced by f (through complex time-frequency masking of y)

The proposed model was trained on triplets of s, x and y, where x and y would be used as model inputs and s was the target audio. The baseline model, which has no additional input, was trained only on pairs of y and s. Note that the actual training target for FullSubNet is a complex ideal ratio mask, which, when applied to the noisy signal y, would produce s (see Section 2.3.2). Referring to s as the *target* is therefore a simplification for readability.

All audio processing in this chapter was done on monoaural audio with a sample rate of 16 kHz (Wideband audio). Audio data after preprocessing (see Chapter 3) is referred to as *raw data*.

5.1 Data Augmentation

In our scenario of overdub choir recordings (described in Section 1.1), the clean audio s and reference audio x include choir voices (s and x) as well as instrumental accompaniment (only x). To approximate s and x for a real-world scenario, raw data for training and validation was taken from choir/instrumental excerpts synthesised for the Cantāmus app (see Table 2). Therefore, the choir excerpts used during training are not taken from real singing voice recordings. Below, individual instances of raw audio will also be referred to as *tracks*.

We start by describing how data for s and x was combined. Hereafter, we will refer to the following definitions (next to the ones defined above).

$\boldsymbol{r}$ A raw audio track with $r \in R$

$\boldsymbol{R}$ The set of all raw audio tracks (see Table 2)

$\boldsymbol{\mathcal{T}(r)}$ The voice type of a given raw audio track r. The voice types include *Soprano, Alto, Tenor, Bass* and *Accompaniment*

$\boldsymbol{\mathcal{P}(r)}$ The choir piece that a raw audio track r was originally synthesised for. Given the polyphonic nature of choir music, multiple tracks from R can belong to the same piece

$\boldsymbol{l}$ The sample length (duration) of subsamples. All subsamples of a data point triplet (s, x, y) have the same length l

$\boldsymbol{t}$ The time index t, with $0 \leq t < l$

$\boldsymbol{t_r}$ The subsample offset within a signal r. Thus, $r(t_r + t)$ defines a subsample of r, starting at t_r and with a length of l

To create an individual data point (triplet of s, x and y), we needed to first find suitable candidates for s and x before creating a noisy sample y from those. To retrieve a candidate for s, we would search for a voiced subsample of a random vocal track.

$$s(t) = r(t_s + t) \quad \text{with} \quad \mathcal{T}(r) \neq \textit{Accompaniment} \tag{5.1}$$

where t_s is the start of the voiced subsample in r. As described below, the retrieval of voiced subsamples was done with a naive algorithm. In the use case, the reference x would originate from a mix of tracks of various voice types (including the voice type of s itself) and instruments, which are all time-aligned with each other and with s. Naturally, the reference audio would be from the same piece as s. Hence, we could define our candidate for x as follows:

$$x(t) = \sum_{r \in \hat{R}} r(t_s + t) \quad \text{with} \quad \hat{R} \subseteq \left\{ r \in R \mid \mathcal{P}(r) = \mathcal{P}(s) \right\} \tag{5.2}$$

Note that $x(t)$ is time aligned with s as t_s is the sample start for each r. The pool of raw files for x, $\hat{R}$, is defined as a subset of all tracks from piece $P(s)$, which

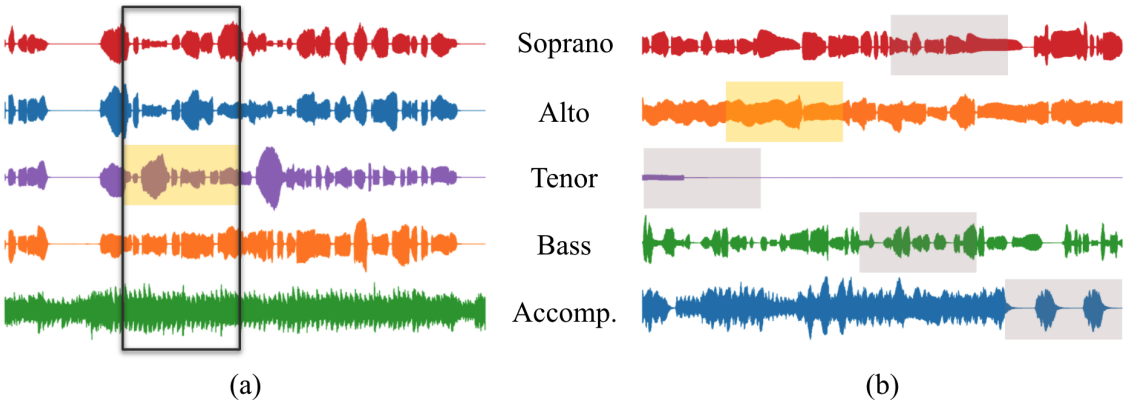


Figure 5: Sampling scenarios: **(a) Aligned:** A voiced subsample (yellow) of the Tenor voice is found and used as target audio s; material for reference audio (black window, all voices) must be from the same piece and cover the same time frame as s. **(b) Random:** Voiced subsamples are found for random excerpts of all voice types; one of them is chosen as foreground audio (yellow), all of them (grey, yellow) are eligible as material for the reference audio.

includes s itself. Defining $\hat{R}$ as a subset of all tracks from a piece simulates user-specific selection of individual reference tracks, which is a feature in the Cantāmus app. For a visual representation of this process, please refer to the *aligned* scenario in Figure 5. However, to achieve greater variety in the training data, this use-case oriented aligned scenario was reinforced with a scenario of mixing samples regardless of their alignment. Here, voiced subsamples for x were derived from random tracks of disjunct voice types.

$$\begin{gathered} x(t) = \sum_{r \in \hat{R}} r(t_r + t) \\ \text{with} \quad \hat{R} \subseteq R \quad \text{and} \quad \forall r, r' \in \hat{R}.\, r \neq r' \implies \mathcal{T}(r) \neq \mathcal{T}(r') \end{gathered} \tag{5.3}$$

Note that t_r is not fixed, as we were specifically looking to retrieve voiced subsamples. Also note that no sample of voice type $\mathcal{T}(s)$ other than $s(t_s + t)$ was allowed in $\hat{R}$. This scenario is not musically meaningful. It provides combinations of x and y that are harmonically and rhythmically less coherent than the use case data, which implies smaller correlation in the time domain and frequency domain. In our implementation, the source selection for x was performed by randomly selecting five synthesised tracks, one from each voice group. A subset of those five would then be selected and randomly positioned voiced parts would be found for each. The

process is visualized in Figure 5 in the *random* scenario. For s, we used only voiced excerpts, which is explained below in the augmentation step *subsampling*. For each instance of s, x would either be derived from other time-aligned choir voices/accompaniment from the same piece as s or from randomly sampled voiced excerpts of the raw data.

To be able artificially create a noisy signal y from s and x, x was degraded with a room filter h and then mixed with s (see Equation 2.4). RIRs and MIRs from various sources (see Table 3) had been pre-selected to simulate h by means of convolution. Therefore, various RIRs and MIRs were convolved with each other in a chain. Let H_r be the set of all RIRs and H_m be the set of all MIRs, then the filter h applied on a signal x can be defined as:

$$\begin{aligned} h(x) &= h_{m,0} * \ \ldots * h_{m,M} * h_{n,0} * \ \ldots * h_{n,N} * x \\ \text{with} \quad h_{r,i} &\in H_r;\ h_{m,i} \in H_m;\ M \in \{0,1,2\};\ N \in \{0,1\};\ M+N \neq 0 \end{aligned} \tag{5.4}$$

Then, in analogy to Equation 2.4, a noisy user recording y was approximated:

$$y = s + h(x) \tag{5.5}$$

Note that we assume the leakage in the mixture, $h(x)$, and x to be time-aligned. In a real world scenario, the problem of time-alignment needs to be solved before or during application of the proposed system. To preserve the property of time alignment between $h(x)$ and x, the transients of all impulse responses were aligned to start at $t = 0$. Moreover and contrary to Section 4 we assume that the recording scenario happens in a quiet room, i.e. no room degradation is applied to s.

After defining the relations and nature of s, x and h, we can proceed to describe the augmentation/processing stack. Augmentation was dynamically applied. For each augmentation technique, the percentage of samples it was applied to is given next to a more detailed description. For all random numbers mentioned, discrete numbers would be drawn from a uniform distribution within the given range.

- **Silent target. 1%:** To simulate the scenario of unvoiced parts in the user

recording, the target s is silent. Since $s = 0$, it follows that $y = h(x)$.

- **Target in reference. 10%:** It was ensured that $s \in x$ in only 10% of the cases. The most problematic case in training was when $s = x$.

- **Degradation filtering. 100%:** As described above, combinations of RIRs and MIRs were convolved with each other and x to form $h(x)$ (see Equation 5.4). The numbers of MIRs and RIRs in each degradation were randomly chosen from hard-coded options, $M \in (0, 1, 2)$ and $N \in (0, 1, 1)$. If both values were 0, an RIR was applied regardless. It follows that an RIR was included in h in $\frac{7}{9}$ of the cases, while the number of MIRs in h was uniformly distributed between 0 and 2.

- **Levelling. 100%:** Between target s and noisy signal y, a random SNR between 6 dB and 20 dB was established to form y. Note that the SNR range includes negative values, opening the possibility for very noisy scenarios. Thereafter, y was levelled to a random level between -35 dBFS and -15 dBFS; of course, x was leveled proportional to the levelling of y to ensure correctness of Equation 2.2. Here, dBFS is the amplitude-oriented measure of decibels relative to full scale.

- **Sampling scenarios. 50%/50%:** For each data point, one of the sampling scenarios *aligned* and *random* (see Figure 5) was randomly selected and applied.

- **Voiced Subsampling:** For each audio sample r from the raw data selection $\hat{R}$, a randomly positioned voiced snippet of 3 seconds in length was taken. The aforementioned retrieval of voiced subsamples was achieved with a naive algorithm. It would jump to randomly positioned time frames in the tracks and analyse the absolute amplitude; based on the portion of individual samples surpassing a certain threshold, the signal would then be classified as voiced. A track is disregarded if there is no voiced candidate found after 10 tries. In the aforementioned (see Figure 5) *aligned* scenario, this algorithm was only be

applied to s, in the *random* scenario it was applied to all tracks that give rise to s and x.

5.2 Architecture

As stated before, a major difference in the problem statements of denoising and leakage removal is the existence of knowledge about the signal x causing the degradation. With that in mind, we propose a model that is a simple adaptation of FullSubNet (see Section 2.3.2) for two inputs (reference audio x and noisy recording y). Given the results on the denoising task (Section 6.1), FullSubNet was considered a promising candidate for such an adaptation.

Full-band and sub-band models both essentially stay the same as for the original FullSubNet, with two LSTM layers followed by a linear (fully connected) layer (and a ReLU activation in the case of full-band). Input features for full-band and sub-band processing are analogous to the original, but are concatenated with information about the additional input, thus generally being of larger sizes in our adaptation. Normalization is treated for each input source separately, which implies the necessity of handling normalization before handing the respective input features to the full-band and sub-band models. The adapted workflow is explained below and depicted in Figure 6.

Hereafter, we will be using the following definition for the STFT magnitude spectral feature of a signal z:

$$\widetilde{Z} = (Z_1, \ldots, Z_t, \ldots, Z_T) \in \mathbb{R}_{\geq 0}^{T \times F} \tag{5.6}$$

Where $Z_t \in \mathbb{R}_{\geq 0}^{F}$ is the magnitude spectrum of an STFT frame at time t, with F frequency bins. T is the number of STFT frames or time steps. Before full-band processing, STFT magnitude features $\widetilde{X}$ and $\widetilde{Y}$ of x and y are normalized by pointwise division with their respective mean scalar value (i.e. the new mean value of all scalars in $\widetilde{X}$ would be 1). The full-band input $N_{full,\,t}$ at a point in time t would then be a concatenation of the (normalized) STFT magnitude spectra for X_t and Y_t

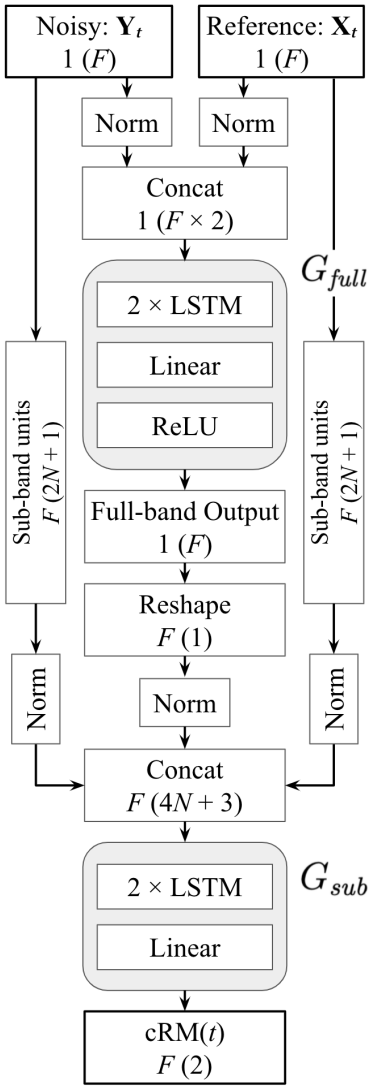


Figure 6: The proposed modified FullSubNet architecture. It is fed with the magnitude spectral features $\widetilde{X}$ and $\widetilde{Y}$. This figure shows the workflow for the tth STFT frame. The second line in each rectangle describes the dimensions of the data at the current stage. For example, "1 (F)" represents one F-dimensional vector. "F ($4N + 3$)" represents F independent ($4N + 3$)-dimensional vectors. Grey boxes indicate submodels G_{full} and G_{sub}.

(corresponding to x and y):

$$N_{full,\,t} = [X_t(0),\, \ldots\, , X_t(F-1), Y_t(0),\, \ldots\, , Y_t(F-1)]^T \in \mathbb{R}_{\geq 0}^{2\times F} \tag{5.7}$$

$$N_{full} = (N_{full,\,1},\, \ldots\, , N_{full,\,t},\, \ldots\, , N_{full,\,T}) \in \mathbb{R}_{\geq 0}^{T\times(2\times F)} \tag{5.8}$$

Here, N_{full} is the full-band input aggregate over all time frames, in analogy to Equation 5.6. Full-band output $G_{full}(N_{full})$ is achieved by applying the full-band model G_{full} to N_{full}. Full-band output at time t is then defined $G_{full}(N_{full})_t$. Essentially recreating a spectral output, full-band output $G_{full}(N_{full}) \in \mathbb{R}^{T\times F}$ yields the same shape as its inputs $\widetilde{X}$ and $\widetilde{Y}$ and is normalized in the same way again.

To retrieve the sub-band feature N_{sub}, the sub-band features $\widetilde{x}$ and $\widetilde{y}$ would need to be derived from $\widetilde{X}$ and $\widetilde{Y}$. The time-aggregated sub-band feature $\widetilde{z}$ for a signal z is defined as:

$$\widetilde{z} = (\widetilde{z}_1,\, \ldots\, , \widetilde{z}_t,\, \ldots\, , \widetilde{z}_T) \tag{5.9}$$

Here, $\widetilde{z}_t$ is the sub-band feature for time t. It can be derived from its STFT magnitude feature Z_t as described in Equation 2.6. Aggregated sub-band features $\widetilde{x}$ and $\widetilde{y}$ are each normalized, again by point-wise division with their mean scalar value. The time-aggregated sub-band input N_{sub} is derived by frequency-wise concatenation of $\widetilde{x}$, $G_{full}(N_{full})$ and $\widetilde{y}$. For a given time t and for a given center frequency f, an entry $N_{sub,\,t}(f)$ can therefore be described as:

$$\begin{aligned} N_{sub,\,t}(f) = [&X_t(f-N),\, \ldots\, , X_t(f+N),\\ &G_{full}(N_{full})_t(f),\\ &Y_t(f-N),\, \ldots\, , Y_t(f+N)]^T \in \mathbb{R}^{4\times N+3} \end{aligned} \tag{5.10}$$

This is in analogy to Equation 2.7. Here, N is the number of neighbouring frequencies above and below f. For a center frequency f, the sub-band input aggregate $N_{sub}(f)$ is retrieved in analogy to Equation 5.6:

$$N_{sub}(f) = \Big(N_{sub,\,1}(f),\, \ldots\, , N_{sub,\,t}(f),\, \ldots\, , N_{sub,\,T}(f)\Big) \in \mathbb{R}^{T\times(4\times N+3)} \tag{5.11}$$

Corresponding to the number of frequency bins in $\widetilde{X}$ and $\widetilde{Y}$, F independent sub-band input representations are created. The sub-band model G_{sub} is then applied on $N_{sub}(f)$ for each center frequency $f \in F$. Hence, by aggregating F iterations of $G_{sub}(N_{sub}(f)) \in \mathbb{R}^{T\times 2}$, the complex ratio mask for $\hat{s}$, $cRM_{\hat{s}} \in \mathbb{R}^{T\times F\times 2}$ is generated.

5.3 Training and Experimental Setup

Training data was retrieved and augmented as described above (see Section 5.1). Therein, raw audio files for x were batch-convoluted, such that individual convoluted files could then be combined with each other (distributive property of convolution). This way, multiple combinations of s and x could be derived from a single set of raw data, combinations both varying in subsample positions and raw sample selection for x (referring to the values t_s, t_r and $\hat{R}$ from Section 5.1). This procedure was meant to accelerate the augmentation process.

After each training epoch, objective validation scores were computed on a validation set and closely monitored for manual selection of a best model. Multiple objective validation scores were computed on a validation set after each training epoch and closely monitored to select a best model manually. For the scores, we used the metrics of SI-SDR and PESQ, which are described in section 6. To ensure comparability between epochs and in contrast to the training process, the validation set, consisting of 97 data triplets (noisy/reference/target), was pre-computed. Therefore, raw files (choir excerpts and IR; see tables 2, 3) were coupled and augmented. Augmentations were conducted in the same way as for training data, but disregarding the 'silent target' technique (see Section 5.1), which would otherwise inhibit the correct computation of the SI-SDR metric. Furthermore, subsample lengths of 5 seconds and the *aligned* sampling scenario was chosen. The number of 97 data points corresponds to 97 voice recordings existent in the raw data for validation.

5.4 Model Configuration

The proposed model was trained with what was essentially the same configuration as for the original FullSubNet (see Section 4.4). For the two LSTM layers of full-band and sub-band separation blocks, a unit size of 384 and 512 was used, respectively. This configuration yields about 6.21 million parameters, which, due to changes in input sizes of full-band and sub-band models is slightly more than for the original FullSubNet. Audio was transformed to an STFT representation, utilizing Hanning windows of FFT size 512 samples and a hop size of 256 samples. Together with the aforementioned subsample lengths, this would produce 188 frames per sample. A look-ahead of 2 frames was chosen to predict the current frame, which therefore describes a semi-causal approach. Training was performed with a batch size of 16. The Adam optimizer with a learning rate of 0.001, decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and gradient norm clipping of 10 was applied. Training objective was the mean squared error (MSE) measured on the predicted complex ideal ratio mask for each STFT frame and per batch ($16 \times 188 \times 512 \times 2$; batch size $\times$ frames $\times$ FFT size $\times$ complex axis). The proposed model was implemented with Pytorch; a link to the implementation is given in Section 7.4.

5.5 Baseline

As stated before, the Baseline model was the original FullSubNet model. In this, the additional information on x would be ignored during training and the model would be trained on just one input ($f(y) = \hat{s}$). The configuration was the same as above, but with smaller input sizes, such yielding about 5.64 million parameters and being compliant to the denoising setup from Section 4.4.

5.6 Evaluation

Objective and subjective evaluations have been conducted on two pre-computed test sets with real recordings from Cantoría (see Section 3.2). Therefore, in general, unseen raw audio files (choir excerpts and IR; see tables 2, 3) were coupled and

augmented. Augmentations were conducted in the same way as for training data, but disregarding the 'silent target' technique (see Section 5.1), which would otherwise inhibit the correct computation of the SI-SDR metric. To model the use case, only the *aligned* sampling scenario from Section 5.1 was chosen.

Objective evaluation was performed with respect to different SNR levels, so six individual test sets with the SNR settings of *0dB, 5dB, 10dB, 15dB, 20dB* and *mixed* were applied; 48 data triplets have been generated for each. The metrics of interest in objective testing were SI-SDR and PESQ, which are commonly used for AEC tasks. In contrast to Section 4.5, the STOI metric was not used as it was found to be disturbed by the presence of multiple voice signals.

Subjective evaluation was conducted with the same 20 volunteers mentioned in Section 4.5. The test set for subjective evaluation consisted of 6 data triplets of 5 seconds in length. The MUSHRA-like test described in Section 4.5 was performed, with s as hidden reference and the noisy signal y as lower reference point. The other test conditions were the FullSubNet baseline and the proposed model.

Chapter 6

Results & Discussion

In this section, results of our work are presented and discussed. This is done task by task. For each task, the results will be presented followed by a short interpretation. The evaluation methodology to obtain results is explained in sections 4.5 and 5.6. Subjective evaluations were conducted with a population of 20 volunteers, of which all have answered our user study and returned their results.

Metrics used for evaluation are:

SI-SDR Scale-Invariant Signal-to-Distortion Ratio: An adaption of the SNR metric that is robust to scaling errors [54], which is measured in dB.

PESQ Perceptual Evaluation of Speech Quality: An objective metric originally designed to evaluate quality of speech transmitted through telecommunication networks [55]. Although first published in 2001, its efficacy on audio coding and source separation has recently been demonstrated [46]. The value range can vary between implementations. In our case, values lie between 1.02 and 4.56 (worst to best).

STOI Short-Time Objective Intelligibility: This objective measure is highly correlated with subjective intelligibility of human speech [47]. Values lie between 0 and 1 (worst to best).

ACR Absolute Category Rating: A subjective rating of audio quality within the categories *Bad, Poor, Fair, Good, Excellent*, corresponding to values from 1 to 5.

BAQ Basic Audio Quality: A subjective judgement on differences between a reference recording and a test item [52]. In this work, BAQ was measured between the attributes *Very Different* and *No Difference*, on a scale from 0 to 100. Please refer to Section 4.5 for details.

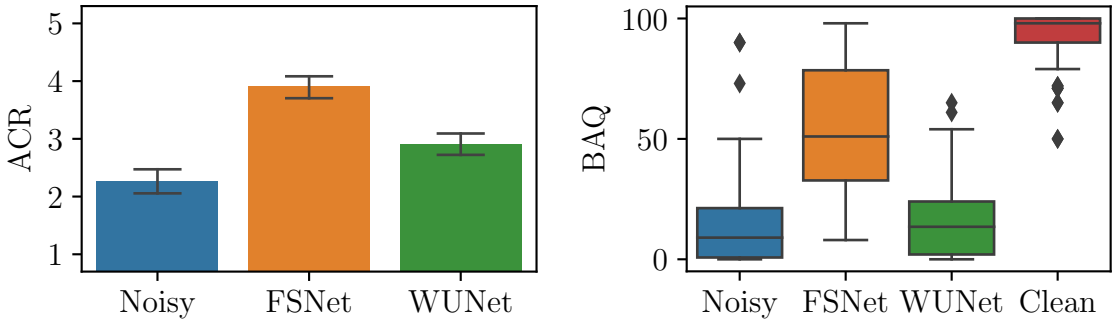
6.1 Denoising

Starting with the task of Denoising, Table 5 shows the results for objective evaluation on a synthesised test set, on metrics PESQ, STOI and SI-SDR. Information on the nature of this dataset is provided in Section 4.5.

Model	PESQ	STOI	SI-SDR
Noisy	1.91	0.72	11.41
Wave-U-Net	2.29	0.73	16.67
FullSubNet	**3.23**	**0.79**	**19.72**

Table 5: Objective evaluation results for denoising: Measured on a synthesised test set, enhanced with Wave-U-Net and FullSubNet. Noisy signal evaluated for reference.

Figure 7 shows results for subjective evaluations on real user recordings (described in Section 3.2) as well as the aforementioned synthesised test set. The real-world data and synthesised data had been judged on ACR (see Figure 3) and BAQ (see Figure 4) metrics, respectively. We can observe that both Wave-U-Net and FullSubNet seem to successfully remove noise. This observation is affirmed mainly by results on the PESQ, SI-SDR and ACR metric. However, only FullSubNet seems to consistently achieve quality improvements on all given combinations of test set and metric. Given its results on STOI and BAQ, such a property could be doubted for Wave-U-Net. However, the BAQ results show a flooring effect with positive skewness for results on noisy and Wave-U-Net enhanced signals. Similarly, the category for clean signals was affected with a ceiling effect. This can lead to a violation of the property of



(a) (b)

Figure 7: User evaluation of the denoising task. **(a):** Absolute Category Rating on real user recordings. **(b):** Basic Audio Quality on synthesised recordings (similar to MUSHRA). Conditions (with label): Noisy; FullSubNet (FSNet); Wave-U-Net (WUNet); Clean.

normality even in a well-balanced MUSHRA testing setup [56]. The high correlation that we experience between noisy and Wave-U-Net enhanced signals may likely be caused by such an effect, impeding a direct comparison between the two categories. The most salient observation is, that, by all measures, FullSubNet achieves the best results on this specific task of denoising user singing voice recordings.

6.2 Leakage Removal

For the task of leakage removal, Table 6 shows the results for objective evaluation on multiple synthesised test sets, on metrics PESQ and SI-SDR. The test sets were all synthesised with the same methodology, but given different SNR target levels. Please refer to Section 5.6 for more information. Figure 8 shows results for subjective evaluations on a synthesised test set. The test set is described in Section 5.6. Performance had been judged on the BAQ metric (see Figure 8). Figure 9 shows exemplary results as spectrograms on one data point of the synthesised test set.

We can observe that, generally, both the proposed model and baseline model seem to succeed at enhancing subjective and objective quality on the test set. While, for the proposed model, this becomes very clear by all measures taken, improvements achieved by the baseline model tend do be insignificant in some occasions. This is especially true for the 0dB SNR category, which both models were not trained

Metric	Model	0dB	5dB	10dB	15dB	20dB	mixed
SI-SDR	Noisy	0.00	4.99	12.07	15.00	21.61	13.04
	Baseline	2.10	12.13	15.03	18.45	22.30	16.41
	Proposed	**10.91**	**22.27**	**22.86**	**25.21**	**29.51**	**23.05**
PESQ	Noisy	1.50	2.25	2.24	2.45	3.16	2.65
	Baseline	1.52	2.34	2.52	2.98	3.63	2.91
	Proposed	**2.30**	**3.37**	**3.77**	**4.14**	**4.35**	**3.88**

Table 6: Objective evaluation results for leakage removal: Measured on synthesised test sets of various SNR levels and one with mixed SNR; enhanced with the baseline FullSubNet architecture and its proposed adaption. The noisy signal was evaluated for reference.

for (they were trained on SNRs of 6 dB to 20 dB). In subjective testing, the aforementioned flooring and ceiling effects are noticeable, but it can be stated that the proposed model's outputs are closer to the clean reference than to the noisy input. The opposite holds true for the baseline model.

Spectrograms in Figure 9 indicate that the additional model input improves the removal of leakage, especially in places where the foreground voice is currently silent. Frequencies that are particularly close to the harmonics of the foreground signal seem to pose a problem for both models. They are reduced, but tend to yield new artifacts for both models. This effect is less noticeable for the proposed model.

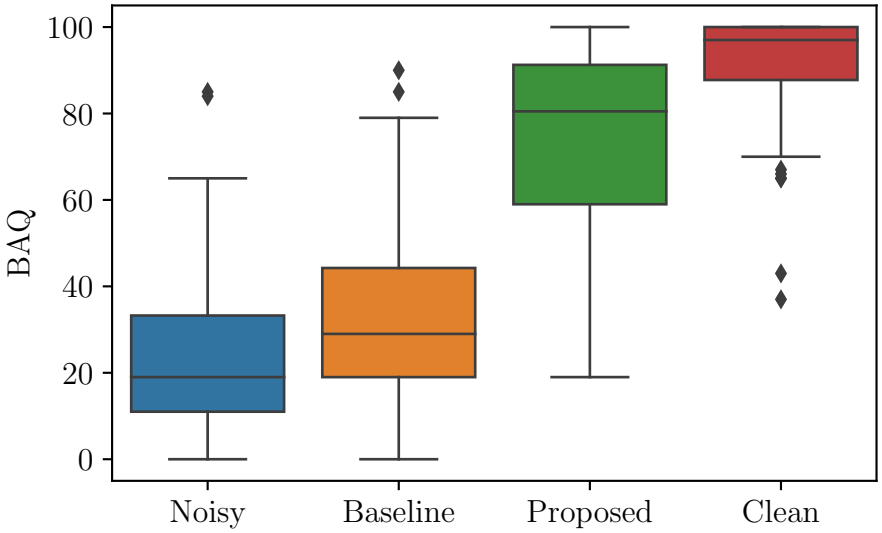


Figure 8: User evaluation of the leakage removal task: Basic Audio Quality on synthesised recordings. Conditions correspond to Section 5: Noisy and clean references were used to compare the proposed model against a baseline model.

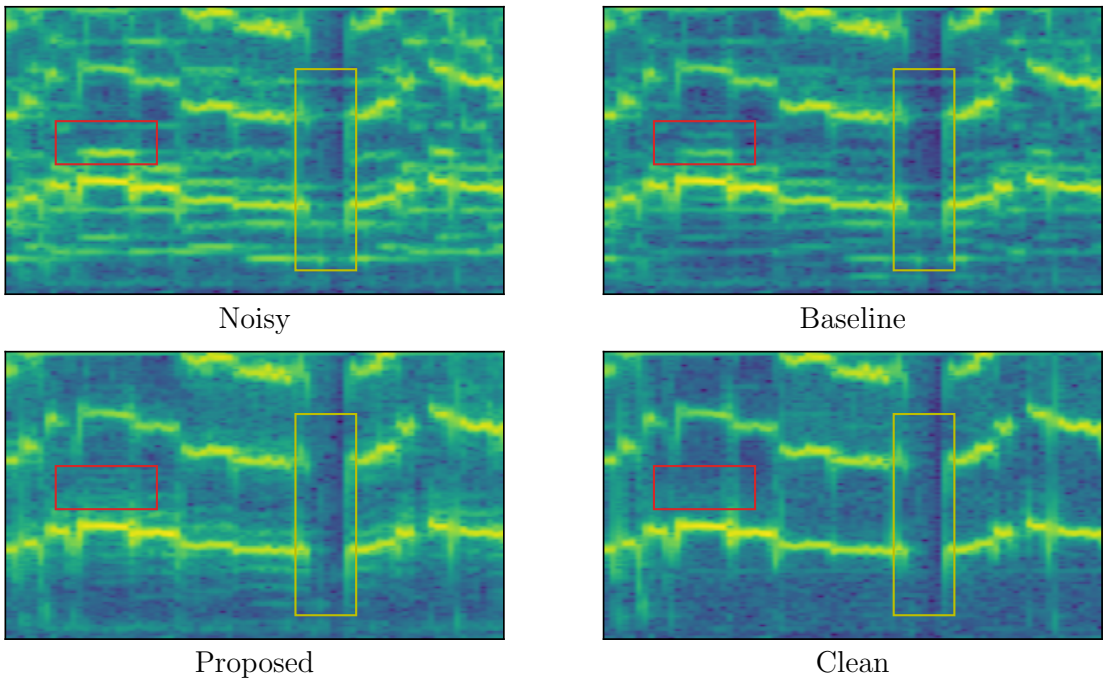


Figure 9: Four spectrogram excerpts for leakage removal: Noisy and clean references in comparison to enhancement with the proposed model and baseline model. **Red:** Highlighted differences in harmonic filtering and artifacts. A close look reveals that both models introduce a new harmonic peak that does not exist in the noisy signal. **Yellow:** Highlighted differences for an unvoiced scenario.

Chapter 7

Conclusions

7.1 Conclusions

In this thesis, we have studied neural network approaches for speech denoising in their adaption to singing enhancement tasks. These include denoising and leakage removal of user recordings in the context of a choir rehearsal software. We have found that the denoising capabilities of Wave-U-Net and FullSubNet on speech translate effectively to the domain of singing voice. Clearly, the best results on this task were achieved by FullSubNet. Furthermore, we have proposed and evaluated a modification of FullSubNet for leakage removal. Introduction of an additional model input with reference audio resulted in significant improvements over a blind separation approach.

7.2 Contributions

We have elaborated and implemented a training methodology with which we can effectively train and compare FullSubNet and Wave-U-Net architectures on denoising of singing recordings (see Chapter 4). Furthermore, we have implemented a modification of FullSubNet that allows for effective leakage removal (see Chapter 5). A training methodology for leakage removal with this network was elaborated and implemented. Both training methodologies could be used to train other neural net-

work architectures in their respective domain. Next to these contributions in singing enhancement, the modified FullSubNet architecture could present a novel approach in AEC, which is yet to be examined (see Section 7.3).

7.3 Future Work

At this point, we have explored enhancement of user singing recordings on the aspects of leakage and background noise. Generally, other degradations like room coloration/reverberation or microphone distortion can be subject to further research. Furthermore, the separation of tasks could be repealed. The network presented in [2] examplifies, how AEC and denoising are performed simultaneously. To target a combination of coloration, reverberation and background noise degradation types, a dataset like DR-VCTK [15] with real-world device-recorded audio could be used for training.

In this work, the leakage removal task was trained to handle perfectly synchronized pairs of reference audio and noisy recording. In a real-world scenario, this can only be ensured to a certain extent, e.g. by compensating for a device's round trip time and/or performing cross-correlation on the audio pair after recording. To effectively cope with leakage in a real-world setup, the model would need to be trained on cases with a slight delay between reference audio input and noisy recording or vice versa. Furthermore, the leakage removal task has not been evaluated on real-world data. This could be done on user recordings coupled with information on the reference audio used.

Given the similarities of leakage removal and AEC, the proposed modified FullSubNet architecture could be evaluated against other AEC systems on leakage removal. Likewise, it could be trained and evaluated against other AEC models in the domain of AEC.

7.4 Reproducibility

All code needed to reproduce our experiments is available on online[1]. This resource includes the following materials:

- Code for Chapter 3: Scripts for preprocessing and preselection of audio data
- Audio for Chapter 3.4: Laptop noise profiles retrieved from Freesound
- Code for Chapter 4 on denoising: An identical training process for application with both Wave-U-Net and FullSubNet. This includes data augmentation and objective evaluation methods.
- Code for Chapter 5 on leakage removal: Implementation of the proposed architecture together with the specified training process. This includes data augmentation and objective evaluation methods.
- Code for subjective testing: Modifications on the BeaqleJS framework according to Section 4.5.
- Results: Audio examples and trained weight configurations for all assessed combinations of models and tasks

However, due to the use of proprietary datasets, results are not entirely reproducible. The training process for denoising relies on singing data from Voctro Labs. This could be replaced by the similar data from VocalSet (to compare both datasets, please refer to Section 3.1). Similarly, the training process for leakage removal relies on multitrack choir singing data from Voctro Labs. For the unaligned case from Section 5.1, this data could simply be replaced by data from a non-multitrack singing resource like, again, VocalSet. However, data for the aligned scenario must be derived from multitrack resources. Two such publications are the Dagstuhl ChoirSet [57] and the Choral Singing Dataset [58]. In combination, they would provide five multitrack-recorded choir pieces sung by 29 singers of four voice types. A comprehensive list of more such resources is provided in [57].

[1] wimmerb: Quality Enhancement of Overdub Singing Recordings `https://github.com/wimmerb/singing-quality-enhancement`

List of Figures

1 Source separation as performed by Wave-U-Net [21]. Dashed arrows depict so-called *skip connections*. 9

2 The FullSubNet architecture for denoising of speech signals [20]. It represents a fusion of a full-band model G_{full} and a sub-band model G_{sub}, which both rely on LSTM layers and dense layers. 11

3 ACR testing: Participants were asked to listen through all **Test Items** and then rate them based on perceived quality and absence of noise. Note that the slider 'snaps' to discrete positions according to the given categories. 28

4 MUSHRA-like testing: After listening through the **Reference** and all **Test Items**, participants were asked to rate any and all detected differences between the reference and the test item on a scale from *Very Different* (0) to *No Difference* (100). 29

5 Sampling scenarios: **(a) Aligned:** A voiced subsample (yellow) of the Tenor voice is found and used as target audio s; material for reference audio (black window, all voices) must be from the same piece and cover the same time frame as s. **(b) Random:** Voiced subsamples are found for random excerpts of all voice types; one of them is chosen as foreground audio (yellow), all of them (grey, yellow) are eligible as material for the reference audio. 33

6 The proposed modified FullSubNet architecture. It is fed with the magnitude spectral features $\widetilde{X}$ and $\widetilde{Y}$. This figure shows the workflow for the t th STFT frame. The second line in each rectangle describes the dimensions of the data at the current stage. For example, "1 (F)" represents one F-dimensional vector. "F $(4N+3)$" represents F independent $(4N+3)$-dimensional vectors. Grey boxes indicate submodels G_{full} and G_{sub}. 37

7 User evaluation of the denoising task. **(a):** Absolute Category Rating on real user recordings. **(b):** Basic Audio Quality on synthesised recordings (similar to MUSHRA). Conditions (with label): Noisy; FullSubNet (FSNet); Wave-U-Net (WUNet); Clean. 44

8 User evaluation of the leakage removal task: Basic Audio Quality on synthesised recordings. Conditions correspond to Section 5: Noisy and clean references were used to compare the proposed model against a baseline model. 46

9 Four spectrogram excerpts for leakage removal: Noisy and clean references in comparison to enhancement with the proposed model and baseline model. **Red:** Highlighted differences in harmonic filtering and artifacts. A close look reveals that both models introduce a new harmonic peak that does not exist in the noisy signal. **Yellow:** Highlighted differences for an unvoiced scenario. 46

List of Tables

1 Segmentation of singing and speech data; raw data for experiments on denoising. Source labels according to Section 3.1. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$. 16

2 Segmentation of choir singing data; raw data for experiments on leakage removal. Source labels according to Section 3.2. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours. 18

3 Segmentation of RIR and MIR data; raw data for experiments in denoising (RIR) and leakage removal (RIR+MIR). Source labels according to Section 3.1. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$. 20

4 Segmentation of noise data; raw data for experiments on denoising. Source labels according to Section 3.4. Preprocessing steps are indicated. **#:** number of files; **len:** total length of audio in hours; **bal:** over/undersampling proportion $\frac{segmented}{preprocessed}$. 22

5 Objective evaluation results for denoising: Measured on a synthesised test set, enhanced with Wave-U-Net and FullSubNet. Noisy signal evaluated for reference. 43

6 Objective evaluation results for leakage removal: Measured on synthesised test sets of various SNR levels and one with mixed SNR; enhanced with the baseline FullSubNet architecture and its proposed adaption. The noisy signal was evaluated for reference. 45

Bibliography

[1] Zwicker, E., Flottorp, G. & Stevens, S. S. Critical Band Width in Loudness Summation. *J. Acoust. Soc. Am.* **29**, 5 (1957).

[2] Westhausen, N. L. & Meyer, B. T. Acoustic Echo Cancellation with the Dual-Signal Transformation LSTM Network. In *ICASSP*, 7138–7142 (IEEE, 2021).

[3] Xia, B. & Bao, C. Speech enhancement with weighted denoising auto-encoder. In *INTERSPEECH*, 3444–3448 (ISCA, 2013).

[4] Isik, U. *et al.* PoCoNet: Better Speech Enhancement with Frequency-Positional Embeddings, Semi-Supervised Conversational Data, and Biased Loss. In *INTERSPEECH*, 2487–2491 (ISCA, 2020).

[5] Luo, Y. & Mesgarani, N. TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation. In *ICASSP*, 696–700 (IEEE, 2018).

[6] Luo, Y. & Mesgarani, N. Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation. *IEEE ACM Trans. Audio Speech Lang. Process.* **27**, 1256–1266 (2019).

[7] Cohen-Hadria, A., Roebel, A. & Peeters, G. Improving singing voice separation using Deep U-Net and Wave-U-Net with data augmentation. In *EUSIPCO*, 1–5 (IEEE, 2019).

[8] Li, Q., Gao, F., Guan, H. & Ma, K. Real-time Monaural Speech Enhancement With Short-time Discrete Cosine Transform. Preprint at `https://arxiv.org/abs/2102.04629` (2021).

[9] Zhao, Y., Wang, D., Xu, B. & Zhang, T. Monaural Speech Dereverberation Using Temporal Convolutional Networks With Self Attention. *IEEE ACM Trans. Audio Speech Lang. Process.* **28**, 1598–1607 (2020).

[10] Panayotov, V., Chen, G., Povey, D. & Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In *ICASSP*, 5206–5210 (IEEE, 2015).

[11] Thiemann, J., Ito, N. & Vincent, E. The Diverse Environments Multi-Channel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings. *J. Acoust. Soc. Am.* **133**, 3591 (2013).

[12] Jeub, M., Schäfer, M. & Vary, P. A binaural room impulse response database for the evaluation of dereverberation algorithms. In *DPS*, 1–5 (IEEE, 2009).

[13] Fujimura, T., Koizumi, Y., Yatabe, K. & Miyazaki, R. Noisy-target Training: A Training Strategy for DNN-based Speech Enhancement without Clean Speech. Preprint at `https://arxiv.org/abs/2101.08625` (2021).

[14] Mysore, G. J. Can we Automatically Transform Speech Recorded on Common Consumer Devices in Real-World Environments into Professional Production Quality Speech? - A Dataset, Insights, and Challenges. *IEEE Signal Process. Lett.* **22**, 1006–1010 (2015).

[15] Sarfjoo, S. S. *et al.* Transformation of low-quality device-recorded speech to high-quality speech using improved SEGAN model. Preprint at `https://arxiv.org/abs/1911.03952` (2019).

[16] Sondhi, M. M. An adaptive echo canceller. *Bell Syst. tech.* **46**, 497–511 (1967).

[17] Peng, R., Cheng, L., Zheng, C. & Li, X. ICASSP 2021 Acoustic Echo Cancellation Challenge: Integrated Adaptive Echo Cancellation with Time Alignment

and Deep Learning-Based Residual Echo Plus Noise Suppression. In *ICASSP*, 146–150 (IEEE, 2021).

[18] Zhang, H. & Wang, D. Deep Learning for Acoustic Echo Cancellation in Noisy and Double-Talk Scenarios. In *INTERSPEECH*, 3239–3243 (ISCA, 2018).

[19] Ma, L., Huang, H., Zhao, P. & Su, T. Acoustic Echo Cancellation by Combining Adaptive Digital Filter and Recurrent Neural Network. Preprint at `https://arxiv.org/abs/2005.09237` (2020).

[20] Hao, X., Su, X., Horaud, R. & Li, X. Fullsubnet: A Full-Band and Sub-Band Fusion Model for Real-Time Single-Channel Speech Enhancement. In *ICASSP*, 6633–6637 (IEEE, 2021).

[21] Stoller, D., Ewert, S. & Dixon, S. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. In *ISMIR*, 334–340 (2018).

[22] Guimarães, H. R., Nagano, H. & Silva, D. W. Monaural speech enhancement through deep wave-u-net. *Expert Syst. Appl.* **158**, 113582 (2020).

[23] Macartney, C. & Weyde, T. Improved Speech Enhancement with the Wave-U-Net. Preprint at `https://arxiv.org/abs/1811.11307` (2018).

[24] Bedoya Molina, J. D. *Speech Enhancement using the Wave-U-Net with Spectral Losses*. Master's thesis, Universitat Pompeu Fabra (2020).

[25] Kim, J. & Chang, J. Attention Wave-U-Net for Acoustic Echo Cancellation. In *INTERSPEECH*, 3969–3973 (ISCA, 2020).

[26] Kolbæk, M., Tan, Z., Jensen, S. H. & Jensen, J. On Loss Functions for Supervised Monaural Time-Domain Speech Enhancement. *IEEE ACM Trans. Audio Speech Lang. Process.* **28**, 825–838 (2020).

[27] Fu, S., Liao, C. & Tsao, Y. Learning With Learned Loss Function: Speech Enhancement With Quality-Net to Improve Perceptual Evaluation of Speech Quality. *IEEE Signal Process. Lett.* **27**, 26–30 (2020).

[28] Pandey, A. & Wang, D. On Adversarial Training and Loss Functions for Speech Enhancement. In *ICASSP*, 5414–5418 (IEEE, 2018).

[29] Jansson, A. *et al.* Singing Voice Separation with Deep U-Net Convolutional Networks. In *ISMIR*, 745–751 (2017).

[30] Li, X. & Horaud, R. Online Monaural Speech Enhancement Using Delayed Subband LSTM. In *INTERSPEECH*, 2462–2466 (ISCA, 2020).

[31] Williamson, D. S., Wang, Y. & Wang, D. Complex Ratio Masking for Monaural Speech Separation. *IEEE ACM Trans. Audio Speech Lang. Process.* **24**, 483–492 (2016).

[32] Reddy, C. K. A. *et al.* The INTERSPEECH 2020 Deep Noise Suppression Challenge: Datasets, Subjective Testing Framework, and Challenge Results. In *INTERSPEECH*, 2492–2496 (ISCA, 2020).

[33] Wilkins, J., Seetharaman, P., Wahl, A. & Pardo, B. VocalSet: A Singing Voice Dataset. In *ISMIR*, 468–474 (2018).

[34] Veaux, C., Yamagishi, J. & MacDonald, K. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92). `https://doi.org/10.7488/ds/2645` (2019).

[35] TROMPA: Participatory concert. `https://trompamusic.eu/index.php/choir-singers/cantoria-esmuc--concert`. Accessed: 2021-08-31.

[36] Eaton, J., Gaubitch, N. D., Moore, A. H. & Naylor, P. A. The ACE challenge - Corpus description and performance evaluation. In *WASPAA*, 1–5 (IEEE, 2015).

[37] Jeub, M. *et al.* Do We Need Dereverberation for Hand-Held Telephony? In *Int. Congr. Acoust.*, 1–7 (2010).

[38] REVERB challenge: Welcome to the REVERB challenge. `https://reverb2014.dereverberation.com/`. Accessed: 2021-08-31.

[39] REVERB challenge: Details on challenge data. `https://reverb2014.dereverberation.com/data.html`. Accessed: 2021-08-31.

[40] Ko, T., Peddinti, V., Povey, D., Seltzer, M. L. & Khudanpur, S. A study on data augmentation of reverberant speech for robust speech recognition. In *ICASSP*, 5220–5224 (IEEE, 2017).

[41] AudioLabs Erlangen: RIR-Generator. `http://micirp.blogspot.com/p/about-micirp.html`. Accessed: 2021-08-31.

[42] Microphone Impulse Response Project: About MicIRP. `http://micirp.blogspot.com/p/about-micirp.html`. Accessed: 2021-08-31.

[43] Freesound. `https://freesound.org`. Accessed: 2021-08-31.

[44] haoxiangsnr: Wave-U-Net-for-Speech-Enhancement. `https://github.com/haoxiangsnr/Wave-U-Net-for-Speech-Enhancement`. Accessed: 2021-08-31.

[45] haoxiangsnr: FullSubNet. `https://github.com/haoxiangsnr/FullSubNet`. Accessed: 2021-08-31.

[46] Torcoli, M., Kastner, T. & Herre, J. Objective Measures of Perceptual Audio Quality Reviewed: An Evaluation of Their Application Domain Dependence. *IEEE ACM Trans. Audio Speech Lang. Process.* **29**, 1530–1541 (2021).

[47] Taal, C. H., Hendriks, R. C., Heusdens, R. & Jensen, J. An Algorithm for Intelligibility Prediction of Time-Frequency Weighted Noisy Speech. *IEEE Trans. Speech Audio Process.* **19**, 2125–2136 (2011).

[48] Gupta, C., Li, H. & Wang, Y. Perceptual evaluation of singing quality. In *APSIPA*, 577–586 (IEEE, 2017).

[49] Kraft, S. & Zölzer, U. BeaqleJS: HTML5 and JavaScript based Framework for the Subjective Evaluation of Audio Quality. In *LAC*, 26 (2014).

[50] HSU-ANT: BeaqleJs. `https://github.com/HSU-ANT/beaqlejs`. Accessed: 2021-08-31.

[51] ITU-T. Recommendation P.808: Subjective evaluation of speech quality with a crowdsourcing approach. Tech. Rep., ITU (2021).

[52] ITU-R. Recommendation BS.1534-3: Method for the subjective assessment of intermediate quality level of audio systems. Tech. Rep., ITU (2015).

[53] Zieliński, S., Hardisty, P., Hummersone, C. & Rumsey, F. Potential biases in MUSHRA listening tests. *J. Audio Eng. Soc.* **123**, 7179 (2007).

[54] Roux, J. L., Wisdom, S., Erdogan, H. & Hershey, J. R. SDR - Half-baked or Well Done? In *ICASSP*, 626–630 (IEEE, 2019).

[55] Rix, A. W., Beerends, J. G., Hollier, M. P. & Hekstra, A. P. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *ICASSP*, 749–752 (IEEE, 2001).

[56] Mendonça, C. & Delikaris-Manias, S. Statistical tests with MUSHRA data. *J. Audio Eng. Soc.* **144**, 10006 (2018).

[57] Rosenzweig, S. *et al.* Dagstuhl choirset: A multitrack dataset for MIR research on choral singing. *Trans. Int. Soc. Music. Inf. Retr.* **3**, 98–110 (2020).

[58] Cuesta, H., Gómez, E., Martorell, A. & Loáiciga, F. Choral Singing Dataset. `https://doi.org/10.5281/zenodo.1286570` (2018).