Speech Separation in Noisy Reverberant Acoustic Environments

William Ravenscroft

Primary Supervisor: Prof. Thomas Hain

Secondary Supervisor: Dr. Stefan Goetze

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy in Computer Science

in the

Department of Computer Science

January 24, 2023

Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: William Ravenscroft

Date: 24/01/2023

Acknowledgments

Before proceeding with this thesis, I would like to acknowledge the efforts of my supervisors and colleagues who helped shape this work.

First and foremost, I would like to thank my supervisors, Prof. Thomas Hain and Dr. Stefan Goetze. Without their continued insights, expertise, editorial contributions and reviews of my work, this thesis might have much less to offer the reader. In addition, I would like to thank the Speech and Language Technology Centre for Doctoral Training at The University of Sheffield for providing the funding and opportunity to undertake the work presented in this thesis.

Secondly, I want to acknowledge Dr Mark Fuhs and Dr Monika Woszczyna from 3M Health Information Systems, who helped fund this project. I am very grateful for their help arranging for me to go and work at 3M towards the end of my project. This provided the opportunity to work on unique problems and helped form the basis of work in Chapter 10. Additionally, I would like to thank Anurag Chowdhury, Mohammad Soleymanpour and Abhinav Misra from 3M for their supervision and input, which helped to shape some of the work in Chapter 10.

Finally, I would like to express my gratitude to my family for their unwavering support throughout my PhD journey, especially during the COVID-19 pandemic, which hit during the early phases of my project.

Abstract

Speech separation remains a vital area of research for many modern technologies. The ubiquitous spread of deep neural networks (DNNs) in many areas of signal processing (SP) and machine learning (ML) research over the past decade has resulted in significant improvements in single-channel speech separation on a number of benchmark datasets, particularly for anechoic speech separation. Speech separation and enhancement in noisy and reverberant acoustic environments remains challenging, particularly in single-channel models. This is the area this thesis primarily focuses on.

In recent years, the temporal convolutional network (TCN) has been a popular sequence model, particularly for speech separation. Arguably, the most popular of such models, the convolutional time-domain audio separation network (Conv-TasNet) model, is analysed in the second part of this thesis for how well it performs in dereverberation tasks with a view to improving the combined speech enhancement and separation performance. It is shown that the network's optimal receptive field varies depending on the reverberation time of the data being dereverbed. Further to this, the weighted multi-dilation temporal convolutional network (WD-TCN) is proposed as an improvement to the TCN and is shown to give consistent improvements in dereverberation and separation tasks using various parts of a noisy reverberant speech mixtures corpus known as WHAMR. The WD-TCN allows the TCN to dynamically adjust the focus of its receptive field to focus on more or less localized temporal context. An alternative improvement which adjusts the receptive field itself at the frame level is also proposed. This model is referred to as the deformable temporal convolutional network (DTCN) as it uses deformable convolution in place of vanilla convolution which allows the convolutional kernel to have a fully adaptive receptive field by using a linear interpolation function.

The TCN model is typically considered a computationally lightweight model compared to other DNN models; however, it doesn't model global sequence information beyond the mean and variance. Transformer networks that process high-level global context have led to breakthroughs in performance across various areas of machine learning, including speech separation. The third part of this thesis starts by looking at how incorporating multihead attention (MHA), the main mechanism used in Transformers, can be leveraged to encode

global context in the encodings of the Conv-TasNet model. A multihead self-attention (MHSA) encoder is proposed and shows notable improvements in performance across a number of acoustic conditions. A series of MHA decoders are also proposed with some but less notable improvements. Following on from this, a study on training signal lengths (TSLs) is performed on the dual-path (DP) Transformer-based speech separation model known as SepFormer. This study demonstrated that the choice of TSL is non-trivial and depends on the signal length distribution of the training data in particular but also the data segmentation strategy and the model architecture. Following on from this study, a DNN model referred to as the time domain Conformer (TD-Conformer) is proposed. This model is proposed as a more optimal analogue of the DP Transformer model, particularly for noisy reverberant speech separation. The Conformer is also notably more efficient to train than the DP Transformer in the context of the work presented in this thesis. It is shown that the TD-Conformer model, which uses convolution in place of Transformers for processing local context leads to improved performance for combined speech separation and enhancement with a reduction in computational complexity on shorter signal lengths. Evidence is also provided that this is likely due to a large model size resulting in improved generalization properties of the network. A further chapter is dedicated to analysing the differences between DP models and the conformer models by proposing a model which combines the two, referred to as the convolutional separation Transformer (ConSepT) model. The analysis of this model highlights the strength of the larger model size of the Conformer models in helping the model to generalize and further demonstrates some of the redundancies in the SepFormer model.

In the final part of this thesis, the application of speech separation to multi-speaker automatic speech recognition (ASR) is explored. A novel method is proposed to fine-tune the TD-Conformer model for ASR in a transcription-free fashion. This is done by leveraging the embeddings of pre-trained ASR models and computing the differences between them. The proposed method is shown to result in a reduction in word error rate (WER) on a dataset of real doctor-patient conversations.

Contents

List of Symbols

Constants & Hyperparameters

Γ	Maximum norm in the gradient clipping algorithm, cf. Algorithm 2
∇	Gradient symbol used in gradient clipping definition
Φ	Number of possible speaker permutations
A	Generalized feature dimension of an attention head, e.g. $\frac{D}{G_{\text{heads}}}$
A_k	Feature dimension of attention heads for the key input $\mathbf{K}$, i.e. $\frac{D_k}{G_{\text{heads}}}$
A_q	Feature dimension of attention heads for the query input $\mathbf{Q}$, i.e. $\frac{D_q}{G_{\text{heads}}}$
A_v	Feature dimension of attention heads for the value input $\mathbf{V}$, i.e. $\frac{D_v}{G_{\text{heads}}}$
B	Number of output channels in bottleneck (Conv-TasNet, WD-TCN, DTCN & TD-Conformer)
C	Number of speakers (or sources)
C'	Number of rows in non-negative basis matrix $\mathbf{W}_c$
D	Number of input features or channels
D_k	Feature dimension size of $\mathbf{K}$
D_q	Feature dimension size of $\mathbf{Q}$
D_v	Feature dimension size of $\mathbf{V}$
E	Number of output features or channels or filters
f	Dilation factor of a convolutional layer
G	Number of groups, in group normalization of a convolutional layer
G_{split}	Signal splitting hyperparameter, cf. Section 7.3.1
G_{heads}	Number of attention heads in a multi-head attention layer, cf. (2.65)
H	Number of hidden features inside a convolutional block (Conv-TasNet, WD-TCN, DTCN)
J	Block length (or kernel size) of TasNet encoder
K	Stride
L	Length of an arbitrary sequence

L_{lim}	Training signal length limit measured in samples, cf. Chapter 7
$L_{\mathbf{x}}$	The length of $x[i]$ encoded in block lengths J for 50% overlap, cf. Section 2.9.1
L_U	Related to the chunking in the SepFormer model, cf. Figure 2.5
L_x	The length of time domain signal $x[i]$ in samples
M	Number of microphones
N	Output dimension of encoder network in any TasNet model architecture
P	Kernel size
Q	Number of parallel depthwise convolutions in a WD-TCN network block, cf. (4.2)
R	Number of repeats of stacks of convolutional blocks or times a sequence is passed through the same stack when using shared weights (Conv-TasNet, WD-TCN, DTCN)
R_{conf}	Number of Conformer layers in the TD-Conformer model
R_{DPT}	Number of dual-path Transformer layers in the SepFormer model
R_{TD}	Number of Transformer decoder layers in a Transformer network, cf. Figure 2.5
R_{TE}	Number of Transformer encoder layers in a Transformer network, cf. Figure 2.5
S	Number of subsampling layers in TD-Conformer model
T	Final time index of an audio segment
T_{lim}	Training signal length limit in seconds (s)
U	Chunk size of SepFormer, cf. Figure 2.9
V	Number of repeats of the dual-path Transformer network in SepFormer, cf. Figure 2.5
W	Batch dimension
X	Number of convolutional blocks or layers in a stack ((2.45), Conv-TasNet, WD-TCN, DTCN)
Y	Number of dimension, briefly used in Section 2.5.6
Z	Output feature dimension of SepFormer mask estimation network bottleneck layer

Functions

$\mathcal{A}(\cdot)$	Dot-product attention function
$\mathcal{B}(\cdot)$	Parameter count function
$\mathcal{B}_{\text{Conv1D}}(\cdot)$	Parameter count function
$\mathcal{B}_{\text{D-Conv}}(\cdot)$	Parameter count of D-conv layer
$\mathcal{B}_{\text{DS-Conv}}(\cdot)$	Parameter count of DS-Conv layer
$\mathcal{B}_{\text{P-Conv}}(\cdot)$	Parameter count of P-Conv layer

$\mathcal{C}(\cdot)$	Convolutional layer
$\mathcal{C}_{\text{1D}}(\cdot)$	1D convolutional layer
$\mathcal{C}_{\text{D-Conv}}(\cdot)$	1D depthwise convolutional layer
$\mathcal{F}(\cdot)$	Feedforward neural network layer
$\mathcal{G}(\cdot)$	Global layer normalization
$\mathcal{G}_{\text{cLN}}(\cdot)$	Cumulative layer normalization
$\mathcal{G}_{\text{CN}}(\cdot)$	Channel-wise layer normalization
$\mathcal{G}_{\text{GN}}(\cdot)$	Group normalization
$\mathcal{H}(\cdot)$	Generic activation function
$\mathcal{H}_{\text{enc}}(\cdot)$	Encoder activation function
$\mathcal{H}_{\text{GLU}}(\cdot)$	GLU activation function
$\mathcal{H}_{\text{PReLU}}(\cdot)$	PReLU activation function
$\mathcal{H}_{\text{ReLU}}(\cdot)$	ReLU activation function
$\mathcal{H}_{\text{SiLU}}(\cdot)$	SiLU activation function
$\mathcal{H}_{\text{softmax}}(\cdot)$	Softmax activation function
$\mathcal{H}_{\sigma}(\cdot)$	Sigmoid activation function
$\mathcal{K}_{\text{D-Conv}}(\cdot)$	A single depthwise convolution operation kernel
$\mathcal{K}_{\text{DD-Conv}}(\cdot)$	A single deformable depthwise convolution operation kernel
$\mathcal{L}(\cdot)$	Arbitrary loss function
$\mathcal{L}_{\text{LD}}(\cdot)$	Logit difference loss function
$\mathcal{L}_{\text{SISDR}}(\cdot)$	Scale-invariant signal-to-distortion loss function, cf. (2.72)
$\mathcal{L}_{\text{ST-MAS}}(\cdot)$	Absolute difference of short-time absolute magnitude spectral loss function
$\mathcal{L}_{\text{ST-MSE}}(\cdot)$	Short-time spectral mean square error loss function
$\mathcal{L}_{\text{TD-L1}}(\cdot)$	L1 time domain loss function
$\mathcal{L}_{\text{TD-L1}}(\cdot)$	L2 time domain loss function
$\mathcal{L}_{\text{WE}}(\cdot)$	Whisper encoder loss function
$\mathcal{M}(\cdot)$	Multihead attention layer
$\mathcal{P}(\cdot)$	Probability function
$\mathcal{Q}(\cdot)$	Positional encoding function, cf. (2.66)
$\mathcal{R}(\cdot)$	Receptive field function
$\mathcal{R}_{\text{TCN}}(\cdot)$	Total receptive field of a Conv-TasNet style temporal convolutional network, measured in input frames
$\mathcal{R}_{\text{total}}(\cdot)$	Total aggregated receptive field of a stack of arbitrary convolutional layers in input frames

$\mathcal{R}_{\mathrm{TT}}(\cdot)$	Total receptive field of a Conv-TasNet style temporal convolutional network, measured in time (seconds)
$\mathcal{S}(\cdot)$	1D depthwise-separable convolutional layer
$\mathcal{T}(\cdot)$	Time-complexity function
$\mathcal{Z}(\cdot)$	Recurrent neural network layer
$\mathrm{Var}\,(\cdot)$	Variance

Miscellaneous

$'$	Symbol used to ease differentiating between different variables with similar properties, i.e. x' is not $x \in \mathbb{R}$ but is also real-valued and of the same dimensionality

Matrices & Tensors

$\hat{\mathbf{S}}$	Spectral representation of estimated speech signal $\hat{s}$
$\mathbf{\Phi}$	Permutation matrix
$\mathbf{W}$	Matrix of encoded feature sequence
$\mathbf{B}$	Basis matrix in non-negative matrix factorization (NMF) model
$\mathbf{S}$	Spectral representation of clean speech signal s
$\mathbf{W}$	Weight matrix in non-negative matrix factorization (NMF) model
$\mathbf{\Theta}$	Generalised matrix of tensor of trainable weights
$\mathbf{\Theta}_{\mathrm{1D}}$	Trainable weights in pointwise convolution layer $D \times P \times E$
$\mathbf{\Theta}_{\mathrm{D\text{-}Conv}}$	Trainable weights in 1D depthwise convolution layer of size $D \times 1 \times E$
$\mathbf{\Theta}_{\mathrm{ff}}$	Trainable weights in a feedforward layer or the feedforward component of a recurrent layer, of shape $E \times E$
$\mathbf{\Theta}_{\mathrm{P\text{-}Conv}}$	Trainable weights in 1D pointwise convolution layer of shape $D \times P \times E$
$\mathbf{\Theta}_{\mathrm{rec}}$	Trainable weights in the recurrent part of a recurrent neural network layer, of shape $D \times E$
$\mathbf{\Theta}_{\mathrm{key},g}$	Trainable weights for the gth"key" projection layer in multihead attention, cf. (2.65)
$\mathbf{\Theta}_{\mathrm{query},g}$	Trainable weights for the gth "query" projection layer in multihead attention
$\mathbf{\Theta}_{\mathrm{value},g}$	Trainable weights for the gth "value" projection layer in multihead attention
$\mathbf{A}$	The mixing matrix of the ICA model
$\mathbf{B}$	The TasNet decoder weights, also described as basis signals in Eq. (2.76) of shape $N \times J$
$\mathbf{D}$	Use in multi-head attention formulation
$\mathbf{E}$	Euclidean distance matrix, used in Algorithm 1
$\mathbf{H}$	Intermediate features of DP Network

$\mathbf{I}$	Identity matrices
$\mathbf{K}$	The *"keys"* feature matrix in the multihead attention layer
$\mathbf{M}$	Mask estimation network output matrix of shape $L_{\mathbf{x}} \times N$
$\mathbf{Q}$	The *"queries"* feature matrix in the multihead attention layer
$\mathbf{S}$	A matrix of time domain clean speech signals
$\mathbf{U}$	TasNet encoder weights of shape $J \times N$
$\mathbf{V}$	The *"values"* feature matrix in the multihead attention layer
$\mathbf{W}$	Encoded feature matrix of shape $L_{\mathbf{x}} \times N$
$\mathbf{X}$	A speech mixture matrix, cf. Eq. (2.9)
$\mathbf{Y}$	Arbitrary feature matrix of size $L \times D$, typically used for defining neural network layers and operations, c.f. Section 2.5.2
$\mathbf{Y}_{\ell,f,P}$	A dilated view of $\mathbf{Y}$ for the ℓth convolution in a 1D convolutional layer of shape $P \times D$
$\mathbf{Z}$	Whitened mixture matrix in ICA model

Number Sets

$\mathbb{C}$	Complex numbers
$\mathbb{N}$	Natural numbers
$\mathbb{N}_0$	Natural numbers including 0
$\mathbb{R}_+$	Positive real numbers only
$\mathbb{R}$	Real numbers

Operators

$(\cdot)^\top$	Matrix or tensor transpose operator
:	Used for denoting integer index ranges for sub-vectors and sub-matrices, i.e. $\mathbf{x}_{4:7} = [\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7]^\top$
$*$	Convolution operator
$\cap$	Joint operator (for probabilities)
$\cup$	Union operator (for probabilities)
$\mathcal{E}(\cdot)$	Expectation operator
$\odot$	Elementwise matrix multiplication/Hadamard product

Scalars & Variables

α	Attentuation factor, cf. the reverberant signal model in (2.4)
β	Bias parameter scalar
ℓ	Frame index used for any sequence of length $L > 1$ dimension
ϵ	Machine precision rounding error correction value

η	Learning rate
$\hat{\phi}$	Predicted permutation index
$\hat{r}$	Cross-correlation estimate
$\hat{s}$	Estimated speech signal
$\hat{s}_c$	The estimated cth speaker signal of C speakers
λ	Trainable parameter in PReLU function
e	The eth position in a vector/matrix of length E. Not to be confused with the exponent e.
μ	Mean value
$\mu_{\mathbf{Y}}$	Mean value of matrix $\mathbf{Y}$
$\mu_{k\leq\ell}$	Aggregated mean value from frame 0 up to ℓ
ν	Noise signal, cf. (2.2)
ω	Angle or used in angular frequency
ϕ	Permutation index
π	Mathematical constant; the ratio of a circle's circumference to its diameter ≈ 3.14159
σ	Standard deviation
$\sigma_{\mathbf{Y}}$	Standard deviation value of matrix $\mathbf{Y}$
$\sigma_{k\leq\ell}$	Aggregated standard deviation value from frame 0 up to ℓ
τ	Time delay
$\tau_{\ell,p}$	The deformable offset at frame ℓ and the pth kernel position
θ	Weight parameter scalar
ν_{dist}	Distortion signal vector
ξ	The number of possible unique training samples for a signal of length L_x and training signal length limit of L_{lim}, cf. (7.2)
ζ_d	Learnable parameter in the PReLU non-linearity function, cf. Section 2.5.4.5
a	Index for a vector of size A
a_q	The qth of Q attention value resulting from the squeeze-and-excite layer in utterance weighted multi-dilation convolution, cf. Eq. (4.2)
$a_{m,c}$	The mth audio channel and cth source in the ICA model, cf. (2.8)
b	Used as an index in Algorithm 1
c	Speaker index, cf. (2.1)
d	Feature index for feature vector of size D, cf. Section 2.5.2
e	Exponent. Not to be confused with e

g	Used as a feature group and attention head index
h	Impulse response
h_c	The room impulse response corresponding to the $c$the speaker
i	Discrete time index
j	Imaginary unit $\sqrt{-1}$ of a complex number
k	Used as an alternate sequence index to ℓ
m	Index for a vector of size M, typically used in a microphone array.
n	Feature index, typically used to differentiate from d
p	Used as a kernel position index, i.e. $p \in \{1, \ldots, P\}$
r	Resultant of the cross-correlation function
s	Clean speech signal
s_c	The cth speaker signal in a mixture of speakers
s_{dir}	Direct path (of speech signal from speaker to receiving microphone), cf. (2.4)
s_{early}	Early reflections of speech, cf. (2.6)
s_{late}	Late reflections of speech, cf. (2.6)
s_{rev}	Reverberant components of speech at microphone, cf. (2.4)
$s_{c'}$	Any speaker signal note equal to the $c$the speaker signal in a mixture of speakers
t	Continuous time index
u	Used as an index in the deformable convolutional kernel formulations
v	Used as an index in Algorithm 1
x	Used to denote any noisy and/or reverberant speech mixture signal, c.f (2.2)
x_{clean}	Used to denote any clean speech mixture signal, c.f (2.1)
x_m	Speech mixture signal picked up at microphone m

Vectors

$\hat{\mathbf{s}}_c$	The cth estimated speech signal vector of C speakers, of size L_x or L_s
$\hat{\mathbf{t}}_c$	Predicted transcription of the cth speaker, cf. Section 2.11
w_ℓ	The ℓth encoded feature vector of input signal $x[i]$
$\boldsymbol{\beta}$	Bias vector of size E for neural network layer with output feature dimension E
$\boldsymbol{\theta}$	Vector of trainable weights
$\boldsymbol{\theta}_{\text{LN}}$	Vector of trainable weights used in normalization layers, of size D, cf. Section 2.5.5
$\mathbf{g}_\ell$	The ℓth output of an RNN layer, of size E
$\mathbf{h}$	Feature vector in SepFormer model of size Z, cf. Section 2.10.2
$\mathbf{m}$	Mask vector in a TasNet model of size N

$\mathbf{m}_\ell$	The ℓth mask vector of a sequence in the TasNet model of size N
$\mathbf{s}$	Vector of clean speech signal
$\mathbf{s}_c$	The cth clean speech signal vector of C speakers, of size L_x or L_s
$\mathbf{t}_c$	Reference transcription of the cth speaker
$\mathbf{v}$	Argsort vector used in Algorithm 1
$\mathbf{w}$	Vector of encoded speech mixture signal x
$\mathbf{x}$	Vector of speech mixture signals
$\mathbf{y}$	An arbitrary feature vector of size D used in defining feature vector, c.f. Section 2.5.2
$\mathbf{y}_\ell$	The ℓ feature vector or feature sequence $\mathbf{Y} \in \mathbb{R}^{L \times D}$
$\mathbf{z}$	Argsort vector used in Algorithm 1

List of Acronyms

AC	acoustic condition
ANN	artificial neural network
ASR	automatic speech recognition
BLSTM	bidirectional long short term memory
BS	batch size
BSS	blind source separation
CD	convolutional decoder
CN	channel-wise layer normalization
CSM	clean speech mixtures
cLN	cummulative layer normalization
Conformer	convolution augmented Transformer
Conv-TasNet	convolutional time-domain audio separation network
ConSepT	convolutional separation Transformer
CP-WER	concatenated minimum-power word error rate
CTC	connectionist temporal classification
CV	computer vision
DAE	denoising autoencoder
DANet	deep attractor network
DC	deep clustering
DE	density estimation
DL	deep learning
DM	dynamic mixing
DNN	deep neural network
DPRNN	dual path recurrent neural network model
DPTNet	dual path Transformer network
D-Conv	depthwise convolution

DD-Conv	deformable depthwise convolution
DDS-Conv	deformable depthwise-separable convolution
DP	dual-path
DS-Conv	depthwise separable convolution
DTCN	deformable temporal convolutional network
E2E	end-to-end
ED	epoch duration
ESTOI	extended short-time objective intelligibility
FLOPS	floating operations per second
FFT	fast Fourier transform
gLN	global layer normalization
GN	group normalization
GLU	gated linear unit
GRU	gated recurrent unit
GPIT	guided permutation invariant training
GPU	graphics processing unit
ICA	independent component analysis
IN	instance normalization
IPD	inter-phase difference
IR	impulse response
L	large
LD	logit difference
LGRU	light gated recurrent unit
LN	layer normalization
LSTM	long short term memory
LTI	linear time-invariant
LUFS	loudness units relative to full scale
M	medium
MACs	mutiply-accumulate operations
MHA	multihead attention
MHSA	multihead self-attention
MHCA	multihead cross-attention
MIMO	multiple-input multiple-output
ML	machine learning

MOS	mean opinion score
MPGT	multi-phase gammatone
MR	mask refinement
MRD	mask refinement decoder
MSE	mean square error
MVDR	minimum-variance distortionless response
MT	machine translation
NMF	non-negative matrix factorization
NSM	noisy speech mixture
NRSM	noisy reverberant speech mixture
OOM	out of memory
ORC-WER	optimal reference combination word error rate
PCA	principal component analysis
PE	positional encoding
PESQ	perceptual evaluation of speech quality
PIT	permutation invariant training
PM	post-masking
PMD	post-masking decoder
PReLU	parametric rectified linear unit
P-Conv	pointwise convolution
POLQA	perceptual objective listening quality assessment
QDPN	quasi-dual-path network
QRNN	quasi-recurrent neural network
ReLU	rectified linear unit
RF	receptive field
RIR	room impulse response
RNN	recurrent neural network
RSM	reverberant speech mixture
RTF	real time factor
RQ	research question
S	small
SA	self-attention
SAD	self-attention decoder
SAE	self-attention encoder

SC	skip connection
SE	squeeze-and-excite
SepFormer	separation Transformer
SISDR	scale-invariant signal-to-distortion ratio
SRMR	speech-to-reverberation modulation energy ratio
SDR	signal-to-distortion ratio
SiLU	sigmoid linear unit
SISO	single-input single-output
SNMF	sparse non-negative matrix factorization
SNR	signal-to-noise ratio
SOT	serialized output training
SOTA	state-of-the-art
SP	signal processing
SSR	speech-to-speech ratio
SSSR	self-supervised speech representation
STFT	short-time Fourier transform
STOI	short-time objective intelligibility
SVCCA	singular vector canonical correlation analysis
SW	shared weights
TasNet	time-domain audio separation network
TC	time-complexity
TD-Conformer	time domain Conformer
TCN	temporal convolutional network
TF	time-frequency
TPU	tensor processing unit
TSE	target speaker extraction
TSL	training signal length
uPIT	utterance-level permutation invariant training
UPGMA	unweighted pair group method with arithmetic mean
WD-Conv	weighted multi-dilation depthwise convolution
WD-TCN	weighted multi-dilation temporal convolutional network
WE	Whisper encoder
WER	word error rate
WPE	weighted prediction error

XL extra-large

List of Figures

List of Tables

Part I

Introduction & Background

Chapter 1

Introduction

1.1 Overview

Source separation techniques aim to separate an additive mixture of data, typically from the same domain (e.g. audio), into its individual summative components. Speech separation is the application of source separation techniques specifically for the separation of overlapping speakers, commonly referred to as *speech mixtures*. A popular example of this is the so-called *cocktail party problem* (Haykin & Chen, 2005). Speech separation has many applications, from assistive hearing devices, e.g. hearing aids, to multi-speaker ASR, e.g. meeting recognition, (Haeb-Umbach et al., 2021; Graetzer et al., 2021; Chang et al., 2019). Over the past decade, the ubiquitous application of deep neural networks (DNNs) in research for trying to solve a wide range of traditional machine learning and signal processing problems has resulted in the rapid improvement of single-channel speech separation performance across many benchmark evaluation corpora (Wang et al., 2018; Luo & Mesgarani, 2019; Subakan et al., 2021; Wang et al., 2023a). Furthermore, a knock-on effect of this has been improvements in multi-channel array-based speech separation and enhancement (Ochiai et al., 2020) as well as multi-speaker ASR (Chang et al., 2019).

This thesis is primarily concerned with single-channel speech separation. In the final chapter, its application to multi-speaker ASR is explored as well. Single-channel speech separation models are only able to use one microphone channel of input information to make inferences from the audio data, making it a particularly challenging task, especially when the overlapping speech sources are additionally corrupted by delayed speech reflections off acoustic boundaries, referred to as *reverberation*, or noise (Cord-Landwehr et al., 2022; Maciejewski et al., 2020; Naylor & Gaubitch, 2010). Many different types of noise exist; in this thesis, the primary type addressed is non-stationary noise, i.e. noise sources that change over time. Multichannel separation models can exploit information from multiple information streams as well as cross-channel spatial information of multiple microphones,

neither of which are available in the single-channel separation problem. Despite its obvious benefits, the suitability of multichannel separation solutions is highly dependent on both the practicality of implementing a microphone array on the device for which the model will be deployed as well as the availability of suitable data for training the model. Single-channel speech separation is, therefore, a highly important field with many real-world applications (Haeb-Umbach et al., 2021).

This thesis opens with an overview of speech separation research from the past to present day. In Chapter 2, several prominent model types used for speech separation are introduced, including independent component analysis (ICA), NMF and DNNs in order to highlight their key features and discuss their differences. The former two models were more popular in earlier research, but in more recent years, DNN models have come to dominate speech separation benchmarks (Hyvärinen, 2013; Le Roux et al., 2015; Wang et al., 2018; Luo & Mesgarani, 2018b; Wang et al., 2023b). Further information about these model types, as well as detailed descriptions of two popular single-channel DNN models highly relevant to this thesis, namely Conv-TasNet and SepFormer, are given in Chapter 2.

In Chapter 3 to Chapter 5, the robustness of the Conv-TasNet model (Luo & Mesgarani, 2019) to (in particular) reverberant but also noisy acoustic conditions is investigated. Conv-TasNet was initially proposed for single-channel speech separation using anechoic data (Luo & Mesgarani, 2019) and while it has been evaluated on speech enhancement tasks, not a huge amount of analysis has been done on the network structure and how different configurations respond to differing acoustic conditions. A key feature of the Conv-TasNet model is what is known as its *receptive field*. The receptive field is essentially the amount of temporal information (measured in seconds or samples) the model can observe to produce a single output frame. Not much attention has been paid to this property and how it interacts with temporal features of the environment acoustics, such as reverberation time, i.e. how much time it takes for environmental reflections to decay by a certain amount of energy. The main contributions of this thesis begin in Chapter 3 by investigating how Conv-TasNet performs in reverberant conditions. This is done by reformulating the separation model as a denoising autoencoder (DAE) for performing dereverberation (Ravenscroft et al., 2022b). In Chapter 4, an improvement to the Conv-TasNet model is proposed to enable the model to adapt the focus of its receptive field to different acoustic conditions using a computationally lightweight utterance-level attention network. In Chapter 5, this general idea is taken further by proposing a type of convolution known as deformable convolution (Dai et al., 2017), which enables the network to have a fully adaptive receptive field at the frame level. This model consistently improves speech separation in noisy and reverberant conditions with only a minor increase in model size.

A potential limitation of fully convolutional DNN models, such as Conv-TasNet, is their inability to model global context due to the limitation of the receptive field. As such, from

Chapter 6 to Chapter 9 combining convolutional structures in time-domain audio separation networks (TasNets) with multi-head attention or Transformer layers (Vaswani et al., 2017) is investigated in a number of different ways. Attention is a type of DNN layer used to apply greater weight to features with greater similarity. In Section 6.1, dot-product attention is reformulated as a cross-correlation function to demonstrate that it gives greater weighting to features in a sequence that have higher correlations to one another, and hence, it is argued that attention has inherent denoising properties, assuming the noise is uncorrelated. In Chapter 6, this denoising idea is applied to the time-domain encodings in the aforementioned Conv-TasNet model. Novel encoder and decoder networks are proposed, which incorporate MHA. This idea is shown to be particularly effective in the encoder as it focuses the feature representation on highly correlated features in the speech sequence prior to the separative mask estimation network. In Chapter 7, a study is performed to investigate the impact of training signal lengths (TSLs) on the separation performance of the aforementioned Conv-TasNet and SepFormer models. The insights gained from this section are then used in the remaining chapters for training novel separation models. A time-domain Conformer model, referred to as the TD-Conformer, is proposed in Chapter 8 as an alternative model to the dual-path (DP) network structure of the aforementioned SepFormer model. This model performs better than the SepFormer model, specifically on noisy and reverberant data, with a favourable computational expenditure on shorter signal lengths and shorter training times. The following chapter, Chapter 9, takes insights gained about the performance of the SepFormer and TD-Conformer models on both anechoic and noisy reverberant data and aims to combine the model structure to provide a balanced trade-off in terms of performance between both model structures.

The final contributory chapter is Chapter 10. In this final chapter, the application of the TD-Conformer model for multi-speaker ASR is explored. A novel technique for transcription-free adaptation of separation models for ASR is proposed using connectionist temporal classification (CTC) features instead of a transcription-dependant loss function such as CTC. The proposed method is demonstrated to give increases in WER performance consistently.

The final two chapters, Chapter 11 and Chapter 12, are given to provide some final discussion, ideas for future work and some closing remarks. The remainder of this section is given to outlining the research questions and contributions contained within this thesis.

1.2 Research Questions and Contributions

This section introduces the research questions addressed in this thesis as well as the contributions made. Not all contributions relate to one of the specified research questions and are sometimes a byproduct of the work seeking to address the specified questions. Where a contribution has been published, the reference to the publication is also given.

1.2.1 Research Questions

In this thesis, five main research questions (RQs) are posed. The first four directly address the problem of speech separation in noisy reverberant environments. The final question addresses speech separation more generally, but in the chapters addressing this question, emphasis is put on the noisy reverberant scenario.

RQ1 Given the temporal variations introduced by (non-stationary) noise and reverberation, how can time-domain audio separation networks (TasNets) be extended to better model temporal context in noisy and reverberant environments?

RQ2 Given the importance of the receptive field in controlling the amount of temporal information temporal convolutional networks can process, what relationship, if any, exists between the size of the receptive field of fully convolutional speech dereverberation models and reverberation time?

RQ3 Relating to the previous question, if a relationship can be found between the receptive field of convolutional derverberation networks and the temporal structure of reverberation, how can the receptive fields of those networks be made adaptive to varying acoustic conditions, in particular for combined speech enhancement and separation tasks?

RQ4 Given the inherent denoising properties of dot-product attention for uncorrelated noise in features, how can attention be applied in TasNet encoders to yield less noisy encoded speech features for TasNet separation models?

RQ5 Given the high time and computational requirements in training DP Transformer TasNet models, what methods and models can be found or derived that can deliver similar or better performance with a lower cost to train?

1.2.2 Contributions

The majority of contributions in this thesis have been published in a series of conference and journal papers (Ravenscroft et al., 2022b,c, 2023a, 2022a, 2023b,c). The contributions in Chapter 9 have been accepted for publication at ICASSP 2024 (Ravenscroft et al., 2024), and the contents of Chapter 10 have not yet been published due to the recency of this work. The following list is a set of descriptions of the contributions in this thesis and how they relate to research questions with associated citations or chapter references.

1. A study in Chapter 3 on how the receptive fields of TCNs affect their performance for monaural speech dereverberation, published in Ravenscroft et al. (2022b). This study addresses RQ2 with a view to address RQ1. In Chapter 3, a popular speech separation and enhancement TCN model known as Conv-TasNet is reformulated for dereverberation

tasks. The model is evaluated across many different model configurations to gain insights on how varying the receptive field influences dereverberation performance.

2. A TCN based structure incorporating an utterance-weighted multi-dilation convolutional block is introduced in Chapter 3 for improved dereverberation performance. The findings of this study address RQ1 and RQ3, and are published in Ravenscroft et al. (2022c). This approach proposes allowing TCNs to dynamically weight their focus within their receptive field using a lightweight convolutional attention mechanism known as *squeeze-and-excite*. The already published work in Ravenscroft et al. (2022c) is also added to in Chapter 4 by additionally evaluating the proposed model speech separation tasks.

3. A deformable TCN for speech separation in noisy reverberant environments along with a publicly available toolkit for computing 1D deformable convolution proposed and evaluated in Chapter 5. This work addresses RQ1 and RQ3 again and was published in Ravenscroft et al. (2023a). The deformable convolutional layer proposed in this work allows TCNs to have a fully dynamic receptive field at the frame level. This technique is shown to be particularly useful for noisy and reverberant data.

4. A method for analysing encoded features in TasNets based on depthwise Euclidean distance of the encoded features is proposed in Algorithm 1. This method was published as a part of (Ravenscroft et al., 2022a).

5. A multihead self-attention (MHSA) encoder for TasNets resulting in improved performance in noisy reverberant environments is proposed in Chapter 6. A series of MHA encoders and decoders are also investigated in this chapter with varying performance. The contents of this study were published in Ravenscroft et al. (2022a). This contribution addresses RQ1 and RQ4.

6. Novel findings are reported in Chapter 7 on TSL limits for training speech separation models where it is demonstrated that the signal length distribution of a training dataset can notably alter the overall performance when TSL limits are employed due to random sampling. In addition, it is shown that using TSL limits results in much faster training times for the SepFormer model with no negative impact on performance. The findings of this study were also published in Ravenscroft et al. (2023b) and address RQ1 and RQ5.

7. A subsampling-based TD-Conformer model for speech separation in anechoic and noisy reverberant environments is proposed in Chapter 8. This work was published in Ravenscroft et al. (2023c) and addresses RQ1 and RQ5. In this work, the computational attributes of convolution augmented Transformers (Conformers) are investigated in comparison to DP Transformers. Based on these insights, a new separation network

is proposed that can outperform recently state-of-the-art (SOTA) models on noisy reverberant speech mixtures.

8. A further study on combining Conformer and DP Transformers to investigate the generalisation properties on real mixtures is performed in Chapter 9. This work is accepted for publishing in Ravenscroft et al. (2024).

9. As a part of the previous contribution, a method for evaluating speech separation performance on real mixtures using intrusive metrics is also proposed, again also to be published in Ravenscroft et al. (2024).

10. A method for transcription-free fine-tuning of speech separation models for multi-speaker ASR is proposed in Chapter 10. This work is as yet unpublished, primarily due to the recency of it. In this work, it is shown that *frozen* ASR encoder network embeddings can be used to compute an embedding difference function, which can be backpropagated through speech separation networks for purposes of fine-tuning them towards multi-speaker speech recognition tasks.

1.3 Notation

Before proceeding, some guidance is given on the notation used in this thesis so that it might aid the reader. Where possible, the following conventions are followed for assigning notation:

- Scalar variables are written using lowercase symbols, e.g. x
- Constants are denoted using uppercase symbols, e.g. X. Note this includes model hyperparameters, as they remain constant per the model configuration in question.
- Vectors are written using lowercase boldface, e.g. $\mathbf{x}$
- Matrices and higher dimensional tensors are written using uppercase boldface, e.g. $\mathbf{X}$
- The above conventions apply to Greek symbols as well, e.g. $\theta, \Theta, \boldsymbol{\theta}, \boldsymbol{\Theta}$
- Frequency-domain variables are written using a sans-serif font, e.g. $\mathsf{x}, \mathsf{X}, \mathbf{\mathsf{x}}, \mathbf{\mathsf{X}}$
- All vectors should be assumed to be column vectors by default.

Chapter 2

Speech Separation Background

The work in this thesis is primarily concerned with speech separation methods for applications in adverse acoustic scenarios, i.e. in the presence of non-stationary background noise and reverberation. Reverberation is the effect that occurs when speech reflects off boundaries in an acoustic environment, resulting in delayed reflections of the speech also being picked up at the same microphone as the direct path of the speech from speaker to microphone. As will be discussed further in this thesis, this effect is often detrimental to speech separation performance (Maciejewski et al., 2020; Cord-Landwehr et al., 2022). In this thesis, *speech separation* is considered to fall under the category of *speech enhancement* technology, although sometimes *speech separation and enhancement* is used to emphasise models that jointly perform speech separation and some combination of noise and reverberation removal from the speech signal, referred to as denoising and dereverberation respectively (Maciejewski et al., 2020; Subakan et al., 2021; Rixen & Renz, 2022; Ravenscroft et al., 2023a). Another terminological point is that speech separation here specifically refers to models that aim to reconstruct *more than one* speaker signal, if more than one is present, from a mixture of speakers. Recently, many *target speaker extraction (TSE)* models have been proposed (Žmolíková et al., 2019; Delcroix et al., 2020) which have many similarities to speech separation. However, TSE is explicitly referred to in this thesis as a different technology to separation, as it aims to extract only one speaker from a mixture of speakers at a time.

Speech separation research has its roots in early blind source separation (BSS) methods (Jutten & Herault, 1991; Benesty, 2000; Bell & Sejnowski, 1995; Burel & Rondel, 1993; Yilmaz & Rickard, 2004; Virtanen, 2007). The goal in BSS is to separate C data sources denotes $s_c, c \in \{1, \ldots, C\}$ from an additive mixture of the data, denoted x, with no prior information about the sources s except what can be ascertained from the mixture x. Source separation research has long been dominated by machine learning and signal processing techniques such

Some of the contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2022a).

as ICA and artificial neural networks (ANNs) (Jutten & Herault, 1991; Herault et al., 1985; Richard et al., 2023). More recently, DNN models have heavily dominated the field due to improved access to data combined with improvements in parallel computing technologies (e.g. graphics processing units (GPUs) and tensor processing units (TPUs)) enabling large models with millions or more parameters to be trained on datasets in excess of hundreds of hours of data. A particular variant of DNN models, known as time-domain audio separation networks (TasNets) (Luo & Mesgarani, 2018b) have dominated benchmark datasets used for evaluating speech separation and enhancement models up until recently. The original TasNet (Luo & Mesgarani, 2018b) is composed of a single-layer convolutional neural network audio encoder, a recurrent neural network mask estimation network that filters the encoded audio and a decoder that decodes the filtered encoded audio back into the separated time domain audio signals. A number of variants have been proposed, with significant research interest in the mask estimation network in particular (Luo & Mesgarani, 2019; Tzinis et al., 2022b; Chen et al., 2020a; Li et al., 2022a; Subakan et al., 2021; Rixen & Renz, 2022). Two of note in this thesis are the Conv-TasNet (Luo & Mesgarani, 2019) and SepFormer (Subakan et al., 2021) models. These are highly cited models and are widely used separation networks in related research (Tzinis et al., 2022b; Maciejewski et al., 2020; Cord-Landwehr et al., 2022). They are both used widely as baselines and for evaluating methods throughout this thesis. Most of the information described in the remainder of this thesis typically relates to one or more of these models in some way, and as such, both are described fully in Section 2.9 and Section 2.10.

The remainder of this chapter proceeds as follows. The signal models used for the speech separation and enhancement models in this thesis are introduced in Section 2.1. Some widely used datasets used for creating and evaluating speech separation and enhancement models are introduced and described in Section 2.2. Some discussion of formulations is then given to earlier solutions to solving speech separation. The ICA model is briefly described in Section 2.3 and the NMF model in Section 2.4. In Section 2.5, DNN models are introduced along with some foundational information about different components of DNN models, such as training procedures, layer types and commonly used objective functions. In Section 2.7, some commonly used evaluation metrics of speech enhancement and separation models, particularly those used in this thesis, are introduced. From Section 2.8 to Section 2.10, significant discussion is given to so-called time-domain audio separation network (TasNet) models (Luo & Mesgarani, 2018b). These are a type of DNN model that, up until recently, typically dominated most speech separation benchmarks in terms of raw separation performance (Rixen & Renz, 2022; Wang et al., 2023b). The Conv-TasNet model (Luo & Mesgarani, 2019) is described in-depth in Section 2.9 along with some baseline results. This model has been highly popular for a number of different applications leveraging speech separation models (Ochiai et al., 2020; Delcroix et al., 2020) due to its reliable performance and comparatively lightweight and efficient design to other TasNet models (e.g. Chen et al. (2020a)). It is introduced here as

it is the main point of investigation from Chapter 3 to Chapter 5. The SepFormer model (Subakan et al., 2021) is introduced in Section 2.10. Up until mid-2022, SepFormer remained state-of-the-art on numerous benchmarks and has been widely researched for downstream applications (Luo et al., 2022; Cord-Landwehr et al., 2022). It is used as a baseline model for investigations and evaluations throughout Chapter 7 to Chapter 9. As such, a detailed description is provided in this chapter first before proceeding. The application of speech separation to multi-speaker ASR is described more thoroughly in Section 2.11. Some final discussion on other related topics that might be useful to the reader, such as dereverberation and unsupervised speech separation, are given in Section 2.12.

2.1 Signal Models

The speech separation problem is commonly defined as trying to separate source speech signals $s_c[i], c \in 1, \ldots, C$ from a mixture

$$x_{\text{clean}}[i] = \sum_{c=1}^{C} s_c[i]. \tag{2.1}$$

for discrete-time index i. Thus, the goal of using speech separation methods here is to attain an estimate $\hat{s}_c[i]$ for each of the C source signals $s_c[i]$. In this traditional formulation (Schmidt & Olsson, 2006), the signal model assumes no other disturbances to the speech signal.

In realistic applications of speech separation, noise and reverberation often impact the signal quality (Haeb-Umbach et al., 2021; Maciejewski et al., 2020). Therefore it is often suitable to include these in the signal model, i.e.

$$x[i] = \sum_{c=1}^{C} h_c[i] * s_c[i] + \nu[i] \tag{2.2}$$

where $*$ denotes the convolution operator, $h_c[i]$ is the room impulse response (RIR) corresponding to speaker c and $\nu[i]$ denotes additive noise.

In some parts of this thesis (Chapter 3 and Chapter 4), the focus is primarily on single-speaker speech dereverberation with a later view to evaluating combined speech separation and dereverberation models. To this view, a discrete single-channel reverberant single speaker signal for $C = 1$ is defined as

$$x_{\text{reverb}}[i] = h[i] * s[i] \tag{2.3}$$

$$= s_{\text{dir}}[i] + s_{\text{rev}}[i] \tag{2.4}$$

where $h[i]$ is the RIR of the single speaker. The reverberant speech signal $h[i] * s[i]$ can be

decomposed into the direct-path speech signal $s_{\text{dir}}[i] = \alpha s[i - i_0]$, with a delay i_0 and possible attenuation by a factor α, and reverberant part $s_{\text{rev}}[i] = (h[i] * s[i]) - s_{\text{dir}}[i]$. In dereverberation tasks in Chapter 3 and Chapter 4, the aim in this thesis is to estimate $s_{\text{dir}}[i]$, denoted $\hat{s}[i]$. Using the reverberation model in (2.4), the reverberant part of (2.2) can also be formulated as a purely additive problem such that

$$x[i] = \sum_{c=1}^{C} (s_{\text{dir},c}[i] + s_{\text{rev},c}[i]) + \nu[i] \tag{2.5}$$

where $s_{\text{dir},c}[i]$ denotes the direct path of the cth speaker to the receiving microphone and $s_{\text{rev},c}[i]$ denotes the remaining impulse train of the RIR $h_c[i]$, i.e. typically the reflections from walls and surfaces in the acoustic environment. The signal model (2.5) is often more relevant

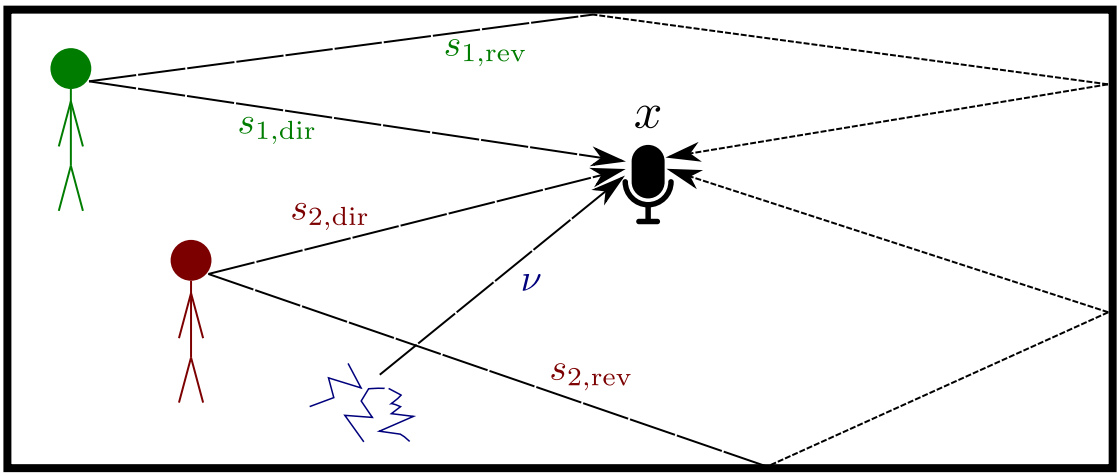


Figure 2.1: *Visualisation of the noisy reverberant speech separation problem (with $C = 2$ speakers) as described by Eq. (2.2).*

to simulated speech corpora such as WHAMR (Maciejewski et al., 2020) (described later in Section 2.2.1) where $s_{\text{dir},c}[i]$ is typically used as the target signal for training speech separation models. The signal model in (2.5) is depicted in Figure 2.1 where $C = 2$ speakers are seen with the direct paths of their speech reaching the microphone without any prior interference except for additive noise in the background until the reflected speech path reaches the microphone. Note that in Figure 2.1, only one reflection path is displayed. Many speech separation models that aim to solve this problem are also trained or constructed to suppress the noise or extract noise as an additional source (Wichern et al., 2019; Maciejewski et al., 2020; Subakan et al., 2021; Tzinis et al., 2022a). As speech reflections are highly correlated with their respective speech sources, not all speech separation approaches seek to suppress reverberant effects. Some approaches aim to perform speech dereverberation prior to separation or after separation (Zhang et al., 2020a; Ueda et al., 2021).

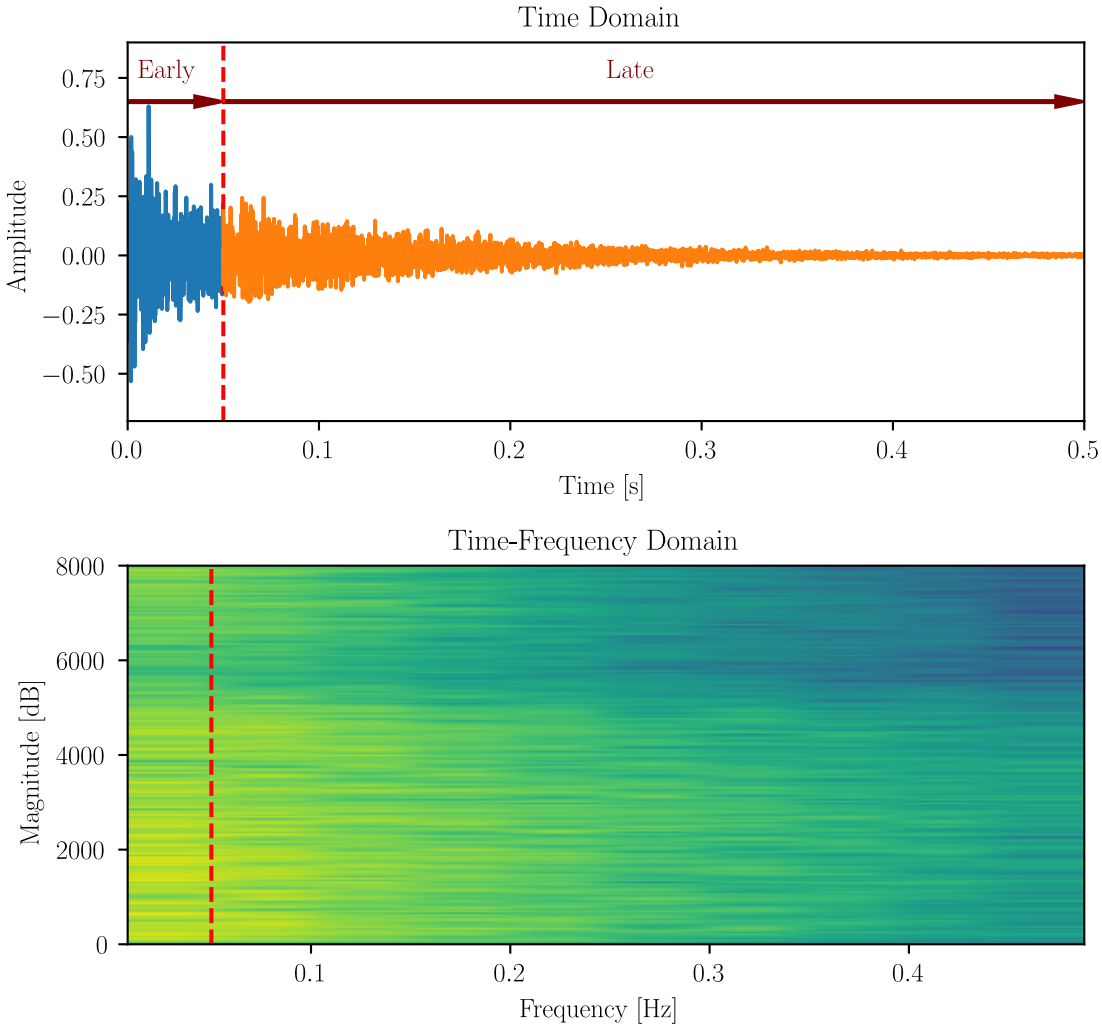


Figure 2.2: *Example RIR $h[i]$ depicted using time and time-frequency representations. Arrows annotate the early and late portions. The example shown is from* `air_type1_air_binaural_lecture_1_5.wav` *in the OPENRIR database (Ko et al., 2017).*

Another approach is to suppress only late reflections of speech (typically $>$ 50ms) and retain early reflections ($\leq$ 50ms) (Drude et al., 2019; Cord-Landwehr et al., 2022). The separate early and late parts of an RIR are visualised in Figure 2.2. This is motivated by research demonstrating the benefits of early reflections for speech intelligibility (Bradley et al., 2003; Bradley & Sato, 2002). This approach is formulated as

$$x[i] = \sum_{c=1}^{C} \left(s_{\text{early},c}[i] + s_{\text{late},c}[i] \right) + \nu[i] \tag{2.6}$$

where $s_{\text{early},c}[i]$ includes the direct path and early speech reflections and $s_{\text{late},c}[i]$ denotes late speech reflections. The signal model in (2.6) is a special case of (2.5), and thus, the mixture signal $x[i]$ is consistent between the two.

2.2 Datasets, Corpora & Challenges

Before discussing more technical details of speech separation models, some discussion is first given to the data available for training such models. As most ML speech separation models "learn" their behaviour from data, thus data is required to create them. The term *far-field* is sometimes used in this section to describe speech data. Far-field refers to the microphone being relatively far away from the speaker. Far-field speech data typically encompasses very noisy and reverberant speech signals, hence its usage in this section and subsequent chapters.

A number of corpora are available that have some application to the noisy reverberant multi-speaker scenario on which this project is focused. Some are simulated, and some are "real". The term "real" in this work refers to speech corpora where the signals used for training or evaluation are not artificially mixed with other speech or noise sources and, for reverberant speech, the speech was not artificially reverbed using real or simulated RIRs. Some corpora are entirely single channel, but most contain some kind of multichannel configuration. They also vary greatly regarding scenario, i.e. the microphone configurations, acoustic environment, speaker characteristics and linguistic content. Settings can vary from dinner party settings (Barker et al., 2018; Segbroeck et al., 2019) to meeting rooms (Carletta et al., 2006) to pedestrian areas (Barker et al., 2015; Vincent et al., 2016; Wichern et al., 2019; Maciejewski et al., 2020).

Some older datasets widely used for training multi-party transcription models, also called meeting recognition, are the NIST Rich Transcription Evaluations (RTE) series (NIST, 2020, 2009). The NIST datasets have not been widely explored in the modular separation and recognition approach described and explored later in this thesis but remain relevant to the field nonetheless. Around the same time as the NIST RTE series, the AMI meeting corpus was developed (Carletta et al., 2006; Hain et al., 2006). The AMI corpus has seen widespread use in a number of applications that utilise speech separation, including speech recognition and diarization (Chang et al., 2019; Morrone et al., 2023). Neither of these corpora is used in this thesis, but both are historically significant in multi-speaker speech processing, and AMI still regularly appears as a training and evaluation set in modern multi-speaker processing research (Chang et al., 2019; Radford et al., 2023).

The REVERB challenge (Kinoshita et al., 2011) presented real and simulated multichannel datasets intended for training and evaluating speech dereverberation & enhancement models. The simulated data is created from the WSJCAM0 (Robinson et al., 1995) corpus, and the *real* data is from the MC-WSJ-AV (Lincoln et al., 2005) corpus which is also used later in

this thesis in Chapter 9 to evaluate speech separation models in far-field environments. The CHiME 3 & 4 challenges (Barker et al., 2015; Vincent et al., 2016) similarly used real and simulated data for far-field audio recordings in a number of public places. Speakers were asked to record utterances from the WSJ0 corpus (Paul & Baker, 1992) in public places to create the *real* data for these corpora. This challenge also came in multiple multi-channel microphone configurations. The successor challenges to CHiME 3 and 4, the CHiME 5 and 6 challenges utilised a corpus of dinner parties with speakers being recorded from multiple microphone arrays and moving between rooms (Barker et al., 2018; Watanabe et al., 2020). The most recent CHiME 7 challenge had two sub-tasks: distant automatic speech recognition and unsupervised domain adaptation for speech enhancement (UDASE). The CHiME 7 datasets contained a diverse set of real and simulated data with the goal of speech recognition in far-field environments and unsupervised adaptation of speech enhancement models to realistic data (Cornell et al., 2023; Leglaive et al., 2023). Some other recent challenges of relevance to this project include the SPEAR and Clarity challenges (Guiraud et al., 2022; Graetzer et al., 2021). The SPEAR challenge was a binaural (2-channel) speech enhancement challenge for augmented reality headsets. There have been many versions of the Clarity challenge, but all have been concerned with speech enhancement for hearing aid devices. The reader is encouraged to refer to the relevant references for more information on these challenges and accompanying datasets.

Many simulated corpora have been proposed for training speech separation and speaker extraction models. One of the most commonly used datasets for anechoic (non-reverberant noise-free) speech separation is the WSJ0-Mix corpus (Isik et al., 2016), which consists of artificially mixed speech signals from the Wall Street Journal corpus. This corpus has 2-speaker and 3-speaker datasets, referred to as WSJ0-2Mix and WSJ0-3Mix, respectively. The WHAM dataset (Wichern et al., 2019) is an extension of WSJ0-2mix that combines non-stationary ambient noise recorded in real environments with the artificial speech mixtures of WSJ0-2Mix. The WHAMR dataset (Maciejewski et al., 2020) takes this a step further again by reverberating the speech mixtures using simulated RIRs. Another similar dataset is the LibriMix dataset (Cosentino et al., 2020). Again, this is a simulated dataset that uses utterances drawn from the open-source LibriSpeech dataset (Panayotov et al., 2015). It also uses the ambient noise in the WHAM dataset to create noisy mixtures. The WSJ0-Mix, WHAM, WHAMR and LibriMix datasets have been instrumental benchmarks in advancing the state-of-the-art in speech separation and related fields (Wang et al., 2018; Luo & Mesgarani, 2019; Subakan et al., 2021; Wang et al., 2023b). More in-depth details of these datasets are given in the following two sections, as these are all used extensively in this thesis.

2.2.1 WSJ-Mix & WHAMR

The WSJ0-Mix dataset was described first in Hershey et al. (2016) and Isik et al. (2016). In this corpus, simple speech mixtures are generated by mixing utterances at varying signal-to-noise ratios (SNRs) from 0-5dB. For each training example there is a speech mixture sample and two or three reference (target) speaker samples. As previously mentioned WSJ0-Mix comes with two speaker configurations for 2-speaker (WSJ0-2Mix) and 3-speaker (WSJ0-3Mix) mixtures. The WSJ0-Mix and the following WHAM and WHAMR corpora come in 8 kHz and 16 kHz configurations. In addition, there are another two configuration options referred to as *min* and *max*. The *min* configuration truncates all utterances to the length of the shortest utterance in the mixture. The *max* configuration pads the signal to the longest utterance.

A drawback of WSJ0-2Mix is it neither includes additional noise nor reverberation as targeted in this work in Eq. (2.2). To incorporate additional noise, the WHAM (WSJ0 Hipster Ambient Mixtures) corpus was introduced in (Wichern et al., 2019), and to incorporate reverberation, the WHAMR dataset was proposed by (Maciejewski et al., 2020) as a noisy reverberant extension to WSJ0-2Mix. Noise clips were sampled from a number of urban environments and these are mixed with the speech mixtures at a randomly selected SNR value from a uniform distribution between -6 and $+3$ dB. RIRs are also simulated from room acoustic parameters, and microphone and speaker positions that are randomly generated from uniform distributions specified by the authors of the corpus Maciejewski et al. (2020). An RIR is generated for each speaker from the same simulated room environment. The RIRs have a reverberation time RT60 ranging from 0.1 s to 1 s are generated using the pyroomacoustics software package (Scheibler et al., 2018). RT60 is a measure in seconds, of the amount of time it takes for an impulse response (IR) to decay by 60 dB.

For all WSJ0-Mix derived corpora mentioned in this section, the training set consists of 20000 training examples, resulting in an overall 58.03 hours of speech; the validation set consists of 5000 training examples, equalling 14.65 hours of speech, and the test set consists of 3000 examples resulting in 9 hours of speech.

2.2.2 LibriMix

LibriMix is a simulated 2-speaker (Libri2Mix) and 3-speaker (Libri3Mix) mixture corpus derived from the LibriSpeech and WHAM corpora (Cosentino et al., 2020). Speech samples come from the LibriSpeech corpus (Panayotov et al., 2015), and noise samples come from the WHAM corpus. Instead of speech-to-speech ratios (SSRs), LibriMix uses loudness units relative to full scale (LUFS) measured in dB to set the loudness of speakers and noise in the mixtures. Speakers have a loudness between -25 and -33 LUFS and noise between -38 and -30 LUFS.

The source clean LibriSpeech corpus has subsets of *train-100*, *train-360*, *dev* and *test*. For

training, LibriMix maintains the two separate *train-100* and *train-360* subsets from the parent LibriSpeech corpus (Panayotov et al., 2015). For Libri2Mix, the *train-360* set comprises 50,800 mixtures and 212 hours of speech and the *train-100* set comprises 13,900 mixtures and 58 hours of speech. For Libri3Mix, the *train-360* set comprises 33,900 mixtures and 146 hours of speech and the *train-100* set comprises 9,300 mixtures and 40 hours of speech. For all speaker configurations the *dev* and *test* sets both consist of 3,000 utterance and 11 hours of speech mixtures.

2.3 Independent Component Analysis (ICA)

Independent component analysis (ICA) was one of the first highly popular solutions to BSS. As such, this section briefly introduces the ICA source separation model, first proposed in Jutten & Herault (1991). The popularity of ICA was due to the simplicity of the approach and given computational restraints at the time. It is introduced here as it is a foundational BSS formulation (Richard et al., 2023) and also for the purposes of comparison to the NMF and DNN methods discussed in Section 2.4 and Section 2.5. It has been shown that using ICA formulations can solve the problem of noisy speech separation in addition to "clean" speech separation (Hongyan & Guanglong, 2010). ICA was first proposed and formalized as a multichannel problem and the solution requires a multichannel formulation. As such, in the following, the model is introduced as a multichannel model and then in the later part of the section, it is explained how it can be adapted for single-channel source separation. In the ICA model, a multichannel mixture signal

$$\mathbf{X} = \begin{bmatrix} x_1[1] & x_1[2] & \cdots & x_1[L_x] \\ x_2[1] & x_2[2] & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ x_M[1] & x_M[2] & \cdots & x_M[L_x] \end{bmatrix} \tag{2.7}$$

with L_x discrete signal samples for M channels is composed of weighted sums of *hidden factors* $s_c[i]$ such that

$$x_m[i] = \sum_{c=1}^{C} a_{m,c} s_c[i] \tag{2.8}$$

where $a_{m,c}$ is a constant value in a *mixing matrix* (Hyvärinen, 2013). The ICA model is typically formulated as the matrix equation

$$\mathbf{X} = \mathbf{AS} \tag{2.9}$$

where the mixing matrix $\mathbf{A} \in \mathbb{R}^{M \times C}$ comprises the mixing parameters $a_{m,c}$ and speech matrix

$\mathbf{S} \in \mathbb{R}^{C \times L_x}$. Thus the solution is to solve for the estimated speech matrix $\hat{\mathbf{S}}$ by calculating the *unmixing matrix* $\mathbf{A}^{-1}$, where $\hat{\mathbf{S}} = \mathbf{X}\mathbf{A}^{-\mathbf{1}}$ (Benesty, 2000). The formulation stated above relies on spatial information, i.e. $M > 1$ microphone channels, in order to solve the source separation problem reliably. However, the single-channel source separation problem ($M = 1$) can be solved using ICA by artificially increasing M with delayed copies of x_1.

To solve for the unmixing matrix $\mathbf{A}^{-1}$, a number of assumptions are made about the sources $\mathbf{S}$. The main assumption is that any source $\mathbf{s}_c$ is non-Gaussian and statistically independent of any other source $\mathbf{s}_{c'}$ (Hyvärinen & Oja, 1999; Tharwat, 2021). Sometimes, other assumptions such as uncorrelatedness or the orthogonality of sources are also assumed (Hyvärinen & Oja, 1999). Mathematically, the independence of two sources is described by the likelihood

$$\mathcal{P}\left(\mathbf{s}_c \cap \mathbf{s}_{c'}\right) = \mathcal{P}\left(\mathbf{s}_c\right)\mathcal{P}\left(\mathbf{s}_{c'}\right). \tag{2.10}$$

If statistical independence is true, then the expectation

$$\mathcal{E}\left(\mathbf{s}_c\mathbf{s}_{c'}\right) = \mathcal{E}\left(\mathbf{s}_c\right)\mathcal{E}\left(\mathbf{s}_{c'}\right), \tag{2.11}$$

i.e. the covariance of the two sources is zero (Ingle et al., 2005),

$$\operatorname{cov}\left(\mathbf{s}_c, \mathbf{s}_{c'}\right) = \mathcal{E}\left(\mathbf{s}_c\mathbf{s}_{c'}\right) - \mathcal{E}\left(\mathbf{s}_c\right)\mathcal{E}\left(\mathbf{s}_{c'}\right). \tag{2.12}$$

In most ICA algorithms, a *whitening* of $\mathbf{X}$ is typically performed before the actual ICA estimation (Hyvärinen, 2013). The whitening process typically involves transforming $\mathbf{X}$ by some whitening matrix $\mathbf{Z}$ where

$$\frac{1}{L_x}(\mathbf{Z}\mathbf{X})(\mathbf{Z}\mathbf{X})^\top = \mathbf{I}, \tag{2.13}$$

where $(\cdot)^\top$ denotes the matrix transpose and $\mathbf{I}$ is an identity matrix. The matrix $\mathbf{Z}$ is fairly trivially found using principal component analysis (PCA) (Hyvärinen, 2013; Wold et al., 1987). The ICA model still holds following this stage as

$$\mathbf{Z} = \mathbf{A}'\mathbf{S} = \mathbf{V}\mathbf{A}\mathbf{S} \tag{2.14}$$

with the consequence that now the quantity $\mathbf{A}' = (\mathbf{V}\mathbf{A})$ is to be estimated instead of just $\mathbf{A}$ in (2.9). $\mathbf{A}'$ is also now an *orthogonal* matrix, i.e. its inner and outer product with itself is an identity matrix. After applying whitening, the estimation of the mixing matrix can be constrained to the space of orthogonal matrices (Hyvärinen, 2013). This, in turn, reduces the number of unused (or *free*) parameters in the model. A major motivation for the whitening process is that if the matrix $\mathbf{A}'$ is orthogonal, this typically results in faster and more stable

estimation.

The final stage is an optimization process where $(\mathbf{A}')^{-1}$ is estimated. This is done by maximising some criterion referred to as the objective function. More common approaches for the optimization stage include the maximum-likelihood estimation and the minimization of mutual information (Pham & Garat, 1997; Eriksson & Koivunen, 2004; Comon, 1994; Hyvärinen & Oja, 2000). There are a number of possible approaches to solving the ICA problem for speech separation. As this is superfluous information to the remaining chapters of this thesis, greater detail is not provided here. However, some descriptions of notable algorithms specifically for single channel source separation using ICA can be found in (Davies & James, 2007; Barry et al., 2005). Some notable ICA algorithms include *FastICA* (Hyvarinen, 1999), *projection pursuit* (Hyvärinen & Oja, 2000), and *Infomax* (Hyvärinen & Oja, 2000).

2.4 Non-negative Matrix Factorisation (NMF)

NMF is another well-established technique in BSS research (Benesty, 2000; Le Roux et al., 2015; Cauchi et al., 2016). NMF forms part of the transition from signal-processing-based source separation to learning-based approaches Roux et al. (2019). It also inspires the later formulation of the TasNet model in Section 2.8 (Luo & Mesgarani, 2018b). A key feature of NMF is the non-negativity constraint, which enforces the input data only to be represented by additive combinations of *weighted basis signals*. Commonly, speech processing models often use purely positive features to achieve their goal, e.g. magnitude and power spectral representation. Thus, the NMF model continues in this fashion by enforcing non-negative representation learning using basis functions. This is a useful feature of the model, particularly as it aids its interpretability. A key difference of using NMF as an alternative to ICA is that it can recover the sources without requiring explicit knowledge or making assumptions about the signal statistics.

In NMF, a non-negative matrix representation of the mixture signals, such as absolute magnitude or power spectral representations, $\mathbf{X}$ of shape $N_{\text{spec}} \times L_{\mathbf{X}}$, where N_{spec} is the number of spectral features and $L_{\mathbf{X}}$ is the sequence length, is defined as

$$\mathbf{X} \approx \mathbf{BW} \tag{2.15}$$

where $\mathbf{B} \in {\mathbb{R}_+}^{N_{\text{spec}} \times C}$ defines the basis of the data and $\mathbf{W} \in {\mathbb{R}_+}^{C \times L_{\mathbf{X}}}$ contains the energy for each basis. The non-negativity constraint is enforced by $\mathbf{B}, \mathbf{W} \geq 0$. The matrices $\mathbf{B}$ and $\mathbf{W}$ are intialised to positive values estimated from example data in the training stage (Abdali & NaserSharif, 2017). A common approach in the case of speech separation of a mixture of C

speakers, i.e.

$$\mathbf{X} = \sum_{c=1}^{C} \mathbf{S}_c \tag{2.16}$$

where $\mathbf{S}_c$ is the spectral representation of the cth speaker, is to define the basis matrix as a concatenated matrix of C submatrices (Schmidt, 2009) such that

$$\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_C] \tag{2.17}$$

where $\mathbf{B}_c \in \mathbb{R}^{N_{\text{spec}} \times C'}$ and $\mathbf{C}'$ is an arbitrary dimension size. Correspondingly, for the weight matrix, this means using concatenation of sparse matrices, i.e.

$$\mathbf{W} = \left[\mathbf{W}_1^\top, \ldots, \mathbf{W}_C^\top\right] \tag{2.18}$$

where $\mathbf{W}_C \in \mathbb{R}^{C' \times L_\mathbf{X}}$. By enforcing the sparsity of the weight sub matrices $\mathbf{W}$, the approximation for each individual source can be found by

$$\hat{\mathbf{S}}_c \approx \mathbf{B}_c \mathbf{W}_c, c \in \{1, \ldots, C\}. \tag{2.19}$$

This approach is generally referred to as *sparse non-negative matrix factorization (SNMF)* (Schmidt, 2009; Peharz & Pernkopf, 2012).

To estimate $\mathbf{B}$ and $\mathbf{W}$ an iterative optimization algorithm is typically used (Schmidt & Olsson, 2006; Peharz & Pernkopf, 2012). One of the simplest solutions to this optimization problem is to minimize the difference between $\mathbf{X}$ and $\mathbf{BW}$ subject to the non-negativity constraint, i.e.

$$\min_{\mathbf{B},\mathbf{W}} \left(\|\mathbf{X} - \mathbf{BW}\|\right) \qquad \text{s.t.} \qquad \mathbf{B}, \mathbf{W} \geq 0. \tag{2.20}$$

To enforce the sparsity constraint in NMF, an additional term for $\mathbf{W}$ is often included which is a weighted sum of its contents, i.e.

$$\min_{\mathbf{B},\mathbf{W}} \left(\|\mathbf{X} - \mathbf{BW}\| + \lambda \sum_{c,\ell} \mathbf{W}_{c,\ell} \right) \qquad \text{s.t.} \qquad \mathbf{B}, \mathbf{W} \geq 0 \tag{2.21}$$

where λ controls the sparsity. More in-depth descriptions of optimization algorithms used for NMF and SNMF can be found in Schmidt & Olsson (2006); Schmidt (2009); Peharz & Pernkopf (2012).

2.5 Deep Neural Networks (DNNs)

A significant drawback of ICA and NMF is that they are limited by their linearity and inability to model complex, in particular, non-linear relationships in the data, i.e. (2.9) and (2.15) are linear models. The further development of DNN models, with their ability to learn non-linear hierarchical representations of the data, has led to significant improvements in separation performance.

The rise in deep learning (DL) research around the early 2010s in a number of different fields (Krizhevsky et al., 2012; Hinton et al., 2012a; Graves, 2013) similarly led to a rise in interest in applying similar methods to speech separation. This section gives an overview of the foundational DNN-based contributions to speech separation research. In addition, a mathematical description of some of the more common network layers and loss functions, particularly those used heavily in this thesis, are given.

The earliest work on ANNs for modelling computational data dates back to the 1940s (McCulloch & Pitts, 1943). Since then, a number of different types of models and strategies for *training* neural network models have been developed. The training process of a neural network model refers to optimising the model's various multiplicative and additive *parameters*, often called *weights* and *bias*, respectively. This process is how models *learn* their behaviour, usually from example data. In this thesis, the primary focus is on *supervised* neural network models. Supervised models are models that can be trained on *labelled data* (Haykin, 2009). Labelled data refers to datasets where each training input sample has a paired reference output sample (the "label") that can be used to compute an error term between the reference output sample and the predicted output of the neural network model. The function used to compute the error term is called the *loss function*. Depending on context, the loss function is sometimes referred to by its negative value, the *objective function*. Thus, the aim in supervised training is usually to either *minimise* the loss function or *maximise* the objective function; the former phrasing is typically used due to the nature of the optimization algorithms commonly used for training neural network models. The optimization algorithm used throughout this thesis is known as *error backpropagation* (Rumelhart et al., 1986). Section 4.4-4.8 of Haykin (2009) provides a thorough mathematical description of error backpropagation for neural networks. In error backpropagation, the output of the loss function (the error term) is used to optimise the parameters of the neural network by backpropagating error gradients backwards layer-by-layer through the neural network. The network parameters are then updated via *gradient descent*, a widely used optimization technique described in Haykin (2009). Gradient descent involves iteratively adjusting model parameters in multiple steps until the model reaches a minimum criterion of the loss function, ideally the minimum, but usually manually specified as some condition by which the loss appears to no longer decrease after a set number of epochs of training. The amount the parameters are adjusted is controlled by a hyperparameter

called the learning rate, denoted η. The overall schematic for supervised training via error backpropagation is visualized in Figure 2.3.

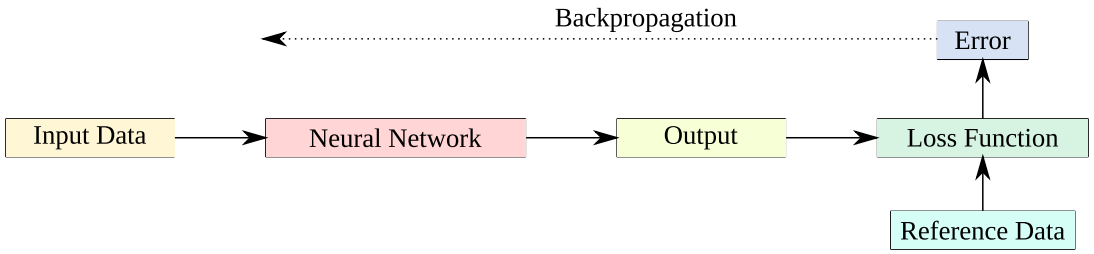


Figure 2.3: *Diagram of the supervised approach to training deep neural networks using error backpropagation. Solid lines indicate information flow. The dashed line indicates direction of gradient propagation.*

The loss function is often computed across multiple training examples and averaged across the so-called *batch* of training examples before being backpropagated through the network. This approach typically speeds up training as *foward passes* for each example in the batch can be computed in parallel using floating point acceleration hardware such as GPUs (Krizhevsky et al., 2012).

In the methods proposed in this thesis, the error backpropagation optimization algorithm is mostly taken for granted. The main point to emphasize on the backpropagation algorithm is that *any neural network operation must be differentiable* else it will not be possible to backpropagate errors through the network properly, and thus, the model will be untrainable. This is a particularly important point later in Chapter 5 for the deformation algorithm that dynamically changes seemingly discrete sample points in convolutional DNN layers (introduced in Section 2.5.6). Many issues arise in training neural networks, often relating to the gradients. A common issue is the *vanishing or exploding gradient problem* (Hochreiter, 1998) with very deep networks or inputs with very long sequences. Every additional backpropagation of a gradient through another layer or sequence step can result in exponentially small or large gradients. For example, if all gradients are < 1, they tend towards 0 with every additional sequence step or network layer. Conversely, if gradients are > 1, they tend towards ∞ with every additional sequence step or network layer. When using discrete floating point numbers on a computer, this can result in the error gradients becoming zero-valued, in which case the parameters will not be updated, or the gradients can become infinite valued, often leading to infinite valued parameters and other unintended behaviour.

2.5.1 DNNs for Speech Separation

Separation techniques using neural network models have been around since the early days of BSS research (Herault et al., 1985; Jutten & Herault, 1991). Although DNNs had been

dominating many areas of research since the early 2010s, they were not popularized in speech separation research until around 2015 (Roux et al., 2019; Hershey et al., 2016). This was primarily due to the *permutation problem*, i.e. which model output $\hat{s}_c[i]$ should be mapped to which reference signal $s_c[i]$ in the loss, known as the *permutation problem* (cf. Section 2.6.2), and the unknown number of sources. With the introduction of *deep clustering (DC)* (Hershey et al., 2016), DNNs models became a lot more dominant in the field of speech separation (Chen et al., 2017; Luo & Mesgarani, 2019; Wang et al., 2023a). This approach developed an objective function to enable the use of the unsupervised k-means clustering algorithm to separate out deep embeddings into separate classes from spectral inputs, i.e. each speaker's embeddings are clustered individually. This method proved effective, outperformed an NMF baseline and implicitly solved the permutation problem. Permutation invariant training (PIT) (Yu et al., 2017) was proposed around a similar time to DC as another approach to solving the permutation problem with DNN models. PIT solves the permutation by minimizing the loss pairs in a permutation matrix and assumes the minimum loss permutation is the most accurate way to solve the problem. This is formulated mathematically later in Section 2.6.2. A deep attractor network (DANet) was proposed shortly following DC and PIT (Chen et al., 2017). This method proposed using clustering combined with a masking network and similarity functions to assign frequency bins to each specific speaker. This method required less preprocessing than the DC approach and gave better performance as well. The Chimera and Chimera++ networks (Luo et al., 2017; Wang et al., 2018) similarly aimed to perform separation using a two-headed network, one DC subnetwork and one time-frequency masking subnetwork. This approach resulted in performance improvements over the DC approach. The Chimera++ network at the time of publishing was SOTA on the WSJ0-2Mix dataset (outperforming both DANet and DC). A further improvement to the PIT algorithm referred to as utterance-level permutation invariant training (uPIT) was proposed which gave comparable results to Chimera++ and DC-based models. The uPIT algorithm also improves the efficiency of PIT by solving the permutation at the utterance level as opposed to the frame level. Nowadays it is common to see uPIT simply referred to as PIT as there is no benefit to using PIT as it was formulated in Yu et al. (2017) over uPIT. For all the models trained in this thesis, uPIT is used and often referred to simply as PIT.

All of the DNN models discussed in the previous paragraph operate in the time-frequency (TF) domain. However, as will be discussed in Section 2.8 and beyond, a learnable time-domain representation alternative can outperform TF models and give SOTA performance in many cases (Luo & Mesgarani, 2019; Subakan et al., 2021; Rixen & Renz, 2022). This was first proposed in the later discussed TasNet model (Luo & Mesgarani, 2018b) by using a convolutional neural network to encode the time domain signal. Other hand-crafted features such as multi-phase gammatone (MPGT) filterbanks have also been proposed as an alternative to TF representations (Ditter & Gerkmann, 2020).

Most of the previously mentioned and other DNN separation models can be separated into one of two classes, namely, mapping or masking. Mapping networks take some encoded speech mixture $\mathbf{W} \in \mathbb{R}^{L \times N}$, of sequence length L and feature dimension N, and directly map the mixtures to C encoded speech signals $\mathbf{V}_c \in \mathbb{R}^{L \times N}$ for $c \in \{1, \ldots, C\}$. Masking networks, on the other hand, take the encoded mixture $\mathbf{W}$ and produce a sequence of masks $\mathbf{M}_c \in \mathbb{R}^{L \times N}$ for each speaker. The masks are then multiplied with the original encoded mixture $\mathbf{W}$ using the Hadamard product, denoted $\odot$, to give the encoded features for each speaker, i.e. $\mathbf{V}_c = \mathbf{M}_c \odot \mathbf{W}$. It has been suggested that masking networks give better intelligibility of speech while mapping networks retain better quality, but the research on mapping vs. masking networks is as yet fairly inconclusive (Nossier et al., 2020). Many examples of masking networks will be discussed in greater depth later in this thesis, such as Conv-TasNet (Luo & Mesgarani, 2019) and SepFormer (Subakan et al., 2021). Some notable examples of mapping networks in more recent separation research include the Wavesplit model (Zeghidour & Grangier, 2021) and the TF-GridNet model (Wang et al., 2023a).

The remainder of this section introduces commonly used network layers and loss functions used in speech separation models with particular emphasis on those most relevant to this thesis. The background of the aforementioned TasNet models is discussed in their own sections later in this Section 2.8. Two of these, the Conv-TasNet model and the SepFormer model, are given their own section as they form a significant backbone of much of the work done in the proceeding chapters. As such, it is seen as necessary to give each of these a more in-depth explanation and analysis than some of the previously mentioned models.

2.5.2 Network Layers

A number of different network layers have been proposed, typically due to how they model spatial or sequential information differently. The following subsections introduce the foundational layer types relevant to the remainder of this thesis. According to their relevance to this thesis, some types receive more in-depth descriptions than others. The first type of network layer, historically referred to as the perceptron (Rosenblatt, 1958; Haykin, 2009) but nowadays more commonly referred to as a *feedforward* or *linear* layer, is a simple mathematical or computational abstraction of how neurons fire in the human brain originally proposed as a probabilistic model for memory storage. In addition to this layer structure, a number of types of network layers are introduced in Section 2.5.3 to Section 2.5.8, including recurrent layers, convolutional layers and the more recent Transformer layer, along with some discussion of the difference between each type. Most of the layers discussed take an input sequence of length L with feature dimension D. In some cases, particularly for higher dimensional convolutional layers, the feature space may be multi-dimensional, i.e. $D \times \ldots \times D'$. In this thesis, higher dimensional layers are never used and, as such, are not included in the following as essential

information. For more information, however, the reader is referred to Section 4.17 in Haykin (2009).

Comments on notation In the proceeding subsections describing each layer, the input features of any layer are drawn from a L length sequence of features of dimension D denoted

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_L \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,D} \\ y_{2,1} & y_{2,2} & \cdots & y_{1,D} \\ \vdots & \vdots & \ddots & \vdots \\ y_{L,1} & y_{L,2} & \cdots & y_{L,D} \end{bmatrix} \in \mathbb{R}^{L \times D} \tag{2.22}$$

The feature (row) vector of $\mathbf{Y}$ at temporal frame index $\ell \in \{1, \ldots, L\}$ is denoted $\mathbf{y}_\ell$ with the ℓ subscript omitted, i.e. just $\mathbf{y}$, sometimes if it applies $\forall \ell \in \{1, \ldots, L\}$. Sometimes scalar features are used with the subscripts $\ell \in \{1, \ldots, L\}$ for sequence indexing and feature indexing $d \in \{1, \ldots, D\}$, i.e. $y_{\ell,d}$. Commonly, the ℓ index is also omitted from $y_{\ell,d}$, i.e. y_d, if the layer applies the same to all L feature vectors independent of ℓ. Greek notation (θ, β, etc.) is used primarily for trainable model parameters in these sections (the main exclusions to this rule include standard deviation σ and mean μ which are used for statistics). The bias terms $\boldsymbol{\beta}$ can often be considered as "optional" or nonessential to the layer and consequently are omitted in more complex formulations, such as depthwise separable convolution (cf. (2.49)). Sometimes, this is due to the nature of the layer being described, but it is also common that bias terms are instead effectively replaced by normalization layers, as is the case in the later described Conv-TasNet model (cf. Section 2.9). The notations for mean, μ, and standard deviation, σ, always refer to the input features drawn from $\mathbf{Y}$. A final comment is that the batch dimension is left out from the above notation in the following subsections. This is because none of the layers described below perform batch-wise operations, e.g. batch normalization (Ioffe & Szegedy, 2015), so this dimension is unnecessary to include. The reader should assume all layer functions are applied identically to each sample in a batch, with no information traversing samples in the batch.

2.5.3 Feedforward & Linear Layers

Feedforward layers (sometimes referred to as linear layers) are the most straightforward type of DNN layer. In its simplest form, it projects a set of features $\mathbf{y} \in \mathbb{R}^D$ of dimension D into a different feature space of dimension E, i.e. $\mathcal{F} : \mathbb{R}^D \to \mathbb{R}^E$ (Haykin, 2009). This is done using a set of network parameter weights $\boldsymbol{\Theta}_{\text{ff}} \in \mathbb{R}^{D \times E}$ and biases $\boldsymbol{\beta} \in \mathbb{R}^E$, i.e.

$$\mathcal{F}(\mathbf{y}, \boldsymbol{\Theta}_{\text{ff}}, \boldsymbol{\beta}) = \mathbf{y}\boldsymbol{\Theta}_{\text{ff}} + \boldsymbol{\beta}. \tag{2.23}$$

2.5.4 Nonlinearties and Activations Functions

On their own, the feedforward layers in (2.23) are only useful for mapping linear relationships in data (Haykin, 2009). Nonlinearities, also in some contexts referred to as activation functions, are a necessary component of neural networks in order to model the complex nonlinear hierarchical relationships in data such as far-field speech, cf. Section 2.2 for the meaning of far-field speech. In this subsection, only the nonlinearities most relevant to the remainder of this thesis are introduced. However, many variants have been proposed and are widely used (Maas et al., 2013; He et al., 2015; Elfwing et al., 2018; Dauphin et al., 2017; Hendrycks & Gimpel, 2016; Haykin, 2009).

2.5.4.1 Hyperbolic Tangent

One of the most prominent activation functions is the hyperbolic tangent function, defined as

$$\mathcal{H}_{\tanh}\left(y_d\right) = \tanh(y_d) \tag{2.24}$$

where $\tanh : \mathbb{R} \mapsto (-1, 1)$. This function has a sigmoidal shape, normalising output features on the interval $(-1, 1)$. This activation is only used in one instance in this thesis, the gating mechanism for the SepFormer model described later in Section 2.10, cf. Figure 2.9.

2.5.4.2 Sigmoid

The sigmoid function has many uses in ML algorithms, particularly due to its common application in logistic regression (Haykin, 2009). A favourable quality of the function is that it results in feature space normalized in the interval $(0, 1)$. The sigmoid function for a scalar feature y_d in feature vector $\mathbf{y} \in \mathbb{R}^D$, $d \in \{1, \ldots, D\}$ is defined as

$$\mathcal{H}_{\sigma}(y_d) = \frac{1}{1 + e^{y_d}}. \tag{2.25}$$

2.5.4.3 Softmax

The softmax function is an extension of the sigmoid function which normalises a feature vector $\mathbf{y}$ on a probabilistic space by dividing the exponent of each feature y_d by the sum of the total of the exponents of all the features, i.e.

$$\mathcal{H}_{\text{softmax}}(\mathbf{y}, u) = \frac{e^{y_u}}{\sum_{d=1}^{D} e^{y_d}} \tag{2.26}$$

for $u \in \{1, \ldots, D\}$. It is often most useful in speech processing for classification and recognition tasks such as the CTC multi-speaker ASR models introduced later in Chapter 10.

2.5.4.4 Rectified Linear Unit (ReLU)

The rectified linear unit (ReLU) function was proposed as an alternate to the sigmoidal activation functions previously described. It is defined as

$$\mathcal{H}_{\text{ReLU}}(y_d) = \begin{cases} 0, & \text{for } y_d < 0 \\ y_d, & \text{for } y_d \geq 0 \end{cases} \tag{2.27}$$

for a scalar feature $y_d \in \mathbb{R}$. It is more efficient to compute than sigmoidal functions and has other favourable qualities, such as a derivative of 1 for positive values, which can aid in the so-called *vanishing gradient problem* described in Hochreiter (1998) and explained earlier in this chapter. It is used in the encoder of the Conv-TasNet and SepFormer models described later in Section 2.9 and Section 2.10, respectively.

2.5.4.5 Parametric ReLU (PReLU)

The parametric rectified linear unit (PReLU) is a proposed improvement to the ReLU function. Originally proposed to improve the accuracy of image classification, it is defined as

$$\mathcal{H}_{\text{ReLU}}(y_d, \zeta_d) = \begin{cases} \zeta_d y_d & \text{for } y_d < 0 \\ y_d & \text{for } y_d \geq 0 \end{cases} \tag{2.28}$$

where ζ_d is a trainable parameter. It is widely used in the aforementioned Conv-TasNet model (Luo & Mesgarani, 2019).

2.5.4.6 Sigmoid Linear Unit (SiLU)

The sigmoid linear unit (SiLU) (sometimes also referred to as Swish) function proposed in (Elfwing et al., 2018) is a so-called *self-gated* activation function (Ramachandran et al., 2018). It is defined as

$$\mathcal{H}_{\text{SiLU}}(y_d) = y_d \mathcal{H}_{\sigma}(y_d) = \frac{y_d}{1 + e^{y_d}}. \tag{2.29}$$

It has been reported to result in faster convergence than ReLU (Gulati et al., 2020) and better performance than ReLU when applied to deeper networks (Ramachandran et al., 2018).

2.5.4.7 Gated Linear Unit (GLU)

The gated linear unit (GLU) function was proposed in (Dauphin et al., 2017). It is related to the SiLU function, but instead of features gating themselves, half the feature vector gates the

other half. GLU is defined as

$$\mathcal{H}_{\text{GLU}}(\mathbf{y}) = \mathbf{y}_{1:\frac{D}{2}} \odot \mathcal{H}_{\sigma}(\mathbf{y}_{\frac{D}{2}:D}) \tag{2.30}$$

where D is an even number. Note that this activation function reduces the input feature dimension size by 2 at the output, i.e. $\mathcal{H}_{\text{GLU}} : \mathbb{R}^D \mapsto \mathbb{R}^{\frac{D}{2}}$

2.5.5 Normalization Layers

Normalization layers are typically used to improve training accuracy and speed (Wu & He, 2018; Ba et al., 2016). Most normalization layers compute the mean and variance across feature vectors, channels, *groups* of channels, samples and/or batches of samples and then subtract the mean from the features and divide them by the standard deviation, often with additional feature weights $\boldsymbol{\theta}_{\text{LN}} \in \mathbb{R}^D$ and biases $\boldsymbol{\beta} \in \mathbb{R}^D$ applied to the normalized features to scale them accordingly.

2.5.5.1 (Global) Layer Normalization

The simplest and most common layer normalization is formulated as

$$\mathcal{G}\left(\mathbf{Y}, \boldsymbol{\theta}_{\text{LN}}, \boldsymbol{\beta}\right) = \frac{\mathbf{Y} - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}} \odot \boldsymbol{\theta}_{\text{LN}} + \boldsymbol{\beta} \tag{2.31}$$

where $\odot$ is the Hadamard product, and $\mu_{\mathbf{Y}} = \mathcal{E}\left(\mathbf{Y}\right)$ and $\sigma_{\mathbf{Y}} = \sqrt{\text{Var}\left(\mathbf{Y}\right)}$ are the mean and standard deviation of features $\mathbf{Y}$, respectively. In subsequent chapters, this formulation is also referred to as global layer normalization (gLN) to differentiate it from channel-wise layer normalization (CN) and cummulative layer normalization (cLN) also used in Conv-TasNet models (Luo & Mesgarani, 2019).

2.5.5.2 Channel-wise Layer Normalization

Channel-wise layer normalization (CN) is used in many public implementations of the Conv-TasNet model (such as Li et al. (2021); Ravanelli et al. (2021)) introduced later in Section 2.9. The CN layer for each block of a feature vector $\mathbf{y}_\ell$ is defined as

$$\mathcal{G}_{\text{CN}}\left(\mathbf{y}_\ell, \boldsymbol{\theta}_{\text{LN}}, \boldsymbol{\beta}\right) = \frac{\mathbf{y}_\ell - \mu_{\mathbf{y}_\ell}}{\sigma_{\mathbf{y}_\ell}} \odot \boldsymbol{\theta}_{\text{LN}} + \boldsymbol{\beta} \tag{2.32}$$

where $\mu_{\mathbf{y}_\ell}$ and $\sigma_{\mathbf{y}_\ell}$ define the mean and standard deviation of $\mathbf{y}_\ell$, and as before, $\boldsymbol{\theta}_{\text{LN}} \in \mathbb{R}^D$ and $\boldsymbol{\beta} \in \mathbb{R}^D$ are trainable weights and basis, respectively.

2.5.5.3 Cumulative Layer Normalization

The cLN layer was proposed to address the non-causal nature of the layer normalization (LN) formulation in 2.31 due to the computation of the mean over the entire sequential axis (of size L). The way the cLN layer addresses this is by aggregating the mean as ℓ increases across the sequential axis, i.e. $\ell \in \{1, 2, \ldots, L\}$. Therefore, the cLN function is defined as

$$\mathcal{G}_{\text{cLN}}\left(\mathbf{y}_\ell, \boldsymbol{\theta}_{\text{LN}}, \boldsymbol{\beta}\right) = \frac{\mathbf{y}_\ell - \mu_{k\leq\ell}}{\sigma_{k\leq\ell}} \odot \boldsymbol{\theta}_{\text{LN}} + \boldsymbol{\beta} \tag{2.33}$$

where $\mu_{k\leq\ell} = \mathcal{E}\left(\mathbf{y}_{k\leq\ell}\right)$ is the mean of all $\mathbf{y}_k, k \leq \ell$ and $\sigma_{g\leq\ell} = \sqrt{\text{Var}\left(\mathbf{y}_{k\leq\ell}\right)}$ is the standard deviation of all $\mathbf{y}_k, k \leq \ell$, for all $\mathbf{y}_\ell \in \mathbf{Y}$.

2.5.5.4 Group Normalization and Instance Normalization

An extension of LN is group normalization (GN) where subgroups of the feature vectors of $\mathbf{Y}$ are grouped into groups of size G, where G is constrained such that $G < D$ and D is perfectly divisible by G, i.e. $D - G\left\lfloor\frac{D}{G}\right\rfloor = 0$. The GN layer is defined as

$$\begin{aligned}\mathcal{G}_{\text{GN}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\text{LN}}, \boldsymbol{\beta}, G\right) = [\mathcal{G}\left(\mathbf{Y}_{1:G}, \boldsymbol{\theta}_{\text{LN},1:G}, \boldsymbol{\beta}_{1:G}\right), \mathcal{G}\left(\mathbf{Y}_{G+1:2G}, \boldsymbol{\theta}_{\text{LN},G+1:2G}, \boldsymbol{\beta}_{G+1:2G}\right), \\ \ldots, \mathcal{G}\left(\mathbf{Y}_{D-G:D}, \boldsymbol{\theta}_{\text{LN},D-G:D}, \boldsymbol{\beta}_{D-G:D}\right)].\end{aligned} \tag{2.34}$$

Note that if $G = 1$, then μ and σ are only computed along the sequential axes of $\mathbf{Y}$. This is referred to commonly as instance normalization (IN) (Wu & He, 2018).

2.5.6 Convolutional Layers

Convolutional layers can be viewed as an extension of the feedforward layer (cf. Section 2.5.3) to higher dimensions, e.g. sequential, temporal or spatial. In 1D convolutional layers, typically the convolutional *kernel*, i.e. the parameters, are used to process sequential information simultaneously (also known as *depthwise* information) in addition to the feature axis (also referred to as *pointwise* information). In this simpler case, there are technically 2 dimensions of information, the feature axis and the sequential axis, but one of them (the feature axis) has a fixed dimension D and the other (the sequential axis) is of an arbitrary dimension L. Hence, the "1" in 1D refers to the sequential axis. Following this logic, the 2D convolutional kernel processes the feature dimension and two sequential dimensions, also often discussed as *spatial information* (Chen et al., 2021a). Following this logic, the shape of the input information of a Y-dimensional convolutional kernel is $D \times L_1 \times \ldots \times L_Y$. The output shape of a kernel can depend on the hyper-parameters of the kernel, such as its *kernel size*, *stride* and *dilation factor*, as well as engineering design choices such as the use of padding or truncating the input

data. In this section, E is still used as the output feature dimension of the convolutional layer but this is sometimes referred to as its number of *output channels*, similarly the input feature dimension D is often referred to as the number of *input channels*. As these are convolutional models they can be viewed similarly to *linear time-invariant (LTI)* systems (Proakis & Manolakis, 1996). Because of this, it is also sometimes the case that these are referred to as the number of input and output filters or just the number of filterbank features in the case of the output feature dimension. All of these terms are consistent and interchangeable but often vary significantly in the literature depending on the chosen domain, e.g. audio processing, image processing, biosignal processing and so on.

A 1D convolutional kernel of kernel size P has weights $\boldsymbol{\Theta}_{\text{1D}} \in \mathbb{R}^{E\times P\times D}$ and biases $\boldsymbol{\beta} \in \mathbb{R}^{E}$ where

$$\boldsymbol{\Theta}_{1D} = \left[\boldsymbol{\Theta}_1^\top, \ldots, \boldsymbol{\Theta}_E^\top\right]^\top \tag{2.35}$$

and $\boldsymbol{\Theta}_{\text{e}} \in \mathbb{R}^{P\times D}$. For convolutional layers, the input sequence of features is non-trivial to formulate as the *dilation factor* f and *stride* K hyperparameters alter what view of the input matrix is processed by the convolutional layer. The dilation factor controls whether and by how much the convolutional kernel skips intermediate features of an input sequence $\mathbf{Y} = \left[\mathbf{y}_1^\top, \ldots, \mathbf{y}_L^\top\right]^\top$, i.e. the ℓth convolution in a sequence using a kernel with dilation factor f and kernel size P would process the input subsequence

$$\mathbf{Y}_{\text{dil},\ell} = \left[\mathbf{y}_\ell^\top, \mathbf{y}_{\ell+f}^\top, \mathbf{y}_{\ell+2f}^\top, \ldots, \mathbf{y}_{\ell+(P-1)f}^\top\right]^\top \in \mathbb{R}^{P\times D}. \tag{2.36}$$

Note that this formulation omits the stride, which will be explained in the following. The stride controls the level of subsampling on the output sequence, i.e. with subsampling indices

$$\ell \in \left[1, 1+K, 1+2K, \ldots, 1+K\left\lfloor\frac{L}{K}\right\rfloor\right] \tag{2.37}$$

ignoring the kernel size P and dilation factor f (although note this is correct for the pointwise convolutional layers defined further down in this section). If dilation factor f and P are factored in then the subsampling indices become

$$\ell \in \left[1, 1+K, 1+2K, \ldots, 1+K\left\lfloor\frac{(L-f(P-1)}{K}\right\rfloor\right]. \tag{2.38}$$

The stride also changes the length of the output signal, such that the output signal is of length

$$L_{\text{out}} = \left\lfloor\frac{(L-f(P-1)}{K}\right\rfloor + 1. \tag{2.39}$$

Zero-padding is often used for a stride of $K = 1$ (i.e. when subsampling is undesirable) to

ensure that the input and output sequences are of the same length, $L_{\text{out}} = L$. Simply put, zero-padding is the practice of appending zeros to the beginning or end of a sequence. The view of the input matrix to the 1D convolutional layer is therefore defined as $\mathbf{Y}_{\text{1DView}} \in \mathbb{R}^{L_{\text{out}} \times P \times D}$ where

$$\mathbf{Y}_{\text{1DView}} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_{1+f} & \cdots & \mathbf{y}_{1+(P-1)f} \\ \mathbf{y}_{1+K} & \mathbf{y}_{1+f+K} & \cdots & \mathbf{y}_{1+(P-1)f+K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{1+K(L_{\text{out}}-1)} & \mathbf{y}_{1+K(L_{\text{out}}-1)+f} & \cdots & \mathbf{y}_{1+K(L_{\text{out}}-1)+(P-1)f} \end{bmatrix} \tag{2.40}$$

$$= \begin{bmatrix} \mathbf{Y}_{\text{dil},1} \\ \mathbf{Y}_{\text{dil},2} \\ \vdots \\ \mathbf{Y}_{\text{dil},1+K(L_{\text{out}}-1)} \end{bmatrix}. \tag{2.41}$$

Using this, the 1D convolutional operation is defined as

$$\mathcal{C}_{\text{1D}}(\mathbf{Y}, \mathbf{\Theta}_{\text{1D}}, \boldsymbol{\beta}, f, P, K) = \begin{bmatrix} \sum (\boldsymbol{Y}_{\text{dil},1} \odot \mathbf{\Theta}_1) & \sum (\boldsymbol{Y}_{\text{dil},1} \odot \mathbf{\Theta}_2) & \cdots & \sum (\boldsymbol{Y}_{\text{dil},1} \odot \mathbf{\Theta}_E) \\ \sum (\boldsymbol{Y}_{\text{dil},1+K} \odot \mathbf{\Theta}_1) & \sum (\boldsymbol{Y}_{\text{dil},1+K} \odot \mathbf{\Theta}_2) & \cdots & \sum (\boldsymbol{Y}_{\text{dil},1+K} \odot \mathbf{\Theta}_E) \\ \vdots & \vdots & \ddots & \vdots \\ \sum \left(\boldsymbol{Y}_{\text{dil},1+K(L_{\text{out}}-1)} \odot \mathbf{\Theta}_1\right) & \cdots & \cdots & \sum \left(\boldsymbol{Y}_{\text{dil},1+K(L_{\text{out}}-1)} \odot \mathbf{\Theta}_E\right) \end{bmatrix} + \boldsymbol{\beta} \tag{2.42}$$

where $\odot$ denotes the Hadamard product, $\sum(\cdot)$ is a short-hand for the total sum of the resulting matrix, i.e.

$$\sum (\boldsymbol{Y}_{\text{dil},\ell} \odot \mathbf{\Theta}_{\text{e}}) = \sum_{p=1}^{P} \sum_{d=1}^{D} (\boldsymbol{Y}_{\text{dil},\ell} \odot \mathbf{\Theta}_{\text{e}})_{p,d} \tag{2.43}$$

and the output is of shape $L_{\text{out}} \times E$. The higher dimensional convolutional layers thus have an output shape of $L_{\text{out},1} \times L_{\text{out},2} \times \ldots \times E$, i.e. an additional variable length dimension for each additional dimension of the convolutional layer. An often discussed quantity of convolutional layers is the receptive field (RF). This is the *width* of input frames from a sequence that are observed to produce one output frame. This quantity depends primarily on the kernel size P and dilation factor f. It is defined as

$$\mathcal{R}(P, f) = 1 + f(P - 1). \tag{2.44}$$

The receptive field can also be formulated for multiple stacked 1D convolutional layers by

starting at the final output layer and observing back through the network the minimum and maximum points in the input sequence the layer was able to process data from to produce one output frame. The formula for the output layer is the same as in (2.44) but for each additional layer lower in the network an additional $f(P-1)$ frames can be observed. This is formulated as

$$\mathcal{R}_{\text{total}}\left(\{P_1,\ldots,P_X\},\{f_1,\ldots f_X\}\right)=\mathcal{R}(P_X,f_X)+\sum_{i=1}^{X-1}f_i(P_i-1) \tag{2.45}$$

for a stack of X 1D convolutional layers with varying dilation factors f_i and kernel sizes P_i. This formulation and the concept of a receptive field is particularly relevant in Chapter 3, Chapter 4 and Chapter 5.

A number of variants of the standard convolutional kernel have been proposed. One used widely in this thesis is known as depthwise separable convolution (DS-Conv) (Chollet, 2017). The idea behind DS-Conv is to reduce the computational cost of *vanilla* convolutional operations shown in Eq. (2.42) by factorising out the operation to perform convolution first across the feature axis (referred to as *pointwise* convolution) and then across the temporal/sequential axis (referred to as *depthwise* convolution).

The depthwise convolution (D-Conv) layer groups the layer parameters by each output channel, where the term *groups* has a similar connotation to the feature groups described with respect to GN in (2.34). In D-Conv, the number of groups G is fixed to the number of input and output channels, i.e. $G=D=E$. The convolution operation is performed solely across the sequential axis for each group. The weight tensor for the operation is defined as $\boldsymbol{\Theta}_{\text{D-Conv}}=\left[\boldsymbol{\theta}_1^\top,\ldots,\boldsymbol{\theta}_D^\top\right]^\top\in\mathbb{R}^{D\times P}$ and the D-Conv operation for a 1D kernel is defined as

$$\mathcal{C}_{\text{D-Conv}}(\mathbf{Y},\boldsymbol{\Theta}_{\text{D-Conv}},\boldsymbol{\beta},f,P,K)=$$
$$\begin{bmatrix}\mathbf{y}_{\text{dil},1,1}\boldsymbol{\theta}_1 & \mathbf{y}_{\text{dil},1,2}\boldsymbol{\theta}_2 & \cdots & \mathbf{y}_{\text{dil},1,D}\boldsymbol{\theta}_D\\ \mathbf{y}_{\text{dil},1+K,1}\boldsymbol{\theta}_1 & \mathbf{y}_{\text{dil},1+K,2}\boldsymbol{\theta}_2 & \cdots & \mathbf{y}_{\text{dil},1+K,D}\boldsymbol{\theta}_D\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{y}_{\text{dil},1+1+K(L_{\text{out}}-1),1}\boldsymbol{\theta}_1 & \cdots & \cdots & \mathbf{y}_{\text{dil},1+1+K(L_{\text{out}}-1),D}\boldsymbol{\theta}_D\end{bmatrix}+\boldsymbol{\beta} \tag{2.46}$$

where

$$\mathbf{y}_{\text{dil},\ell,\text{d}}=\left[y_{\ell,\text{d}},y_{\ell+f,\text{d}},y_{\ell+2f,\text{d}},\ldots,y_{\ell+(P-1)f,\text{d}}\right]^\top\in\mathbb{R}^P. \tag{2.47}$$

The pointwise convolution (P-Conv) is a convolutional layer with kernel size $P=1$. Note that the dilation factor f is irrelevant here due to the kernel size of 1. This layer is equivalent to the feedforward layer defined in (2.23). In the 1D convolutional formula in (2.42) the P-Conv weights would be defined as $\boldsymbol{\Theta}_{\text{P-Conv}}\in\mathbb{R}^{E\times 1\times D}$. Note that if the middle dimension

is collapsed, then $\boldsymbol{\Theta}'_{\text{P-Conv}} \in \mathbb{R}^{E \times D}$, which is equivalent to the feedforward weights in (2.23) via its transpose, i.e. ${\boldsymbol{\Theta}'_{\text{P-Conv}}}^{\top} = \boldsymbol{\Theta}_{\text{ff}}$.

Ignoring the bias terms ($\boldsymbol{\beta}$), which are often omitted and replaced by LN layers (Luo & Mesgarani, 2019), the 1D convolutional kernel can be factorised as

$$\boldsymbol{\Theta}_{\text{1D}} = \begin{bmatrix} \boldsymbol{\Theta}^{\top}_{\text{D-Conv}} \odot \boldsymbol{\Theta}'_{\text{P-Conv,1}} \\ \boldsymbol{\Theta}^{\top}_{\text{D-Conv}} \odot \boldsymbol{\Theta}'_{\text{P-Conv,2}} \\ \vdots \\ \boldsymbol{\Theta}^{\top}_{\text{D-Conv}} \odot \boldsymbol{\Theta}'_{\text{P-Conv,E}} \end{bmatrix} \in \mathbb{R}^{E \times P \times D}. \tag{2.48}$$

Thus, the DS-Conv layer is defined as

$$\mathcal{S}\left(\mathbf{Y}, \boldsymbol{\Theta}_{\text{P-Conv}}, \boldsymbol{\Theta}_{\text{D-Conv}}, \boldsymbol{\beta}, f, K, P\right) = \mathcal{C}_{\text{1D}}\left(\mathcal{C}_{\text{D-Conv}}\left(\mathbf{Y}, \boldsymbol{\Theta}_{\text{D-Conv}}\right), f, K, P\right), \boldsymbol{\Theta}_{\text{P-Conv}}, \boldsymbol{\beta}, 1, 1, 1\right). \tag{2.49}$$

The motivation for using this factorized approach is a reduction in mode size and also the so-called *time-complexity*, i.e. the computational complexity as the sequence length increases. Ignoring bias terms as these are identical for both 1D convolution and DS-Conv, the parameter count of the 1D convolutional layer, is defined as

$$\mathcal{B}_{\text{Conv1D}}(P, D, E) = P \times D \times E \tag{2.50}$$

The parameter count of the P-Conv layer is defined as

$$\mathcal{B}_{\text{P-Conv}}(D, E) = D \times E \tag{2.51}$$

and of the D-Conv layer is defined as

$$\mathcal{B}_{\text{D-Conv}}(P, E) = P \times E. \tag{2.52}$$

Consequnetly, the parameter count of the DS-Conv layer is defined as

$$\mathcal{B}_{\text{DS-Conv}}(E, P, D) = E \times (P + D). \tag{2.53}$$

Thus the parameter count of the DS-Conv operation reduces the parameter count of the model by $\frac{E \times P}{E + P}$ when $E \gg P$ (Luo & Mesgarani, 2019). Another consequence of this is a reduction in computational expenditure, i.e. less mutiply-accumulate operations (MACs) for equal sequence lengths L when using DS-Conv. This is demonstrated in Figure 2.4 where it is shown that for all dimensions D and kernel sizes P the complexity of the standard 1D convolutional layer is significantly higher.

There are a number of other improvements which have been proposed to convolutional

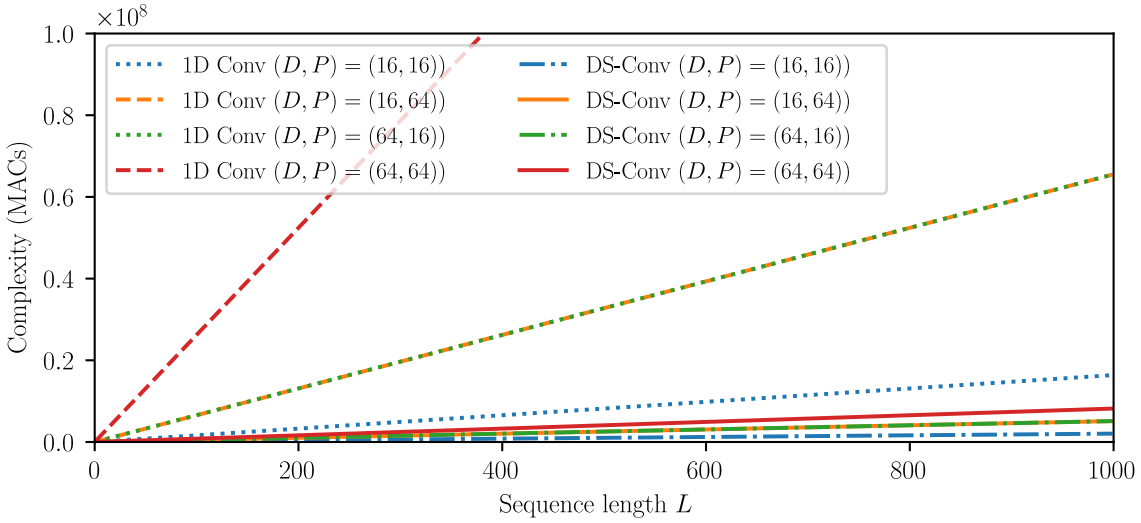


Figure 2.4: *The DS-Conv complexity compared to complexity of standard 1D convolutional layers as sequence length L increases. The input feature dimension is denoted as D and the kernel size is denoted as P.*

kernels such as gating mechanisms (Zhang et al., 2020b), dynamic convolution (Chen et al., 2020b) and deformable convolution (Dai et al., 2017). Dynamic convolution (Chen et al., 2020b) uses multiple weighted and summed convolution kernels to give more flexibility in the parameter weights based on specific input data. A variation on this is proposed in Chapter 4 and will be described in greater depth accordingly in this chapter. Deformable convolution (Dai et al., 2017) was proposed to enable convolutional kernels to have dynamic/adaptive receptive fields. This is achieved using a subnetwork to compute offsets which then alter the positional indexes of the parameter weights using linear interpolation operations. A more complete and mathematical description of deformable convolution is given later in Chapter 5 where the deformable temporal convolutional network (DTCN) is proposed.

2.5.7 Recurrent Layers

Recurrent neural network (RNN) layers were proposed as an extension of the feedforward layer described in Section 2.5.3 for modelling sequences of features as opposed to singular feature vectors in feedforward layers. They also have a benefit over convolutional layers in that they can model global context as opposed to the convolutional layer that can only process context within its receptive field.

The simplest RNN layer is a weighted sum of the current feature vector $\mathbf{y}_\ell$ in the sequence denoted $\mathbf{y}_\ell$ and the previous output of the RNN layer denoted $\mathbf{g}_{\ell-1}$. The layer is formulated as

$$\mathcal{Z}\left(\mathbf{y}_\ell, \mathbf{g}_{\ell-1}, \boldsymbol{\Theta}_{\text{ff}}, \boldsymbol{\Theta}_{\text{rec}}, \boldsymbol{\beta}\right) = \mathbf{g}_\ell = \mathbf{y}_\ell \boldsymbol{\Theta}_{\text{ff}} + \boldsymbol{\beta} + \mathbf{g}_{\ell-1} \boldsymbol{\Theta}_{\text{rec}} + \boldsymbol{\beta}_{\text{rec}} \tag{2.54}$$

where $\boldsymbol{\Theta}_{\text{ff}} \in \mathbb{R}^{D\times E}$ and $\boldsymbol{\Theta}_{\text{rec}} \in \mathbb{R}^{E\times E}$ are the parameter matrices of the feedforward and recurrent parts of the layer and $\boldsymbol{\beta}_{\text{rec}} \in \mathbb{R}^{E}$ are the biases for the recurrent part. A number of variants of RNNs have been proposed. Note that the longer the sequence is, the less weight is given to the earlier context because of the way the layer is formulated. One approach that can mitigate this somewhat is to use *bidirectional* layers that process the sequence in both directions, forwards, $\ell \in \{0, \ldots, L\}$, and backwards, $\ell \in \{L, \ldots, 0\}$. This ensures more even weighting at either extreme of the sequence but is still suboptimal for longer sequences. The long short term memory (LSTM) layer was proposed by Hochreiter & Schmidhuber (1997) to address this issue more directly in the mechanisms used inside the network *cell*. LSTMs use a memory cell that can retain historical information or release it from memory using two gating mechanisms in the cell. Another variant of RNN is the gated recurrent unit (GRU) (Cho et al., 2014) that aimed to achieve the same result as the LSTM but with a reduction in computation via an improved gating mechanism. Further research has been done in this direction but this is beyond the scope of this project as RNNs are not the main focus in this thesis. However, some notable examples for further reading include the light gated recurrent unit (LGRU) (Ravanelli et al., 2018) and the quasi-recurrent neural network (QRNN) (Bradbury et al., 2017).

Since the introduction of the non-causal Transformer model by Vaswani et al. (2017) (discussed in the following sections), RNN models have been replaced in many domains in order to achieve SOTA performance on benchmark datasets (Radford et al., 2023; Takase & Kiyono, 2023). This network structure outperforms RNNs in most cases but comes with an unfortunately high computational cost due to quadratic sequential complexity, for L indices in a sequence L^2 kernel operations are performed. More recently (at the time of writing this thesis), state-space models (Gu & Dao, 2023) have become prominent in the research field of sequence models. Linear state-space models, however, are RNN-like in terms of their causal structure but show promising performance improvements over the Transformer, notably so on audio processing tasks related to those in this thesis (Gu & Dao, 2023).

2.5.8 Attention & Transformer Layers

One of the most significant contributions to the DL field over the past decade has been the attention revolution and its consequences: dot-product attention (Luong et al., 2015), self-attention (SA) (Lin et al., 2017), multihead attention (MHA) (Vaswani et al., 2017) and Transformers (Vaswani et al., 2017). The idea behind the attention mechanism was to design a function or algorithm that would apply *more weight to more relevant features* and implicitly less weight to inconsequential or mostly irrelevant features. In practice, this typically means computing a correlation function across two vectors or axes and applying more weight to the features which have higher correlations. Other attention operations and networks do exist

that operate slightly differently, such as the computationally efficient squeeze-and-excite (SE) attention network Hu et al. (2018) introduced in Chapter 4, but these are not the focus of this section. In terms of computational expenditure, Transformer layers consume a lot of memory for computing the attention matrix and have quadratic time-complexity. Both of which are a drawback compared to RNNs. One key benefit of this structure however is the lack of any causality dependence meaning the layer computations are much more parallelizable and can have faster inference in some contexts.

This section focuses primarily on describing the Transformer and its related attention functions, starting from the ground up with dot-product attention. As the name suggests, dot-product attention is the result of performing the dot-product between two feature matrices or tensors, referred to as the *keys* and *querys* in its most common formulation. The terms come from machine translation (MT) tasks (Bahdanau et al., 2015) and bear little practical meaning in speech processing, perhaps making the model more complicated to comprehend at first. It is perhaps more useful to articulate these as input matrices, which are to be compared in terms of their correlation or similarity. Each of these is of shape $L_k \times D_k$ for the key and $L_q \times D_q$ for the query. The dot-product is computed as

$$\mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{L_q \times L_k}. \tag{2.55}$$

Note that this requires the constraint that $D_k = D_q$ so that the result is complete. The result of the dot-product is then passed through a softmax function, computed along the axis of length L_k, to normalize the weights on a probabilistic-like space (between 0 and 1). This gives a sequence of vectors of normalized weights. Typically, the dot-product is scaled by $\sqrt{D_k}$ as this has been found to improve training stability (Vaswani et al., 2017). This is where the term *scaled dot-product* when discussing Transformers comes from (Vaswani et al., 2017). In the final step, the dot-product is computed between the resulting weight matrix and the *value* matrix of input features that is to have attention applied to it, denoted $\mathbf{V} \in \mathbb{R}^{L_v \times D_v}$ where $L_v = L_q$ to be complete. Again, this terminology bears little meaning in most speech processing usage and is possibly more useful to think of the value matrix as simply as the input sequence of features that one wants to filter or add weight to using the output of (2.55). The final *scaled dot-product attention* mechanism is defined as

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_q}}\right)\mathbf{V}. \tag{2.56}$$

The terms *self-attention* and *cross-attention* have become common vernacular in DL research communities Radford et al. (2023). As the name suggests, self-attention is used when attention is computed from the same two inputs, i.e. $\mathbf{Q} = \mathbf{K}$. Conversely, *cross-attention* refers to the case where the two inputs are explicitly different, i.e. $\mathbf{Q} \neq \mathbf{K}$. From a

signal processing perspective, self-attention is akin to computing a pointwise (cross-feature) auto-correlation function of the input matrix, and cross-attention is akin to computing a pointwise cross-correlation function between the two input matrices. This will be demonstrated mathematically later in Section 6.1. Furthermore, this argumentation provides the basis for the work in Chapter 6 where attention mechanisms are employed to implicitly denoise noisy and reverberant encoded speech features.

2.5.8.1 Multihead Attention Layer

The following introduces multihead attention (Vaswani et al., 2017) as an extension to scaled dot product attention.

Linear projections and attention heads The notation of $\mathbf{V} \in \mathbb{R}^{L_v \times D_v}, \mathbf{K} \in \mathbb{R}^{L_k \times D_k}$ and $\mathbf{Q} \in \mathbb{R}^{L_q \times D_q}$ are also used in this section, as defined in the previous section. The first stage in the MHA layer is to linearly project the inputs into a lower dimensional space. This is achieved by multiplying the input sequences by 3 trainable weight matrices,

$$\mathbf{\Theta}_{\text{key},g} \in \mathbb{R}^{D_k \times A_k} \tag{2.57}$$

$$\mathbf{\Theta}_{\text{query},g} \in \mathbb{R}^{D_q \times A_q} \tag{2.58}$$

$$\mathbf{\Theta}_{\text{value},g} \in \mathbb{R}^{D_v \times A_v} \tag{2.59}$$

for each attention head $g \in \{1, ..., G_{\text{heads}}\}$ where G_{heads} is the number of attention heads and $A = D/G_{\text{heads}}, \{A, D\} \in \{\{D_k, A_k\}, \{D_q, A_q\}, \{D_v, A_v\}\}$ is the reduced dimension size. The motivation for reducing the dimensionality is that this retains roughly the same computational cost of using a single attention head with full dimensionality while allowing for using multiple attention mechanisms. The choice of G_{heads} is normally motivated by empirical analysis. Each of these weight matrices are used to compute $\mathbf{K}_g, \mathbf{Q}_g$ and $\mathbf{V}_g$ for each attention head $g \in \{1, ..., G_{\text{heads}}\}$ such that

$$\mathbf{K}_g = \mathbf{K}\mathbf{\Theta}_{\text{key},g} \in \mathbb{R}^{L_k \times A_k} \tag{2.60}$$

$$\mathbf{Q}_g = \mathbf{Q}\mathbf{\Theta}_{\text{query},g} \in \mathbb{R}^{L_q \times A_q} \tag{2.61}$$

$$\mathbf{V}_g = \mathbf{V}\mathbf{\Theta}_{\text{value},g} \in \mathbb{R}^{L_v \times A_v}. \tag{2.62}$$

For each attention head, the attention function is computed using (2.56) such that

$$\mathbf{D}_g = \mathcal{A}(\mathbf{Q}_g, \mathbf{K}_g, \mathbf{V}_g) \tag{2.63}$$

where $\mathbf{D}_g$ is the gth attention head.

Multihead Attention The final stage is connecting the attention heads by concatenating along the A length dimension and projecting the features using a linear layer defined by a weight matrix

$$\boldsymbol{\Theta}_{\text{out}} \in \mathbb{R}^{AG_{\text{heads}} \times D} = \mathbb{R}^{D \times D} \tag{2.64}$$

The combined concatenation and linear projection is defined by the Multihead Attention function

$$\mathcal{M}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{D}_1, \ldots, \mathbf{D}_{G_{\text{heads}}}] \, \boldsymbol{\Theta}_{\text{out}}. \tag{2.65}$$

2.5.8.2 Transformer Layers

Transformer layers are an applied use of multihead attention (MHA) in a larger network module with additional linear and normalization layers as well as what is known as positional encoding (PE). Originally proposed in (Vaswani et al., 2017), Transformers are normally composed of two-layer variants: Transformer encoders and Transformer decoders. The Transformer network is visualised in Figure 2.5. Each stack of Transformer layers is preceded by a PE function defined as

$$\mathcal{Q}(\ell, d) = \begin{cases} \sin\left(\frac{\ell}{10000^{\frac{2d}{D}}}\right) & d \in \{0, 2, 4, 6, 8, \ldots\} \quad d \leq D \\ \cos\left(\frac{\ell}{10000^{\frac{2d}{D}}}\right) & d \in \{1, 3, 5, 7, 9, \ldots\} \quad d \leq D \end{cases} \tag{2.66}$$

for $\ell \in \{1, \ldots, L\}$ in the case of the $\mathbf{Y} \in \mathbb{R}^{L \times D}$ feature sequence previously defined in (2.22). The PE function adds a sinusoidal shape to the input features to provide positional sequence information in the network (Vaswani et al., 2017). Although the MHA layer models global context, it does not capture positional information explicitly, hence the need for PE.

Transformer encoder layers are primarily composed of a multihead self-attention (MHSA) layer with a residual connection, a normalisation layer, a linear layer, and another normalization layer. The residual connection is where the input and output of the layer are summed at the output; this helps with gradient stability in training (He et al., 2016). When concerning speech processing, particularly speech enhancement and separation, it is common to only see the Transformer encoder layers used, i.e. the transform from $\mathbf{Y}$ to $\mathbf{Y}'$ in Figure 2.5. This is because the input and output features are encoded from and decoded to the same domain. Typically, the inputs and outputs are also of the same dimension and size.

For tasks such as speech recognition, it is more common to see the Transformer decoder being used in addition to Transformer encoder. This is due to highly mismatched sequence lengths of speech audio signals and text and very different input and output feature spaces, e.g. audio spectral features as inputs and probabilities of character labels as outputs. As the output dimension is governed by the input $\mathbf{V}$ in (2.65), this is the reasoning behind

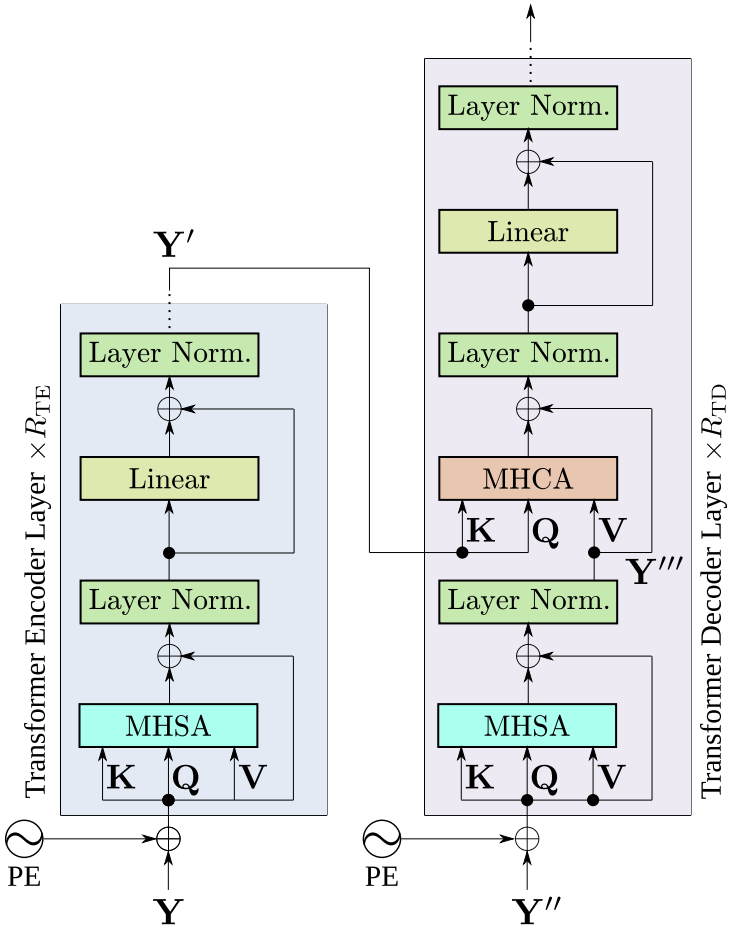


Figure 2.5: *Diagram of a Transformer network, consisting of R_{TE} Transformer encoder layers and R_{TD} Transformer decoder layers.*

cross-attention in the following. The decoder Transformer layer is composed of a MHSA layer with a residual connection, a normalization layer, a multihead cross-attention (MHCA) layer with a residual connection, another normalization layer, a linear layer and a final normalization layer. The first decoder operates on a different sequence $\mathbf{Y}'' \in \mathbb{R}^{L \times D'}$ from the input to the Transformer encoder $\mathbf{Y} \in \mathbb{R}^{L \times D}$ and its output $\mathbf{Y}' \in \mathbb{R}^{L \times D'}$. This input $\mathbf{Y}''$ governs the decoder's final output dimension, i.e. D', will be, but it must be of the same sequence length as $\mathbf{Y}'$. The output of the MHSA part of the decoder is denoted $\mathbf{Y}'''$. The next MHA layer is cross-attentive, the attention weights are computed on the output from the encoder layers, but attention weights are still applied to the output of the preceding MHA layer, i.e. $\mathcal{M}(\mathbf{Y}', \mathbf{Y}', \mathbf{Y}'''')$, where $\mathbf{Y}''''$ is the output of the decoder layer, or in the case of the first decoder layer the input sequence, i.e. $\mathcal{M}(\mathbf{Y}', \mathbf{Y}', \mathbf{Y}'')$.

2.6 Loss Functions & Permutation Solving

2.6.1 Loss Functions

A number of loss functions have been proposed for speech separation and enhancement (Close et al., 2023; Kolbæk et al., 2020; Braun & Tashev, 2021; Close et al., 2024). Typically, most loss functions compute some kind of *difference* function between the estimated speech signal in the time domain or time-frequency domain.

A commonly used loss function for speech separation and enhancement tasks is the averaged L2 norm (Défossez et al., 2021), i.e. MSE between time-domain signals,

$$\mathcal{L}_{\text{TD-L2}}(\mathbf{s}, \hat{\mathbf{s}}) = \frac{1}{L_s} \sum_{i=1}^{L_s} (\hat{s}[i] - s[i])^2 = \frac{1}{L_s} \|\mathbf{s} - \hat{\mathbf{s}}\| . \tag{2.67}$$

Alternatively, the L1 norm can also be used,

$$\mathcal{L}_{\text{TD-L1}}(\mathbf{s}, \hat{\mathbf{s}}) = \frac{1}{L_s} \sum_{i=1}^{L_s} |\hat{s}[i] - s[i]| = \frac{1}{L_s} |\mathbf{s} - \hat{\mathbf{s}}| . \tag{2.68}$$

Similarly, the MSE of the absolute values of short-time spectral representations (e.g. short-time Fourier transforms (STFTs)), can be used as a loss function, e.g.

$$\mathcal{L}_{\text{ST-MSE}}\left(\mathbf{S}, \hat{\mathbf{S}}\right) = \frac{1}{L_{\mathbf{S}} N_{\text{spec}}} \sum_{\ell=1}^{L_{\mathbf{S}}} \sum_{n=1}^{N_{\text{spec}}} \left(|\hat{\mathbf{S}}_{\ell,n}| - |\mathbf{S}_{\ell,n}|\right)^2 \tag{2.69}$$

where N_{spec} is the number of spectral bins (e.g. real fast Fourier transform (FFT) size). Similar to the time-domain losses in the previous paragraph, the L1 of the absolute magnitude

spectra distance can be used for computing the loss, i.e.

$$\mathcal{L}_{\text{ST-MAS}}\left(\mathbf{S}, \hat{\mathbf{S}}\right) = \frac{1}{L_{\mathbf{S}} N_{\text{spec}}} \sum_{\ell=1}^{L_{\mathbf{S}}} \sum_{n=1}^{N_{\text{spec}}} \left| |\hat{\mathbf{S}}_{\ell,n}| - |\mathbf{S}_{\ell,n}| \right| \tag{2.70}$$

Other time-domain loss functions have been popularized in recent years, in particular, the SISDR loss (Luo & Mesgarani, 2018b, 2019; Luo et al., 2020; Subakan et al., 2021; Chen et al., 2020a). As will be discussed in Section 2.7, SISDR is the main metric used for evaluating speech separation models in more recent research, including the contents of this thesis. SISDR is based of the signal-to-distortion ratio (SDR) function, i.e.

$$\text{SDR}\left(\mathbf{s}, \boldsymbol{\nu}_{\text{dist}}\right) = 20 \log \left(\frac{\|\mathbf{s}\|}{\|\boldsymbol{\nu}_{\text{dist}}\|} \right) \tag{2.71}$$

where $\boldsymbol{\nu}_{\text{dist}}$ is the vector the time domain disortion signal. In the case of the distortion signal in training a speech enhancement or separation system this equates to $\boldsymbol{\nu}_{\text{dist}} = \hat{\mathbf{s}} - \mathbf{s}$. The SISDR loss function is thus defined as

$$\mathcal{L}_{\text{SISDR}}\left(\hat{\mathbf{s}}, \mathbf{s}\right) := -10 \log_{10} \frac{\|\mathbf{s}_{\text{target}}\|^2}{\|\boldsymbol{\nu}_{\text{sdist}}\|^2}, \tag{2.72}$$

where

$$\mathbf{s}_{\text{target}} = \frac{\langle \hat{\mathbf{s}}, \mathbf{s} \rangle \mathbf{s}}{\|\mathbf{s}\|^2}. \tag{2.73}$$

in (2.72) is the scale invariant target speech and

$$\boldsymbol{\nu}_{\text{sdist}} = \hat{\mathbf{s}} - \mathbf{s}_{\text{target}} \tag{2.74}$$

is the residual distortion present in the estimated speech after rescaling the reference. The value $\mathbf{s}_{\text{target}}$ is a rescaled version of $\mathbf{s}$ to make the difference (or distortion) calculation $\boldsymbol{\nu}_{\text{dist}}$ invariant to scale mismatches between the estimate $\hat{\mathbf{s}}$ and references $\mathbf{s}$. The term *scale* here refers to amplitude scaling, i.e. the scale invariance is introduced to make the loss term more agnostic to the difference in signal energy between the reference signal and the estimated signal. While this results in faster convergence in training, it does often result in estimated signals being in abnormal amplitude ranges (e.g. $\hat{s}[i] > 1$), but this can be handled via clipping functions on the model output or regularizing loss terms.

2.6.2 Permutation Invariant Training

Determining which speaker should be assigned to which output channel $\hat{s}_c[i]$ is a non-trivial problem and often referred to as the *permutation problem* (Huang et al., 2019; Kolbaek et al., 2017). Some proposed approaches to solving this problem have involved assigning the louder

speaker or male speakers to the first output channel (Huang et al., 2019). However, these solutions start to fail when the speaker energies are very similar, or the speakers are of the same gender (Huang et al., 2019).

PIT solves this permutation assignment problem in an agnostic fashion by matching the targets to the estimates rather than trying to force the estimator to match estimated signals to the pre-assigned reference signals. The permutation-solving formula in PIT is defined as

$$\hat{\phi} = \underset{\phi \in \mathbf{\Phi}}{\arg\min} \sum_{c=1}^{C} \mathcal{L}\left(\hat{\mathbf{s}}_c^{(\phi)}, \mathbf{s}_c\right) \tag{2.75}$$

where ϕ is some permutation in the set $\mathbf{\Phi}$ of $\Phi = C!$ possible permutations where $\hat{\mathbf{s}}_c^{(\phi)}$ denotes the ϕth possible permutation of $\hat{\mathbf{s}}_c$. $\mathcal{L}$ denotes an arbitrary loss function here but in the specific case of this chapter (and many proceeding chapters) $\mathcal{L} = \mathcal{L}_{\text{SISDR}}$. $\hat{\phi}$ denotes the estimated best-matching permutation according to the formula.

2.7 Evaluation Metrics

A number of evaluation metrics have been proposed for analysing speech separation and enhancement models (Goetze et al., 2014; Kolbæk et al., 2020; Close et al., 2022, 2023; Liu et al., 2015; Luo & Mesgarani, 2018b; Tzinis et al., 2022b; Cord-Landwehr et al., 2022). These vary in complexity from simple signal and spectral level difference measures to measures that emphasise the *quality* or *intelligibility* of separated speech (Deng et al., 2020). Quality typically refers to naturalness, pleasantness and clarity of speech, while intelligibility refers to how understandable the speech is (Gul et al., 2022).

As with evaluating most DL models in other domains, the loss function $\mathcal{L}$ used for training can similarly provide insights about overall model performance. This is typically an approach used within speech enhancement and separation research but often with supplementary metrics that provide greater insights. In the case of speech separation, it is very common in recent research trends to see **SISDR** (2.72) being used both as an objection function and a primary evaluation metric (Luo & Mesgarani, 2018b; Rixen & Renz, 2022; Wang et al., 2023b). Typical ranges for SISDR in speech separation evaluation are within -20dB to 30dB (Maciejewski et al., 2020; Wang et al., 2023a). Subjectively, the just noticeable difference value of the metric is a value is around values excess of 2dB SISDR difference, also referred to in this thesis as Δ SISDR.

The **SDR** evaluation metric was previously introduced in (2.71). SDR is a generalized SNR metric that measures the amount of energy in the signal compared with the energy in the combined residual noise, artefacts and interference. SDR has been widely used in assessing source separation models in general (Stoller et al., 2018; Luo & Mesgarani, 2019; Tzinis et al.,

2022b).

PESQ was proposed by (Rix et al., 2001a) as an objective measure for speech quality assessment. The design of PESQ is supposed to offer similar results to mean opinion score (MOS) by using psychoacoustically motivated filter models. The measure ranges from -0.5 to 4.5, with -0.5 being considered the lowest quality. PESQ is often used for assessing general denoising and dereverberation tasks. It has also been used for assessing speech separation performance (Wang et al., 2014; Deng et al., 2020). **Perceptual objective listening quality assessment (POLQA)** is a proposed improvement to PESQ by Gaoxiong & Wei (2012) for wideband and narrowband speech but it has not seen as much popular use in the speech enhancement literature.

Short-time objective intelligibility (STOI) is an intelligibility metric proposed by (Taal et al., 2010) which uses correlation ratios between clean and degraded signals to assess the intelligibility of the degraded signal with a score between 0 and 1. STOI has been commonly used for assessing general speech enhancement tasks but has also been used in assessing speech separation models (Deng et al., 2020). **ESTOI**, proposed in (Jensen & Taal, 2016), is an extension of STOI that has been widely used in speech ehancement literature (Li et al., 2020; Naithani et al., 2018)

In some areas, especially when speech enhancement and separation systems are being utilised for downstream speech recognition tasks, **WER** is often used. In its simplest form, word error rate (WER) computes the word-edit distance between the output utterance of an ASR model and a reference transcription. Multi-speaker implementations of this metric have also been proposed more recently (Sklyar et al., 2022; Watanabe et al., 2020) and are used extensively in Chapter 10.

SRMR (Falk et al., 2010; Santos et al., 2014) is a measure that is able to estimate reverberation energy in speech signals. It has seen use in a variety of speech enhancement literature (Fu et al., 2022; Ernst et al., 2018).

A less commonly used in the speech separation literature but highly useful metric is mean opinion score (MOS) (ITU-T Rec. P.800.1) (Möller & Köster, 2017; ITU, 2003). MOS is a subjective metric which uses real listeners to assess speech quality. Listeners are asked to give ratings between 1 & 5, and the average value is obtained as the final metric.

2.8 Time-domain Audio Separation Networks (TasNets)

The ever-increasing accessibility of high-performance computing resources has led to more memory-consuming and computationally complex DL models. This is true in many machine learning domains including speech separation (Vaswani et al., 2017; Tzinis et al., 2022b).

In speech enhancement research, which here includes speech separation and extraction, denoising, dereverberation, and beamforming technology (among others), time-domain ap-

proaches (as opposed to STFT or frequency-domain based methods) have gained much-increased interest in recent years due to their notable performance improvements on (in particular) speech separation benchmarks (Luo & Mesgarani, 2018b, 2019; Subakan et al., 2021; Zhao & Ma, 2023). Luo & Mesgarani (2018b) published a paper on time-domain audio separation networks (TasNets) which drew from ideas in previous research on non-negative matrix factorization for speech enhancement as well as SOTA speech separation models using DNNs (Le Roux et al., 2015; Kolbaek et al., 2017; Wang et al., 2018). In their technique, they train a convolutional layer to learn encoded mixture weights (similar to the basis signals learned in NMF models), which are then processed by a BLSTM network to compute "speaker masks". The encoded features are then *masked* accordingly to attain encoded representations for each speaker. These representations are then transformed back into the time domain using a transposed convolutional layer. They further refined their design into a fully convolutional architecture by using temporal convolutional networks (TCNs) as a replacement for the BLSTM layers in Luo & Mesgarani (2018b) with superior performance; this is later described in depth in Section 2.9. The original bidirectional long short term memory (BLSTM) network in Luo & Mesgarani (2018b) is often referred to simply as TasNet. In this thesis the term TasNet is used to apply to a whole family of models that use this same network structure but typically have a different mask estimation network only (Luo & Mesgarani, 2019; Subakan et al., 2021; Rixen & Renz, 2022; Tzinis et al., 2022b).

An NMF based problem formulation informs the intuition behind TasNet models and thus is elaborated on in the following. It is assumed that there exists some matrix of basis functions $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N]^\top \in \mathbb{R}^{N \times J}$ where N is the number of basis functions to be used and that a block of size J of signal $x[i]$ denoted $\mathbf{x}_\ell \in \mathbb{R}^J$ can be represented by some weighted sum of the basis functions such that

$$\mathbf{x}_\ell = \mathbf{w}_\ell \mathbf{B} \tag{2.76}$$

where $\mathbf{w}_\ell \in {\mathbb{R}_+}^N$ is non-negative. It is also therefore assumed that it is possible to find some weight vector $\mathbf{v}_c \in [0, \infty)^{1 \times N}$ for each of the C sources in $\mathbf{x}$ such that

$$\mathbf{s}_c = \mathbf{v}_c \mathbf{B} \qquad \mathbf{w} = \sum_{c=1}^{C} \mathbf{v}_c. \tag{2.77}$$

Note that this formulation of the model assumes no noise or reverberation is present in the signal $\mathbf{x}$. In later sections, this assumption is dropped.

Using this formulation, it is possible to derive mask-like vectors $\mathbf{m}_c = (\mathbf{v}_c \oslash \mathbf{w}) \in \mathbb{R}^N$ for each of C speakers that represent the relative amount of energy each speaker c is contributing to the energy for each basis function representation in $\mathbf{w}$. This can be derived by performing

elementwise operations across the vectors such that

$$\mathbf{w} = \sum_{c=1}^{C} \mathbf{w} \odot (\mathbf{v}_c \oslash \mathbf{w}) := \mathbf{w} \odot \sum_{c=1}^{C} \mathbf{m}_c \tag{2.78}$$

where $\odot$ and $\oslash$ are the elementwise multiplication and division respectively and therefore

$$\mathbf{v}_c = \mathbf{m}_c \odot \mathbf{w}. \tag{2.79}$$

The original TasNet is composed of 3 subnetworks: an encoder, mask estimation network and decoder, which are visualised in Figure 2.6. Time domain speech mixtures are first encoded

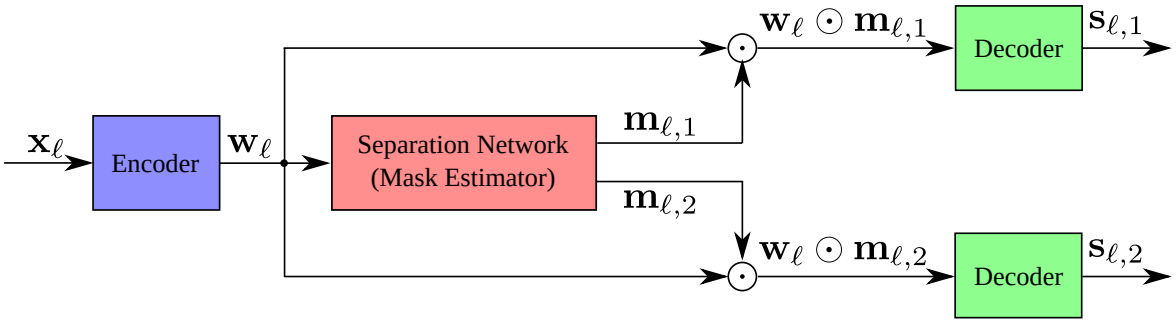


Figure 2.6: *Generalised TasNet Model Schematic (exemplarily shown for two channels, $C = 2$)*

using a 1D convolutional layer and an activation function (or gated activation function in the case of the original TasNet (Luo & Mesgarani, 2018b)) to produce a sequence of features. The mask estimation takes the sequence of features as an input and produces two sequences of masks. These masks are then multiplied with the encoded features to produce an encoded feature sequence for each speaker. The sequence is then decoded back into a time domain using a transposed 1D convolutional layer. The original TasNet used a reasonably simple deep BLSTM network for the mask estimation component. At the time, this model gave SOTA performance on the anechoic WSJ0-2Mix benchmark.

2.9 Conv-TasNet

In this section, the Conv-TasNet model is described in depth. The Conv-TasNet model is the main baseline model for Chapter 3 to Chapter 5. It is a comparatively lightweight separation model that has been utilised in a number of areas relating to speech separation (Ochiai et al., 2020; Delcroix et al., 2020; Mošner et al., 2022). It was proposed as an improvement to the LSTM TasNet model. The model replaces the deep LSTM with a so-called temporal convolutional network (TCN). TCNs are constructed of stacked convolutional blocks of

increasing dilation factors. Each additional block in the stack increases the dilation factor by a factor of 2. The stack is also repeated to reduce the extent to which the receptive field is increased with each additional block, i.e. at the bottom of each stack, the dilation factor is smallest ($f = 1$).

The Conv-TasNet model gained a lot of popularity in the research community due to its reliable performance combined with a comparatively small computational footprint (typically measured in terms of floating operations per second (FLOPS) or MACs) and small model size, making it useful for downstream tasks such as ASR (Ochiai et al., 2020), which typically have a large computational footprint of their own. Previous work has investigated adaptations to Conv-TasNet with additional modifications such as multi-scale convolution and gating mechanisms applied to the outputs of convolutional layers, but these significantly increase the computational complexity (Zhang et al., 2020b). While these modifications improve performance, they come at a significant cost to model size (Zhang et al., 2020b) and overall computational complexity.

In other applications, Kinoshita et al. (2020) adapted the Conv-TasNet architecture to denoise speech for ASR tasks with notable reductions in relative WER improvement on both simulated and real data. This approach also consistently outperformed the same network using frequency rather than time domain features. Ochiai et al. (2020) proposed an minimum-variance distortionless response (MVDR) beamforming approach that used Conv-TasNet to estimate power spectral densities for speakers and noise in the signal. Another multichannel speech separation approach based on TasNet was proposed in Gu et al. (2019). This approach computed inter-phase differences (IPDs) and provided them as an input to the TCN separation network. The multichannel approach led to significant increases in SISDR performance for 2-speaker separation on the WSJ0-2Mix corpus. Luo et al. (2019) proposed a low-latency adaptive beamforming model that adapted the Conv-TasNet model into a multi-channel enhancement and separation system that performed denoising and separation. This approach improved performance for both joint denoising and dereverberation, and speech separation of noisy echoic speech over using either model separately. Furthermore, it showed the adaptability of the Conv-TasNet model for more than just monaural enhancement and separation systems.

The remainder of this section describes the Conv-TasNet speech separation model in detail. As with the original LSTM TasNet model, the network is primarily composed of three sub-networks: an encoder network, a mask estimation network, and a decoder network. The generalised schematic of the overall network structure is the same as that visualised in Figure 2.6 where the network is configured to produce $C = 2$ output speaker signals. The mask estimation network formulated in this section follows the implementation that can be found in the open-source SpeechBrain (Ravanelli et al., 2021) and ESPnet (Li et al., 2021) software toolkits. This implementation differs slightly from the original model proposed by (Luo & Mesgarani, 2019) by leaving out a set of skip connections (SCs) and replacing cLN

with CN. The removal of the SCs and its consequences are discussed in further detail in Section 2.9.7.

In the same way as the original TasNet model (Luo & Mesgarani, 2018b), the encoder network transforms the overlapping blocks of the time domain signal $\mathbf{x}_\ell$ into a sequence of features. The encoder is a convolutional neural network where the layer weights are learned in an end-to-end (E2E) fashion. Another way this is sometimes described is a *learned filterbank* (Pariente et al., 2020; Ditter & Gerkmann, 2020). The mask estimation network takes the output of the encoder network $\mathbf{w}_\ell$ and uses it to estimate a set of mask-like vectors $\mathbf{m}_{\ell,c}$ for each of the C speakers. These mask-like vectors are then multiplied with the encoded signal vector $\mathbf{w}_\ell$, producing a masked weight vector for each speaker. The decoder in the original Conv-TasNet approach (Luo & Mesgarani, 2019) is a transposed 1D convolutional layer that decodes the masked encoded mixture, $\mathbf{w}_\ell \odot \mathbf{m}_{\ell,c}, c \in \{1, \ldots, C\}$, back into the time domain to result in C separated source estimates $\hat{\mathbf{s}}_{\ell,c}$. The goal of the decoder is theoretically to perform the inverse function of the encoder although this is not explicitly enforced in the network structure (Luo & Mesgarani, 2018b, 2019).

2.9.1 Encoder

The first stage of the network is to encode the input mixture signal $x[i]$. The mixture signal $x[i]$ (defined in (2.2)) of length L_x is processed in $L_{\mathbf{x}}$ blocks of length J with a 50% overlap and block index ℓ defined as

$$\mathbf{x}_\ell = [x[0.5(\ell-1)J], \ldots, x[0.5(1+\ell)J-1]]\,. \tag{2.80}$$

The encoder is constructed using a 1D convolutional filter of kernel size J with 1 input channel and N output channels and a nonlinear encoder activation layer denoted by $\mathcal{H}_{\text{enc}} : \mathbb{R}^{\cdots \times N} \to \mathbb{R}^{\cdots \times N}$. For a piece of audio of length L_x this results in $L_{\mathbf{x}}$ frames and N output channels such that the network produces $L_{\mathbf{x}}$ encoded mixture vectors $\mathbf{w}_\ell \in \mathbb{R}^{1\times N}$ given by

$$\mathbf{w}_\ell = \mathcal{H}_{\text{enc}}\left(\mathbf{x}_\ell \mathbf{U}\right) \tag{2.81}$$

where $\mathbf{U} \in \mathbb{R}^{J\times N}$ represents a matrix of the trainable convolutional weights. In the implementation used in this section, the nonlinear activation used is chosen as a ReLU function, i.e. $\mathcal{H}_{\text{enc}}(\mathbf{x}_\ell) = \mathcal{H}_{\text{ReLU}}(\mathbf{x}_\ell)$. The encoded signal mixture for all frames ℓ is defined as $\mathbf{W} \in \mathbb{R}^{L_{\mathbf{x}}\times N}$ such that

$$\mathbf{W} = \left[\mathbf{w}_1^\top, \ldots, \mathbf{w}_{L_{\mathbf{x}}}^\top\right]^\top. \tag{2.82}$$

2.9.2 Channel Sorting for Visualising Encoded Representations

The following describes a proposed approach for visualizing the encoded representations of TasNet models originally published in Ravenscroft et al. (2022a). While time-frequency approaches for speech separation based on masking spectrogram representations are often easy to interpret, for visualization of the encoded signal $\mathbf{W}$, sorting over the output convolutional channels n is beneficial. When visualising the encoded representations in this work, the encoded signals' channels are thus reordered according to the sorting algorithm defined in Algorithm 1 based on depthwise Euclidean distance. In the Conv-TasNet paper, (Luo & Mesgarani, 2019) propose using unweighted pair group method with arithmetic mean (UPGMA) to sort the channels by Euclidean filter similarity. The proposed Algorithm 1 was found to be preferable in many cases to (Luo & Mesgarani, 2019)'s approach as it leads to a less granular representation with most of the speech energy being located in the lower region of the representation, making it easier to observe lower energy noisier regions within the encoded signal. By this, the proposed channel sorting algorithm results in visualisations more similar to well-known spectrogram-like time-spectral representations. The key difference of the proposed sorting algorithm is that (Luo & Mesgarani, 2019)'s method uses filter similarity to sort channels whereas the proposed method sorts channels according to encoded feature similarity. The use of UPGMA which is based on a clustering approach to sort the channels, is also not clearly motivated by (Luo & Mesgarani, 2019); hence, in the proposed approach, the channels are sorted by decreasing similarity from the most similar channels measured in Euclidean feature similarity. This is premised on the assumption that the most similar channels will contain the most amount of speech energy. The strength of this assumption gradually weakens as the signal-to-noise ratio (SNR) of the encoded signal decreases, particularly as it passes 0dB SNR. The distance matrix $\mathbf{E}$ in line 1 of Algorithm 1 is composed of Euclidean distances between

Algorithm 1 Channel sorting algorithm

Input: $\mathbf{W} = [\mathbf{w}_1^\top, \ldots, \mathbf{w}_{L_\mathbf{x}}^\top] \in \mathbb{R}^{N \times L_\mathbf{x}}$ `# cf.` (2.82)

1: $\mathbf{E} \leftarrow \text{Distance}(\mathbf{W}, \mathbf{W}) \in \mathbb{R}^{N \times N}$ `# cf.` (2.83)
2: $\mathbf{E} \leftarrow \mathbf{E} + \mathbf{I} \cdot \max(\mathbf{E})$
3: $\{v, b\} \leftarrow \text{argmin}(\mathbf{E})$
4: $\mathbf{E} \leftarrow \mathbf{E} - \mathbf{I} \cdot \max(\mathbf{E})$
5: $\mathbf{v} \leftarrow \mathbf{E}_{v,:}$
6: $\mathbf{z} \leftarrow \text{argsort}(\mathbf{v})$
7: **return z**

the encoded channel representations, calculated element-wise by

$$\mathbf{E}_{i,j} = \|\mathbf{W}_{:,n_i} - \mathbf{W}_{:,n_j}\|_2 \tag{2.83}$$

$$= \sqrt{\sum_{\ell=1}^{L_{\mathbf{x}}} (\mathbf{W}_{\ell,n_i} - \mathbf{W}_{\ell,n_j})^2} \tag{2.84}$$

with $\mathbf{w}_{:,n_i}$ and $\mathbf{w}_{:,n_j}$ being the channels i and j, respectively, i.e. the i^{th} and j^{th} row of $\mathbf{W}$. Since the Euclidean distances are zero on the main diagonal of $\mathbf{E}$, these are replaced by the maximum value of $\mathbf{E}$ in line 2. Here, $\mathbf{I}$ is the identity matrix. Next, the most similar channels are identified in line 3. These are enumerated as the v^{th} and b^{th} channel. Zeros on the main diagonal of $\mathbf{E}$ are restored in line Algorithm 4 to make sorting by distance possible for any given row in $\mathbf{E}$. In line 5 the v^{th} row of $\mathbf{E}$ is selected. This row contains every distance between all N channels and channel v such that $\mathbf{v} \in \mathbb{R}^N$. The indices of vector $\mathbf{v}$ are then sorted in order of increasing distance from channel v such that $\mathbf{z} = \text{argsort}(\mathbf{v}) = [v, b, \ldots, \text{argmax}(\mathbf{v})]$.

A comparison of the original encoded NRSM signal with its channels reordered according to Algorithm 1 can be seen in Figure 2.7. It can be seen that the right panels of Figure 2.7 are closer to what you would expect from a spectrogram representation and thus allow for more insight when analysing signals and masking. By contrast, the method used by Luo and Mesgarani in (Luo & Mesgarani, 2019) seemingly provides a large number of small clusters containing speech energy across the entire vertical axis. Note that values larger than $0.05 \times \max(\mathbf{W})$ in Figure 2.7 have been normalized to 1 to deal with a few extraneously large values (which are less than 1 % of the values).

2.9.3 Mask Estimation Network

The separation network is visualised in Figure 2.8. It uses a TCN which consists of X layers of convolutional blocks (horizontal and coloured in Figure 2.8 (a)) which are repeated R times (vertical in Figure 2.8 (a)). The features are first normalized using CN as defined in (2.32). Following this, a P-Conv layer acts as a bottleneck layer and produces B channels as input for the successive convolutional blocks. At the output of the mask estimation network, a set of masks are produced in a single vector, one each speaker at each frame (cf. also Section 2.9.3.3). This is done using a single P-Conv that changes feature dimension from B to CN.

2.9.3.1 Convolutional Blocks

Each of the convolutional blocks consists of a P-Conv layer proceeded by a DS-Conv operation as visualized in Figure 2.8 (b) resulting in H channels within the convolutional block. Each subsequent convolutional block has an increasing dilation factor $f = 2^0, 2^1, \ldots, 2^{X-1}$ which widens the temporal context of the network for every additional block. This implementation

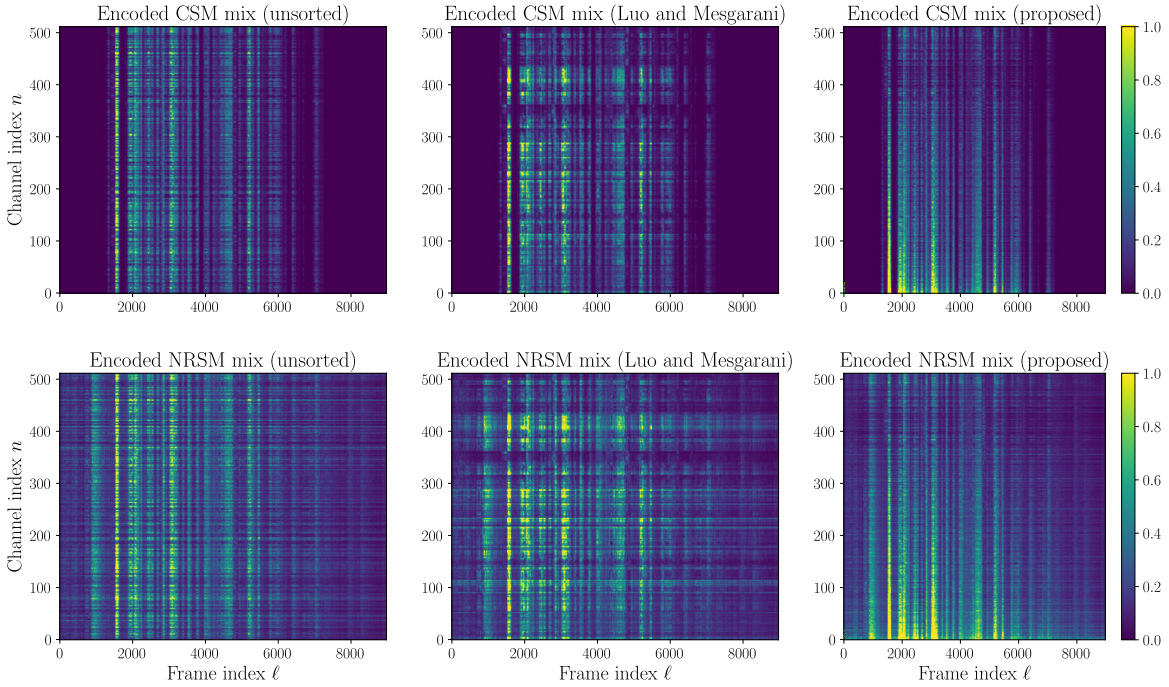


Figure 2.7: *Encoded signal matrix* $\mathbf{W}$ *for clean speech mixtures (CSM) (top) and noisy reverberant speech mixture (NRSM) (bottom). From left to right: unsorted; sorted using (Luo & Mesgarani, 2019)'s method; sorted using the proposed method in Algorithm 1.*

of the Conv-TasNet TCN follows that which is used in popular research frameworks such as SpeechBrain[1] (Ravanelli et al., 2021) and ESPnet (Watanabe et al., 2018; Li et al., 2021). This implementation is in contrast to the original Conv-TasNet proposed by (Luo & Mesgarani, 2019) which includes an additional skip connection from a parallel convolutional layer at the output of the convolutional blocks. As will be reported in Table 2.1, these skip connections are redundant; adding significantly to the model size and computational expenditure with no positive impact on model performance.

Conv-TasNet was originally proposed in both causal and non-causal implementations. In the causal implementation cumulative layer normalization is proposed by (Luo & Mesgarani, 2019) for the normalization layers in the convolution blocks. In the implementations and results in the following sections, the focus is on the non-causal model, which utilises gLN (2.31) for normalizing intermediate layers inside the convolutional blocks.

A PReLU activation function (2.28) is used after the initial P-Conv as well as the in the DS-Conv, denoted by $\mathcal{S}$ in Figure 2.8 (b), cf. also (2.49). The TCN takes an $L_{\mathbf{x}} \times N$ dimensional input and produces a $L_{\mathbf{x}} \times CN$ dimensional output. The input sequences to

[1]Conv-TasNet implementation in SpeechBrain: `https://github.com/speechbrain/speechbrain/blob/develop/speechbrain/lobes/models/conv_tasnet.py`

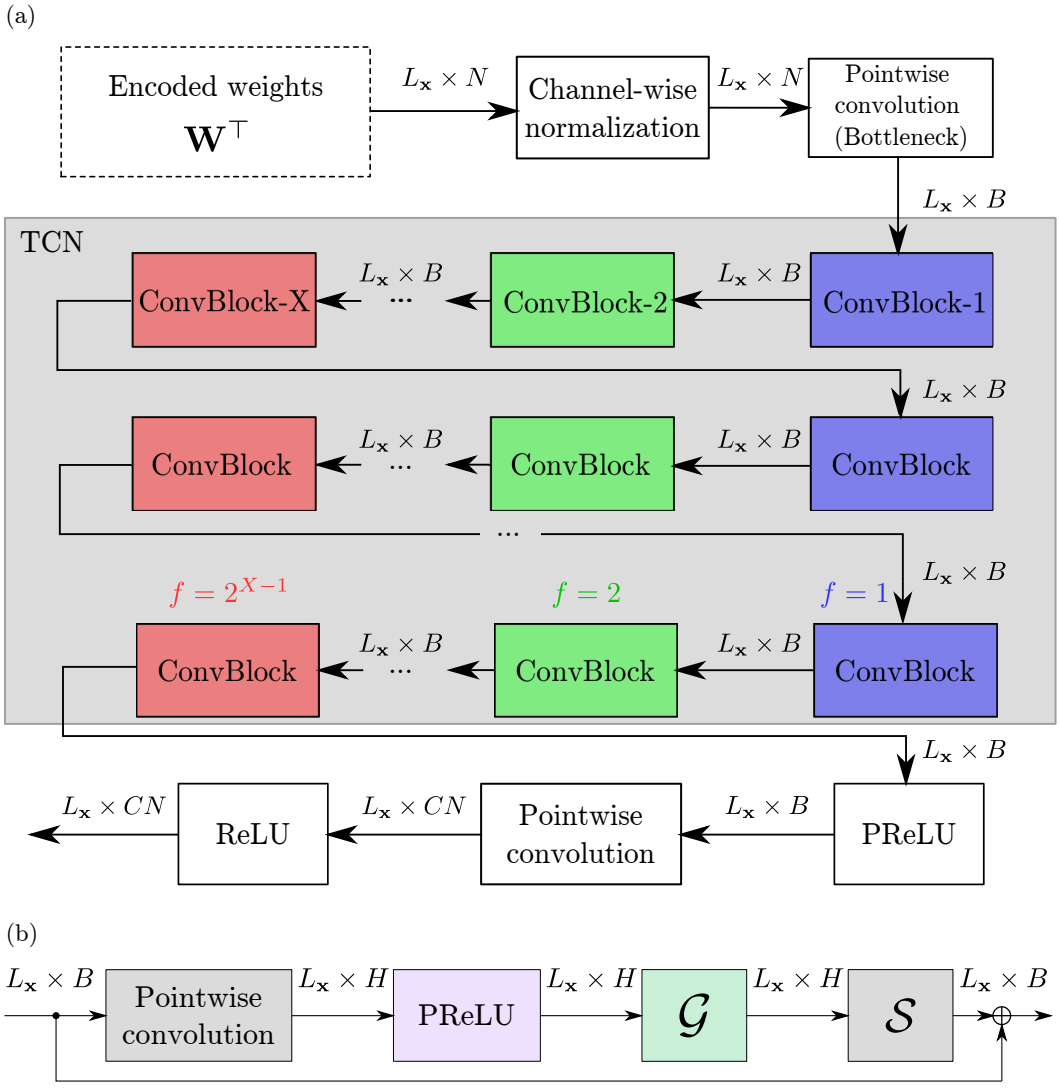


Figure 2.8: *(a) Temporal Convolutional Mask Estimation Network. (b) Network layers inside convolutional block in Figure 2.8 (a).* $\mathcal{G}(\cdot)$ *denotes the layer normalisation as defined in* (2.31) *and* $\mathcal{S}(\cdot)$ *is the DS-Conv as defined in* (2.49).

the DS-Conv layers are zero-padded, i.e. zeros are appended to the beginning and end of the sequence, such that the output sequences are always of the same length as the input sequences.

2.9.3.2 Receptive Field & Temporal Context

This subsection expands upon the receptive field of convolutional layers as described by (2.85) in Section 2.5.6. The TCN has a fixed window of depthwise inputs that the output layer is able to observe for a given output block. This window of data points is of interest particularly as the input speech data to the network can be modelled as a causal system with long-term

dependencies particularly with reverberant speech signals for which the room impulse response $h_c(t)$ in Eq. (2.2) significantly increases long term dependencies. As discussed earlier in Section 2.5.6, the receptive field of a convolutional network refers to the number of frames in a sequence that can be simultaneously observed by the network at the final convolutional layer in a deep convolutional network. This can be measured in terms of the number of input frames or time (e.g. in seconds) when dealing with time-domain signals. The receptive field measured in frames for the TCN used in Conv-TasNet depends on the number of convolutional blocks defined by blocks repetitions X and R as well as the kernel size P and can be defined as

$$\mathcal{R}_{\text{TCN}}(R, X, P) = 1 + R(P-1)\sum_{i=1}^{X} 2^{X-i}. \tag{2.85}$$

The intuition behind this formula is that for every additional layer in the network, it is possible to observe an additional $(P-1) * f_i$ frames, where f_i is the ith dilation factor in a stack of X convolutional blocks, can be observed. As $f_i, i \in 1, \ldots, X$ is equal to 2^{X-i} for the TCN hence the resulting formula is (2.85). Note that R is multiplicative as it simply repeats the same structure of X convolutional blocks in each stack.

It is possible to use the receptive field formula in (2.85) to measure the total receptive field in seconds. This is also sometimes referred to as the temporal context (Luo & Mesgarani, 2019). Given the sample rate f_s and the encoder block size J, the receptive field measured in seconds is

$$\mathcal{R}_{\text{TT}}(J, R, X, P) = \frac{J}{2\text{f}_s}\left(1 + R(P-1)\sum_{i=1}^{X} 2^{X-i}\right) + \frac{J}{2\text{f}_s} \tag{2.86}$$

$$= \frac{1}{\text{f}_s}\left(J + \frac{RJ}{2}(P-1)\sum_{i=1}^{X} 2^{X-i}\right). \tag{2.87}$$

2.9.3.3 Output Masks

The output features of the TCN network for each frame ℓ is a concatenated vector of estimated masks, which is defined as

$$\left[\mathbf{m}_{\ell,1}^{\top}, \ldots, \mathbf{m}_{\ell,C}^{\top}\right]^{\top} \in {\mathbb{R}_+}^{CN} \tag{2.88}$$

where $\mathbf{m}_{\ell,c} \in {\mathbb{R}_+}^{1\times N}$ and $c \in \{1, \ldots, C\}$ such that there is a set of mask vectors for each source signal c. Note that in this work, one should consider the mask-estimation stage complete when the mask-like features in (2.88) of shape $L_{\mathbf{x}} \times CN$ are split into C features matrices of shape $L_{\mathbf{x}} \times N$ and thus all computation proceeding from this stage is considered part of the decoder.

2.9.4 Decoder

The input of the decoder is an element-wise multiplication of the masks $\mathbf{m}_{\ell,c}$ and the encoded mixture $\mathbf{w}_\ell$ from (2.81). Estimates for the source signals $\hat{\mathbf{s}}_{\ell,c}$ are then obtained from performing a transposed 1D convolution operation defined as

$$\hat{\mathbf{s}}_{\ell,c} = (\mathbf{w}_\ell \odot \mathbf{m}_{\ell,c})\mathbf{B} \tag{2.89}$$

where $\mathbf{B} \in \mathbb{R}^{J\times N}$ represents a set of learned basis vectors to be convolved with the masked mixture. $\hat{\mathbf{s}}_{\ell,c} \in \mathbb{R}^{1\times J}$ is the estimated segment ℓ of for each audio source c. The matrix $\mathbf{B}$ is ideally the same as the $\mathbf{B}$ in the original TasNet formulation, cf. (2.76), s.t

$$\mathbf{x}_\ell = \mathbf{w}_\ell \mathbf{B}. \tag{2.90}$$

where this operation ideally performs the inverse operation of $\mathbf{U}$; however, no restraints are put on the model training to enforce this condition so that this only assists the understanding of how the model is expected to learn and hence can be a useful approach in interpreting the model.

2.9.5 Loss Function

The loss function used for training the Conv-TasNet is negative SISDR (cf. (2.72)) with a PIT wrapper (Section 2.6.2). This loss function, previously discussed in Section 2.6, was originally proposed for the BLSTM-TasNet model (Luo & Mesgarani, 2018b). Its key features are that it uses the SDR evaluation metric as an objective for training but includes scale invariance as this accelerates convergence because it decreases the penalty for inaccurate energy scaling.

2.9.6 Training Setup & Configurations

All Conv-TasNet models in the remainder of this thesis are trained over 100 epochs with an initial learning rate of 10^{-3}, identical to the original implementation of Conv-TasNet (Luo & Mesgarani, 2019). The learning rate is fixed for the first 50 epochs, after which it is halved if there is no improvement in performance after 3 epochs. Gradient clipping is used to improve training stability (Mikolov, 2012).

Gradient clipping is designed to address exploding gradients when training neural networks. The method works by thresholding the maximum norm, denoted Γ, of the gradients in the network. The algorithm for clipping a gradient, denoted ∇, by a maximum norm value by its norm is visualised in Algorithm 2. It is a very simple algorithm, whereby if ∇ is larger than the maximum specified norm Γ then it is normalized by its norm, i.e. $\frac{\nabla}{\|\nabla\|}$, and multiplied by the maximum norm Γ.

Algorithm 2 Gradient clipping by maximum norm algorithm. ∇ denotes the gradient and Γ denotes the maximum norm specified.

Input: ∇, a parameter error gradient, and Γ, the maximum norm hyperparameter
1: **if** $\|\nabla\| \geq \Gamma$ **then**
2: $\quad \nabla \leftarrow \Gamma \frac{\nabla}{\|\nabla\|}$
3: **end if**

In the Conv-TasNet model, a maximum norm of $\Gamma = 5$ is selected for this (Luo & Mesgarani, 2019). For most experiments involving Conv-TasNet-style TCNs the default encoder configuration uses a block (or kernel) size of $J = 16$ and stride of 8 samples (50% overlapping blocks). The output feature dimension of the encoder is $N = 512$. The TCN based mask estimation network has a bottleneck feature dimension of $B = 128$. It has $X = 8$ stacked convolutional blocks of increasing dilation factors repeated $R = 3$ times. The internal feature dimension of the convolutional blocks is $H = 512$ with a kernel size of $P = 3$.

Baseline Evaluation Metrics SISDR is reported for all experiments as this is used for model training as well as being the most prevalent metric currently used in speech separation literature (Luo & Mesgarani, 2018b, 2019; Tzinis et al., 2022b; Subakan et al., 2021; Rixen & Renz, 2022; Wang et al., 2023b; Roux et al., 2019).

2.9.7 Baseline Results

In this subsection, some initial baseline results are provided, which are used in the later chapters for further analysis and development of the Conv-TasNet architecture. Results are provided for the simulated noisy reverberant WHAMR corpus (cf. Section 2.2.1) of two speaker mixtures as well the anechoic WSJ0-2Mix speech mixtures (cf. Section 2.2.1) for completeness.

The original Conv-TasNet with SCs included is compared to the baseline implementation of Conv-TasNet which neglects the SCs of the original as they are shown to drastically increase the model size with no improvement in SISDR performance. The Conv-TasNet implementation without the SCs actually outperforms the original Conv-TasNet model even though it has fewer parameters. For this reason, the SpeechBrain (Ravanelli et al., 2021) & ESPnet (Li et al., 2021) style implementation, with no SCs, is used as the primary baseline in subsequent chapters.

	WSJ0-2Mix		**WHAMR**		
Model	**SISDR (dB)**	**Δ SISDR (dB)**	**SISDR (dB)**	**Δ SISDR (dB)**	**Model Size**
Raw Mixture $x[i]$	0	N/A	-6.1	N/A	N/A
Conv-TasNet (Luo & Mesgarani)	15.3	15.3	3.3	9.4	5.1M
Conv-TasNet (w/o SCs)	15.3	15.4	3.5	9.7	3.6M

Table 2.1: *Comparison of Conv-TasNet original implementation with SpeechBrain implementation without SCs. The model size is reported in the number of parameters*

2.10 SepFormer

In this section, the separation Transformer (SepFormer) model (Subakan et al., 2021) is introduced. This model was formerly SOTA on the WHAMR and WSJ0-2Mix benchmarks up until recently (Subakan et al., 2021, 2023). It is a foundational model highly relevant to the work in Chapter 7 to Chapter 9 so a detailed description is given in the following sections. SepFormer is a type of dual-path (DP) network. DP networks sequentially process local and global information using different tensor views. Section 2.10.1 first gives an overview of DP networks and the background literature behind SepFormer. Further details of the SepFormer network structure, training configuration and some baselines results are given in Section 2.10.2.

2.10.1 Dual-Path Networks

Dual-path (DP) networks are a type of sequence model proposed as an improvement to the TasNet and Conv-TasNet models. The dual path recurrent neural network model (DPRNN) was the first DP network to be proposed (Luo et al., 2020). The DPRNN model first chunks the input sequence from a shape of $L \times D$ to a shape of $\frac{L}{L_{\text{chunk}}} \times L_{\text{chunk}} \times D$. The first *local* RNN layer then process each $\frac{L}{L_{\text{chunk}}}$ chunks of length L_{chunk} individually. The output of these parallelizable RNN operations is the same dimensionality and shape as the input to the RNN layers. In the next stage, the sequential and chunk axes are swapped such that the representation is of shape $L_{\text{chunk}} \times \frac{L}{L_{\text{chunk}}} \times D$. Then the L_{chunk} sequences of length $\frac{L}{L_{\text{chunk}}}$ are passed through the second *global* RNN layer. This second process is equivalent to passing a dilated view of the input sequence through an RNN layer where the dilation factor is equal to L_{chunk}. The best performing DPRNN configuration also used very small block sizes in the encoder, $J = 2$ at 8kHz equivalent to 0.125ms of audio. However, this resulted in less efficient inference and training for the model due to the high temporal resolution of the model (8× larger than the best-performing Conv-TasNet model (Luo & Mesgarani, 2019; Luo et al., 2020)).

Some notable further improvements to the DP method were to replace the RNNs with Transformers. The Dual path Transformer network (DPTNet) model was the first such network to incorporate this technique. This model did not do away with RNNs entirely as it replaced the sinusoidal PE with an RNN layer. While both DPTNet (Chen et al., 2020a) and DPRNN (Luo et al., 2020) gave SOTA performance and had a small model size, less than 3M parameters, they both came with notable computation inefficiencies in terms of real time factor (RTF) and computational expenditure. The SepFormer model was the second DP Transformer network to be proposed. SepFormer maintained the convention sinusoidal PE used in Transformers (Vaswani et al., 2017), reducing the computational expenditure. It also operates on a lower temporal resolution than the DPRNN model, which in turn reduces the

computational expenditure, particularly for Transformers that have quadratic time-complexity. However, the model size of the best-performing SepFormer configuration is 10× larger than DPRNN or DPTNet.

2.10.2 SepFormer Network

The SepFormer model is described in this section. SepFormer is chosen because it is a large Transformer model that is among the state-of-the-art models on several speech separation benchmarks (Subakan et al., 2021, 2023). The model uses the same encoder, mask estimation network, and decoder structure as the TasNet models in the previous two sections, cf. Figure 2.6. As with the Conv-TasNet, DPRNN and other TasNet models (Luo & Mesgarani, 2019; Luo et al., 2020; Luo & Mesgarani, 2018b), the SISDR loss function (2.72) with a PIT wrapper is used for training the network.

2.10.2.1 Encoder & Decoder

The SepFormer uses an identical convolutional encoder to the Conv-TasNet encoder in Section 2.9.1. Similarly, the decoder is a single transposed 1D convolutional layer as discussed in Section 2.9.1. As the encoder and decoders are the same, the same $\mathbf{W}$ notation is used for the encoded features and $\mathbf{M}_c$ for the masking features of the cth speaker.

2.10.2.2 Mask Estimation Network

A SepFormer mask estimation network diagram is shown in Figure 2.9. This diagram shows both the overall network architecture (left) as well as a detailed schematic of the DP Transformer layers (right). In the first stage of the mask estimation network, the encoded features $\mathbf{W} \in \mathbb{R}^{L_{\mathbf{x}} \times N}$ are transformed using a linear layer and LN layer to give features $\mathbf{H} \in \mathbb{R}^{L_{\mathbf{x}} \times Z}$, where Z is the internal feature dimension used in the rest of the mask estimation network. As previously discussed, the mask estimation network uses a dual-path structure (Luo et al., 2020) whereby a series of Transformer encoder layers are stacked such that each alternating layer computes multihead attention over the sequence's local or global context. Please note that throughout the rest of this section, for the sake of brevity, Transformer encoder layers are shortened to just *Transformer layers* as Transformer decoder layers are not used in this model, cf. Section 2.5.8.2. Please see Figure 2.5 for a visualization of the Transformer encoder. The local processing is achieved by first splitting the features $\mathbf{H} \in \mathbb{R}^{L_{\mathbf{x}} \times Z}$ into 50% overlapping chunks of a predetermined size U turning a 2D tensor of shape $\mathbf{H} \in \mathbb{R}^{L_{\mathbf{x}} \times Z}$ into a 3D tensor

$$\mathbf{H}_{\text{chunk}} = [\mathbf{h}_{\text{chunk},1}, \ldots, \mathbf{h}_{\text{chunk},L_U}] \in \mathbb{R}^{L_U \times U \times N}. \tag{2.91}$$

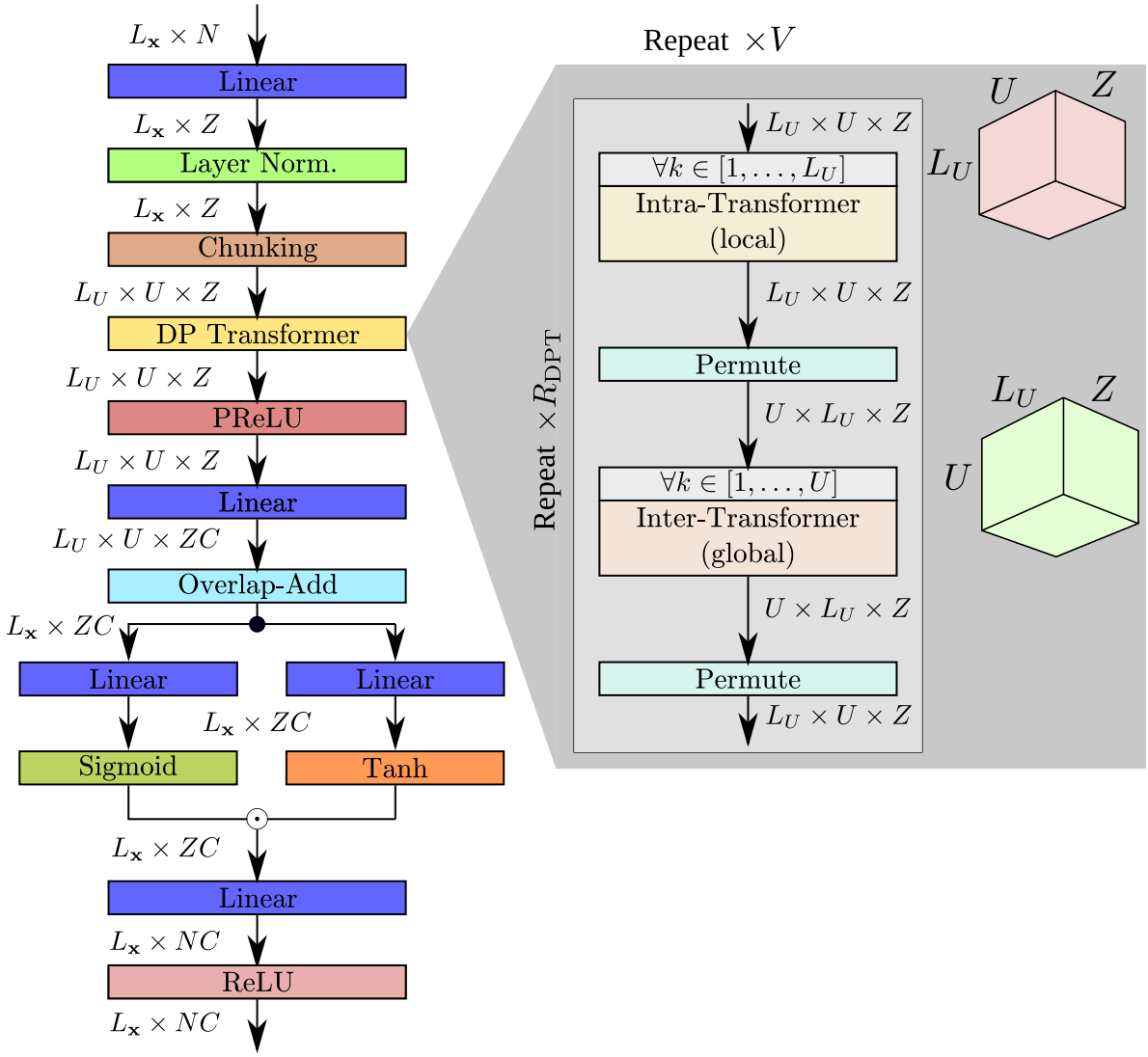


Figure 2.9: *Diagram of the SepFormer mask estimation network (left) and dual path (DP) Transformer layers (right).*

where $L_U = \frac{2L_{\mathbf{x}}}{U}$. Note that the batch dimension is omitted for brevity and simplicity in this description. The chunked representation is then passed through the first self-attentive Transformer layer iteratively, i.e.

$$\mathcal{M}\left(\mathbf{h}_{\text{chunk},k}, \mathbf{h}_{\text{chunk},k}, \mathbf{h}_{\text{chunk},k}\right) \forall k \in \{1, \ldots, L_U\} \tag{2.92}$$

This first Transformer is referred to in the source literature (Subakan et al., 2021) as the *intra-Transformer*. It is used for modelling short-term, also referred to as *local*, dependencies (or context). The output of the local Transformer layer is another 3D tensor of the same shape which is then reshaped by swapping the 1st and 2nd axes of the chunked features (of

size L_U and U respectively), denoted $\mathbf{H}'_{\text{chunk}} \in \mathbb{R}^{U \times L_U \times Z}$, before being passed through the global Transformer layer, i.e.

$$\mathcal{M}\left(\mathbf{h}'_{\text{chunk},k}, \mathbf{h}'_{\text{chunk},k}, \mathbf{h}'_{\text{chunk},k}\right) \forall k \in \{1, \ldots, U\}. \tag{2.93}$$

The 1st and 2nd axes are then swapped back. The DP network consists of R_{DPT} DP layers (i.e. a DP layer being defined as a combined intra-Transformer and inter-Transformer). The entire output of the DP network is repeated (i.e. the output features are fed back into the input of the network) V times.

On the output of the DP network is a PReLU activation with a linear layer. The output of this layer, denoted $\mathbf{H}''_{\text{chunk}} \in \mathbb{R}^{L_U \times U \times ZC}$, is reconstructed to a 2D tensor of shape $L_{\mathbf{x}} \times ZC$ using the overlap-add method, denoted

$$\mathbf{H}'' = \begin{bmatrix} \frac{1}{2}\mathbf{h}''_{\text{chunk},1,1:\frac{U}{2}} \\ \frac{1}{2}\left(\mathbf{h}''_{\text{chunk},1,\frac{U}{2}:U} + \mathbf{h}''_{\text{chunk},2,1:\frac{U}{2}}\right) \\ \frac{1}{2}\left(\mathbf{h}''_{\text{chunk},2,1:\frac{U}{2}} + \mathbf{h}''_{\text{chunk},3,\frac{U}{2}:U}\right) \\ \vdots \\ \frac{1}{2}\left(\mathbf{h}''_{\text{chunk},L_U-1,\frac{U}{2}:U} + \mathbf{h}''_{\text{chunk},L_U,1:\frac{U}{2}}\right) \\ \frac{1}{2}\mathbf{h}''_{\text{chunk},L_U,\frac{U}{2}:U} \end{bmatrix} \in \mathbb{R}^{L_{\mathbf{x}} \times Z}. \tag{2.94}$$

The reconstructed feature matrix $\mathbf{H}''$ is then gated using two linear layers, one with a tanh activation and another with a Sigmoid activation (cf. the bottom left network in Figure 2.9). The output of this gating module is then projected into C sets of masking features with correct feature dimension, i.e. $\mathbf{M} \in \mathbb{R}^{L_{\mathbf{x}} \times CN}$ where $\mathbf{M}_c \in \mathbb{R}^{L_{\mathbf{x}} \times N}$ for each speaker, using a linear layer and ReLU activation function similar to the final layer of the mask estimation network in Conv-TasNet (cf. Section 2.9.3.3). The encoded features are then masked and decoded back into the time domain using the same decoder network as the Conv-TasNet model: a transposed 1D convolutional layer (cf. Section 2.9.1).

2.10.3 Training Configuration & Data Setup

This subsection describes the standard configuration used to train the best-performing SepFormer model in (Subakan et al., 2021). The encoder uses a kernel size of $J = 16$ and output feature dimension of $N = 256$, the internal feature dimension of the DP network is $Z = 1024$, the chunk size is $U = 250$, the number of DP Transformer layers is $R_{\text{DPT}} = 8$ and the number of network cycles is $V = 1$.

Subakan et al. (2021) used a technique known as dynamic mixing (DM) where, instead of using one pre-simulated training dataset, new speech mixtures are simulated for each epoch to increase the data diversity and thus model generalization. Subakan et al. (2021) also take this a step further by using speed perturbation training to augment the data further, i.e. the pre-simulated speech samples are sped up or slowed down ±5%.

2.10.4 Baseline Results

Baseline results for the Sepformer model on the WSJ0-2Mix and WHARM datasets are shown in Table 2.2. Note that these are reproduced results and differ slightly from those

Table 2.2: *Baseline SepFormer results on the WSJ0-2Mix (Isik et al., 2016) and WHAMR (Maciejewski et al., 2020) datasets, with and without Dynamic Mixing (DM) (as implemented in Subakan et al. (2021) and Subakan et al. (2023))*

		WSJ0-2Mix		WHAMR	
Model	**DM**	**SISDR (dB)**	**Δ SISDR (dB)**	**SISDR (dB)**	**Δ SISDR (dB)**
SepFormer	✗	20.4	20.4	5.4	11.5
SepFormer	✓	22.3	22.3	7.9	14.0

reported by the original authors in Subakan et al. (2023) where SepFormer achieves 11.5 dB on WHAMR but 11.4 in Subakan et al. (2023). The introduction of DM notably improves the performance of the model on both datasets. As such, this method is employed on all the proposed separation models later in Chapter 4, Chapter 5, Chapter 8 and Chapter 9. Incidentally, it is also found later in Chapter 9 that using slightly different hyperparameters, particularly the learning rate, yields improved performance. Further analysis on improving the training time of SepFormer by truncating the training signal length (TSL) limits is provided in Chapter 7.

2.11 Speech Separation & Multi-Speaker Speech Recognition

In Chapter 10, the application of speech separation to multi-speaker Speech Recognition is explored. As such, in this chapter, an overview is provided of recent developments in multi-speaker ASR using DNN separation networks. Numerous approaches to multi-speaker ASR have been proposed, often including a variety of speech technology subcomponents including but not limited to multichannel and monaural speech enhancement, speech separation, speaker diarization and speech recognition itself (Watanabe et al., 2020). More recently, serialized output training (SOT) was proposed, a technique that enables the design of explicitly multispeaker ASR models with no subcomponents (Kanda et al., 2020). While there is a benefit to having a simplified pipeline that requires no fine-tuning on subcomponents, this can come at the expense of model interpretability. At the point in time that this thesis is being

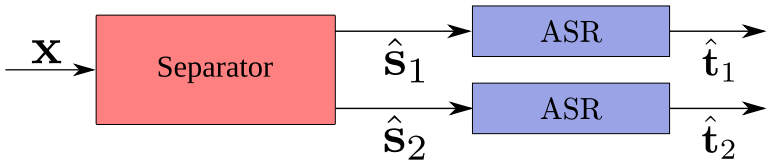


Figure 2.10: *Visualization of a modular multi-speaker ASR system, combining speech separation and single-speaker ASR model, for $C = 2$ speakers. The ASR model is the same model for both speakers.*

written, as far as the author is aware, there is no in-depth study on whether the SOT approach or the *modularized* approach typically gives better overall performance when controlling for computational expenditure. In this thesis, the primary interest is in a modularized *separate-and-recognize* approach, i.e. using a speech separation model to first separate the input audio signal $x[i]$ in C estimated speech signals $\hat{s}_c[i]$ for C speakers and then feeding each separated signal individually into a speech recognition module to attain predicted transcription of discrete character tokens $\hat{\mathbf{t}}_c$ (Cord-Landwehr et al., 2022). The modular multi-speaker ASR approach is visualised in Figure 2.10. One of the weaknesses of the modularized approach is that the speech separation module often introduces distortions into the speech signals that the speech recognizer is not trained to be robust to. One proposed solution is to mask the model artefacts with noise as described in Cord-Landwehr et al. (2022). Another option is to design a speech recognition model that generalizes very well to a large variety of speech. The recently proposed Whisper model is probably the best example of a model trained for this purpose (Radford et al., 2023). This model was trained via weak supervision, i.e. the model is trained on large amounts of poorly labelled data rather than smaller amounts of very well-labelled data, resulting in better overall ASR performance regardless of the acoustic qualities of the speech. In von Neumann et al. (2023a), the authors have claimed SOTA performance on meeting recognition benchmarks using the Whisper model combined with the currently SOTA TF-GridNet speech separation model (Wang et al., 2023b). The final approach to mitigating the training mismatch between separators and recognizers is to jointly train the separation and recognition elements using a recognition-based loss function. Qian et al. (2018) proposed a combined feature separation and recognition network that could be jointly trained using a PIT loss function from the output of the recognition model. This was possible because every operation in and between the two subnetworks was fully differentiable, enabling the use of the error backpropagation (Rumelhart et al., 1986) for training the separative part of the network as well, cf. Figure 2.3. Chang et al. (2019) expanded upon this idea into a multichannel model that used a so-called minimum-variance distortionless response (MVDR) beamformer where the speakers were separated in TF domain. Again, this method was possible because the STFT computation of the TF features and the MVDR formulation are both differentiable

functions. Around the same time Xu et al. (2019) proposed a similar method, using an STFT beamformer for separating the speakers that could be trained jointly with so-called *acoustic model* in an ASR system. Single channel variations of this approach have also been proposed (Chang et al., 2020). All these proposed end-to-end methods require reference transcriptions to do end-to-end training. In Chapter 10, it is explored whether these models can be jointly trained without requiring reference transcriptions by instead using feature representations from DNN ASR encoder models.

2.12 Related Work & Further Reading

In this final section, some related works are covered to provide a broader context for the modern state of speech separation and enhancement technology. In subsection Section 2.12.1 an overview of derverberation research particularly with respect to DNN models is provided. This information is particularly relevant to the dereverberation experiments Chapter 3. In subsection Section 2.12.2, some discussion is given to the various approaches that have been proposed in recent speech separation performance to achieve either SOTA or create more computationally efficient models and how these two approaches to designing models often come at the expense of one another. This is provided to aid some of the discussion in later chapters relating to computational efficiency and how it impacts performance. The final section, Section 2.12.3, discusses more recent unsupervised approaches to training speech separation networks. This is primarily for the readers' own interest, but it relates to the proposed fine-tuning method in Chapter 10, which in some sense could be considered partially unsupervised if pretraining the ASR encoder network was not considered part of the training scheme for the separation network.

2.12.1 Dereverberation Literature

Although the overall goal in this thesis is combined speech enhancement and separation models, the earlier parts of this thesis (Chapter 3 and Chapter 4) focus extensively on the sole dereverberation task to gain insights about the Conv-TasNet model. As such, a brief overview of some relevant literature on, particularly, DNN based dereverberation methods is given.

Dereverberation of speech signals has been studied thoroughly over the past decades (Naylor & Gaubitch, 2010; Cauchi et al., 2015; Oppenheim et al., 1968) based on machine learning models and SP techniques (Haeb-Umbach et al., 2021; Habets, 2007). Typically, SP approaches model reverberant speech as a mixture of the anechoic speech signal summed with delayed, exponentially decaying weighted sums of itself. SP methodologies for suppressing reverberant content in speech signals range from a number of techniques, with the more prominent approaches in recent work using spectral suppression or linear predictive modelling

(Nakatani et al., 2010; Park et al., 2018; Zhang et al., 2020a). Recent advances in speech dereverberation performance in a number of domains have been driven by DNN models (Kinoshita et al., 2017; Ernst et al., 2018; Wang & Wang, 2020; Wang et al., 2021; Zhao et al., 2021). DL models have mostly surpassed pure SP approaches for enhancing reverberant speech signals on objective measures such as WER or PESQ (Purushothaman et al., 2020; Kinoshita et al., 2017; Zhao et al., 2020). The original BLSTM-TasNet (Luo & Mesgarani, 2018b) was also proposed for dereverberation (Luo & Mesgarani, 2018a) with promising results. However, this evaluation was quite limited. A dereverberation network using a TCN, similar to Conv-TasNet, with self-attention was proposed in (Zhao et al., 2020) which demonstrated that TCN models give competitive results with other state-of-the-art dereverberation techniques such as DNN weighted prediction error (WPE) models (Kinoshita et al., 2017).

2.12.2 Lightweight Models vs. SOTA Peformance

The DP models discussed in Section 2.10.1 typically perform well but have a drawback of being poor in terms of computational performance with issues such as slow training times and high real time factors (RTFs) often due to high computational complexity and memory consumption (Tzinis et al., 2022b). This is especially the case with DP Transformer models (Subakan et al., 2021; Chen et al., 2020a) due to the quadratic time complexity of Transformer layers. Time domain (i.e. TasNet) models, in general, tend to have much higher computational expenditure than TF methods as well. This is normally due to the very high temporal resolution often used to achieve optimal results. The DPRNN model is a key example of this where the proposed previous SOTA configuration has an encoder kernel size of $J = 2$ with a stride of $K = 1$ (yielding $L_{\mathbf{x}} = 8000$ frames at 8kHz). While time domain methods have up until recently remained SOTA in terms of SISDR performance on popular benchmarks, there has been scientific criticism of using this as the sole approach to evaluating these models. In Cord-Landwehr et al. (2022), the authors firstly show that with careful modifications, the TF methods can reach parity, and, in some cases, improve upon time-domain methods on multi-speaker ASR evaluations. Furthermore, Cord-Landwehr et al. (2022) show that the PIT-BLSTM model (Kolbaek et al., 2017) published in 2017 can match the performance of the SepFormer model, questioning the real-world utility in the gains found using more recently proposed approaches such as TasNet and SepFormer. Very recent developments in speech separation research have proposed using TF features again with demonstrable SOTA results (Wang et al., 2023a). However, these models still have comparably high computational complexity to time domain models (Wang et al., 2023a; Tzinis et al., 2022b).

The Conv-TasNet model can generally be considered “lightweight” in terms of computational expenditure when compared to other TasNet models such as SepFormer; however, its performance is lacking in terms of raw SISDR performance on many popular benchmarks

such as WHAMR (Maciejewski et al., 2020) and LibriMix (Cosentino et al., 2020). There has been notable research interest in having so-called lightweight models without the drop in performance typically found when using them. Some notable example of these works include the SuDO-RM-RF and SuDO-RM-RF++ models (Tzinis et al., 2022b, 2020), the Skipping Memory LSTM (SkiM) model Li et al. (2022a) and the work on Tiny SepFormers (Luo et al., 2022). The techniques employed in these networks typically involve carefully modifying pre-existing models and network structures, U-Net (Ronneberger et al., 2015) in the case of Tzinis et al. (2022b), LSTMs (Hochreiter & Schmidhuber, 1997) in the case of Li et al. (2022a) and SepFormer (Subakan et al., 2021) in the case of Luo et al. (2022). The approach in all of these newer models is usually to identify and remove computational redundancies in the original model structures while maintaining overall performance.

2.12.3 Supervised vs. Unsupervised

The emphasis in this thesis is on supervised models, but in recent years, there has been an increased interest in unsupervised methods due to the limitations of using labelled data where targets always typically contain some imperfections (Tzinis et al., 2022a; Cornell et al., 2023). Unsupervised methods also yield the possibility of doing away with labelled data altogether and enable training on real speech mixtures instead of simulated speech mixtures. While unsupervised methods have improved over recent years, supervised approaches still typically give the best overall performance. As this isn't the focus of this thesis, no more detail is provided but for more on unsupervised speech separation, the reader is encouraged to refer to the MixIt unsupervised models proposed in (Wisdom et al., 2020), the RemixIT model (Tzinis et al., 2022a), the CHiME-7 UDASE challenge (Leglaive et al., 2023) and the most recently proposed UNSSOR model (Wang & Watanabe, 2023).

Part II

Temporal Convolutional Networks for Speech Separation

Chapter 3

Receptive Field Analysis for Dereverberation

This chapter and the following two chapters are primarily concerned with gaining insights about the Conv-TasNet model in order to develop technical improvements and then implement and evaluate those improvements. As alluded to in Chapter 2, convolutional models, such as Conv-TasNet, are typically less computationally expensive in a number of ways than most recurrent or Transformer-based models due to their purely feed-forward structure. Thus, for the purpose of developing higher-performing but lighter-weight models (computationally speaking), they form a good basis for developing new techniques that improve upon previous models in these areas.

In this chapter, the capabilities of the Conv-TasNet model for performing monaural dereverberation of speech are analysed according to the model configuration and the acoustic characteristics of reverberation. The goal of this research is to attain insights about the Conv-TasNet model which might aid in developing improvements to the model for the purpose of the combined speech enhancement and separation task that is explored more in-depth in the following chapters. In the proceeding content of this chapter, the Conv-TasNet model is reformulated as a denoising autoencoder (DAE) (Luo & Mesgarani, 2018a), i.e. the number of output speakers is reduced to $C = 1$. The Conv-TasNet model is evaluated across a number of configurations with particular attention paid to the receptive field. The main focus is to analyse the interplay between receptive field, model size and RIR length on the capability of Conv-TasNets to dereverb speech. The motivation for this is that both the model and data have a temporal variable: receptive field for the Conv-TasNet model and reverberation time (measured in RT60) in the case of RIRs. The RT60 measure is defined as the amount of time it takes for an IR to attenuate by 60 dB from its maximum excitation point, typically

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2022b).

measured in time (i.e. seconds (s) or milliseconds (ms)).

The remainder of the chapter proceeds as follows, Section 3.1.1 Conv-TasNet is formulated as a DAE, in Section 3.2 the data and experimental setup are discussed, in Section 3.3 results are presented and Section 3.4 concludes the paper.

3.1 Conv-TasNet as a Denoising Auto-Encoder for Dereverberation

In the remaining work in this Chapter, the signal model for a reverberant single-speaker signal is used, cf. (2.4). The aim in this chapter is to estimate the values of the direct path of the speech $s_{\text{dir}}[i]$ from $x_{\text{reverb}}[i]$ denoted as $\hat{s}[i]$, cf. (2.4) for more details on the single-speaker reverberation signal model.

3.1.1 Denoising Auto-Encoder (DAE)

The Conv-TasNet model is reformulated as a DAE again composed of an encoder, a mask estimation network and a decoder (Luo & Mesgarani, 2019; Ravenscroft et al., 2022a), cf. Figure 2.6. The only change is that $C = 1$ and thus there is only one output channel. As in Section 2.9, the audio blocks $\mathbf{x}_\ell$ are encoded into feature vectors $\mathbf{w}_\ell$. The mask estimation network produces a sequence of masks from the encoded signal. The masks $\mathbf{m}_\ell$ are then multiplied with the encoded features $\mathbf{w}_\ell$ to produce a sequence of output features that are decoded back into a time domain signal by the decoder.

The SISDR objective function formulated in (2.72) is used for training the DAE network. Note that for single-speaker dereverberation, the SISDR only needs to be computed for the single-speaker signal and averaged across the batch prior to back-propagation, i.e. there is no need to use PIT as in the previous chapter because $C = 1$ and thus there is only one permutation to consider.

3.2 Data and Experiments

3.2.1 Data

WHAMR (cf .Section 2.2.1) and an extension of WHAMR, referred to as WHAMR_ext in the following, are used for all experiments in this chapter. Only the first speakers' audio clips are used ($c \equiv 1$) since the focus is on single speaker dereverberation. In WHAMR, RT60 values for the RIRs are randomly generated between 0.1s and 1s. To create WHAMR_ext, reverberant speech with larger RT60 values between 1s and 3s were simulated following the same routine as for WHAMR. The RIRs in WHAMR_ext are generated the same as in WHAMR, using the

pyroomacoustics (Scheibler et al., 2018) software framework. Scripts to recreate WHAMR_ext have been published on GitHub[1] for reproducibility. The 8kHz sample rate *min* configuration of both WHAMR and WHAMR_ext (cf. Section 2.2.1) is used for all training and evaluation.

3.2.2 Training Configuration

Training is performed over 100 epochs with an initial learning rate set to 10^{-3} that is halved if there is no improvement in the average SISDR over the validation set after 3 epochs. The number of blocks, X, in the dilated stack of the TCN was varied from 1 to 10 and the number of repeats, R, of the stack itself was varied from 1 to 8. The rest of the network's configuration is fixed. The encoder has a kernel size of $J = 16$ and $N = 512$ output channels. The TCN is configured such that there are $B = 128$ output channels from the bottleneck layer and each convolutional block has $H = 512$ internal convolutional channels and a kernel size $P = 3$.

3.2.3 Evaluation Metrics

Several metrics are used for evaluating models in this chapter. The primary metric used is the objective function, but also PESQ (Rix et al., 2001b), ESTOI (Jensen & Taal, 2016) and SRMR (Falk et al., 2010) are reported for some results in this chapter. Refer to Section 2.7 for further details on these metrics.

3.3 Results

3.3.1 Δ SISDR on WHAMR and WHAMR_ext

The Δ SISDR results for the models trained and evaluated on the WHAMR dataset can be seen in Table 3.1 for varying X and R. These parameters are varied such that they change the receptive field and model size of TCNs where X has more effect on the receptive field (cf. (2.86)) and both increase the number of layers in the network, but R has more effect on the temporal parameter density, i.e. number of parameters per second of the receptive field. Note that one convolutional block has

$$BH + H + HP + HB = 133,120 \text{ parameters.} \tag{3.1}$$

Results in Table 3.1 show that for smaller models ($\lesssim$ 2M parameters) it is preferable to have a larger receptive field, i.e. a higher X value, than a deeper network per the receptive field, i.e. a higher R value. For example, for a model with 12 convolutional blocks the best performing model configuration is $(X, R) = (6, 2)$ where $X = 6$ is the largest possible value for the 4 possible TCN configurations, $\{(X, R)\} = \{(4, 3), (3, 4), (6, 2), (2, 6)\}$. The importance of

[1] Mixing script available online at `https://github.com/jwr1995/WHAMR_ext`

Table 3.1: Δ *SISDR in dB for all TCN configurations trained and evaluated on WHAMR, the best-performing model for a number of convolutional blocks* $(X \cdot R)$ *in TCN is shown in bold.*

		X									
		1	2	3	4	5	6	7	8	9	10
R	1	**1.88**	**2.61**	**3.32**	**4.05**	**4.66**	**5.09**	**5.41**	**5.65**	**5.67**	5.68
	2	2.48	3.58	4.45	5.25	**5.92**	**6.26**	**6.47**	6.45	6.60	6.63
	3	2.95	4.08	4.94	5.94	6.43	**6.80**	**6.88**	6.94	7.02	7.01
	4	3.28	4.46	5.47	**6.53**	**6.97**	**7.01**	**7.16**	**7.23**	7.14	7.11
	5	3.54	4.82	5.86	6.70	**7.06**	**7.31**	7.29	7.32	7.42	7.44
	6	3.74	4.99	6.16	6.87	7.25	**7.37**	7.45	7.51	7.47	7.40
	8	4.09	5.55	6.44	7.12	**7.44**	**7.63**	7.59	7.54	7.48	7.40

Table 3.2: Δ *SISDR in dB for all TCN configurations trained and evaluated on WHAMR_ext, best performing model for number of convolutional blocks* $(X \cdot R)$ *in TCN shown in bold.*

		X									
		1	2	3	4	5	6	7	8	9	10
R	1	**3.04**	**3.85**	**4.67**	**5.79**	**6.84**	**7.68**	**8.09**	**8.55**	**8.69**	8.69
	2	3.65	4.76	6.11	7.44	8.56	**9.19**	**9.52**	**9.64**	9.76	9.79
	3	4.06	5.44	6.98	8.42	9.29	**9.83**	**10.13**	**10.19**	**10.21**	10.15
	4	4.45	6.10	7.62	8.96	9.68	10.11	**10.41**	**10.42**	10.42	10.47
	5	4.70	6.51	8.21	9.36	10.01	10.37	**10.60**	**10.62**	10.54	10.50
	6	4.96	6.85	8.48	9.63	10.15	10.49	**10.74**	10.77	10.67	10.60
	7	5.29	7.14	8.75	9.71	10.34	10.61	10.72	10.68	10.76	10.70
	8	5.45	7.44	9.03	10.02	10.49	**10.80**	**10.81**	10.67	10.68	10.57

widening the receptive field is also apparent in the first row of Table 3.1 where $R = 1$ remains constant for best performance for the first 9 convolutional blocks. This importance of having $X > R$ disappears as the number of convolutional blocks surpasses 36 ($X = 6, R = 6$) at which point the best performance is gained by models with $R > X$, i.e. more importance is given to a deeper network than a wider receptive field.

Comparing the best performing models for increasing the number of convolutional blocks $(X \cdot R)$ trained on WHAMR (cf. Table 3.1) and WHAMR_ext (cf. Table 3.2) indicates that for a dataset containing only larger RT60 values between 1s and 3s it is more preferable to increase the model's receptive field as the number of convolutional blocks in the TCN increases up to 42, i.e. $(X, R) = (7, 6)$. For RT60 values between 0.1s and 1s (WHAMR corpus, Table 3.1) the model only benefits from putting more emphasis when expanding the receptive field up to 36 convolutional blocks.

Table 3.3 shows the best-performing models in terms of SISDR, trained on each training set and evaluated on each test set. These results show that performance improvements in

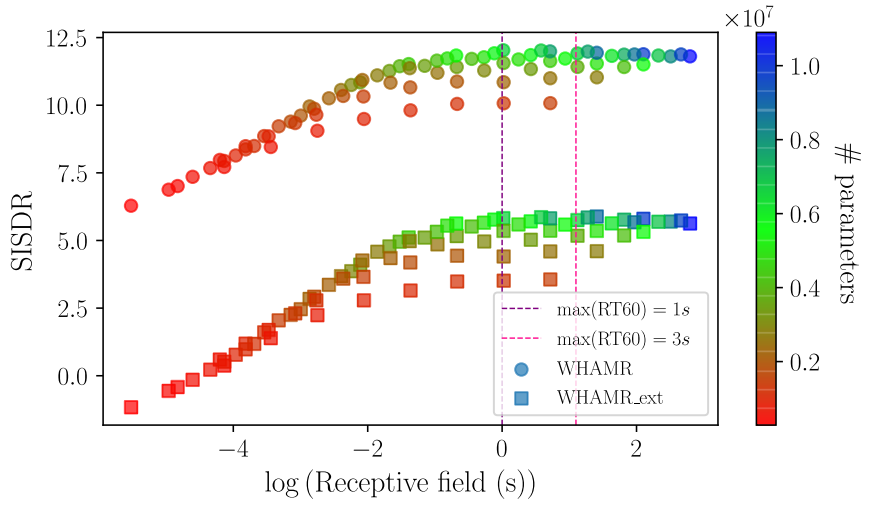
Table 3.3: *Best performing models for models trained on WHAMR and WHAMR_ext evaluated on each test set. The model size is reported in parameter count. The receptive field is denoted by $\mathcal{R}$ and calculated in seconds. Δ indicates the Δ-measure (i.e. improvement over the unenhanced signal) of the measure to its respective left. SISDR measures are reported in dB.*

train	eval	X	R	# params	$\mathcal{R}$ (s)	SISDR	Δ	PESQ	Δ	ESTOI	Δ	SRMR	Δ
WHAMR	WHAMR	6	8	6.6M	1.02	12.03	7.63	3.46	0.91	0.93	0.15	8.7	2.26
WHAMR	WHAMR_ext	8	8	8.8M	4.09	5.89	9.64	2.3	0.94	0.74	0.35	8.48	5.72
WHAMR_ext	WHAMR	6	8	6.6M	1.02	10.79	6.39	3.24	0.69	0.92	0.14	8.81	2.36
WHAMR_ext	WHAMR_ext	7	8	7.7M	2.04	7.07	10.81	2.46	1.11	0.81	0.42	9.18	6.42

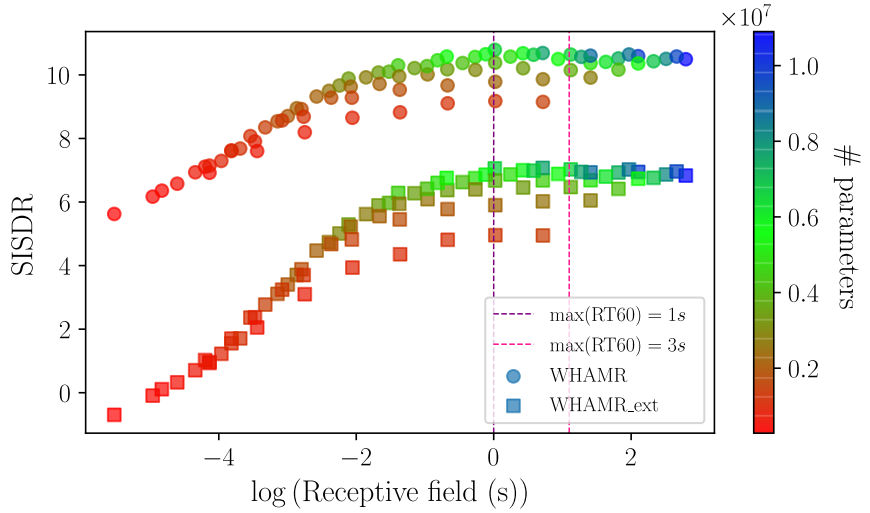
SISDR are similarly replicated in SRMR, PESQ and ESTOI measures and thus correspond to an objective improvement in perceived reverberation, quality and intelligibility of speech. Table 3.1 and Table 3.2 have been calculated also for SRMR, PESQ and ESTOI, which show similar trends. Table 3.3 also indicates that when evaluated on WHAMR the model that was trained on WHAMR_ext gives better dereverberation performance (in SRMR) but was more distorted (in SISDR), c.f. SISDR and SRMR results in rows 1 & 3. It is, however, expected that training on both WHAMR and WHAMR_ext jointly would lead to further generalisation improvements on the WHAMR test set.

3.3.2 Receptive Field and Model Size

Figure 3.1 shows the results for models trained on WHAMR (upper panel) and WHAMR_ext (lower panel) depending on the receptive field and model size. Note that the SISDR measure is used here, as opposed to the Δ SISDR measure, because it is later compared with SRMR in this section. In terms of model size, the SISDR performance that can be achieved by TCNs alone saturates as the number of parameters approaches 6M, for both WHAMR and WHAMR_ext when evaluated on WHAMR. For evaluation on WHAMR_ext, the SISDR performance also saturates as it approaches 6M parameters but the results for the model trained on WHAMR in Table 3.3 indicate it may benefit from the larger model size when having to dereverb larger reverberation times especially when trained only with smaller RT60s. Figure 3.1 further illustrates that the SISDR performance saturates before the receptive field reaches 1s for models trained on WHAMR, i.e. the highest occurring RT60 in WHAMR. The receptive fields of the best-performing models can be seen in Table 3.3. The best model trained and evaluated on WHAMR in terms of SISDR has a receptive field of 1.02s. Analysing the same models but evaluated on WHAMR_ext, the best SISDR performance is attained with a receptive field of 4.09s. This indicates that although the network has not been trained on large RT60s it may still be able to use the larger receptive field to analyse the longer reverberation tails in a meaningful way. For models trained on WHAMR_ext, the optimal model evaluated on WHAMR has a receptive field of 1.02s and a receptive field of 2.04s when evaluated on



(a) *Trained on WHAMR.*



(b) *Trained on WHAMR_ext.*

Figure 3.1: *SISDR depending on the logarithmic receptive field. Circles and squares indicate evaluation on WHAMR and WHAMR_ext test sets, respectively. Maximum RT60s of WHAMR and WHAMR_ext are shown by dashed lines. The colour scale indicates the number of model parameters.*

WHAMR_ext. Figure 3.2 analyses the best-performing models on each of the two datasets by their Receptive fields depending on model size (i.e. a number of convolutional blocks). This indicates that for larger RT60 values TCNs benefit from having a larger receptive field as most of the best performing models evaluated on WHAMR_ext have a larger receptive field than the best performing models evaluated on WHAMR. The SISDR of the estimated signal improves even with RT60s larger than the receptive field as can be seen in Figure 3.1. This implies the network learns how to estimate masks that suppress the characteristic of reverberation, as opposed to trying to perform a blind deconvolution operation based on an

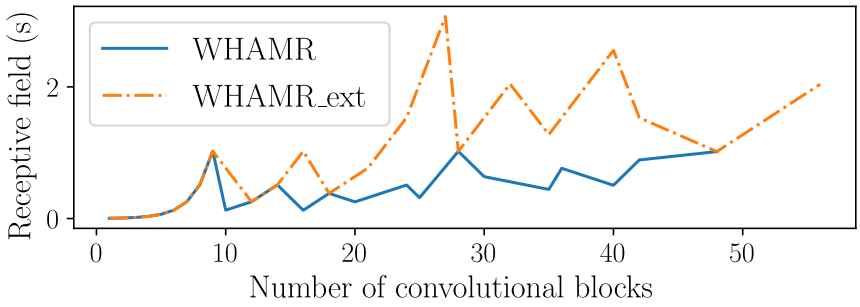


Figure 3.2: *receptive fields for the best performing models in Tables Table 3.1 and Table 3.2 shown by increasing model size measured in number of convolutional blocks (one convolutional block is 133,120 parameters). Line colour and style indicates the training and test set used.*

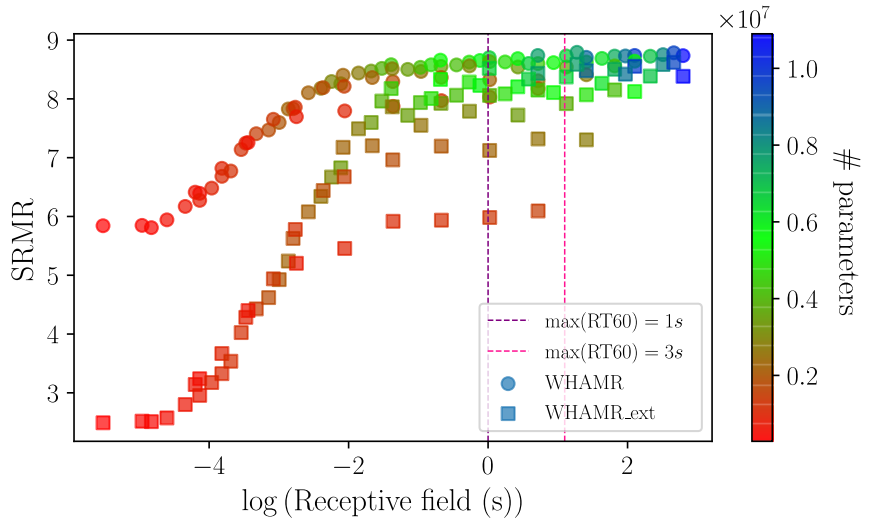
estimated RIR representation in the network. As such, it should be possible to dereverb RIRs of any RT60 value regardless of the receptive field, but Table 3.3 indicates that having a receptive field in a similar range to the maximum RT60 value is optimal and if the model has a receptive field in a similar range to the largest RT60 being evaluated then the SISDR will benefit from this even if it was trained on smaller RT60 values.

3.3.3 SRMR vs SISDR

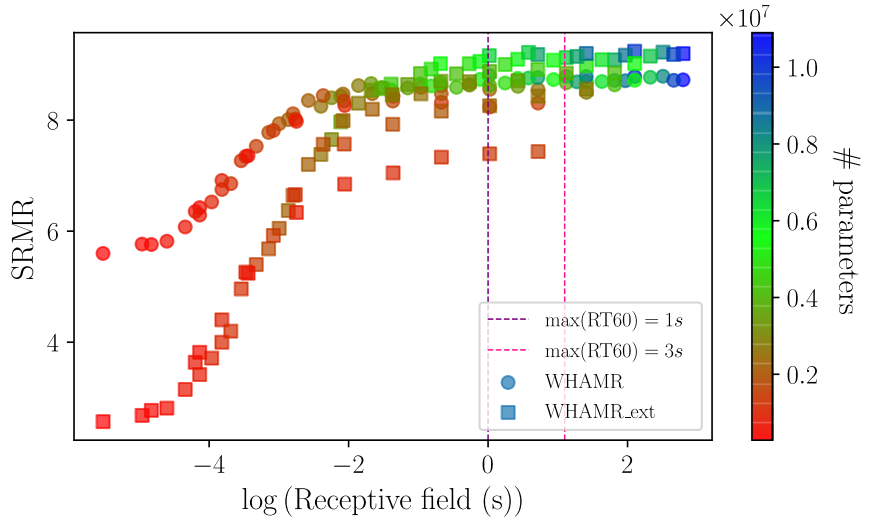
Comparing results of SISDR (Figure 3.1) with SRMR (Figure 3.3) indicates that with sufficiently large model sizes ($\gtrsim$ 4M parameters) much of the residual distortions in the signal are artefacts introduced by the network and not reverberation itself. This also indicates that the larger the reverberation time, the more residual distortions are present in the estimated clean speech signal. Audibly, the distortions present in RT60s $\lesssim 1$ were not particularly noticeable however as the RT60 approaches 3s they become more noticeable as informal listening tests showed. The performance of SRMR also saturates slightly earlier than that of SISDR similarly implying that some of the gain from increasing the model size has more correlation to reducing artefact distortions in $\hat{s}[i]$ than any residual reverberant effects. This is an argument in favour of using the SISDR function over a pure reverberation-based measure like SRMR as the loss function because SRMR is more agnostic to general distortions in the signal.

3.3.4 Improvement on WHAMR vs WHAMR_ext

The Δ SRMR results in Figure 3.4 demonstrate that greater reverberation improvement can be attained on the more reverberant WHAMR_ext dataset but that larger model sizes ($\gtrsim$ 4M parameters) are required to capitalize on this fully. Also, for both WHAMR_ext evaluations,



(a) *Trained on WHAMR.*



(b) *Trained on WHAMR_ext.*

Figure 3.3: *SRMR depending on the logarithmic receptive field. Circles and squares indicate evaluation on WHAMR and WHAMR_ext test sets, respectively. Maximum RT60s of WHAMR and WHAMR_ext are shown by dashed lines. Colour scale indicates the number of model parameters.*

there is a much broader distribution of values as the receptive field increases. This is related to the R variable in the TCN, in other words using a deeper network leads to a more significant improvement when evaluating SRMR improvement on larger RT60s.

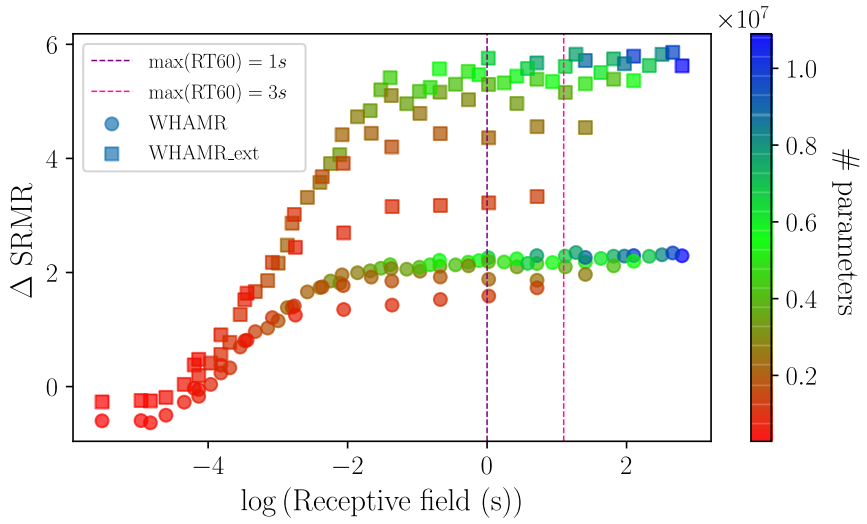


Figure 3.4: Δ *SRMR for model trained on WHAMR depending on logarithmic receptive field. Circles and squares indicate evaluation on WHAMR and WHAMR_ext test sets, respectively. Maximum RT60s of WHAMR and WHAMR_ext are shown by dashed lines. Colour scale indicates the number of model parameters.*

3.4 Conclusion

In this chapter, TCNs were analysed for their application in dereverberation tasks. It was found that for smaller models more emphasis should be put on widening the receptive field of the network than using more layers in a network. The model performance in both SISDR and SRMR starts to saturate around a model size of 4M parameters. It was shown that for larger RT60 values, there is strong evidence that having a wider receptive field is important for achieving optimal performance. This is especially true when the model is trained on smaller RT60s. It was found that SRMR performance was fairly agnostic to the variation in RT60 values but SISDR performance was significantly impacted. This indicates that much of the distortion remaining in the signal maybe modelling errors as opposed to reverberant effects. It was also found that much of the distortions in the dereverberated signal were network artefacts as opposed to reverberant effects. The optimal receptive field for a model tested and evaluated on RT60 values between 0.1s and 1s was found to be 1.02s. A new corpus containing larger RT60 values between 1s and 3s was created and when re-evaluated on this dataset the optimal receptive field was found to be 4.09s. When trained on RT60s between 1s and 3s the optimal receptive was found to be 2.04s regardless of RT60 range in the evaluation set. Given models trained and evaluated on the same RT60 ranges, for the model trained on larger RT60 values it was shown that increasing the receptive field of the model was a more pertinent factor to improving model performance than it was for the model trained on smaller RT60 values.

Chapter 4

Utterance-Weighted Multi-Dilation TCNs

In Chapter 3, it was shown that the best performing TCN models for speech dereverberation tasks typically have a larger receptive field for data with higher reverberation times T60 and a smaller receptive field for data with small T60s (Ravenscroft et al., 2022b) which forms the primary motivation for the proposed method in this chapter.

In this chapter, a TCN architecture is proposed which is able to focus on specific temporal context within its receptive field. This is achieved by using an additional depthwise convolution kernel in the depthwise-separable convolution with a small dilation factor. Inspired by work in dynamic convolutional networks, a *squeeze-and-excite (SE)* attention network is used to select how to weight each of the depthwise kernels (Han et al., 2021; Chen et al., 2020b).

The remainder of this chapter proceeds as follows. Section 4.1 introduces the WD-TCN dereverberation network. Section 4.2 describes the experimental setups and training configurations. Results are presented in Section 4.2.4.

4.1 Weighted Dilation TCN for Monaural Dereverberation

In this section, the monaural speech dereverberation signal model is introduced and the proposed WD-TCN dereverberation model is described. The general WD-TCN model architecture is similar to the reformulation of the Conv-TasNet speech separation model (Luo & Mesgarani, 2019) as a DAE in Chapter 3.

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2023a).

4.1.1 Encoder & Decoder

The encoder is the same 1D convolutional layer used in the Conv-TasNet model (Luo & Mesgarani, 2019) and Chapter 3, see Section 2.9.1 for details. Similarly the decoder is also the Transposed 1D convolutional layer used in Chapter 3 and described in Section 2.9.1.

4.1.2 Mask Estimation using WD-TCNs

A mask estimation network is trained to estimate a sequence of masks $\mathbf{m}_\ell$ that filter the encoded features $\mathbf{w}_\ell$ to produce a dereverbered encoded signal defined as

$$\mathbf{v}_\ell = \mathbf{m}_\ell \odot \mathbf{w}_\ell. \tag{4.1}$$

The $\odot$ operator denotes the Hadamard product. The reader should refer back to Section 2.9.3 for a more detailed description of the TCN used as the baseline structure that is built upon in this chapter.

The conventional TCN consists of an initial stage which normalizes the encoded features $\mathbf{w}_\ell$ and reduces the number of features from N to B for each block using a P-Conv bottleneck layer (cf. Section 2.9). The TCN is composed of a stack of X dilated convolutional blocks that is repeated R times. This structure allows for increasingly larger models with increasingly larger receptive fields as discussed in Chapter 3. The D-Conv layer in the blocks has an increasing dilation factor to the power of two for the X blocks in a stack, i.e. the dilation factors f for each block are taken from the set $\{2^0, 2^1, \ldots, 2^{X-1}\}$ in increasing order. Figure 4.1 (a) depicts the convolutional block as implemented in (Ravanelli et al., 2021; Ravenscroft et al., 2022a). The convolutional block consists primarily of P-Conv and D-Conv layers with PReLU activation functions (He et al., 2015) and gLN layers (Luo & Mesgarani, 2019). The P-Conv and D-Conv layers are structured to allow increasingly larger models to have larger receptive fields. The reader is also reminded that combining these two operations is known as DS-Conv (cf. Eq. (2.49)). More detailed definitions of P-Conv, D-Conv and DS-Conv layers are given previously in Section 2.5.6. It may be useful for the reader to refer back to these definitions when proceeding with the definition of the proposed weighted multi-dilation depthwise convolution (WD-Conv) operations that replace the DS-Conv operations in TCNs. The WD-Conv itself is introduced in Section 4.1.2.1.

4.1.2.1 Weighted Multi-Dilation Depthwise-Separable Convolution (WD-Conv)

The WD-Conv network structure depicted in Figure 4.1 (b) is proposed here as an extension to the DS-Conv operation where it is preferable to allow the network to be more selective about the temporal context to focus on without drastically increasing the number of parameters in the model. The proposed WD-Conv layer incorporates additional parallel D-Conv layers that

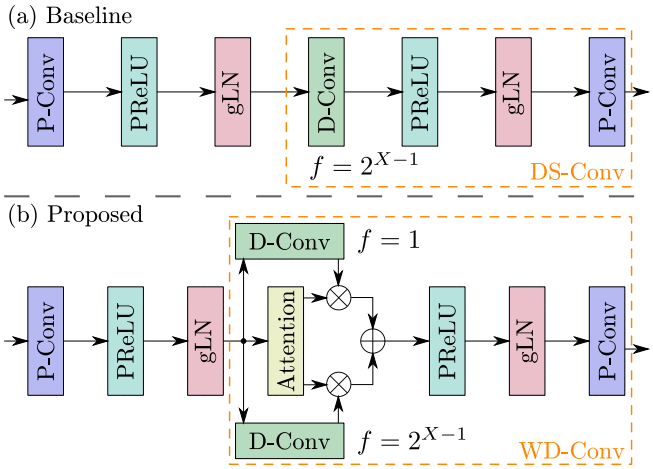


Figure 4.1: *(a) Convolutional block in baseline TCN; (b) Proposed convolutional block. Example for a final block in a stack of conv. blocks for $Q = 2$ with dilation factor $f = 2^{X-1}$. Note that a residual connection around the entire block is omitted for brevity.*

can have different dilation factors, hence the phrase *multi-dilation*. The output features of the D-Conv layers are weighted in a sum-to-one fashion and summed together. This summed output is then passed as the input to a P-Conv. As in Section 2.5.2, in the following, the notation $\mathbf{Y} \in \mathbb{R}^{L \times D}$ is used for an arbitrary sequence of features of length L with dimension D. The WD-Conv operation is formulated as

$$\mathcal{C}_{\text{WD}}\left(\mathbf{Y}, \left(\boldsymbol{\Theta}_{\text{D-Conv}_1}, \ldots, \boldsymbol{\Theta}_{\text{D-Conv}_Q}\right), \boldsymbol{\Theta}_{\text{P-Conv}}\right) = C_{\text{1D}}\left(\sum_{q=1}^{Q} a_q \mathcal{C}_{\text{D-Conv}}\left(\mathbf{Y}, \boldsymbol{\Theta}_{\text{D-Conv}_q}\right), \boldsymbol{\Theta}_{\text{P-Conv}}\right) \quad (4.2)$$

where Q is the number of parallel D-Convs in the WD-Conv and a_q are their corresponding weights that sum-to-one, i.e. $\sum_{q=1}^{Q} a_q = 1$. In the model proposed here the number of D-Convs is set to $Q = 2$; one with a dilation factor $f = 1$ and the other according to the exponentially increasing dilation rule defined previously and used in (Luo & Mesgarani, 2019; Ravenscroft et al., 2022b,a) where f is increasing in powers of 2 with every successive block in a stack of X blocks. Note that the first convolutional blocks of a stack of X blocks in the proposed implementation use an identical dilation of $f = 1$ for each of the D-Conv kernels in the WD-Conv operation.

Inspired by the dynamic convolution kernel proposed in (Chen et al., 2020b), the implementation proposed in this chapter computes the weights for each D-Conv layer using an SE attention network (Hu et al., 2018). The SE attention network is shown in Figure 4.2 and is composed of a global average pooling layer that reduces the sequence dimension from $L_{\mathbf{x}}$ to

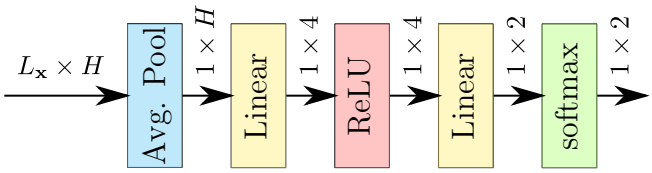


Figure 4.2: *Squeeze and excite attention weighting network. The output dimensionality of each layer is indicated above the arrows.*

1 producing a vector of dimension H, the same as the feature dimension of the input. This feature vector is then compressed using a linear layer, with a ReLU activation, to a dimension of 4 as in (Chen et al., 2020b). The final stage is a linear layer that computes a weight for each of the D-Conv kernels in the WD-Conv structure. In the proposed model, there are two D-Conv kernels, and so the linear layer has an input dimension of 4 and an output dimension of 2. A softmax activation is used to ensure the sum-to-one constraint on the weights of the D-Conv layers.

4.1.3 Objective Function

The objective function used here is the SISDR function (Roux et al., 2019) which is the same as that used to train the baseline TCN (Chapter 3). See (2.72) for the SISDR formulation.

4.2 Experimental Setup

This subsection describes the experimental setup used for first evaluating the WD-TCN model on dereverberation tasks. The dereverbation results have been published in Ravenscroft et al. (2022c). In addition, some novel results using the WD-TCN as a mask estimator in a (TasNet) speech separation model (cf. Figure 2.6), which were not included in the original publication (Ravenscroft et al., 2022c), are also included later in Section 4.3.

4.2.1 Model and Training Configuration

Different model configurations are compared in the following to demonstrate the improvement gained by the proposed WD-TCN model across a range of model sizes. This is done by varying the number of convolutional blocks in a dilated stack X as well as the number of times the dilated stack is repeated R. Based on the previous work Ravenscroft et al. (2022b), the ranges of $X \in \{4, 5, 6, 7, 8\}$ and $R \in \{4, 5, 6, 7, 8\}$ were selected, resulting in 25 different configurations. All other parameters are fixed, i.e. kernel size $J = 16$, number of encoder output channels $N = 512$, number of bottleneck output channels $B = 128$, number of channels inside the convolutional blocks $H = 512$ and the kernel size inside each D-Conv $P = 3$. For

more details on these parameters see Ravenscroft et al. (2022b,a); Luo & Mesgarani (2019) and Section 2.9.

The same training approach as in Ravenscroft et al. (2022b) is used for both the baseline TCN and WD-TCN. Each model is trained for 100 epochs. An initial learning rate of 0.001 is used and is halved if there is no improvement for 3 epochs. A batch size of 4 is used. The training was performed using the SpeechBrain speech processing toolkit (Ravanelli et al., 2021). The implementation of the proposed WD-TCN is available on GitHub[1].

4.2.2 Data

The simulated WHAMR noisy reverberant two-speaker speech separation corpus (Section 2.2.1) is used for the following experiments in this section. For the dereverberation tasks, only the reverberant and clean first speaker data ($c = 1$) is used for the input data $x[i]$ and target data $s_{\text{dir}}[i]$. The 8kHz *min* configuration is used for all experiments.

Some supplementary evaluations are later performed by applying the WD-TCN model to the speech separation task using the WHAMR and WSJ0-2Mix corpora. For these, the standard data setup used for the Conv-TasNet and SepFormer baselines (cf Section 2.9.7 and Table 2.2) are used

4.2.3 Evaluation Metrics

A number of metrics are used to assess a variety of properties in the dereverberated speech. The objective function SISDR is also used to measure distortions in signals. SRMR (Falk et al., 2010) is used to directly measure reverberant effects in the signal. PESQ (Rix et al., 2001b) is used to assess the quality and intelligibility of signals. In addition, Δ and improvement measures are used. Measures denoted with Δ indicate a change in measurement value between the reverberant signal $x[i]$ and the dereverberated signal $\hat{s}_{\text{dir}}[i]$. Improvement measures indicate the improvement of the WD-TCN model over the TCN baseline from (Ravenscroft et al., 2022b).

4.2.4 Results

4.2.4.1 Performance Metrics and Model Size

The average SISDR results on the WHAMR evaluation set for the 25 chosen model configurations of the proposed WD-TCN model are given in Table 4.1 with SISDR improvements over the TCN model in the parenthesis. The bold font indicates the best performance and highest improvement, respectively. These results show that the WD-TCN outperforms the TCN model across all 25 model configurations. The biggest performance gains are seen around

[1]Link to WD-TCN model repository on GitHub: `https://github.com/jwr1995/WD-TCN`

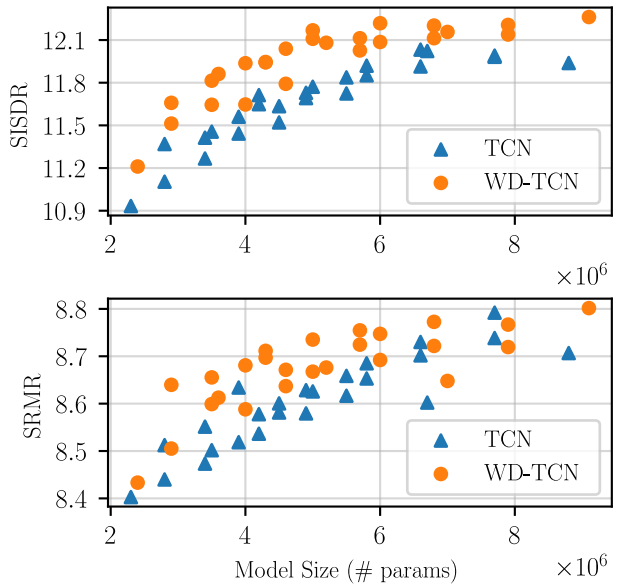


Figure 4.3: *Comparison of baseline TCN and WD-TCN over model size (no. of parameters) in terms of SISDR (top) and SRMR (bottom).*

$\{X, R\} = \{5, 7\}$ and the WD-TCN model with most parameters and largest receptive field, $\{X, R\} = \{8, 8\}$, shows best overall performance, contrary to the TCN model which gave the best SISDR results with $\{X, R\} = \{6, 8\}$. Figure 4.3 (top) shows SISDR performance for all models over the model sizes in numbers of parameters. It can be seen that using the WD-TCN is a more parameter-efficient approach to improving model performance than increasing the number of convolutional blocks (larger X or R values) in a conventional TCN. The SRMR performance against model size (Figure 4.3, lower panel) shows the same findings, i.e. that the WD-TCN is a more parameter efficient approach to improving performance. For some larger models ($>$ 6M parameters) performance differs less. However, the best-performing model in terms of SRMR is still the WD-TCN.

Table 4.2 shows the results of the best performing TCN and WD-TCN models for each of

		X				
		4	5	6	7	8
R	4	11.21 (.28)	11.66 (.29)	11.81 (.40)	11.94 (.38)	12.04 (.40)
	5	11.51 (.41)	11.86 (.41)	11.94 (.23)	12.11 (.42)	12.11 (.39)
	6	11.64 (.38)	11.95 (.30)	12.08 (.31)	12.09 (.23)	12.11 (.20)
	7	11.65 (.20)	12.17 (**.44**)	12.22 (.30)	12.16 (.13)	12.14 (.16)
	8	11.79 (.27)	12.03 (.19)	12.20 (.17)	12.21 (.22)	**12.26** (.32)

Table 4.1: *SISDR performance of WD-TCN with SE attention in dB. Numbers in* $(\cdot)$ *report performance improvement over baseline TCN.*

the chosen performance metrics, highlighted in yellow, compared with the respective other model for the same X and R hyper-parameters. The performance in PESQ is inconclusive as many TCN models outperform their corresponding WD-TCN configurations but the best PESQ score of 3.5 is achieved with the WD-TCN model. The WD-TCN models show slightly better performance in ESTOI in line with the trend already observed in SRMR and SISDR. Note that SRMR is considered a more significant metric for the dereverberation evaluations here evaluation as it is designed to assess reverberation only. The performance increase in SRMR and SISDR are minor but also mostly reflected in ESTOI where these improvements are consistent but also minor.

Table 4.2: *Best performing TCN and WD-TCN models compared corresponding models in SISDR, PESQ, ESTOI and SRMR. Bold indicates best performance per configuration, in terms of the X and R hyper-parameters. Results highlighted in yellow indicate best overall results for each model in each metric.*

Model	X	R	# params	SISDR	PESQ	ESTOI	SRMR
TCN	6	7	5.8M	11.92	3.46	0.930	8.65
WD-TCN	6	7	6.0M	**12.22**	**3.5**	**0.933**	**8.69**
TCN	6	8	6.6M	12.03	**3.46**	0.932	8.70
WD-TCN	6	8	6.8M	**12.20**	3.43	**0.934**	**8.72**
TCN	8	4	4.5M	11.63	**3.48**	0.927	8.60
WD-TCN	8	4	4.6M	**12.04**	3.45	**0.931**	**8.67**
TCN	8	7	7.7M	11.98	**3.46**	0.933	**8.79**
WD-TCN	8	7	7.9M	**12.14**	3.45	**0.935**	8.72
TCN	8	8	8.8M	11.94	**3.46**	0.933	8.71
WD-TCN	8	8	9.1M	**12.26**	3.45	**0.935**	**8.8**

4.2.4.2 Squeeze-and-Excite Attention Analysis and T60 Variation

In the following, the attentive weights a_q in (4.2) in the convolutional blocks are analysed. Note that a_1 corresponds to the attention weight applied to the D-Conv layers with the increasing dilation of $f \in \{1, 2, \ldots, 2^{X-1}\}$ for all convolutional blocks (cf. Figure 4.1 (b)) and $a_2 =$ is the weight corresponding to the D-Conv layers with the more local fixed dilation $f = 1$. The variation of a_1 for three increasingly larger model sizes and for 3 speech samples of increasing T60 is shown in Figure 4.4. Many blocks were trained such that the attention gave balanced mixtures of the two D-Convs, i.e. $a_1 \approx a_2 \approx 0.5$. It is also noticeable that for the larger the model, the greater the variance of a_1 appears to be. It was also observed that samples with low T60 values typically produce less variance in a_1 as the overall model size increases. To analyse whether the SE attention approach was working as intended the attention weights were firstly computed across the entire evaluation set for every WD-TCN model trained in Table 4.1. Mean values of the weights for each model and each sample in

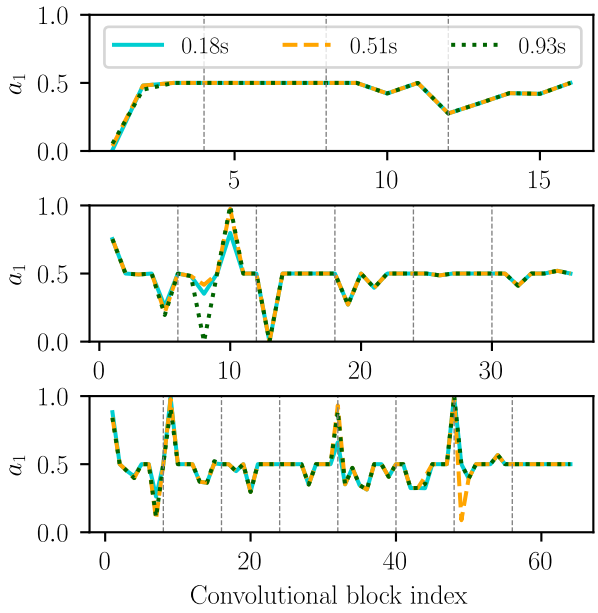


Figure 4.4: *Variation of a_1 through the WD-TCN for three different samples of varying T60 values for three different model configurations $\{X, R\} = \{4, 4\}$ (top), $\{X, R\} = \{6, 6\}$ (middle) and $\{X, R\} = \{8, 8\}$ (bottom). Vertical dashed lines denote blocks with the highest dilation factors $f = 2^{X-1}$.*

the evaluation set were then computed and the evaluation set was divided into increasing T60 ranges from 0.1s up to 1s. The mean for each weight a_q over all models and samples denoted as $\bar{a}_q, q \in \{1, 2\}$, was then computed for each T60 range. Figure 4.5 shows how the mean weight values vary across increasing T60 ranges. As the T60 range increases $\bar{a}_1$ increases. This demonstrates the SE attention approach is working as intended because the network has a less local focus within its receptive field for speech signals with larger reverberation times. Similarly, the mean of the local attention weight $\bar{a}_2$ decreases as the T60 range increases, demonstrating that the network is more focused on local information in its receptive field when the speech has a smaller reverberation time.

4.3 WD-TCN for Speech Separation in Noisy Reverberant Environments

Having validated the WD-TCN mask estimation network's improved capability over the vanilla TCN of Conv-TasNet for dereverbing speech, it is now applied to the combined speech enhancement and separation task using the WHAMR corpus. Evaluations are also performed using the anechoic "clean" WSJ0-2Mix dataset for completeness. Descriptions of both corpora are found in Section 2.2.1. The 8kHz *min* configurations are used for all evaluations. To

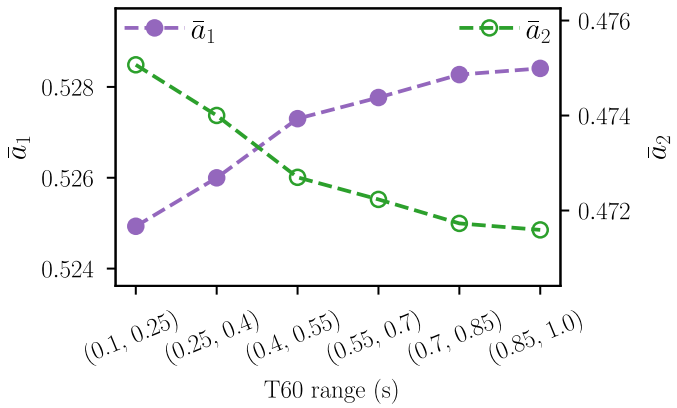


Figure 4.5: *Mean values of attention weights $\bar{a}_q$ across six different T60 ranges in the WHAMR evaluation set over all models with $X \in \{4, \ldots, 8\}$ and $R \in \{4, \ldots, 8\}$.*

extend the model for speech separation, the procedure for turning Conv-TasNet into a DAE is inverted (cf. 3.1.1). The separation network uses the same structure as the TasNet and Conv-TasNet models visualised in Figure 2.6.

4.3.1 Training Configuration

Generally, the same training configuration is used for the derverberation networks in Section 4.1. The only difference is for the network hyperparameters, the number of convolutional blocks in a stack is set to $X = 8$ and the number of stack repeats is set to $R = 3$, identical to the Conv-TasNet baselines in Table 2.1.

4.3.2 Results

Results are provided for the WHAMR corpus and WSJ0-2Mix corpus in Table 4.3. The

Table 4.3: *Comparison of speech separation Δ SISDR performance between WD-TCN model and TCN models on WSJ0-2Mix and WHAMR benchmarks. Results are also shown with and without dynamic mixing (DM). Mean Δ SISDR for the WHAMR corpus is -6.12 dB.*

Model	WSJ0-2Mix	WHAMR	Model Size
TCN	15.4dB	9.7dB	3.4M
TCN+DM	16.6	10.8dB	3.4M
WD-TCN	16.0dB	10.4dB	3.7M
WD-TCN+DM	17.2dB	11.4dB	3.7M

results demonstrate the efficacy of the WD-TCN model with a 0.7dB improvement in SISDR performance over the TCN on the WHAMR dataset without DM. These results suggest the dynamic convolution structure of the WD-TCN is beneficial not only for the dereverberation

task but also for the separation and possibly the denoising aspect of the WHAMR task. This is confirmed by the 0.6dB increase in performance on the WSJ0-2Mix dataset where no dereverberation or denoising is being performed by the network. With DM, further performance gains are seen on both WSJ0-2Mix and WHAMR. These gains are, however, almost in line with the baseline TCN model implying that the WD-TCN does not give any particular benefit in the way of model generalization. Overall these results confirm the efficacy of the multi-dilation approach for not only dereverberation tasks but also speech separation and denoising tasks. This also suggests, perhaps, that having more adaptive receptive fields is beneficial for these tasks as well. This is explored in further depth in Chapter 5.

4.4 Conclusions

This chapter proposed the WD-TCN model for TCN-based speech dereverberation by replacing depthwise-separable convolutions with weight multi-dilation depthwise-separable convolutions. It was shown that the WD-TCN consistently outperformed a conventional TCN across 25 different model configurations and that using the WD-TCN was a more parameter-efficient approach to improving model performance than increasing the number of convolutional blocks in the TCN. New findings applying the WD-TCN mask estimation network to speech separation were also presented, with WD-TCN model showing consistent improvement across both anechoic and noisy reverberant speech separation.

Chapter 5

Deformable TCNs

In Chapter 3 it was shown that the optimal receptive field of TCNs in dereverberation models varies with reverberation time when the model size is sufficiently large (Ravenscroft et al., 2022b). Furthermore in Chapter 4, it was shown that multi-dilation TCN models can be trained implicitly to weight differently dilated convolutional kernels to optimally focus within the receptive field on more or less temporal context according to the reverberation time in the data for dereverberation tasks (Ravenscroft et al., 2022c), i.e. for larger reverberation times more weight was given to kernels with larger dilation factors. Similarly, in this chapter, deformable depthwise convolutional layers (Dai et al., 2017; Bhagya & Suchetha, 2021; Chollet, 2017) are proposed as a replacement for standard depthwise convolutional layers (Luo & Mesgarani, 2019) in TCN based speech separation models for reverberant acoustic conditions. Deformable convolution allows each convolutional layer to have an adaptive receptive field. This is proposed as a replacement for standard convolution in a TCN, which enables the TCN to have a receptive field that can potentially adapt to different acoustic conditions, such as varying reverberation times. Using shared weights (Zhang et al., 2020b) and dynamic mixing (Zeghidour & Grangier, 2021) are also explored as ways to reduce the model size and improve performance. A Python library for training deformable 1D convolutional layers as well as a SpeechBrain (Ravanelli et al., 2021) *recipe* for reproducing results (cf. Section 5.3) are provided.

The remainder of the chapter proceeds as follows. The DTCN is introduced in Section 5.1. Section 5.2 discusses the experimental setup, data and baseline systems. Results are given in Section 5.3. Section 5.4 provides an analysis of the proposed models. Section 5.5 provides some unpublished results on using the DTCN model for single-speaker speech enhancement. Conclusions are provided in Section 5.6.

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2023a).

5.1 Deformable Temporal Convolutional Separation Network

5.1.1 Deformable TCN Architecture

The deformable temporal convolutional network (DTCN) separation network uses the same mask-based approach used in Section 2.9 and Chapter 4, which is briefly reviewed in the following. Readers are encouraged to refer back to Section 2.9.1 for more details. Encoded features $\mathbf{w}_\ell$ are used as the input to a mask estimation network to produce masks $\mathbf{m}_{\ell,c}$ for each speaker $c \in \{1, \ldots, C\}$. The masks are then applied to the encoded features using the Hadamard product, i.e. $\mathbf{w}_\ell \odot \mathbf{m}_{\ell,c}$ resulting in $\mathbf{v}_{\ell,c}$. The encoded estimates $\mathbf{v}_{\ell,c}$ for speaker c are decoded from the same space as $\mathbf{w}_\ell$ back into the time domain using the decoder weights $\mathbf{B} \in \mathbb{R}^{N \times J}$, such that

$$\hat{\mathbf{s}}_{\ell,c} = \mathbf{v}_{\ell,c}\mathbf{B} \tag{5.1}$$

where $\hat{\mathbf{s}}_{\ell,c}$ is the estimated clean speech signal for frame ℓ in the time domain. These frames are then combined using the overlap-add method. The network model diagram is foundationally the same as that shown for a generic TasNet model Figure 2.6.

5.1.2 Mask Estimation Network

In this subsection, the deformable depthwise convolution (DD-Conv) layer is introduced as a replacement for D-Conv layers and then the DTCN network is described in full. The mask estimation network consists of CN and a bottleneck P-Conv layer which transforms the feature dimension from N to B followed by the DTCN which is followed by a P-Conv and ReLU activation to compute a sequence of masks $\mathbf{m}_{\ell,:}$ with dimension CN (Luo & Mesgarani, 2019). The reader may want to refer back to Section 2.5.6 in this section for definitions of layer types such as P-Conv, D-Conv and DD-Conv.

5.1.2.1 Deformable Depthwise Convolution (DD-Conv)

The formulation of DD-Conv in this section is adapted from (Dai et al., 2017) and (Bhagya & Suchetha, 2021). To begin with, the definition of a D-Conv layer is revised for an individual D-Conv operation, i.e. one cell in Eq. (2.46). A single D-Conv operation of kernel size P, dilation factor f and convolutional kernel weights for the dth channel of an input with D channels denoted $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_D] \in \mathbb{R}^{L \times D}$ at the ℓth frame of L frames is defined as

$$\mathcal{K}_{\text{D-Conv}}(\ell, \mathbf{y}_d, \boldsymbol{\theta}_d, f, P) = \sum_{p=1}^{P} \theta_{d,p}\, y_d[\ell + f \cdot (p-1)]. \tag{5.2}$$

The corresponding DD-Conv function with learnable continuous offset of the pth kernel weight denoted $\tau_{\ell,p}$ at frame ℓ is defined as

$$\mathcal{K}_{\text{DD-Conv}}(\ell, \mathbf{y}_d, \boldsymbol{\theta}_d, \tau_{\ell,1:P}, f, P) = \sum_{p=1}^{P} \theta_{d,p}\, y_d[\ell + f \cdot (p-1) + \tau_{\ell,p}]. \tag{5.3}$$

Note that $\tau_{\ell,p}$ only varies temporally and not across channels. It is feasible to vary these values across channels, but in this work, offsets are only varied temporally to reduce computational expenditure. An illustration of the DD-Conv operation is shown in Figure 5.1. To simplify

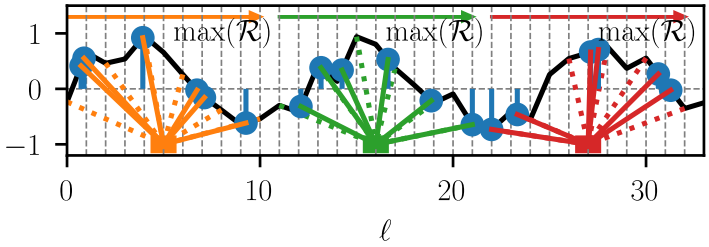


Figure 5.1: *Single channel example of the deformable depthwise convolution (bottom) on pseudo-random signal (shown in black) with a kernel size of 6, dilation factor of 2, and stride of 11. $\mathcal{R}$ denotes the receptive field of the kernel. Dotted lines indicate the original sampling position of kernel weights before deformation.*

notation let $\Delta\ell_p = \ell + f \cdot (p-1) + \tau_{\ell,p}$. Linear interpolation is used to compute values of $y[\Delta\ell_p]$ from input sequence $\mathbf{y}$ such that

$$y[\Delta\ell_p] = \sum_{u=\lfloor \Delta\ell_p \rfloor}^{\lfloor \Delta\ell_p \rfloor + 1} \max(0, 1 - |u - \Delta\ell_p|) y[u]. \tag{5.4}$$

In practice the interpolation function is designed to constrain the deformable convolutional kernel so it cannot exceed a maximum receptive field of $P \cdot (f-1)+1$ by replacing $u = \lfloor \Delta\ell_p \rfloor$ with $u = \min(\lfloor \Delta\ell_p \rfloor, \ell + P \cdot (f-1) - 1)$ in the bottom of the summation of (5.4). This constrains the kernel with the benefit of improving interpretability for the overall scope of the DTCN described in the following subsection.

5.1.2.2 Deformable Temporal Convolutional Mask Estimation Network

The DTCN is formulated in the same way as the Conv-TasNet TCN described in Section 2.9. This implementation again deviates slightly from the original Conv-TasNet (Luo & Mesgarani, 2019) by neglecting the SCs and their associated P-Conv layers. As demonstrated in Section 2.9.7, the SC layers have a negligible impact on performance (≤ 0.1 dB SISDR) while having a significant negative impact on model size ($\approx 35\%$ parameter increase).

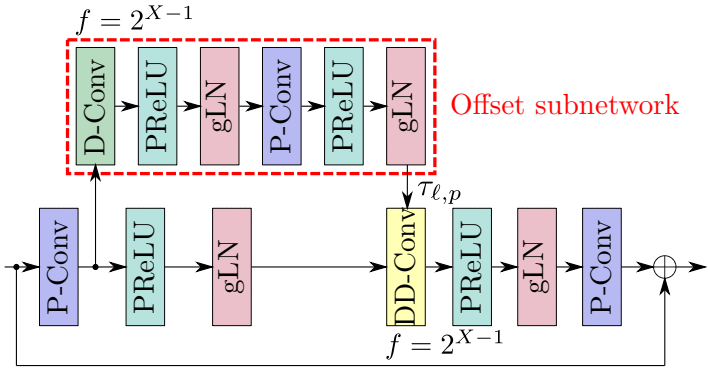


Figure 5.2: *Layers inside deformable temporal convolutional blocks.*

The DTCN is composed of $X \cdot R$ convolutional blocks where $X, R \in \mathbb{Z}_+$ (Ravenscroft et al., 2022a). Each convolutional block consists of a P-Conv which projects the feature dimension from B to H, DD-Conv that performs a depthwise operation across the H channels and another P-Conv layer which projects the feature dimension back to B from H. The DD-Conv proceeded by P-Conv layer forms a deformable depthwise-separable convolution (DDS-Conv) structure. DS-Conv, i.e. a P-Conv proceeded by any D-Conv layer, is used as a replacement for standard convolutional layers as it is more parameter efficient and mathematically equivalent (Luo & Mesgarani, 2019). In each convolutional block the DD-Conv has an increasing dilation factor f for each additional block in a stack of X blocks as in (Ravenscroft et al., 2022a; Luo & Mesgarani, 2019). The dilation factor f increases in powers of two through the stack such that $f \in \{1, 2, \ldots, 2^{X-1}\}$. Note that in D-Conv the dilation factor determines the fixed receptive field whereas in the proposed DD-Conv the dilation factor defines only the maximum possible receptive field of the kernel. The stack of X convolutional blocks is then repeated R times where the dilation factor is reset to 1 at the beginning of each stack. Using shared weights (SW) for each repeat is experimented with as this significantly reduces the model size similar to (Zhang et al., 2020b). The offsets $\tau_{\ell,p}$ are computed using DS-Conv following the initial P-Conv in the block, referred to as the offset sub-network. A PReLU activation is used at the output as this allows for both negative and positive offsets. Residual connections are applied around each of the convolutional blocks similar to the TCN described in (Ravenscroft et al., 2022a). A schematic of the convolutional blocks is shown in Fig. 5.2.

5.1.3 Loss Function

The negative SISDR function with a PIT wrapper is used as the loss function for training the DTCN models. This is the same loss function used to train Conv-TasNet Luo & Mesgarani (2019) and many other TasNet models (Luo et al., 2020; Subakan et al., 2021). For the formulation of SISDR, please refer to (2.72), and for PIT, refer to Section 2.6.2.

5.2 Experimental Setup

5.2.1 Data

Two datasets are used to evaluate and validate the proposed DTCN model. The first is the clean speech WSJ0-2Mix corpus described in Hershey et al. (2016); Isik et al. (2016) and Section 2.2.1. The second is the noisy reverberant speech WHAMR corpus proposed by Maciejewski et al. (2020) and also described in Section 2.2.1. In this chapter, the use of DM is also explored for improving separation model generalization, cf. Section 2.10. As in Table 2.2, speed perturbation training is performed as part of the DM process. The 8kHz *min* configurations of both corpora are used.

5.2.2 Model Configuration

Feature dimensions N, B and H, kernel size P and encoder block size J are fixed:

$$\{N, B, H, P, J\} = \{512, 128, 512, 3, 16\}. \tag{5.5}$$

These values correspond to the optimal TCN network in Luo & Mesgarani (2019). Note J equates to 2 ms at 8kHz. Five DTCN model configurations are evaluated:

$$\{X, R\} \in \{\{3, 8\}, \{4, 6\}, \{5, 5\}, \{6, 4\}, \{8, 3\}\}. \tag{5.6}$$

These configurations are selected as they have a similar or the same number of convolutional blocks to the optimal model configuration in Luo & Mesgarani (2019), i.e. $\{X, R\} = \{8, 3\}$.

Two GitHub repositories have been released in conjunction with this work. The first[1] is a Pytorch library for performing 1D deformable convolution. The second[2][3] is a model and *recipe* for reproducing our results with the DTCN model using the SpeechBrain (Ravanelli et al., 2021) framework.

5.2.3 Evaluation Metrics

Several metrics are used to evaluate the performance of the proposed DTCN models. SISDR and SDR (Roux et al., 2019) are used to measure residual distortion in the signal. PESQ (Rix et al., 2001b) and ESTOI (Jensen & Taal, 2016) are used to measure speech quality and intelligibility, respectively. SRMR (Falk et al., 2010) is used to measure residual speech reverberation for the WHAMR corpus. For more details on evaluation metrics, refer to Section 2.7.

[1] URL to dc1d pip repository: `github.com/jwr1995/dc1d`
[2] URL to DTCN recipe: `github.com/jwr1995/DTCN`
[3] URL to newer PubSep repository that also includes the DTCN recipes: `github.com/jwr1995/PubSep`

5.3 Results

The results for various performance measures against each DTCN configuration's receptive field on the clean speech WSJ0-2Mix evaluation are shown in Fig. 5.3 where they are also compared against their corresponding TCN configurations. Performance improvements can be

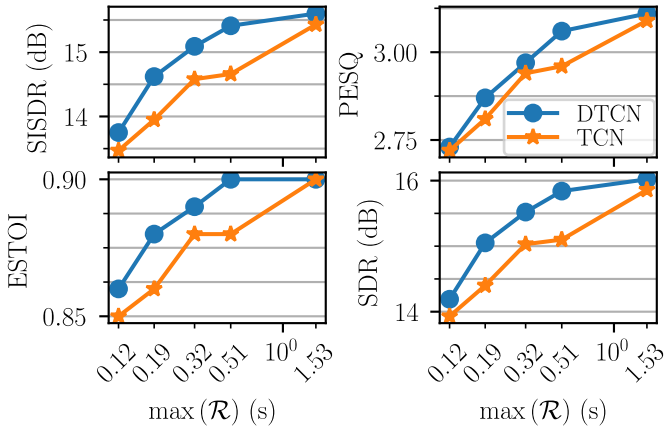


Figure 5.3: *Performance measures over receptive field for WSJ0-2Mix clean speech mixtures.*

seen across all configurations but is more significant with the models which have a receptive field of 0.19s to 0.51s. The improvement in most metrics at the highest receptive field 1.53s is marginal and for the intelligibility metric ESTOI the performance is identical to the TCN.

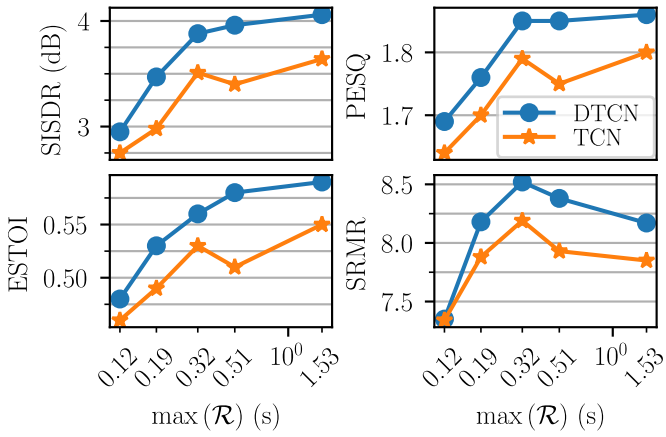


Figure 5.4: *Performance measures over receptive field for WHAMR noisy reverberant speech mixtures.*

Fig. 5.4 shows respective results for each DTCN configuration's receptive field against the performance measures on noisy reverberant WHAMR data. Note that SDR has been replaced

Table 5.1: *Comparison of various DTCN models with other speech separation models of varying size and complexity. DM and SW demote dynamic mixing and shared weights, respectively. Compuational efficiency is expressed in MACs. Where possible, MACs have been estimated on a 5.79s signal (mean signal length of WHAMR evaluation set) using thop (Zhu, 2022).*

	WSJ0-2Mix		**WHAMR**			
Model	**ΔSISDR**	**ΔSDR**	**ΔSISDR**	**ΔSDR**	**Model size**	**GMACs**
Conv-TasNet (Luo & Mesgarani)	15.3	15.6	9.2	-	5.1M	5.2
Conv-TasNet (w/o SC)	15.4	15.7	9.7	9.1	3.4M	**3.5**
Conv-TasNet (w/o SC and $H = 532$)	15.2	15.5	9.8	9.1	3.6M	3.7
WD-TCN (Chapter 4)	16.0	16.2	10.4	9.7	3.7M	3.7
WD-TCN+DM (Chapter 4)	17.2	17.4	11.4	10.6	3.7M	3.7
SkiM-KS8 (Li et al.)	17.4	17.8	-	-	6.0M	5.1
Tiny-SepformerS-32 (Luo et al.)	15.2	16.0	-	-	5.3M	-
SuDoRM-RF++ 1.0x+DM (Tzinis et al.)	17	-	-	-	2.7M	2.7
SuDoRM-RF 0.5x+DM (Tzinis et al.)	15.4	-	-	-	1.4M	1.7
SepFormer+DM (Subakan et al.)	22.3	**22.4**	14.0	**13.0**	26M	59.4
QDPN+DM (Rixen & Renz)	**23.6**	-	**14.4**	-	200M	-
DTCN (proposed)	15.6	15.9	10.2	9.3	3.6M	3.7
DTCN+DM (proposed)	17.2	17.4	11.1	10.3	3.6M	3.7
DTCN+SW (proposed)	15.0	15.3	10.0	9.3	**1.3M**	3.7
DTCN+SW+DM (proposed)	16.1	16.3	10.1	9.5	**1.3M**	3.7

by SRMR to provide a measure of reverberation. The DTCN again shows improvement over the TCN across all measures and model configurations. The performance also increases more consistently as the receptive field increases. The performance convergence seen on the clean speech mixtures in Fig. 5.3 at the largest receptive field, $\mathcal{R} = 1.53$s, is not seen in the results for the noisy reverberant data in Fig. 5.4. These findings suggest that deformable convolution is particularly useful for noisy reverberant data and that if the receptive field is sufficiently large for anechoic mixtures, the use of deformable convolution is most likely redundant.

In 5.1, the proposed DTCN model is compared against other speech separation models in size, efficiency and performance. In the third row, the model size of the Conv-TasNet model without SCs is made the same size as the proposed DTCN by changing H from 512 to 532. Comparing for model size the proposed DTCN outperforms all the Conv-TasNet model baselines including those of equal or larger model size (Luo & Mesgarani, 2019), and the recurrent SkiM-KS8 model (Li et al., 2022a). When DM is used in training, the DTCN outperforms the much larger convolutional SuDo-RM-RF 1.0x++ model (Tzinis et al., 2022b). Using SW reduces the model size by two-thirds but is still able to give comparable performance to the SuDoRM-RF 0.5x model of similar size and much-improved performance when DM is also used.

5.4 Analysis

In the following the offset values of the best-performing model configuration $\{X, R\} = 8, 3$ are analysed with the aim to provide insight as to how temporal offsets $\tau_{\ell,p}$ in (5.3), cf. also Fig. 5.1, behave relative to one another. The 2nd convolutional block of the 2nd repeat in the DTCN (i.e. the 10th block overall) was selected for analysis as it was found to have the highest average offset variance over the WHAMR evaluation set. The motivation for this choice is that it is assumed that blocks with offsets of larger variances are more indicative of the benefits of using deformable convolution. A correlation analysis was performed between

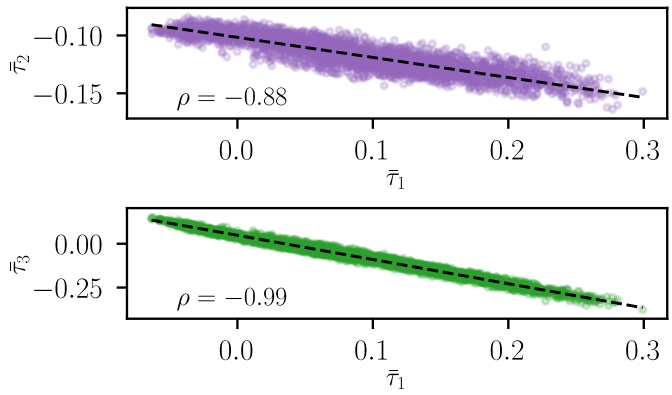


Figure 5.5: *Mean offset values $\bar{\tau}_2$ (top) and $\bar{\tau}_3$ (bottom) of the 10th convolutional block of the DTCN model plotted against the mean offset value of the first kernel weight $\bar{\tau}_1$ for each example in the WHAMR evaluation set. Pearson correlation coefficients are denoted with ρ. Dashed black line indicates line of best fit.*

each of three offset values averaged per utterance $\bar{\tau}_p$, corresponding to the three kernel weight positions. Fig. 5.5 shows scatter plots for the mean of the middle and outermost offsets denoted $\bar{\tau}_2$ and $\bar{\tau}_3$, respectively, against the mean offset value of the first kernel sample point $\bar{\tau}_1$ for every example in the evaluation set. A strong negative correlation ($\rho = -0.99$) can be observed between $\bar{\tau}_1$ and $\bar{\tau}_3$ indicating that the deformation is causing the receptive field of the kernel to shrink and grow more than shifting its focal point. A less strong negative correlation ($\rho = -0.88$) was found between $\bar{\tau}_1$ and $\bar{\tau}_2$ indicating similar behaviour. The comparison of $\bar{\tau}_2$ against $\bar{\tau}_3$ is omitted from Fig 5.5 for brevity but these mean offset values were found to have a positive correlation of $\rho = 0.81$.

5.5 Single-Speaker Speech Enhancement Performance

Some further results on single-speaker denoising, dereverberation and enhancement tasks are provided in Table 5.2 to analyze the DTCN's enhancement performance compared to the

TCN. Results are shown using the 8kHz *min* WHAMR corpus (cf. Section 2.2.1) with $C = 1$. Comparing the DTCN to the TCN there is a minor improvement of 0.3 dB Δ SISDR over the

Table 5.2: *Comparison of TCN and DTCN models on dereverberation, denoising and enhancement tasks with the WHAMR dataset.*

Model	Task	Δ SISDR (dB)	SISDR (dB)
TCN	Dereverberation	6.9	11.3
DTCN	Dereverberation	7.2	11.5
TCN	Denoising	13.9	13.1
DTCN	Denoising	14.1	13.2
TCN	Enhancement	10.8	8.2
DTCN	Enhancement	11.0	8.3

TCN model for dereverberation and a minor improvement of 0.2 dB Δ SISDR for denoising. Similarly, for the combined denoising and dereverberation enhancement task, only a minor gain of 0.2 dB Δ SISDR is observed. Interestingly, this suggests that the biggest gains using the proposed method are seen for the separation task. This is a similar trend also seen in Chapter 4 whereby the main benefit of the WD-TCN was actually seen in the separation task itself, with only relatively minor improvements in dereverberation performance. Some additional work was done to see if a correlation between the DTCN offset variation and reverberation time (T60) for both the dereverberation task and the combined enhancement on separation target but the strongest correlation coefficients ρ that could be found were $\rho = 0.06$ for dereverberation and $\rho = 0.12$ for the combined enhancement and separation task.

5.6 Conclusion

In this chapter, deformable convolution was proposed as a method to improve TCNs for noisy reverberant speech separation. It was shown that the DTCN model is particularly useful for noisy reverberant conditions as performance increases were less consistent in the case of anechoic speech separation with a sufficiently large receptive field. Using shared weights and dynamic mixing led to further performance improvements, resulting in a small model size for the DTCN compared to other separation models, which give comparable performance. It was shown that the DTCN offsets vary the size of the receptive field of convolutional blocks in the network relative to the input data. Finally, it was also shown that the DTCN model is also useful for single-speaker denoising and dereverberation tasks, yielding some improvements over the baseline TCN model.

Part III

Attention, Transformers and Speech Separation

Chapter 6

Attending to TasNet Encodings

The design of the TasNet models discussed up to this point put the majority of the onus for enhancing the signal on the mask estimation network when used without any pre-processing of the input data or post-processing of the separation network output data. The use of multihead attention (MHA) is proposed in this chapter as an additional layer in the encoder and decoder to help the separation network attend to encoded features that are relevant to the target speakers and, conversely, suppress noisy disturbances in the encoded features. As is shown later in this chapter, incorporating MHA mechanisms into the encoder network, in particular, leads to a consistent performance improvement across numerous quality and intelligibility metrics on a variety of acoustic conditions using the WHAMR corpus. The use of MHA is also investigated in the decoder network where it is demonstrated that smaller performance improvements are consistently gained within specific model configurations.

Models that used learned filterbank transforms from the time domain such as TasNets, up until recently (Wang et al., 2023b), have been able to consistently outperform models based on STFT features (Luo & Mesgarani, 2018b, 2019; Luo et al., 2020; Chen et al., 2020a; Subakan et al., 2021). The encoder of TasNets can be interpreted as filterbank features as alluded to earlier in this thesis in Section 2.9. It was shown by (Yang et al., 2019) that combining the learned features of Conv-TasNet's encoder with STFT features leads to a small improvement in performance for clean speech separation tasks. Similarly, Pariente et al. (2020) demonstrated that using complex-valued learnable analytic filterbanks in the encoder and decoder can lead to further performance improvement over the real-valued encoder of Conv-TasNet. (Ditter & Gerkmann, 2020) proposed hand-crafted MPGT filterbank features over the learned filterbank in Conv-TasNet. This approach was effective when just applied to the encoder but the learned decoder of Cont-TasNet proved more effective than their MPGT

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2022a).

based decoder.

Vaswani et al. (2017) proposed MHA as a way to parallelize a single attention mechanism into multiple attention heads while maintaining a similar parameter count to single-headed attention. This work proposes incorporating multihead attention mechanisms into the encoders and decoders of Conv-TasNet to improve the performance on noisy and reverberant speech mixtures where it is assumed that the noisy content of the data is orthogonal (i.e. uncorrelated) to the speech. Some discussion about the relevance of the orthogonality assumption and its relationship to cross-correlation is given to motivate why attention mechanisms are a suitable choice for improving the encoders and decoders. The network structures are evaluated on noisy and reverberant data from the WHAMR corpus (Maciejewski et al., 2020). Although the main goal of this work is to minimize the negative effects of additive noise under the assumption of orthogonality, separation of reverberant speech mixtures, i.e. with convolutive noise (reverberation) are also considered.

The remainder of this chapter proceeds as follows. In Section 6.1, some discussion is provided to demonstrate that dot-product attention is a special case of cross-correlation function and to provide some motivation for using attention to denoise encoded features. In Section 6.2 the proposed multihead attention and the novel encoder and decoder structures are introduced. The training configuration and experiments conducted on the WHAMR corpus are explained in Section 6.3. Further discussion and some conclusions are given in Section 6.4.

6.1 Dot-Product Attention & Cross-Correlation

Some brief discussion is given to how the scaled dot product function in MHA can be formulated as computing a cross-correlation matrix of finite discrete processes across the features of each frame ℓ. This motivates the design of the MHA encoder and decoder designs in the following sections. Using this formulation, it is suggested that the attention mechanism naturally applies more weight across frames that are highly cross-correlated. In this section the notation for $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ is taken from Section 2.5.8 where D is an arbitrary size of a dimension. It may be useful for the reader to refer back to Section 2.5.8 in the following.

The discrete cross-correlation function of two finite processes $q[i]$ and $k[i]$ can be estimated by

$$\hat{r}_{qk}[\kappa] = \sum_{d=1}^{D} q[d]k[d+\kappa]. \tag{6.1}$$

The dot-product $\mathbf{QK}^\top$ in the numerator of the dot-product attention mechanism in (2.56) is

the following matrix of size $L_q \times L_k$ can be written as

$$\mathbf{QK}^T = \begin{bmatrix} \mathbf{q}_1\mathbf{k}_1^T & \mathbf{q}_1\mathbf{k}_2^T & \dots & \mathbf{q}_1\mathbf{k}_{L_k}^T \\ \mathbf{q}_2\mathbf{k}_2^T & \mathbf{q}_2\mathbf{k}_2^T & \dots & \mathbf{q}_2\mathbf{k}_{L_k}^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_{L_q}\mathbf{k}_1^T & \mathbf{q}_{L_q}\mathbf{k}_2^T & \dots & \mathbf{q}_{L_q}\mathbf{k}_{L_k}^T \end{bmatrix} \tag{6.2}$$

where

$$\mathbf{q}_\ell = [q_\ell[1], q_\ell[2], \dots, q_\ell[D_q]] \tag{6.3}$$

$$\mathbf{k}_\ell = [k_\ell[1], k_\ell[2], \dots, k_\ell[D_k]] \tag{6.4}$$

For each cell in the resultant matrix, there is the dot-product of the feature vectors $\mathbf{q}_\ell$ and $\mathbf{k}_\ell$, both of which have a common feature dimension size $D_q = D_k$. As such each cell is written more explicitly as

$$\hat{r}\left(\mathbf{q}_\ell, \mathbf{k}_\ell\right) = \sum_{d=1}^{D_q} q_\ell[d]\mathrm{k}_\ell[d]. \tag{6.5}$$

The formula for the dot-product in (6.5) for $\hat{r}_{\mathbf{q}_\ell\mathbf{k}_\ell}$ is equivalent cross-correlation formulation in (6.1) where $\kappa = 0$ and $D = D_q$. The intuition proposed using this formulation is that because the features resulting from additive noise sources are assumed to be uncorrelated across frames to one another and the encoded speech signal features, very little weight will be given to the noisy features, implicitly denoising the encoded speech feature sequence.

6.2 Multihead Attention (MHA) Encoder and Decoders

In the following, the proposed MHA encoder and decoder designs are introduced as extensions to the Conv-TasNet encoder and decoder, cf. Section 2.9.1. The scaled dot product attention function (Vaswani et al., 2017) and MHA are briefly introduced, and the proposed application of MHA in the TasNet architecture is described. Attention was first proposed by Bahdanau et al. (2015) as a layer in DNN models that can be used to assess the similarity or relevance between two sets of features and thus provide attention to more relevant features. The reader is encouraged to consult Section 2.5.8 for formulations of dot-product attention and multi-head attention used ubiquitously in this chapter.

In the encoders and decoders proposed here, the output of the attention function is used to re-weight a sequence of features according to which features in a sequence have the most pointwise correlation (i.e. correlation across channels as opposed to across discrete time) to one another. There is a twofold assumption in the proposed application of the attention function. The first is that encoded blocks containing speech will have a higher correlation to

one another than blocks containing noise. Note that this is a typical assumption made when discussing noise and speech such as in (Roux et al., 2019). The second assumption is that in the encoded speech mixture of each individual speaker's speech signal, $s_c[i]$ will have a larger pointwise correlation to itself than to any other speaker across all frames.

Figure 6.1 demonstrates the motivation for the proposed approach by calculating the self-attention (Lin et al., 2017) of the transposed encoded signal blocks $\mathbf{W}$ from (2.82), i.e. $\mathbf{K} = \mathbf{Q} = \mathbf{V} = \mathbf{W}^\top$. The lower right panel shows the output of the attention mechanism which is then used to re-weight the encoded mixture in the lower left panel. Not that in Figure 6.1, only the dot-product attention operation is computed, there is no neural network model parameters involved, yet the representation already visually appear less noisy. Futhermore, Figure 6.2 shows the attention-weighted encoded input (middle panel) compared to an encoded noisy reverberant speech mixture (NRSM) features (top panel) as well as the corresponding encoded clean speech mixtures (CSM) features (bottom panel). Attention weighting adds greater emphasis to the features containing speech and, conversely, weights down some of the noisier parts of the encoded features.

6.2.1 MHA Encoder and Decoder architectures

In this section, the MHA encoder and decoder architectures are described. Both the encoder and decoder models use a similar paradigm by applying a multihead attention layer followed by a non-linearity to produce a set of mask-like features, which are then used to weight an encoded mixture.

6.2.1.1 Self-Attention Encoder

For the encoder self-attention (Lin et al., 2017) is used. Self-attention refers to applying attention across a sequence to itself. Therefore the inputs to the MHA layer are defined as visualized in Figure 6.3 such that

$$\mathbf{V} = \mathbf{K} = \mathbf{Q} = \left[\mathcal{H}_{\text{enc}}\left(\mathbf{x}_1\mathbf{U}\right), \ldots, \mathcal{H}_{\text{enc}}\left(\mathbf{x}_{L_\mathbf{x}}\mathbf{U}\right)\right]^T \in \mathbb{R}^{L_\mathbf{x} \times N} \tag{6.6}$$

where every input to the MHA layer is the encoded mixture from a 1D convolutional layer and ReLU activation similarly as in Conv-TasNet, cf. (2.81). The output of the MHA layer is then treated in a mask-like fashion where it is multiplied element-wise with the encoded mixture. This representation is then proceeded by a ReLU activation. Empirically it was found that placing the ReLU activation after the elementwise multiplication as opposed to using the direct output of the MHA layer consistently yielded better performance across all acoustic conditions. The complete network diagram for the MHA encoder is shown in Figure 6.3.

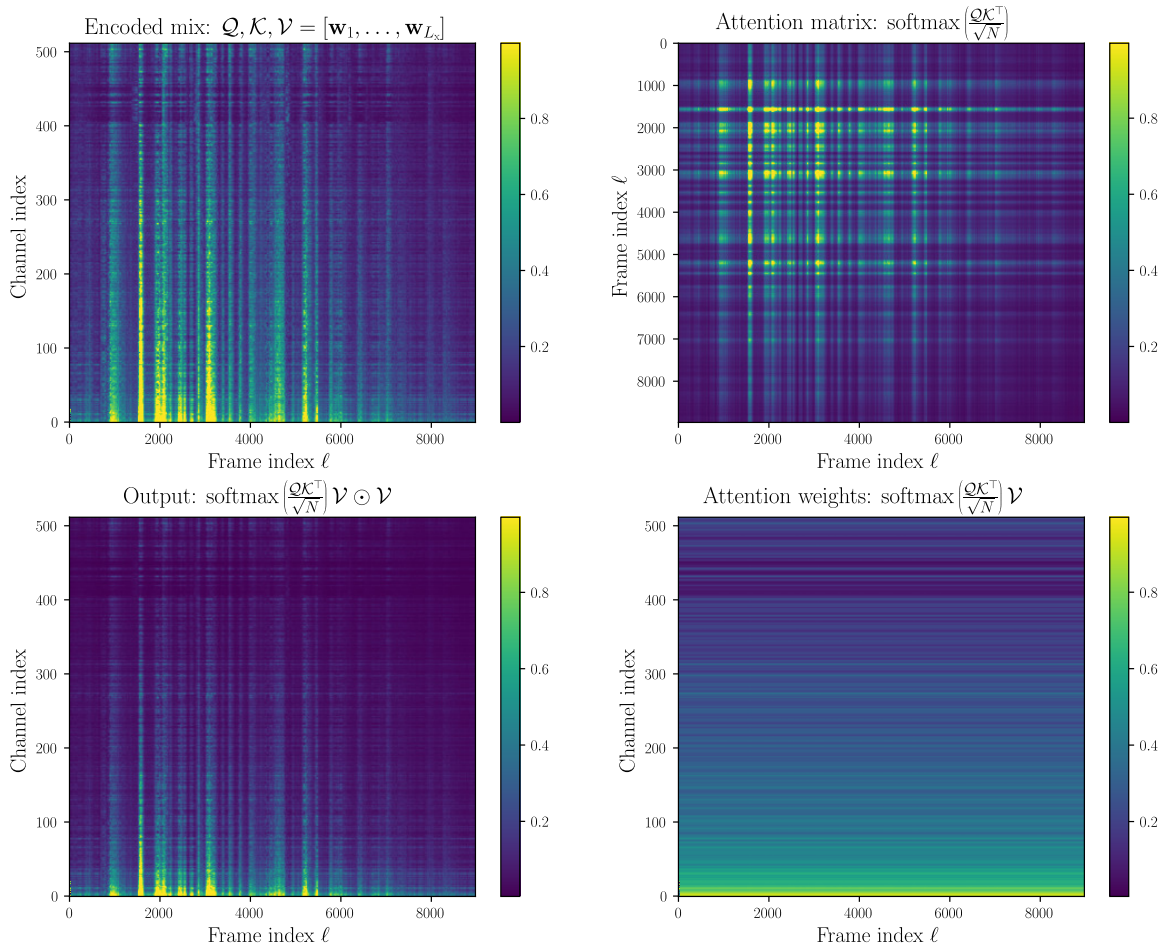


Figure 6.1: *Top left: encoded NRSM signal from the WHAMR corpus (Maciejewski et al., 2020). Top right: Computed attention matrix weights. Bottom right: Scaled dot product attention. Bottom left: encoded NRSM signal re-weighted with attention. The figures on the left have had values above* $0.05\times$ *their maximum values clipped and are normalized between 0 and 1. The figures on the right are normalized between 0 and 1. Channels are sorted using Algorithm 1.*

6.2.1.2 MR and PM Decoders

A number of approaches are proposed. Two encoder-decoder attention (Vaswani et al., 2017) based decoder models are proposed in the following subsection. The first is referred to as MR and the other is referred to PM. Both decoders are composed of a MHA layer proceeded by a ReLU activation function and a transposed 1D convolutional layer. For both architectures,

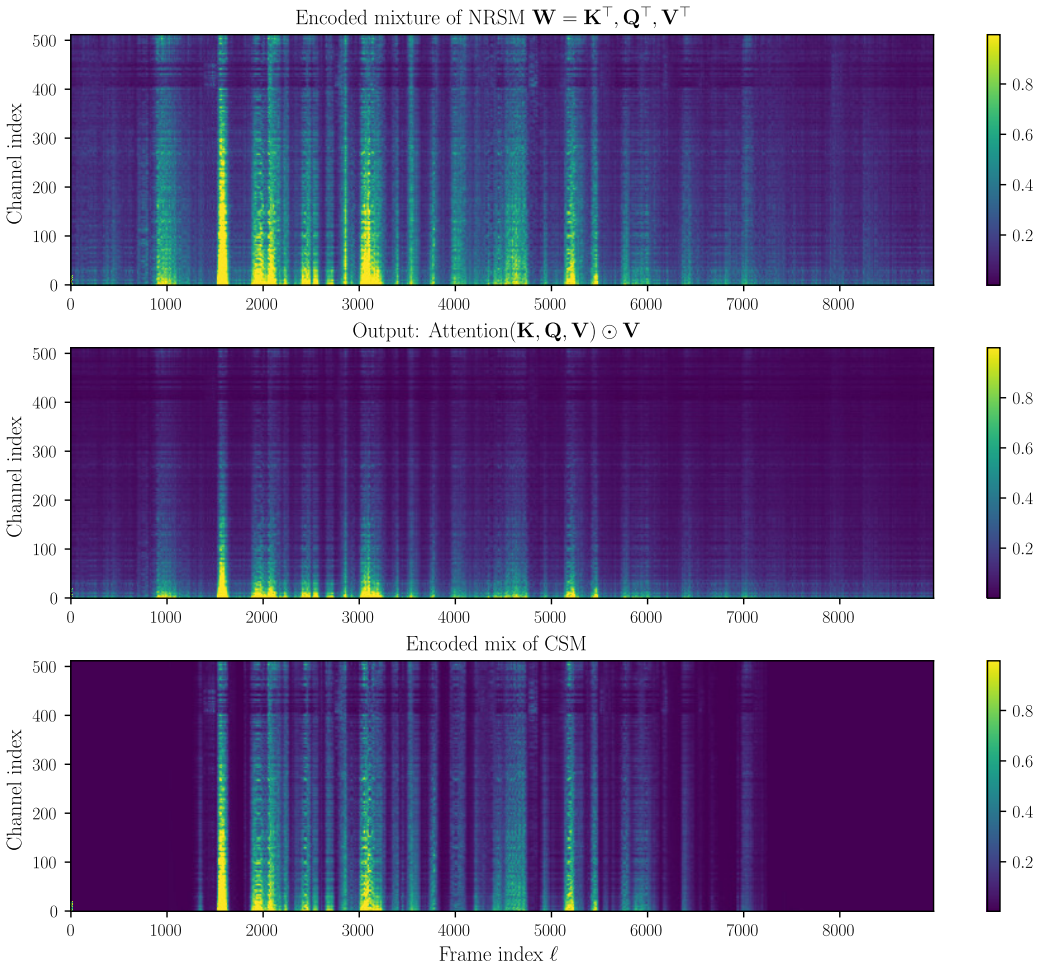


Figure 6.2: *Top: Encoded NRSM signal blocks. Middle: Encoded NRSM signal blocks re-weighted with attention as defined in* ((2.56)). *Bottom: Encoded CSM signal blocks* ($\nu[i] = 0$, $h_c[i] = \delta[i]$, $\forall t \geq 0$). *The top and bottom figures clip values above* $0.05\times$ *the maximum value of the encoded NRSM signal and then normalized between 0 and 1. The middle figure clips values above* $0.05\times$ *its maximum value and is then normalized between 0 and 1.*

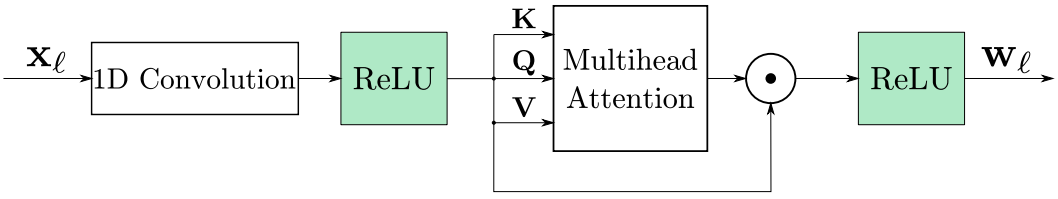


Figure 6.3: *Convolutional MHA encoder diagram*

the input to the MHA layers are defined as

$$\mathbf{V} = [\mathbf{w}_1, \ldots, \mathbf{w}_{L_\mathbf{x}}]^T \in \mathbb{R}^{L_\mathbf{x} \times N} \tag{6.7}$$

$$\mathbf{K}_c = [\mathbf{m}_{1,c}, \ldots, \mathbf{m}_{L_\mathbf{x},c}]^T \in \mathbb{R}^{L_\mathbf{x} \times N} \tag{6.8}$$

$$\mathbf{Q}_c = [\mathbf{w}_1 \odot \mathbf{m}_{1,c}, \ldots, \mathbf{w}_K \odot \mathbf{m}_{L_\mathbf{x},c}]^T \in \mathbb{R}^{L_\mathbf{x} \times N} \tag{6.9}$$

where $c \in \{1, ..., C\}$ and C is the number of target signals. These inputs are defined to combine the principles of encoder-decoder attention, described in Section 3.2.3 of (Vaswani et al., 2017), with those of self-attention as both the key and query contain information from the estimated masks. The same MHA layer is used for each speaker.

The mask refinement (MR) decoder produces a mask from the MHA layer proceeded by a ReLU function which is multiplied by the encoded mixture and this re-masked encoded mixture is then decoded back into the time domain with the transposed 1D convolutional layer. The MR decoder model is depicted in Figure 6.4 (a). The motivation in this design is to use the MHA mechanism to produce a mask that refines the already masked encoded representation such that it attends better to features most relevant to the most present speaker features in the original masked encoded features.

The post-masking decoder (PMD) also uses an MHA layer to produce a new mask but in this model the new mask is used to refine the already masked encoded mixture. The PMD model is shown in Figure 6.4 (b). The motivation in this design is to use the MHA mechanism to produce a new mask by observing speaker information in the masks and masked encoded mixtures to produce an improved hypothesis of what the masks should be by attending to the most prevalent correlated speaker information in both types of representation.

6.2.1.3 SA Decoder

An additional decoder based on self-attention is proposed shown in Figure 6.4 (c). This decoder applies MHA to the masks estimated by the network defined in Section 2.9.3 in a self-attentive manner such that

$$\mathbf{V} = \mathbf{K} = \mathbf{Q} = [\mathbf{m}_1, \ldots, \mathbf{m}_{L_\mathbf{x}}]^T \in \mathbb{R}^{L_\mathbf{x} \times N}. \tag{6.10}$$

The output of the MHA layer is proceeded by a ReLU function to produce a new set of masks. The Hadamard product of the new masks with the encoded mixture is then computed. This masked encoded mixture is then decoded back into the time domain using a transposed 1D convolutional layer.

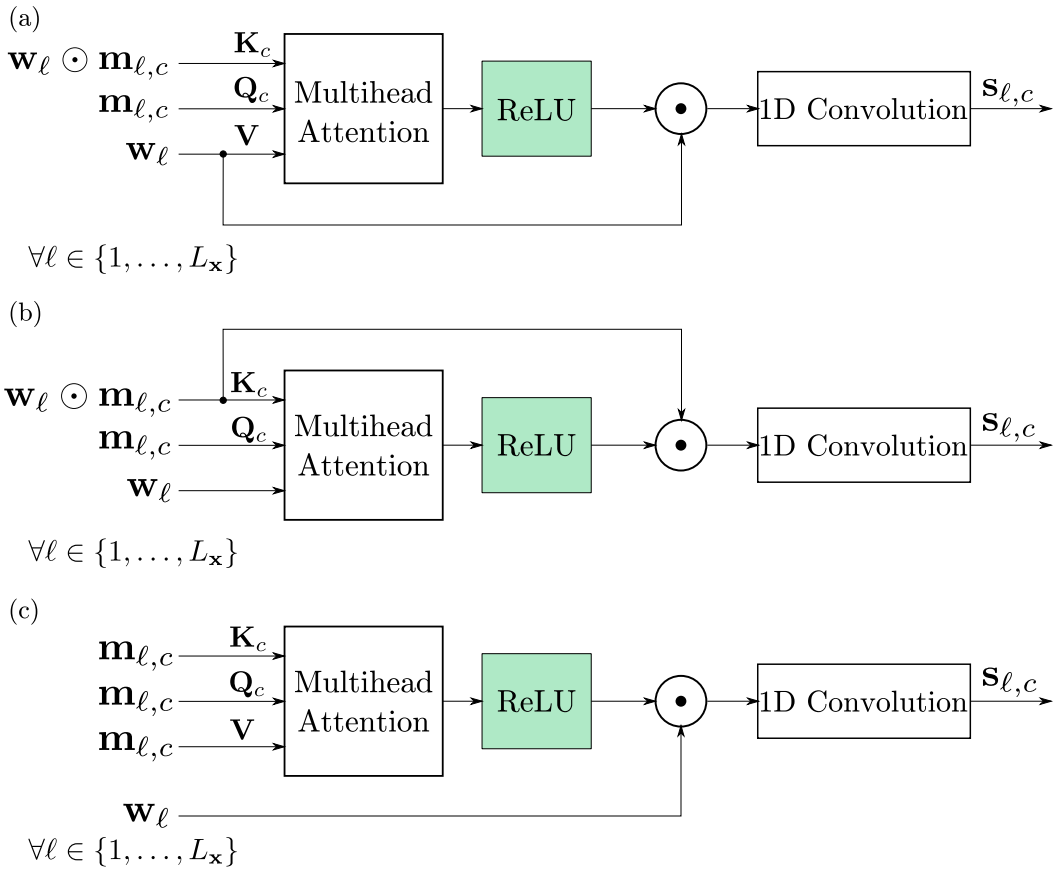


Figure 6.4: *(a) MHA mask refinement (MR) decoder architecture. (b) MHA post-masking (PM) decoder architecture. (c) MHA self-attention (SA) decoder architecture.*

6.2.2 Architecture Complexities

Some brief discussion is given to the model complexities predominantly for reference. The complexities for each of the proposed encoders and decoders as well as the baselines used later in Section 6.3 are given in Table 6.1. The proposed encoder described in Section 6.2.1.1 is

Table 6.1: *Complexity of all encoder and decoder models evaluate including all non-linearities, weights and biases. Best performing results for each acoustic condition shown in bold.*

Model	Complexity
Conv-TasNet encoder (Luo & Mesgarani)	$(J+1)L_{\mathbf{x}}N$
Conv-TasNet decoder (Luo & Mesgarani)	$JL_{\mathbf{x}}N$
Deep-PReLU encoder (Kadıoğlu et al.)	$(J+7)L_{\mathbf{x}}N + 3L_{\mathbf{x}}N^2$
Deep PReLU-decoder (Kadıoğlu et al.)	$(J+6)L_{\mathbf{x}}N + 3L_{\mathbf{x}}N^2$
SA encoder (proposed)	$(J+3)L_{\mathbf{x}}N + L_{\mathbf{x}}N^2 + (1+\frac{1}{G_{\text{heads}}})L_{\mathbf{x}}^2N$
SA decoder, PM decoder, MR decoder (proposed)	$(J+2)L_{\mathbf{x}}N + L_{\mathbf{x}}N^2 + (1+\frac{1}{G_{\text{heads}}})L_{\mathbf{x}}^2N$

more computationally complex than the encoders proposed by Luo & Mesgarani (2019) and Kadıoğlu et al. (2020) however, a significant reason for this is that the attention operation considers the entire sequence length as opposed to operating over a smaller context window as is the case in the other encoders. The same is true of the proposed decoders described in Section 6.2.1.2 and Section 6.2.1.3 compared with the purely convolutional decoders proposed by (Luo & Mesgarani, 2019) and (Kadıoğlu et al., 2020). In future work, ways to alleviate the computational complexity using linear attention (Katharopoulos et al., 2020) and restricted self-attention (Vaswani et al., 2017) can be explored, but this is beyond the scope of the work presented here.

6.3 Experimental Setup

This section presents details on the experimental setup as well as the results performed to evaluate the proposed encoders and decoders in the previous section.

6.3.1 Data

The WHAMR dataset (cf. Section 2.2.1) is primarily used for the evaluations in this section. Speech mixtures are evaluated under four different acoustic conditions (ACs): clean speech mixtures (CSM), i.e. $\nu[i] = 0$ and $h_c[i] = \delta[i]$ in Eq. (2.2), noisy speech mixture (NSM), i.e. $h_c[i] = \delta[i]$ but noise present in Eq. (2.2), reverberant speech mixture (RSM) i.e. $v[i] = 0$ but reverberation present in Eq. (2.2), and noisy reverberant speech mixture (NRSM). 8kHz audio samples are used and truncated to 3-second segments for training. This length constraint

is removed for validation and testing. The impact of TSL limits is explored later in Chapter 7.

6.3.2 Loss Function

The negative SISDR function (cf. (2.72)) with a PIT wrapper (cf. Section 2.6.2) is used for training all models. This is the same strategy used for training the Conv-TasNet model (Luo & Mesgarani, 2019) and the DTCN in Chapter 5.

6.3.3 Training Configuration

The Conv-TasNet model is implemented using the SpeechBrain framework introduced by (Ravanelli et al., 2021). As previously noted, the mask estimation network in SpeechBrain neglects the skip connections in the original Conv-TasNet model proposed by (Luo & Mesgarani, 2019) and implemented in (Maciejewski et al., 2020). A description of the baselines model parameters as well as the CSM SISDR performance and temporal context, reported in seconds (s), is shown in Table 6.2. Note that for this section, the TCN is configured using $\{X, R\} = \{6, 4\}$ instead of the $\{X, R\} = \{3, 8\}$ configuration previously used.

Table 6.2: *Details of the Conv-TasNet configuration compared to (Maciejewski et al., 2020). Proposed baseline SISDR result is shown in bold.*

Variable	Description	Baseline
N	Input channels	512
J	Input block size	16
B	Bottleneck output channels	128
H	Output channels	512
P	Kernel size of conv. block	3
X	Blocks of increasing dilation	6
R	Repeats of dilated layers	4
$\mathcal{R}_{\mathrm{TT}}(\cdot)$	Receptive field (s)	0.51
SISDR	SISDR (dB) on CSM	**14.6**

An initial learning rate of 1×10^{-3} is used, and a learning rate scheduler is used that halves the learning rate if there is no average SISDR improvement of the model for 3 epochs. A batch size of 4 was used. A total of 100 epochs of training were performed.

6.3.4 Evaluation Metrics

Performance is measured using SISDR, SDR, PESQ and STOI. Refer to Section 2.7 for more details on these measures. Δ measures are shown in addition to the absolute metric values to indicate the improvement in quality or intelligibility between the noisy reverberant signal mixture $x[i]$ and the network estimates $\hat{s}_c[i]$ against the reference $s_c[i]$.

6.3.5 Additional Baselines

Some work has already been done to investigate improved encoders and decoders for the Conv-TasNet model. Deeper convolutional encoder and decoder networks were proposed by (Kadıoğlu et al., 2020) for use with Conv-TasNet on speech separation tasks. In this work, this deep convolutional encoder and decoder model is implemented as an additional baseline to the original Conv-TasNet model described in the previous part of this section. The deep convolutional encoder consists of three additional 1D convolutional layers each with a kernel size of 3 and a stride of 1. Each convolutional layer is proceeded by a PReLU activation function. The number of input and output channels are equal to N. Their deep convolutional decoder is similarly constructed of an additional 3 transposed 1D convolutional layers proceeded by PReLU activation functions. Each additional transposed 1D convolutional layer has the same kernel size and stride as the additional encoder layer. Each layer also has N input and output channels. It was found by (Kadıoğlu et al., 2020) that increasing the dilation of the encoder and decoder layers had negligible effects on the SISDR separation performance and so a fixed dilation of 1 is used for each layer.

6.3.6 Results

The following subsections address the speech separation results of the proposed method in comparison to baseline methods on the WHAMR corpus. The MHA encoder is evaluated first and then two subsequent sections analyse the MHA decoder architectures and look at how the number of attention heads affects performance. Every set of results is compared against the original encoder and decoder proposed by (Luo & Mesgarani, 2019) reported as Conv-TasNet and the deep convolutional encoder and decoder model proposed by (Kadıoğlu et al., 2020) is reported as Deep PReLU.

6.3.7 MHA Encoder Results

The MHA encoder model seen in Figure 6.3 is compared to the original Conv-TasNet baseline encoder proposed by (Luo & Mesgarani, 2019) as well as the Deep PReLU approach proposed by (Kadıoğlu et al., 2020). The results for this comparison across all four acoustic conditions can be seen in Table 6.3. The MHA encoder is denoted as the self-attention encoder (SAE) in all results. These results demonstrate a consistent improvement of the MHA encoder over the original baseline purely convolutional encoder. Highest improvement in performance can be observed for the clean speech mixtures (CSM) since this is the easiest task for the network. The MHA encoder achieved slightly more performance improvement on the RSM condition than the NSM condition and the NRSM. The MHA encoder outperformed the Deep PReLU encoder on every acoustic condition. Figure 6.5 shows the intermediate features in

Table 6.3: *Comparison of MHA encoder with 4 attention heads to Original Conv-TasNet encoder across various acoustic conditions. Best performing results for each acoustic condition shown in bold. Δ columns refer to the improvement of the respective measure in the adjacent column to the left.*

AC	Encoder	SISDR (dB)	Δ	SDR (dB)	Δ	PESQ	Δ	STOI	Δ
CSM	Conv-TasNet	14.7	14.7	15.1	15	2.99	1.69	0.94	0.342
	Deep PReLU	14.8	14.8	15.2	15.1	2.96	1.66	0.941	0.344
	SAE	**15.7**	**15.7**	**16.1**	**16.0**	**3.15**	**1.84**	**0.952**	**0.355**
NSM	Conv-TasNet	7.63	12.1	8.28	12.5	1.97	0.838	0.824	0.373
	Deep PReLU	7.83	12.3	8.51	12.7	2.04	0.900	0.840	0.432
	SAE	**8.37**	**12.9**	**9.01**	**13.2**	**2.09**	**0.93**	**0.854**	**0.446**
RSM	Conv-TasNet	5.52	8.81	7.75	7.87	2.20	0.969	0.847	0.312
	Deep PReLU	5.91	9.20	8.09	8.21	2.26	1.04	0.860	0.325
	SAE	**6.39**	**9.67**	**8.57**	**8.68**	**2.34**	**1.10**	**0.874**	**0.339**
NRSM	Conv-TasNet	3.54	9.66	5.48	8.96	1.79	0.656	0.75	0.366
	Deep PReLU	3.63	9.76	5.56	9.05	1.82	0.68	0.76	0.372
	SAE	**4.11**	**10.4**	**6.00**	**9.48**	**1.92**	**0.754**	**0.787**	**0.399**

the MHA encoder encoding an NRSM signal. Comparing the encoded signal after the first convolutional layer in the network to the similar representation in Figure 6.3 it is notable that the convolutional layer has learned to focus on a narrow set of channels. This implies a large number of the channels are, in fact, redundant, a similar finding to the MPGT encoder and convolutional decoder model proposed by Ditter & Gerkmann (2020). The final output of the MHA encoder further narrows the focus of the encoded features. Another interesting finding of the output of the MHA layer is that the mask-like features (top right in Figure 6.5) do not seem to attenuate the signal where there is only noise present as one might expect due noise not being present in the target signal at training. This effect can be seen more clearly when compared to the intermediaries of the CSM signal encoded by the MHA encoder in Figure 6.6.

6.3.8 MHA Decoder Architecture Comparisons

A comparison of the MRD in Figure 6.4 (a), the PMD in Figure 6.4 (b) and the self-attention decoder (SAD) in Figure 6.4 (c) is carried out in the following to analyse which approach, if any, leads to superior decoding performance over the Conv-TasNet baseline (Luo & Mesgarani, 2019) and Deep PReLU decoder (Kadıoğlu et al., 2020). The results are shown in Table 6.4. In each case the number of attention heads is set to $A = 2$. There was a clear performance improvement on clean speech mixtures across all metrics with the MRD in Figure 6.4 (a). Also a noticeable performance increase can be observed for the reverberant speech mixtures but this improvement is not also seen for the noisy reverberant speech mixtures where there was a small drop across all measures except for the STOI measure. The PMD design showed decreased

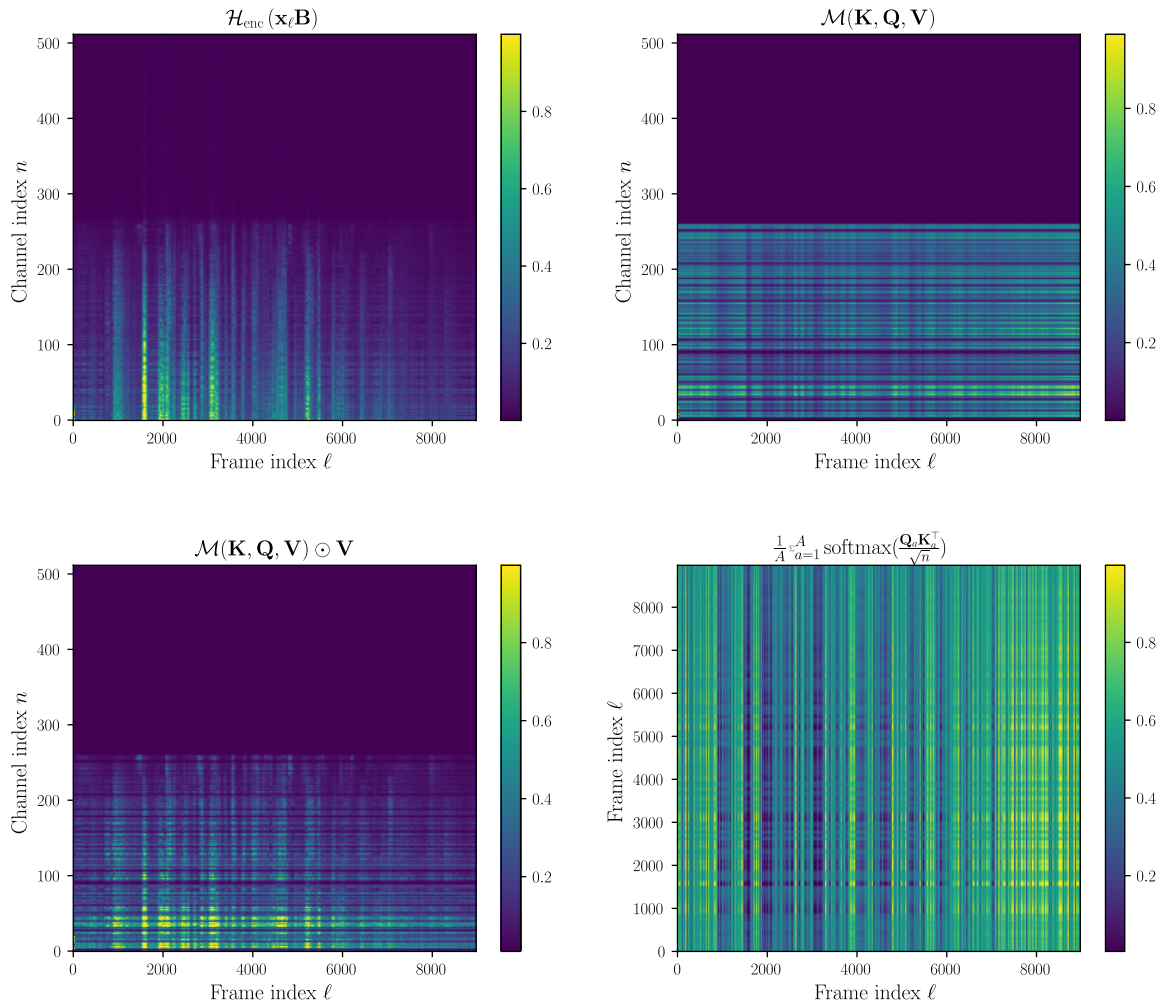


Figure 6.5: *Top left: encoded NRSM features after 1D convolution and non-linearity in MHA encoder sorted using Algorithm 1. Top right: mask-like output of self attentive MHA layer in MHA. Bottom left: output of the MHA encoder. Bottom right: Averaged attention weight matrix across all attention heads,* $G_{\text{heads}} = 4$.

performance across all conditions and metrics. The best performing of the proposed decoders across all conditions was the self-attention decoder. This decoder also outperformed the baseline Deep PReLU decoder with greater success the more challenging the audio became, c.f. SISDR results for CSM, NSM conditions with SISDR results for RSM and NRSM conditions.

6.3.9 Number of Heads Comparison in MHA Decoders

Results shown in Section 6.3.8 demonstrated that the proposed self-attention decoder in Figure 6.4 (c) was more effective than the MR and PM decoders. The MR decoder also showed some potential performance improvement for the CSM condition but this was not

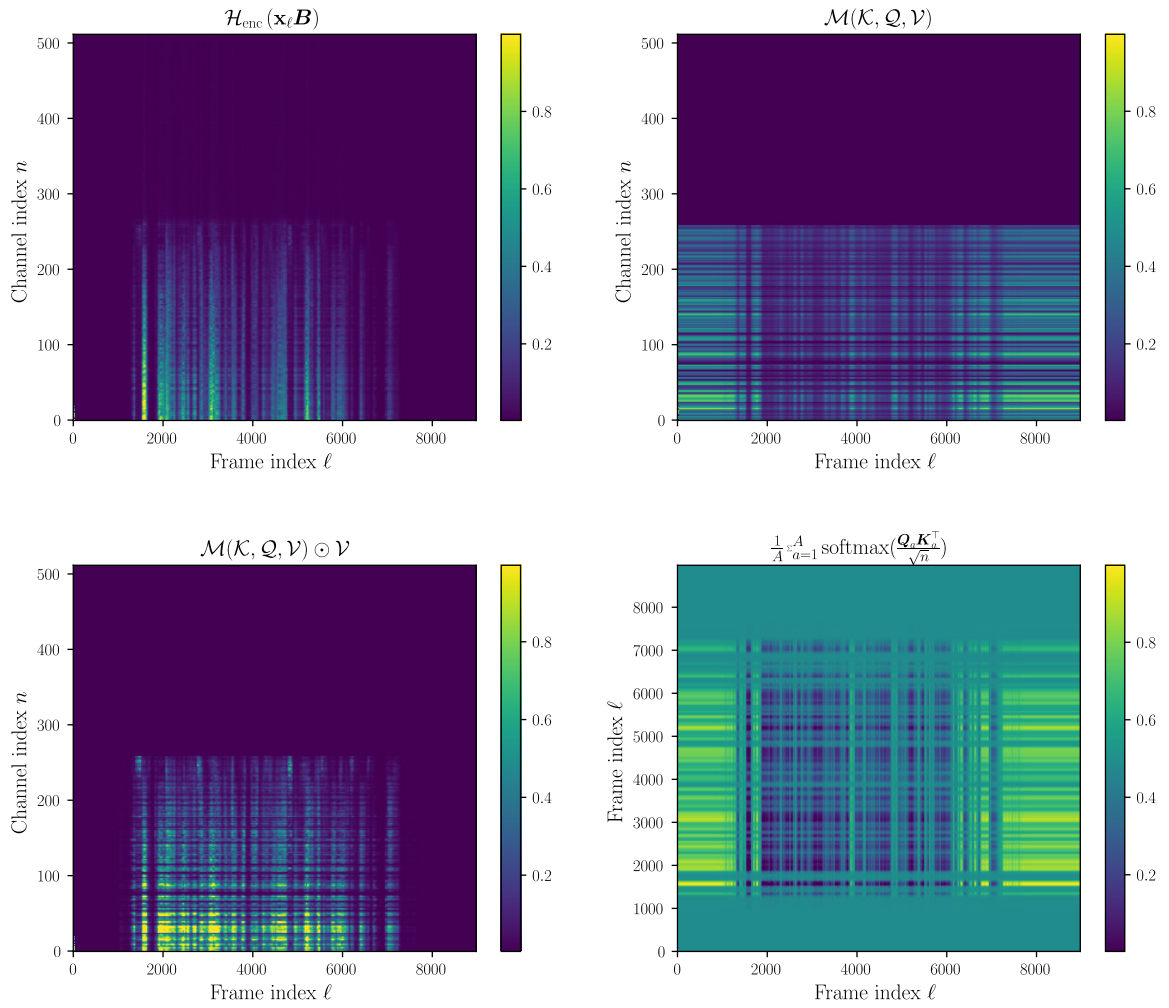


Figure 6.6: *Top left: encoded CSM features after 1D convolution and non-linearity in MHA encoder. Top right: mask-like output of self attentive MHA layer in MHA. Bottom left: output of the MHA encoder. Bottom right: Averaged attention weight matrix across all attention heads,* $G_{\text{heads}} = 4$.

replicated across all conditions. In the following subsection, further analysis is done using the SAD and MRD to observe the effect that using a variable number of heads might have on the model. Experiments were performed using $G_{\text{heads}} = \{2, 4, 8\}$ attention heads for both decoders and are again compared against the Conv-TasNet (Luo & Mesgarani, 2019) and Deep-PReLU baselines (Kadıoğlu et al., 2020). The results in Table 6.5 show that using $G_{\text{heads}} = 4$ attention heads leads to a small but consistent performance increase across all metrics used for MRD over the original Conv-TasNet decoder. The smallest improvement is often close to 0.1 dB SISDR and it is thought that this is not a strong enough improvement beyond the effects of randomized model initialization to confirm that this technique as implemented here is any more effective than the original Conv-TasNet decoder. The SAD again shows consistent

Table 6.4: *Comparison of MRD in Figure 6.4 (a) to PMD in Figure 6.4 (b) across various acoustic conditions. Best performing results for each acoustic condition shown in bold.*

AC	Decoder	SISDR (dB)	Δ	SDR (dB)	Δ	PESQ	Δ	STOI	Δ
CSM	Conv-TasNet	14.7	14.7	15.1	15	2.99	1.69	0.94	0.342
	Deep PReLU	15.0	15.0	15.5	15.3	3.01	1.72	0.943	0.345
	SAD	15.0	15.0	15.5	15.3	3.09	1.78	0.944	0.347
	MRD	**15.1**	**15.1**	**15.6**	**15.4**	**3.06**	**1.76**	**0.946**	**0.348**
	PMD	12.5	12.5	13.1	13.0	2.85	1.55	0.932	0.335
NSM	Conv-TasNet	7.63	12.1	8.28	12.5	1.97	0.838	0.824	0.373
	Deep PReLU	7.87	12.4	**8.55**	**12.8**	**2.05**	**0.913**	0.834	0.426
	SAD	**7.88**	**12.4**	8.53	12.8	2.05	0.9	0.842	0.434
	MRD	7.52	12.0	8.19	12.4	1.98	0.837	**0.837**	**0.429**
	PMD	7.39	11.9	8.08	12.3	1.96	0.82	0.835	0.427
RSM	Conv-TasNet	5.52	8.81	7.75	7.87	2.20	0.969	0.847	0.312
	Deep PReLU	5.85	9.14	7.88	7.99	**2.27**	**1.04**	0.856	0.32
	SAD	**5.92**	**9.2**	**8.07**	**8.19**	2.27	1.03	**0.859**	**0.323**
	MRD	5.77	9.06	7.96	8.07	2.20	0.976	0.855	0.319
	PMD	5.37	8.66	7.32	7.44	2.22	0.986	0.850	0.315
NRSM	Conv-TasNet	3.54	9.66	5.48	8.96	1.79	0.656	0.75	0.366
	Deep PReLU	3.68	9.81	5.54	9.03	1.82	0.681	0.761	0.373
	SAD	**3.87**	**9.99**	**5.74**	**9.22**	**1.88**	**0.718**	**0.774**	**0.385**
	MRD	3.19	9.32	5.12	8.61	1.76	0.62	0.769	0.381
	PMD	3.08	9.20	4.84	8.32	1.76	0.62	0.756	0.368

improvement over the previously demonstrated model with only 2 attention heads for both $G_{\text{heads}} = 4$ and $G_{\text{heads}} = 8$. Typically for both models $G_{\text{heads}} = 4$ leads to best average improvement across all metrics for both the MRD and SAD.

6.3.10 Combined MHA Encoder & Decoder Evaluations

The final set of results given in this section compare the MHA encoder and decoder approach to a deep convolutional encoder and decoder proposed by (Kadıoğlu et al., 2020). A Conv-TasNet model utilising both the proposed MHA encoder and decoder was trained in an E2E fashion. For all results the SAE, SAD and MRD 4 attention heads were used. Similarly, a Conv-TasNet model using both the deep encoder and decoder proposed by (Kadıoğlu et al., 2020) was trained. The SAE with the original Conv-TasNet decoder are reported with the decoder abbreviated to convolutional decoder (CD) for brevity in some results.

Table 6.5: *Comparison of using 2, 4 and 8 attention heads in MRD (Figure 6.4 (a)) against the original Conv-TasNet decoder proposed by (Luo & Mesgarani, 2019). Best performing results for each acoustic condition shown in bold.*

AC	Decoder	G_{heads}	SISDR (dB)	Δ	SDR (dB)	Δ	PESQ	Δ	STOI	Δ
CSM	Conv-TasNet	-	14.7	14.7	15.1	15	2.99	1.69	0.94	0.342
	Deep PReLU	-	15.0	15.0	15.5	15.3	3.01	1.72	0.943	0.345
	MRD	2	15.1	15.1	15.6	15.4	3.06	1.76	0.946	0.348
	MRD	4	15.0	15.0	15.4	15.3	3.07	1.75	0.944	0.347
	MRD	8	14.6	14.6	15.1	14.9	3.02	1.71	0.936	0.338
	SAD	2	15.0	15.0	15.5	15.3	3.09	1.78	0.944	0.347
	SAD	4	**15.3**	**15.3**	15.7	15.5	3.1	1.79	0.946	0.349
	SAD	8	**15.3**	**15.3**	**15.8**	**15.6**	**3.14**	**1.82**	**0.948**	**0.351**
NSM	Conv-TasNet	-	7.63	12.1	8.28	12.5	1.97	0.838	0.824	0.373
	Deep PReLU	-	7.87	12.4	8.55	12.8	2.05	0.913	0.834	0.426
	MRD	2	7.52	12.0	8.19	12.4	1.98	0.837	0.837	0.429
	MRD	4	7.74	12.2	8.40	12.6	2.04	0.87	0.834	0.426
	MRD	8	7.51	12.0	8.17	12.4	2.04	0.873	0.831	0.423
	SAD	2	7.88	12.4	8.53	12.8	2.06	0.9	0.842	0.434
	SAD	4	**7.97**	**12.5**	**8.62**	**12.9**	2.08	0.919	**0.844**	**0.436**
	SAD	8	7.96	12.5	8.61	12.8	**2.09**	**0.931**	0.841	0.433
RSM	Conv-TasNet	-	5.52	8.81	7.75	7.87	2.20	0.969	0.847	0.312
	Deep PReLU	-	5.85	9.14	7.88	7.99	2.27	1.04	0.856	0.320
	MR	2	5.77	9.06	7.96	8.07	2.20	0.976	0.855	0.319
	MR	4	5.58	8.87	7.84	7.96	2.25	1.00	0.846	0.311
	MR	8	5.46	8.75	7.71	7.83	2.21	0.968	0.846	0.306
	SA	2	5.92	9.2	8.07	8.19	2.28	1.03	0.859	0.323
	SA	4	**6.01**	**9.3**	**8.13**	**8.25**	**2.29**	**1.05**	**0.863**	**0.328**
	SA	8	5.99	9.28	8.12	8.24	2.28	1.04	0.862	0.326
NRSM	Conv-TasNet	-	3.54	9.66	5.48	8.96	1.79	0.656	0.75	0.366
	Deep PReLU	-	3.68	9.81	5.54	9.03	1.82	0.681	0.761	0.373
	MR	2	3.19	9.32	5.12	8.61	1.76	0.622	0.769	0.381
	MR	4	3.61	9.73	5.54	9.03	1.87	0.710	0.764	0.376
	MR	8	3.61	9.74	5.53	9.01	1.88	0.714	0.765	0.376
	SA	2	**3.87**	**9.99**	5.74	9.22	1.88	0.718	**0.774**	**0.385**
	SA	4	3.81	9.93	**5.74**	**9.23**	**1.89**	**0.728**	0.766	0.377
	SA	8	3.81	9.93	5.67	9.15	1.88	0.719	0.769	0.38

6.4 Conclusions and Future Work

In this chapter, novel MHA encoder and decoder networks were proposed for improving TasNet models. The proposed self-attention based MHA encoder demonstrated significant improvement over other encoder baselines across SISDR, SDR, PESQ, and STOI metrics. Three MHA decoders, two using encoder-decoder attention approaches and one using a self-attention approach, were proposed. Performance compared to the original Conv-TasNet model (Luo & Mesgarani, 2019) and a Deep PReLU decoder (Kadıoğlu et al., 2020) baselines varied. The Deep PReLU decoder typically performing better under most acoustic conditions than the MR decoders. The self-attention decoder consistently performed better than all the other proposed and baseline decoders. Using the MHA encoder alone yielded better performance

Table 6.6: *Comparison of MHA encoder and decoder against the deep convolutional encoder/decoder Conv-TasNet model proposed in Kadıoğlu et al. (2020). Best performing results for each acoustic condition shown in bold.*

AC	Model	Size	SISDR (dB)	Δ	SDR(dB)	Δ	PESQ	Δ	STOI	Δ
CSM	Conv-TasNet	3.5M	14.7	14.7	15.1	15	2.99	1.69	0.94	0.342
	Deep PReLU	8.2M	14.8	14.8	15.2	15.1	2.96	0.66	0.943	0.345
	SAE & MRD	5.5M	15.2	15.2	15.7	15.5	3.12	1.81	0.946	0.349
	SAE & SAD	5.5M	15.6	15.6	16.0	15.9	**3.16**	**1.85**	0.952	0.355
	SAE & CD	4.5M	**15.7**	**15.7**	**16.1**	**16.0**	3.15	1.84	**0.952**	**0.355**
NSM	Conv-TasNet	3.5M	7.63	12.1	8.28	12.5	1.97	0.838	0.824	0.373
	Deep PReLU	8.2M	8.20	12.7	8.88	13.1	2.07	0.938	0.849	0.441
	SAE & MRD	5.5M	7.97	12.5	8.62	12.9	2.06	0.896	0.839	0.431
	SAE & SAD	5.5M	8.3	12.8	8.94	**13.2**	**2.11**	**0.943**	0.852	0.444
	SAE & CD	4.5M	**8.37**	**12.9**	**9.01**	13.2	2.09	0.93	**0.854**	**0.446**
RSM	Conv-TasNet	3.5M	5.52	8.81	7.75	7.87	2.20	0.969	0.847	0.312
	Deep PReLU	8.2M	6.23	9.51	8.24	8.36	2.32	1.10	0.870	0.334
	SAE & MRD	5.5M	6.13	9.42	8.32	8.44	2.29	1.05	0.869	0.334
	SAE & SAD	5.5M	6.13	9.41	8.33	8.44	2.29	1.05	0.869	0.334
	SAE & CD	4.5M	**6.39**	**9.67**	**8.57**	**8.68**	**2.34**	**1.10**	**0.874**	**0.339**
NRSM	Conv-TasNet	3.5M	3.54	9.66	5.48	8.96	1.79	0.656	0.750	0.366
	Deep PReLU	8.2M	3.81	9.93	5.64	9.12	1.80	0.667	0.760	0.376
	SAE & MRD	5.5M	3.80	9.93	5.69	9.19	1.88	0.717	0.778	0.389
	SAE & SAD	5.5M	3.91	10.0	5.78	9.27	1.9	0.735	0.778	0.39
	SAE & CD	4.5M	**4.11**	**10.42**	**6.00**	**9.48**	**1.92**	**0.754**	**0.787**	**0.399**

than any changes to the decoder even with both an MHA encoder and MHA decoder. Further analysis of the intermediate MHA features in the self-attention encoder showed evidence that the network was being more selective in the features being attended to and that many of the channels in the encoder might be mostly redundant. The best-performing MHA models yield a mean 0.6 dB scale-invariant signal-to-distortion (SISDR) improvement on noisy reverberant mixtures over a baseline 1D convolution encoder. A mean 1 dB SISDR improvement is observed on clean speech mixtures.

There are a number of avenues for further research with the proposed MHA encoder and decoders. The MHA encoder demonstrated reliable performance improvements without the significant increase in model size seen in other encoder and decoder networks proposed for Conv-TasNet (Kadıoğlu et al., 2020). One drawback in any implementation using the MHA layer proposed by Vaswani et al. (2017) is the significant memory usage and computational complexity of these network layers. Work by Katharopoulos et al. (2020) proposed linear attention layers. Linear attention reduces the quadratic sequential complexity $L_{\mathbf{x}}^2$ of the scaled dot-product attention mechanism used by Vaswani et al. (2017) to have a linear complexity of $L_{\mathbf{x}}$. Another avenue for future work is to apply the self-attentive designs proposed in this chapter on other kinds of filterbank features such as the MPGT filterbank features proposed by Ditter & Gerkmann (2020). In this work, the focus was solely on using individual attention

mechanisms to improved performance but particularly with the encoder it is likely that using additional self-attention layers might lead to further improvements in performance across all acoustic conditions.

Chapter 7

On Data Sampling Strategies for TasNet Models

Many DNN speech separation models use LSTM or Transformer layers (Luo et al., 2020; Subakan et al., 2021; Rixen & Renz, 2022) which are typically less efficient than the convolutional layers focussed on heavily in Chapter 3 to Chapter 5. Transformers, in particular, consume large amounts of memory and have quadratic time-complexity, i.e. for L input frames of data the model performs at least L^2 operations (Katharopoulos et al., 2020). This is a particular concern in training when memory requirements are higher due to storing gradients for each operation required in the back-propagation stage (Haykin, 2009). This increased computational load also means longer training times. One way to compensate for the memory requirements is to use a batch size of 1 (Subakan et al., 2021) which leads to even longer training times as more parameter updates have to be performed. This is the case with the SepFormer model described earlier in Section 2.10. Another approach to training that reduces memory requirements and allows for larger batch sizes is to reduce the mixture signal length (Luo & Mesgarani, 2019). This reduces the training time but potentially at the expense of performance.

In this chapter, the first aim is to address if there are TSL limits at which seemingly no additional performance gain can be attained by DNN speech separation models. It is shown that, depending on model and dataset selection, there is a TSL limit at which not only no additional performance gain is attained but actually limiting the TSL to a specific value can lead to notably improved performance on some datasets. This effect is demonstrated to be due to a random sampling of the start index when using TSL limits. Further evaluations show the benefit of having more unique training examples than using the full signal lengths. Finally,

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2023b).

the application of TSL limits used with DM (Zeghidour & Grangier, 2021) is evaluated.

The remainder of this chapter is structured as follows. The separation networks and datasets used are described in Section 7.1.1, and Section 7.1.2.1, respectively. Section 7.2 presents evaluations for varying TSL limit for each separation network and dataset. Section 7.3 explores splitting signals to generate more train examples and whether DM mitigates the gains found using TSL limits. Final conclusions are provided in Section 7.5.

7.1 Experimental Setup

7.1.1 Separation Models

The SepFormer (Subakan et al., 2021) and Conv-TasNet (Luo & Mesgarani, 2019) models are both used in this chapter for the purpose of analysis on different signal lengths. These models are selected due to their contrasting features; Conv-TasNet being a smaller (3.6M parameters), more efficient, fully convolutional model and SepFormer conversely being a much larger (25.8M parameters) less efficient, Transformer based model. The reader is encouraged to refer to Section 2.9 for an in-depth description of Conv-TasNet and Section 2.10 for an in-depth description of Sepformer. The fully convolutional structure of Conv-TasNet means that it is only able to process more localized information within its receptive field (cf. Eq. (2.44)) and the Transformer based architecture of SepFormeris able to model global context.

Both Conv-TasNet and SepFormer have a similar overall structure owing to them both being derived from the TasNet model, an example of which is visualized in Figure 2.6 for $C = 2$ speakers Both are also trained using the PIT SISDR objective function (cf. Section 2.6) (Kolbaek et al., 2017; Luo & Mesgarani, 2019; Subakan et al., 2021; Roux et al., 2019). Models are trained according to the best-performing models in each of their original papers (Luo & Mesgarani, 2019; Subakan et al., 2021) unless stated otherwise. Batch sizes of 2 and 4 are used for SepFormer and Conv-TasNet, respectively, except where otherwise stated.

7.1.2 Datasets & Signal Length Distributions

7.1.2.1 Datasets

Three corpora of 2-speaker mixtures are analysed in this work: WSJ0-2Mix, WHAMR and Libri2Mix. The trends demonstrated later in Sections 7.2 and 7.3 for the 2-speaker scenario are assumed to generalize to higher C values based on findings in Li et al. (2024); Cosentino et al. (2020). The reader should refer back to Section 2.2.1 and Section 2.2.2 for descriptions of these corpora. For the Libri2Mix dataset, the configuration without noise, i.e. $\nu[i] = 0$ in (2.2), is used for the purpose of comparison to WSJ0-2Mix. Furthermore, the signal length distribution of LibriMix differs greatly for WSJ0-Mix derived datasets, an important point of

comparison in the remainder of this chapter. For the Libri2Mix dataset, *train-100* is used for training. The motivation for this selection is faster training times than the larger *train-360* set with near identical signal length distributions. For all corpora, the 8kHz *min* configuration is used.

7.1.2.2 Signal Length Distributions

The distribution, density estimation (DE) and the mean and standard deviation of the mixture signal length in the WHAMR train *tr* and test *tt* sets can be seen in Figure 7.1. WSJ0-2Mix

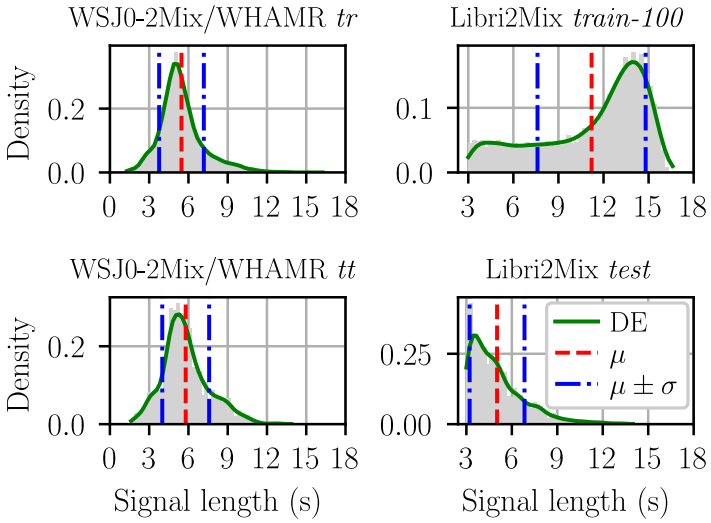


Figure 7.1: *Distributions of mixture signal lengths in WSJ0-2Mix/WHAMR (left) and Libri2Mix (right) for both train (top) and test (bottom) sets. Density estimation (DE) is shown by solid green lines, mean values are indicated by dashed red lines and standard deviation values by dash-dotted blue lines.*

and WHAMR have identical signal distributions as WHAMR is derived from the former. These distributions are shown in the left panel. The train and test sets in WHAMR have similar distributions of signal length with mean values within 0.3s of one another and standard deviations within 0.1s of one another. This contrasts the distributions of the Libri2Mix dataset (Cosentino et al., 2020) also shown in Figure 7.1 where the *train-100* and *test* sets have a difference in mean value of 6.2s and difference in standard deviation of 1.79s.

7.2 Training Signal Length Analysis

Evaluations of varying the TSL limits are presented in this section. For all evaluations, the improvement in SISDR over the input mixture signal, denoted by Δ SISDR, is used as the evaluation metric. More details on SISDR are given in Section 2.6, and the formulation is found in (2.72).

7.2.1 Initial TSL Limit Evaluations

As a first experiment, twelve SepFormer models are trained and evaluated on WSJ0-2Mix, WHAMR and Libri2Mix, each with a different TSL limit. Twelve logarithmically spaced signal limits T_{lim} are selected between 0.5s and 10s:

$$T_{\text{lim}} \in \{0.5, 0.66, 0.86, 1.13, 1.49, 1.95, 2.56, 3.36, 4.42, 5.8, 7.62, 10\}\text{s}. \tag{7.1}$$

The notation L_{lim} is used for the respective discrete sample index, i.e. $L_{\text{lim}} = T_{\text{lim}}\text{f}_s$ for sampling rate f_s. When cutting the training signal lengths such that $L_x \leq L_{\text{lim}}$ the starting sample index of the signal is randomly selected from the discrete uniform distribution of integers $\mathcal{U}(0, 1 + max(0, L_x - L_{\text{lim}}))$. Performance for SepFormer models trained and evaluated on all three datasets is compared in Figure 7.2. For the WHAMR corpus, an increase in overall Δ

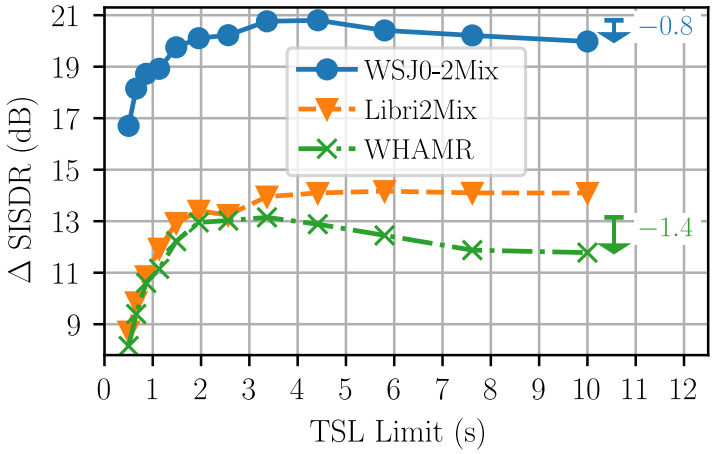


Figure 7.2: *SepFormer results for varying the TSL limit for the anechoic WSJ0-2Mix (top), Libri2Mix (middle) and WHAMR (bottom) test sets.*

SISDR performance from the 0.5s to 1.95s limit can be observed. The optimal TSL is at 3.36s. Between 3.36s and 10s performance decreases again by −1.4dB. This may seem surprising as the general convention with training DNNs is that more data normally results in improved overall performance. For an immediate explanation of this effect refer ahead to Section 7.4. A similar trend is observed for WSJ0-2Mix where there is a notable increase between 0.5s and 3.36s and then a drop in performance of 0.8dB between 4.4s and 10s. For Libri2Mix, the performance saturates before a TSL limit of 4.42s. There is no drop in performance as the TSL limit approaches 10s which is likely due to the Libri2Mix training set having a more uniform distribution below signal lengths of 10s than the WHAMR or WSJ0-2Mix datasets, cf. Figure 7.1. More is elaborated on this effect in later sections.

The results for the WHAMR evaluation set are separated into quartiles of mixture signal length for the following experiment. Δ SISDR results for each quartile are shown in Figure 7.3.

Comparing Q1 to Q4 shows that, with a sufficiently large TSL ($\geq$ 1.95s), the best separation

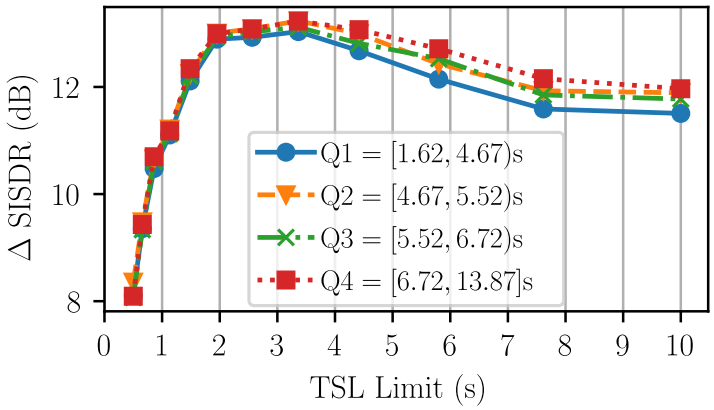


Figure 7.3: *Training signal length (TSL) analysis of the 1st to 4th signal length quartiles in the WHAMR evaluation set.*

performance in SISDR is found on the longest signal lengths, regardless of TSL. A loss in SISDR performance is still observed from 3.36s to 10s regardless of which quartile is evaluated.

7.2.2 Training Time Evaluation

The average training ED for the SepFormer model on WHAMR and Libri2Mix training sets are shown in Figure 7.4. Note the ED for WSJ0-2Mix is omitted for brevity but is similar to WHAMR due to having the same TSL distribution (cf. Figure 7.1). All models were trained on the same hardware to control any impact this has on speed. The average EDs

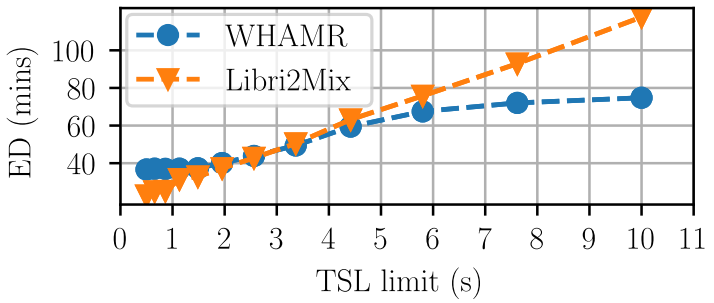


Figure 7.4: *Comparison of average epoch duration (in mins) for the SepFormer model on the Libri2Mix and WHAMR training sets.*

for the WHAMR dataset have a sigmoidal shape due to the majority of the signal lengths being concentrated around the mean signal length of the training set (5.6s, cf. Figure 7.1). Libri2Mix has a more linear relationship between TSL limit and ED due to the more uniform shape of the signal length distribution below the maximum TSL limit of 10s in the *train-100* set, cf. Figure 7.1. Reducing the TSL limit has more benefit in terms of ED for the Libri2Mix dataset when iterating over all training examples.

7.2.3 Fixed vs. Random Start Index

In Section 7.2.1, the start index of each shortened signal was randomly sampled from a uniform distribution. In this section, this is compared to using a fixed start sample. A start sample of 1999 (= 0.25s at 8kHz, zero-indexed) was used for signals where the original mixture signal length was larger than L_{lim}, else the entire signal length was used. The motivation for this was that many training examples contain silence at the beginning of clips. It was considered desirable to omit as much silence to make for a fairer comparison with the randomly sampled clips which are assumed to have a lower likelihood of beginning with silence. Results in Figure 7.5 confirm that the loss in performance from a TSL limit of 3.36s to 10s with WHAMR is due to randomly sampling the start index. The performance saturates at a TSL limit of

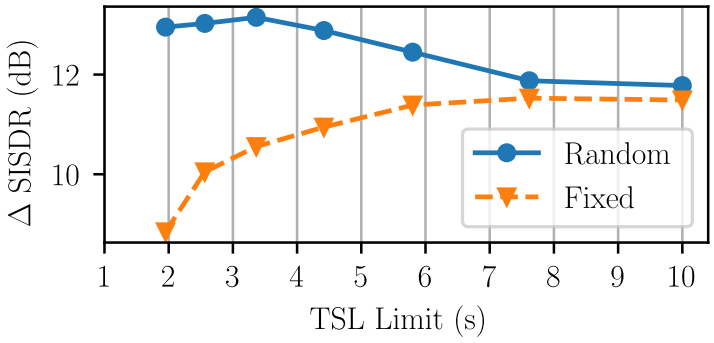


Figure 7.5: *Comparison of TSL variation for the SepFormer model trained and evaluated on the WHAMR datasets on a subset of TSL limits in the range* $[1.95, 7.62]s$.

5.8s when using a fixed start index. This is similar to the performance saturation point of Libri2Mix in Figure 7.2 demonstrating the performance drop in higher TSLs seen before on WHAMR (cf. Figure 7.2) is related to both a non-uniform TSL distribution and the random sampling used.

7.2.4 Transformer vs. Convolutional Model

Results comparing the Conv-TasNet model (cf. Section 2.9) to the SepFormer model (cf. Section 2.10) are shown in Figure 7.6. The loss in performance above 3.36s is not as notable for the Conv-TasNet model. All SISDR results above $T_{\text{lim}} = 1.95$s are within in 0.5dB of each other suggesting Conv-TasNet is more invariant to the TSL limit if the limit is sufficiently large. This is possibly due to the 1.53s receptive field of the Conv-TasNet models being smaller than these particular TSLs limits (Ravenscroft et al., 2022b).

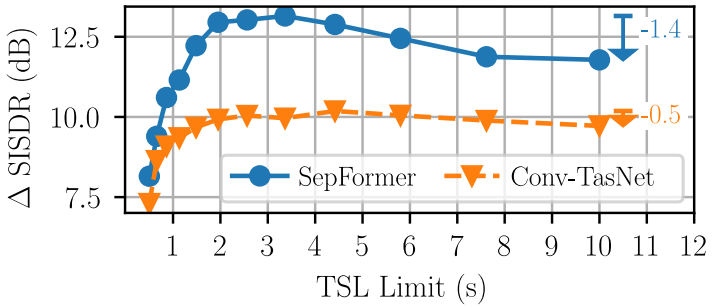


Figure 7.6: *Comparison of SepFormer and Conv-TasNet across TSL limits* $T_{\text{lim}} \in [4.42, 7.62]$ *using the WHAMR corpus.*

7.3 Signal Splitting and Dynamic Mixing

Next, two sampling strategies are evaluated to (i) investigate whether with WHAMR the performance gained by using TSL limits with random sampling on shorter sequences still holds when the same quantity of audio data in terms of length in seconds is used and (ii) whether using TSL limits still result in performance gains with DM, i.e. simulating new speech mixtures for each epoch (Zeghidour & Grangier, 2021).

7.3.1 Signal Splitting

A signal splitting strategy was designed such that a batch of inputs $\mathbf{Y}_{\text{bat}} \in \mathbb{R}^{W \times L_x}$ was reshaped to $\mathbf{Y}'_{\text{bat}} \in \mathbb{R}^{WG_{\text{split}} \times \frac{L_x}{G_{\text{split}}}}$ for batch size W. Signal length L_x is still limited such that $L_x \leq L_{\text{lim}}$. The motivation of this method is to evaluate the importance of training on the entire sequence length compared to the raw data quantity used in seconds. Computational complexity in training is also reduced. TSLs for $T_{\text{lim}} \in [4.42, 10]$s are analysed for $D = 2$. Figure 7.7 shows that $DG_{\text{split}} = 2$ improves performance for shorter TSL limits ($T_{\text{lim}} \leq 5.8$s) compared to $G_{\text{split}} = 1$ (the original shape). However for $T_{\text{lim}} \geq 7.62$s the performance is similar to $G_{\text{split}} = 1$. As in Figure 7.2 this is likely due to the TSL distribution of WHAMR.

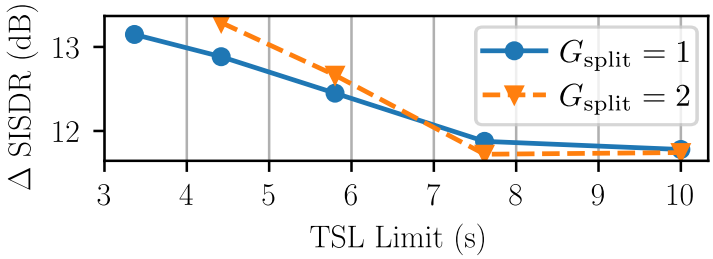


Figure 7.7: *Comparison of split signal and batch reshape training* $G_{\text{split}} = 2$ *(orange) against full signal training* $G_{\text{split}} = 1$ *(blue) for the SepFormer model.*

7.3.2 Dynamic Mixing

As discussed in previous chapters (cf. Chapter 5), dynamic mixing (DM) has been shown to consistently improve the performance of various speech separation models (Zeghidour & Grangier, 2021; Subakan et al., 2023). DM often results in a 1.0 to 1.5dB SISDR performance improvement dependant upon the model and dataset (Zeghidour & Grangier, 2021; Ravenscroft et al., 2023a). The random start index sampling used in Section 7.2 is similar to DM in that it provides the model with unique training examples each epoch but without simulating new mixtures. In this section using DM and TSL limits is compared against just using TSL limits to see if further performance gains can be attained using both approaches. The DM results for the WHAMR corpus are shown in Figure 7.8. It can be seen that with DM the drop in performance is less (at 7.62s) than without DM. The best performing model $T_{\text{lim}} = 4.4$s is

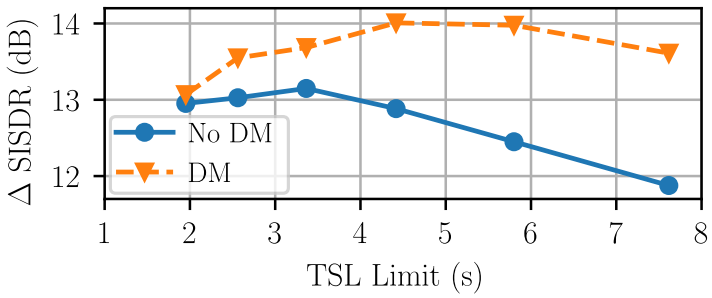


Figure 7.8: *Comparison of SepFormer model training using TSL limits with and without dynamic mixing being used as well on WHAMR evaluation set.*

compared to the Sepformer model with no TSL limit in Table 7.1. The batch size reported for the model with no TSL is the largest it was found possible to train on a 32GB Nvidia V100 GPU. It can be seen that using the TSL limited model is able to match its performance with an average ED reduction of 44%, highlighting the benefit of this approach.

Table 7.1: *Comparison of best performing SepFormer models on WHAMR with and without TSL limits. W denotes batch size and the average epoch duration (ED) is reported in minutes.*

Model	W	T_{lim} (s)	Δ SISDR (dB)	ED (mins)
SepFormer + DM	1	–	**14.0**	85
SepFormer + DM	2	7.62	13.6	74
SepFormer + DM	2	4.42	**14.0**	**59**

7.4 Discussion

When this work was originally published in Ravenscroft et al. (2023b), the paper lacked of formal explanation for why the shorter TSL limits can result in improved performance. Below,

this is outlined using a more mathematical formulation than the verbal explanations given earlier in this chapter. Given the input signal length L_x and signal length of the TSL in samples L_{lim} using TSLs with random sampling means that the data can now be sampled from

$$\xi = L_x - L_{\text{lim}} \tag{7.2}$$

possible unique training examples can be sampled. A consequence of this is that the larger L_{lim} is, the smaller ξ is, i.e. the fewer unique training examples the model is likely to be trained on and the less diverse the training examples will be, due to the higher likelihood a "unique" training example will overlap with a previous training example the larger L_{lim} is. Thus, if the TSL distribution doesn't skew much larger than the TSL limit, as is the case with the 4.4s TSL limit on WHAMR (cf. Section 7.1.2.2) it is favourable to use shorter TSL limits as it improves the overall data diversity.

7.5 Conclusion

In this chapter, it was shown that TSL limits can affect the overall performance for speech separation models in a number of ways. For WSJ0-derived speech separation benchmarks, i.e. WSJ0-2Mix and WHAMR, it is optimal to use shortened training examples randomly sampled from the original examples due to the signal length distribution of these corpora. For the Libri2Mix dataset, the same method led to shorter training times with no notable loss in performance. The SepFormer model was compared to the Conv-TasNet model and it was shown that the Conv-TasNet performance has less variation. Using dynamic mixing and TSL limits with random sampling was shown to be able to match the performance of the SepFormer model trained DM using full sequence lengths on WHAMR with a 44% reduction in training time. With some previous literature opting to limit TSLs (Luo & Mesgarani, 2019; Luo et al., 2020) and others not (Rixen & Renz, 2022; Subakan et al., 2021) while using the same benchmark datasets for evaluating models, the results in this work suggest that this is not necessarily a fair comparison and that TSL limiting is important to factor in when analysing results, particularly for the WSJ0-2Mix and WHAMR benchmarks.

Chapter 8

Time-domain Conformers for Speech Separation

In the Chapter 7, it was shown that the training of the SepFormer model can be significantly improved using TSL limits combined with random sampling with no loss in performance. Even with this modification, the total training time is still over 6 days on the same hardware used in the original SepFormer paper (Nvidia V100 32GB GPU) (Subakan et al., 2021). It is shown in this chapter that a large source of the computational complexity in the SepFormer model is the intra-Transformer used for processing local context with a fixed window of size U. This fixed window size means that, for all L_U chunks the model intra-Transformer processes, it results in a time-complexity function of $L_U \times U^2$. As is demonstrated later in Section 8.1.3, this can have a significant impact on the efficiency of the model in processing shorter signal lengths. This chapter proposes the Conformer (Gulati et al., 2020) model as an analogue of the SepFormer model. The Conformer, originally proposed for speech recognition (Gulati et al., 2020), is notably more efficient for computing shorter signal lengths because it uses convolutional layers for processing local context while still modelling global context via a Transformer layer. It is demonstrated in this chapter that using the Conformer can result in more efficient training with improved performance on the noisy reverberant WHAMR speech separation benchmark dataset.

A STFT-based Conformer model has been proposed for continuous speech separation in (Chen et al., 2021b) which shows reasonable performance on the LibriCSS dataset but does not compare to time-domain models on other popular benchmarks as it is shown later in Section 8.3.3, most likely due to its lower temporal resolution as Cord-Landwehr et al. (2022) similarly demonstrated. Conformers have been proposed widely in other areas of

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2023c).

speech processing. Most closely related to speech separation, a TCN augmented Conformer model has been proposed for speech enhancement (Koizumi et al., 2021) and a time-domain Conformer and a TCN-augmented Conformer model were proposed for speech extraction in Sinha et al. (2022). The TCN model performs best but has a much wider local context window, or receptive field, than the pure Conformer model. Questions remain, however, about whether the full TCN approach is necessary if a convolutional module in a Conformer were to have a sufficiently wide kernel size (Sinha et al., 2022; Rixen & Renz, 2022) and comparable model size, as this was not analysed in Sinha et al. (2022).

This chapter evaluates a single channel TD-Conformer model across different model sizes and computational expenditures from both theoretical and experimental perspectives. While TasNets and Conformer models are well studied, the combination of the two for separation tasks with a corresponding optimisation and performance analysis is as yet missing from the speech separation literature. Furthermore, this chapter demonstrates why it is an oversight in the area of speech separation research to not explore this particular combination in greater depth given the proposed model configuration in this work is able to achieve close to SOTA results with useful trade-offs in computational expenditure for the separation of shorter speech utterances. It is shown that, in the local-layer global-layer paradigm often used in speech separation DNN models (Luo et al., 2020; Chen et al., 2020a; Subakan et al., 2021), Conformer layers can be a more computationally suitable option in terms of efficiency.

In this work, a subsampling method is also introduced, which further reduces the computational TC of the dot product attention in the Transformer layers, similar to the SE-Conformer model (Kim & Seo, 2021). Several evaluations are performed to assess the optimal receptive field (Ravenscroft et al., 2022b) of the convolutional component in the Conformer, the impact of subsampling on performance and computational complexity, and the benefits of the time domain approach over the STFT model proposed in (Chen et al., 2021b). Results are compared to a number of other SOTA models in terms of performance, model size, and computational complexity across multiple benchmarks and acoustic conditions.

The remaining chapter proceeds as follows. In Section 8.1 the proposed TD-Conformer model is described in detail. Training configurations are explained in Section 8.2 and results are presented in Section 8.3. Section 8.4 gives the conclusions for this chapter.

8.1 TD-Conformer Separation Networks

The proposed TD-Conformer, described in the following, is a TasNet composed of the same three main components as all other TasNet models previously discussed: a feature encoder, a mask estimation network and a decoder (cf. Figure 2.6). The encoder is the same learnable filterbank (Luo & Mesgarani, 2018b; Ditter & Gerkmann, 2020) based on 1D convolutional layer with a ReLU function described in Section 2.9.1 and also used in the SepFormer model

(Section 2.10). The decoder is again identical to that used in Conv-TasNet and SepFormer (Section 2.9.1) that performs the inverse function to convert encoded features back into the time domain. The mask estimation network calculates C masks $\mathbf{m}_{\ell,c}$ for each time frame ℓ to obtain estimated frames $\hat{\mathbf{s}}_{\ell,c}$ of the clean input speech signals $\mathbf{s}_{\ell,c}$. Boldface letters indicate vectors capturing respective frames of samples of the quantities in Eq. (2.2).

8.1.1 Encoder & Decoder

Similar to (Subakan et al., 2021; Luo & Mesgarani, 2019), the mixture signal $x[i]$ in Eq. (2.2) is segmented into $L_{\mathbf{x}}$ overlapping blocks $\mathbf{x}_\ell$ of size J which are encoded into $\mathbf{w}_\ell \in \mathbb{R}^{1\times N}$ using a 1D convolutional layer with weights $\mathbf{U} \in \mathbb{R}^{J\times N}$ for N output channels, followed by a ReLU activation function $\mathcal{H}_{\text{enc}} : \mathbb{R}^{1\times N} \to \mathbb{R}^{1\times N}$ producing encoded features

$$\mathbf{w}_\ell = \mathcal{H}_{\text{ReLU}}\left(\mathbf{x}_\ell \mathbf{U}\right) \in \mathbb{R}^N \forall \ell \in [1, \ldots, L_{\mathbf{x}}]. \tag{8.1}$$

The masked encoded mixture $\mathbf{m}_{\ell,c} \odot \mathbf{w}_\ell$ is decoded back into the time domain signal $\mathbf{s}_{\ell,c}$ using a transposed convolutional layer of weights $\mathbf{B}$, the same as that described Section 2.9.1.

8.1.2 Conformer Mask Estimation Network

The mask estimation network is based on the Conformer architectures proposed in (Gulati et al., 2020; Kim & Seo, 2021) and Figure 8.1. A diagram of the Conformer mask estimation network is shown in Figure 8.1. The input sequence of features $\mathbf{w}_\ell$ is normalized using layer normalization (Ba et al., 2016) before being projected from dimension size N to B using a P-Conv layer followed by a PReLU activation. Note that B is the same notation used in the TCN, DTCN and WD-TCN models, as all networks are structured the same up to this point. This results in a sequence of features of shape $L_{\mathbf{x}} \times B$. This sequence is then fed through S subsampling layers each of which is a 1D convolutional layer of a fixed kernel size of 4 and stride of 2 thus reducing the temporal dimension by a factor of 2 giving a sequence of shape $\frac{L_{\mathbf{x}}}{2^S} \times B$. Each subsampling layer has a residual connection to a respective supersampling block composed of a 1D transposed convolutional layer, PReLU and layer normalization. This structure increases the temporal dimension by a factor of 2 to restore the sequence length to $L_{\mathbf{x}}$. In between the subsampling and supersampling blocks are a series of R_{conf} Conformer layers.

The Conformer layers are shown in detail in Figure 8.2 and are composed of a feed-forward module, a convolution module, a MHSA module and another feed-forward module. The convolution module comes before the MHSA module contrasting the original Conformer (Gulati et al., 2020) which has MHSA first. This is so the model processes the local context first similar to the DP models proposed in (Luo et al., 2020; Subakan et al., 2021). The two feed-forward

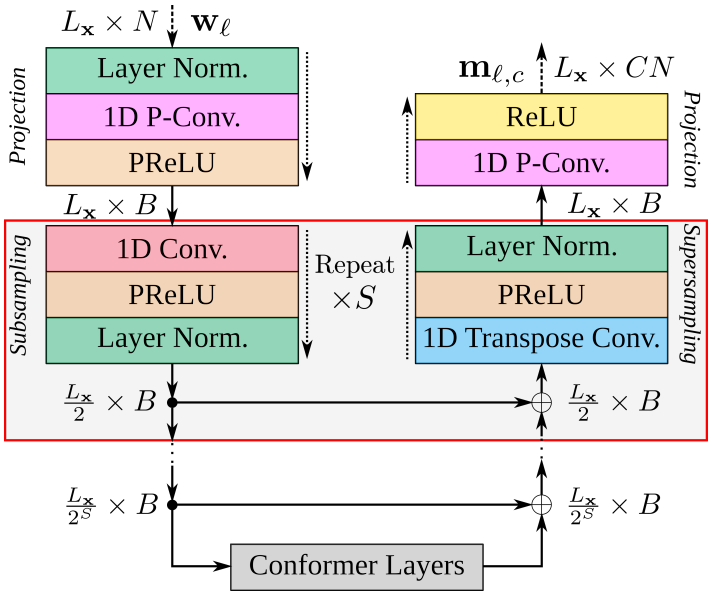


Figure 8.1: *Proposed TD-Conformer mask estimation network structure with subsampling and supersampling layers to reduce and increase the temporal resolution, and also enabling a reduction of the time-complexity (TC) in the Conformer layers.*

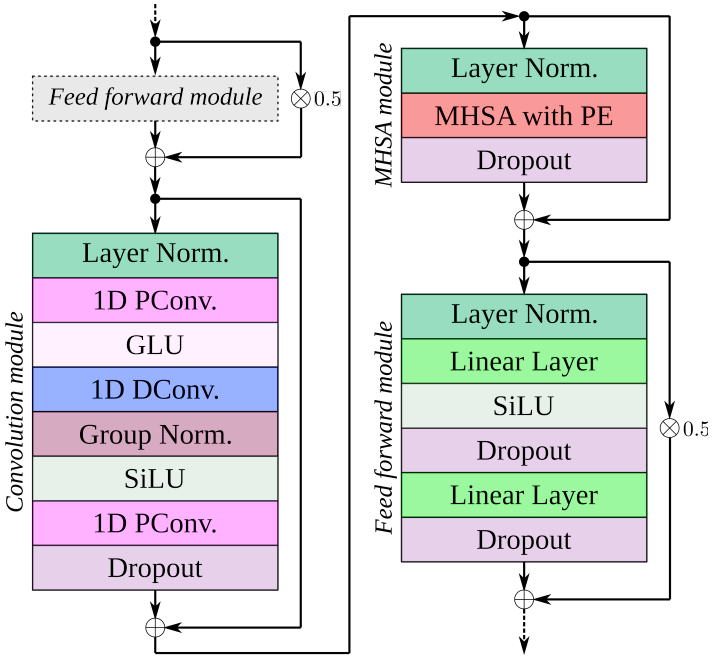


Figure 8.2: *Diagram of a single Conformer layer composed of feed-forward, convolution and MHSA modules. Note the first feed-forward module is identical to the final module but its details are omitted for brevity.*

modules are composed of layer normalization, a linear layer with SiLU activation (Elfwing et al., 2018), dropout (Hinton et al., 2012b) and then another linear layer followed by another

dropout. Dropout (Hinton et al., 2012b) is a method used in DNNs where certain neurons are randomly ignored (or zeroed out) during training, predominantly to improve a model's generalization capabilities and prevent overfitting from occurring. Each feed-forward module has a weighted residual connection from input to output. The convolution module is composed of layer normalization, a P-Conv with GLU activation, a D-Conv with kernel size P, group normalization (Wu & He, 2018), a SiLU activation and lastly a P-Conv layer followed by dropout. For group normalization, the number of groups equals the number of input channels, i.e. instance normalization (IN). A residual connection goes from the input to the output of the module. The MHSA module is composed of layer normalization and MHSA with relative PE (Vaswani et al., 2017) followed by dropout. The MHSA module has a residual connection around the entire module.

8.1.3 Conformers vs. Dual-Path Transformers

The Conformer layers are proposed in analogy to the DP Transformer layers in the widely researched SepFormer model (Subakan et al., 2021; Cord-Landwehr et al., 2022). Note that other DP Transformer layers have been proposed, such as in (Chen et al., 2020a), but these are disregarded here as they are more computationally expensive and less performant than SepFormer (Subakan et al., 2021; Tzinis et al., 2022b). In this section, the respective TC functions of the Conformer layer and the DP Transformer layer are modelled to demonstrate that under certain (often more realistic) signal lengths and feature dimensions, Conformers are less computationally complex than DP Transformers. The intra-Transformer is compared to the convolutional module of the Conformer as a local context layer and the inter-Transformer is compared to MHSA modules of the Conformer as a global context layer. The TC for a Conformer layer is defined as

$$\mathcal{T}_{\text{Conf}} = \frac{L_{\mathbf{x}}}{2^S}\left(PB + B^2\right) + \frac{L_{\mathbf{x}}^2}{2^{2S}}B + B^2\frac{L_{\mathbf{x}}}{2^S}. \tag{8.2}$$

The TC for a dual-path Transformer layer in the style of SepFormer (Subakan et al., 2021) is defined as

$$\mathcal{T}_{\text{DPT}} = \left(\frac{L_{\mathbf{x}}}{2U} + \frac{U}{2}\right)\left(U^2B + B^2U\right) + \frac{L_{\mathbf{x}}}{4U^2}B + B^2\frac{L_{\mathbf{x}}}{2U}, \tag{8.3}$$

where U represents the chunk size (Subakan et al., 2021). The time-complexities for DP Transformer and Conformer layers with a feature sequence from an 8kHz piece of audio encoded with block length $J = 16$ are shown in Figure 8.3 for different feature dimensions B where the DP Transformer features dimension is fixed to $Z = B$. The average and maximum signal lengths in the WHAMR *tt* evaluation set used later in Section 8.3, are 5.79s and 13.87s, respectively (Ravenscroft et al., 2023b). Hence, in evaluations on this dataset, the Conformer layer is the less computationally complex option overall. For larger B values the Conformer

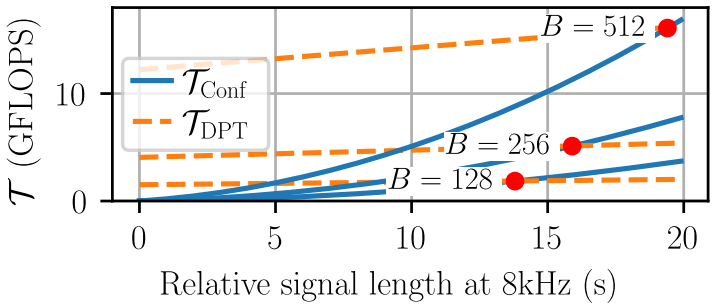


Figure 8.3: *Comparison of TCs measured in GFLOPS for a Conformer layer and a DP Transformer layer for different feature dimensions $B \in \{128, 256, 512\}$, where the feature dimension of the DP Transformer is fixed to $Z = B$, over relative signal length in seconds. Note that $S = 0$ for the Conformer layer here and $P = U = 250$ is used as it is equal to the best performing configuration of the DP model in (Subakan et al., 2021).*

more likely has lower TC than the DP Transformer. Another benefit of the Conformer is that, assuming a small number of subsampling layers, it has a much higher temporal resolution than the DP approach when processing the global context. Figure 8.4 demonstrates that every

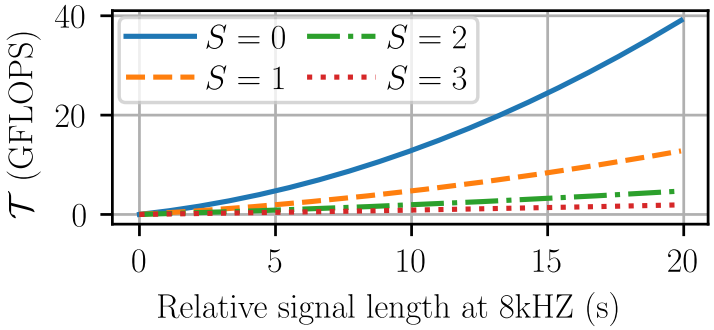


Figure 8.4: *Comparison of Conformer TC function over relative signal length for varying subsampling layers $S \in \{0, 1, 2, 3\}$.*

additional subsampling layer can significantly reduce the TC of the Conformer layer. The impact of increasing the subsampling on overall performance is explored in more depth later in Section 8.3.2. Note that the DP Transformer topology also has its own implicit subsampling strategy akin to a dilated view of the output tensor from the local Transformer layer which reduces the computational complexity but also the temporal resolution significantly (Rixen & Renz, 2022; Subakan et al., 2021).

8.1.4 Objective function

An SISDR objective function (Roux et al., 2019; Luo & Mesgarani, 2018b) with an PIT wrapper (Kolbaek et al., 2017) is used to train all of the models. The formula for SISDR can be found in Eq. (2.72), and a description of PIT can be found in Section 2.6.2.

8.2 Experimental Setup

8.2.1 Data

The WHAMR (Maciejewski et al., 2020) and WSJ0-2Mix (Isik et al., 2016) datasets are used for analysing the proposed TD-Conformer models. Details of each corpus can be found in Section 2.2.1. As the noisy reverberant condition is the main area of interest for this thesis, the WHAMR dataset is used for all results to fine-tune hyperparamaters as well as compare the proposed TD-Conformer to other similar models. WSJ0-2Mix is used as a benchmark for the purposes of comparison to other models in Section 8.3.3. For all evaluations the 8kHz *min* configuration is used.

8.2.2 Network configurations

Four model configurations are proposed to vary the model size in increasing internal mask estimation feature dimension B of 128 (denoted as small (S)), 256 (medium (M)), 512 (large (L)) and 1024 (extra-large (XL)). The encoder has a fixed output dimension of $N = 256$ and a kernel size of $J = 16$ with a 50% stride of 8 samples. The number of Conformer layers is fixed to $R_{\text{conf}} = 8$, the same as the number of DP layers in (Subakan et al., 2021). In the results sections the number of subsampling layers S and Conformer kernel size P are modified to gain a better understanding of their impact on separation performance and computational cost. For all evaluations, a dropout of 10% is used.

8.2.3 Training configurations

All models are trained using an initial learning rate of 5×10^{-5}. Learning rates are fixed for the first 90 epochs and then halved if there is no performance improvement after 3 epochs. Training is performed over 200 epochs. Training examples were limited to 4 seconds. In (Ravenscroft et al., 2023b) it was shown that this signal length is in an optimal range to reduce training time without impacting overall performance for the datasets used in this work. By limiting the TSLs, it enables the use of a batch size of 4 even with the largest XL models proposed in Section 8.2.2. This contrasts the best performing SepFormer model where it has been shown that even with comparable TSL limits the largest batch size it was possible to use was 2 (Ravenscroft et al., 2023b) on the same GPU used in this work, a 32GB Nvidia V100. Further to the discussion in Section 8.1.3, the reason for this difference is that despite both the TD-Conformer XL model and the Sepformer using an internal feature dimension of 1024, the use of MHSA in the Sepformer for processing local information consumes a lot more memory for an utterance of 4s in length. An open-source SpeechBrain (Ravanelli et al., 2021) *recipe* is provided with this work to enable other researchers to reproduce the results in this

chapter.[1]

8.2.4 Evaluation Metrics

The main separation evaluation metric used is SISDR (cf. (2.72)) improvement over the noisy mixture, denoted by Δ SISDR and measured in dB. Where relevant, computational expenditure is report using MACs. MACs are calculated using the *thop* (Zhu, 2022) toolkit on a signal of 5.79s in length, equal to the mean signal length in the WHAMR test set. This is to be more reflective of a realistic input as the TD-Conformer models contain quadratic time complexities. Model size is measured in parameter count where relevant.

8.3 Results

The results below are separated into three subsection; first an investigation into the impact of kernel size on performance, second an investigation into the impact subsampling layers on performance and third optimizing for computational expenditure and performance compared to a number of similar models.

8.3.1 Varying kernel sizes

In this section, the kernel size P of the D-Conv layer in the Conformer layer is varied for different feature dimensions B to evaluate if there exists a common P value across all models that gives optimal performance. Five P values are evaluated: $\{16, 32, 64, 125, 250\}$. The values 16 and 32 are similar to the kernel sizes in the original Conformer model (Gulati et al., 2020) as well as in (Chen et al., 2021b; Kim & Seo, 2021). The values 64, 125 and 250 were selected to give the D-Conv layer a comparable receptive field to the local Transformer layer of popular DP models such as SepFormer (Subakan et al., 2021) where the chunk size is set to $U = 250$ for the equivalent $J = 16$ model configuration. The results in Figure 8.5 show that for all models the performance in Δ SISDR peaks for a kernel size of $P = 32$. For sampling rate f_s, the relative receptive field of each convolution module in seconds is defined as

$$\mathcal{R}_{\text{conv}}(S, P, J, \mathrm{f}_s) = \frac{1}{\mathrm{f}_s}\left(2^{S-1}JP + \frac{J}{2}\right) \tag{8.4}$$

which for the configuration $\{S, P, J, \mathrm{f}_s\} = \{2, 32, 16, 8000\}$ is 0.129s. This receptive field value is used later in Section 8.3.3 for optimising the model hyperparameters.

[1] GitHub link to SpeechBrain Conformer recipe: `https://github.com/jwr1995/PubSep`.

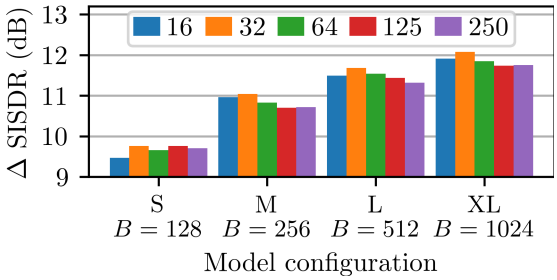


Figure 8.5: *Performance in Δ SISDR for Conformer layer kernel sizes* $P \in \{16, 32, 64, 125, 250\}$ *and different model sizes based on* B.

8.3.2 Varying the number of subsampling layers

Altering the number of subsampling layers changes the temporal resolution of the input encoded features to the Conformer layers in the mask estimation network. It also inversely affects the overall model size, i.e. more subsampling layers result in lower temporal resolution but slightly larger model size. In this second experiment the number of subsampling layers S is varied from 0 to 3. Note that for $S = 0$, a smaller batch size of 2 has to be used due to the increased memory consumption of the Transformer layers. A fixed kernel size of $P = 32$ is used. From Figure 8.6 it is notable that the difference in performance between $S = 0$ and

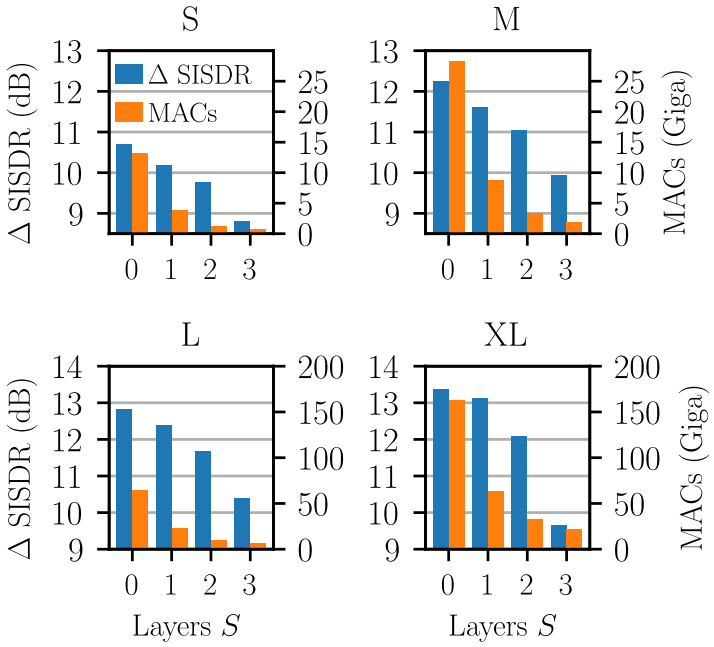


Figure 8.6: *Performance over number of subsampling layers S for all Conformer model sizes (S, M, L & XL) with respective computational cost in MACs exemplary for signal of length* 5.79*s*.

$S = 1$ layers is less than between $S = 1$ and $S = 2$ and likewise then for $S = 2$ and $S = 3$.

It is also noticeable that for the smaller model sizes the reduction in performance for each additional subsampling layer is also smaller. The $S = 0$ configuration gives the best overall performance for both the S and XL TD-Conformer models. This is expected as $S = 0$ gives the MHSA layers in the Conformer the highest temporal resolution when processing the global context. This improvement comes at a significant computational cost however as can be seen from the MACs reported in Figure 8.6 (note the scales vary for each row). This provides some justification for using a small number of subsampling layers at the benefit of significant reductions in computational requirements.

8.3.3 Hyper-parameter optimization and comparison to other models

In the following, TD-Conformer models are compared with several other discriminative supervised speech separation models. To find optimal configurations for the Conformer models, a set of TD-Conformer models with $S = 1$ subsampling layers and kernel sizes $P \in \{64, 125\}$ are trained. Models with no subsampling ($S = 0$) are not evaluated, due to unreasonably long training times on the hardware available for only minor improvements in performance (cf. Section 8.3.2). The kernel size $P = 64$ was specifically chosen as it gives an equal receptive field to the optimal configuration $\{P, S\} = \{32, 2\}$ in Section 8.3.1, cf. (8.4). DM, as implemented in (Zeghidour & Grangier, 2021), is also used to maximise model performance for each of the models. Results are reported with and without DM. MACs are reported for all models for which an open source implementation was available for evaluation (Luo & Mesgarani, 2019; Tzinis et al., 2022b; Ravenscroft et al., 2023a; Subakan et al., 2021). Performance in terms of Δ SISDR for the WSJ0-2Mix anechoic speech mixture dataset (Isik et al., 2016) is shown for completeness.

Table 8.1 shows that all TD-Conformer models outperform the STFT Conformer model proposed in (Chen et al., 2021b). The small TD-Conformer-S models outperform the similarly sized and complex DTCN and SuDoRMRF++ baselines (Ravenscroft et al., 2023a; Tzinis et al., 2022b) with DM on WSJ0-2Mix and show comparable performance on WHAMR without DM. Interestingly, with DM, there was a small performance drop. Later analysis suggested this was likely due to the $\eta = 10^{-5}$ learning rate being too small for the TD-Conformer-S model. The medium TD-Conformer-M models give comparable performance to the similarly sized SkiM baseline (Li et al., 2022a) on the WSJ0-2Mix benchmark but with less than half the number of MACs. The large TD-Conformer-L models give better performance than the SepFormer baseline on the WHAMR benchmark and comparable performance on WSJ0-2Mix with a similar model size but roughly a third of MACs. The TD-Conformer-XL model outperforms the much larger quasi-dual-path network (QDPN) model on WHAMR, with the best model giving 14.6 dB Δ SISDR. The TD-Conformer does not quite reach parity with the most recently SOTA MossFormer-L and TF-GridNet (Zhao & Ma, 2023; Wang et al.,

Table 8.1: *Δ SISDR results for various TD-Conformer models with $S = 1$ compared to other separation models on the WSJ0-2Mix (abbrev. W-2Mix) and WHAMR benchmarks. * indicates results not included in the respective paper cited for a model.*

Model	P	Δ SISDR (dB) WSJ0-2Mix	WHAMR	MACs	Params
Conv-TasNet (Luo & Mesgarani)	-	15.6	9.7*	3.6G	5.1M
STFT-Conformer (Chen et al.)	-	10.8*	6.7*	**1.8G**	57.5M
SuDoRMRF++ +DM (Tzinis et al.)	-	17	-	2.7G	2.7M
WD-TCN (Chapter 4)	-	16.0	10.4	3.7G	3.7M
WD-TCN+DM (Chapter 4)	-	17.2	11.4	3.7G	3.7M
DTCN (Chapter 5)	-	15.6	10.2	3.7G	3.6M
DTCN+DM (Chapter 5)	-	17.2	11.1	3.7G	3.6M
SkiM (Li et al.)	-	18.3	-	19.7G	5.9M
SepFormer (Subakan et al.)	-	20.4	11.5*	60.7G	25.6M
SepFormer+DM (Subakan et al.)	-	22.3	14	60.7G	25.6M
QDPN (Rixen & Renz)	-	22.1	13.1	-	200M
QDPN+DM (Rixen & Renz)	-	**23.6**	14.4	-	200M
MossFormer-L+DM (Zhao & Ma)	-	22.8	16.3	42.8G	42.1M
TF-GridNet (Tab. XIII) (Wang et al.)	-	22.0	-	29.8G	6.8M
TF-GridNet (Wang et al.)	-	23.4	**17.1**	228.2G	14.3M
TD-Conformer-S	64	15.8	10.5	3.7G	**1.8M**
TD-Conformer-S	125	15.9	10.5	3.7G	**1.8M**
TD-Conformer-S+DM	64	17.4	9.7	3.7G	**1.8M**
TD-Conformer-S+DM	125	17.5	9.7	3.7G	**1.8M**
TD-Conformer-M	64	17.7	11.7	8.5G	6.7M
TD-Conformer-M	125	17.8	11.6	8.6G	6.8M
TD-Conformer-M+DM	64	18.1	12.0	8.5G	6.7M
TD-Conformer-M+DM	125	18.8	11.9	8.6G	6.8M
TD-Conformer-L	64	19.5	12.3	21.9G	25.9M
TD-Conformer-L	125	19.7	12.5	22.0G	26.2M
TD-Conformer-L+DM	64	20.3	13.4	21.9G	25.9M
TD-Conformer-L+DM	125	20.2	13.2	22.0G	26.2M
TD-Conformer-XL	64	20.4	13.1	63.6G	102.2M
TD-Conformer-XL	125	20.3	13.0	63.9G	102.7M
TD-Conformer-XL+DM	64	21.1	14.6	63.6G	102.2M
TD-Conformer-XL+DM	125	21.2	14.3	63.9G	102.7M

2023a) models; however, the largest TF-GridNet model has significantly higher computational expenditure, and MossFormer is an augmented version of a Conformer model. Thus the results here help to validate the design choice of this approach.

Table 8.2: *Training time comparison of the TD-Conformer-XL model with* $\{S, P\} = \{1, 63\}$ *with to the SepFormer model. Best results for each metric are indicated in bold.*

Model	Params.	GPU Mem.	TSL	Max. BS	ED	Epochs	Train. Time
SepFormer+DM (Subakan et al.)	**25.8M**	32GB	N/A	1	85 mins	**150**	8.9 days
SepFormer+DM (Table 7.1)	**25.8M**	32GB	4s	2	59 mins	**150**	6.1 days
TD-Conformer-XL+DM	102.7M	32GB	4s	**4**	**39 mins**	200	**5.4 days**

8.3.4 Training Times

The following is supplementary information not included in the original publication relating to this work (Ravenscroft et al., 2023c). In this subsection, a comparison of the TD-Conformer-XL to the SepFormer model is given. The TD-Conformer-XL+DM model with $\{S, P\} = \{1, 64\}$ from Table 8.1 was selected as this gave the best overall performance on WHAMR. The models are all trained on the same GPU as the original SepFormer model with 32GB VRAM available. The models are evaluated in terms of the maximum batch size (BS) it was possible to fit on the GPU without out of memory (OOM) errors, as well as mean epoch duration (ED) and total training time for each model as originally proposed. The results are shown in Table 8.2. Despite the TD-Conformer-XL's much larger model size than SepFormer ($\simeq 4\times$), when using the TSL limit of 4s also used for the SepFormer model in Table 7.1, it was possible to use double the batch size. The consequence of this is a much short mean epoch duration of 39 minutes vs. 59 minutes for the optimised SepFormer training. Furthermore, even with 50 epochs of extra training, the TD-Conformer still took almost a day less to train than the SepFormer model.

8.4 Conclusion

In this paper, a single-channel time-domain Conformer speech separation model was introduced and evaluated. It was shown to reach comparable Δ SISDR performance to SOTA models on the WHAMR and WSJ0-2Mix benchmarks. Using Conformer layers in place of DP Transformer layers was demonstrated to reduce the TC of processing local information whilst increasing the TC for processing global context. A benefit of increased global TC is that it gives the global context layer a higher temporal resolution as demonstrated by varying the number of subsampling layers before the Conformer layers in the proposed network. The proposed TD-Conformer-XL model achieves 14.6 dB Δ SISDR on the WHAMR benchmark. The smallest TD-Conformer-S model outperforms a number of larger and similarly complex models on the WSJ0-2Mix and WHAMR benchmarks. Furthermore, it is demonstrated that despite its larger size, the TD-Conformer-XL model is notably more efficient to train than the SepFormer model while still yielding better performance on the WHAMR evaluation.

Chapter 9

Combining Conformers and Dual-Path Transformers

In Chapter 8, the TD-Conformer was proposed and analysed, particularly with respect to its similarities to the SepFormer, but had interesting trade-offs in both SISDR separation performance and computational expenditure. Most pertinently, the TD-Conformer performed better on noisy reverberant mixtures, and the SepFormer performed better on anechoic clean mixtures. The goal in the following work is thus to analyse these models further by combining them and seeing if it is possible to gain insights about each model and understand if, by combining them, the negative trade-offs between the two network types can be mitigated.

To do this, the ConSepT model is proposed in this chapter. ConSepT is a mixed Conformer and DP Transformer model. The motivation for combining the two variant layers is that, controlling for model complexity, DP Transformer models were shown to be more performant for anechoic speech mixtures (cf. Figure 8.3) and Conformer models have been shown to be more performant on noisy reverberant speech mixtures (cf. Section 8.3.3) (Ravenscroft et al., 2023c; Subakan et al., 2021). This contrast is explored by mixing the two layer types to analyse whether combining them can achieve higher overall performance. The model is structured so that Conformer layers process the earlier features in the network under the assumption that they contain more noise, and thus, the DP Transformer layers are used to process the cleaner features. There are two key contrasts between the two model types; firstly, the Conformer layers result in a larger model size primarily due to the convolutional local context layer in the Conformer block, and secondly, the DP Transformer layers typically have significantly more computational complexity given to processing local context whereas Conformer layers give most of their computational complexity to processing global context as

The contents of this chapter are a revised version of the author's own work found in (Ravenscroft et al., 2024).

they are implemented in this chapter, Chapter 8 and Subakan et al. (2021). The consequences of these characteristics of each layer type is explored with respect to model performance and model generalisation by varying the numbers of each type of layer while keeping the overall number of layers in the network constant. To do this, the generalisation benefits gained from using DM, where new training data is simulated for each epoch, are contrasted for models trained on the simulated WHAMR dataset and evaluated using the real recorded speech mixtures in the MC-WSJ-AV corpus (Lincoln et al., 2005). A preprocessing script for aligning the MC-WSJ-AC recordings is also provided as part of the contribution of this work.

The remainder of this chapter proceeds as follows. Section 9.1 introduces the ConSepT model. In Section 9.2 the training configurations and experimental setup are discussed. Results are given in Section 9.3 and conclusions in Section 9.4.

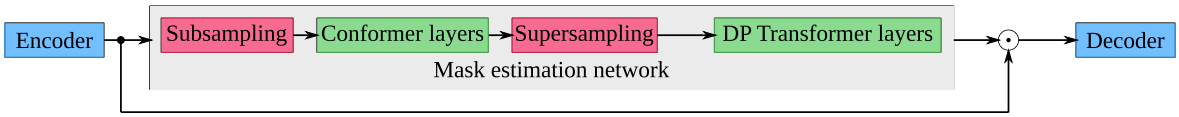


Figure 9.1: *The ConSepT network composed of encoder, mask estimation network and decoder. The $\odot$ symbol denotes the Hadamard product. For more details on the sub-components of the mask estimtation network refer to Figure 2.9, Figure 8.1 and Figure 8.2.*

9.1 ConSepT Separation Network

The proposed ConSepT model is described in the following. The model uses the same general TasNet architecture as the TD-Conformer and SepFormer models, cf. Figure 2.6. The mixture signal $x[i]$ from (2.2) is first chunked into $L_{\mathbf{x}}$ blocks $\mathbf{x}_\ell$ of size J with 50%-overlap. Each block $\mathbf{x}_\ell$ is then encoded into a feature vector $\mathbf{w}_\ell$, which is then passed into a mask estimation network to produce masks $\mathbf{m}_{c,\ell}$ for each speaker. The encoded feature vectors are then masked for each speaker before being decoded back into the time domain.

9.1.1 Encoder & Decoder

The encoder and decoder are the same as that used for the TD-Conformer model, SepFormer model, and most other models discussed thus far in this thesis, excluding Chapter 6. The encoder is composed of a single 1D convolutional layer and ReLU activation function that encodes time-domain blocks of the mixture signal x. the decoder is a transposed 1D convolutional layer that decodes the masked encoded mixture back into the time domain. Readers should refer to Section 2.9.1 for further details.

9.1.2 Mask Estimation Network

The mask estimation network is comprised of two subnetworks processed sequentially. The first is a Conformer network with subsampling layers, based on (Ravenscroft et al., 2023c), and the second is a dual-path Transformer network, based on (Subakan et al., 2021).

The Conformer subnetwork uses subsampling and supersampling layers to reduce the computational complexity of proceeding Transformer layers in Conformer blocks. The subsampling is performed using a projection layer proceeded by a 1D convolutional layer with a kernel size of 4 and a stride of 2, thus reducing the temporal resolution by a factor of 2. The effect of this subsampling on performance is explored in (Ravenscroft et al., 2023c). A set of R_{conf} Conformer layers proceeds the subsampling layer. Each Conformer layer is composed of four modules: a feed forward module with internal feature dimension B, a convolution module with kernel size P and dimension B, a MHSA with PE module of G_{ched} attention heads, and another feed forward module with dimension B. A supersampling layer composed of a transposed 1D convolutional layer that reverses the subsampling layer follows the Conformer layers. A final projection layer in the subnetwork transforms the feature dimension back to N.

The DP Transformer network is composed of a series of alternating local and global Transformer layers with each combined local and global Transformer layer being referred to as a single DP Transformer layer. The output of the supersampling layer of the Conformer subnetwork is first reorganised into overlapping chunks of length U. The chunks are then processed by the local Transformer of dimension Z with G_{intra} attention heads. Following this the axes for the chunk size and the number of chunks are swapped and then the sequence is processed by the global context Transformer of dimension Z with G_{inter} heads. The axes are then swapped back passed through an additional R_{DPT} DP Transformer layers and the entire network is repeated V times.

The final part of the network is a linear layer followed by a ReLU activation function that takes the output of the DP Transformer network to produce a series of masks, $\mathbf{m}_{\ell,c}$.

9.1.3 Objective function

SISDR is used as the objective function for training the networks as it is the same objective used to train the original TD-Conformer and SepFormer models (cf, Ravenscroft et al. (2023c) and Subakan et al. (2021)). A PIT wrapper around the function is used to resolve the speaker permutation problem (Kolbaek et al., 2017). Details on SISDR and PIT can be found in Section 2.6.

9.2 Experimental Setup

9.2.1 Data

The WSJ0-2Mix (Isik et al., 2016) and WHAMR datasets (Maciejewski et al., 2020), as described in Section 2.2.1, are used for training and evaluating models. WSJ0-2Mix is a simulated 2-speaker dataset of anechoic mixtures. WHAMR is a simulated noisy reverberant extension of WSJ0-2Mix. The 8kHz *min* configuration is used for both (cf.Section 2.2.1). Differing from the previous chapters, the far-field MC-WSJ-AV dataset (Lincoln et al., 2005) is also used for evaluating models on out-of-domain unseen data. The *olap* part of this dataset contains real far-field multi-channel recordings of 2-speaker mixtures. The 20k subset of the dataset is used. The 1st channel of array 1 is used as the input mixture, and headset microphones are used as reference signals. Preprocessing steps were performed to make the data suitable for evaluation. First, the audio is downsampled from 16kHz to 8kHz as MC-WSJ-AV was recorded at 16kHz. The headset recordings were both aligned to the array signal using a cross-correlation method for computing time-delays (Azaria & Hertz, 1984). The loudness of the array channel was adjusted to match that of the sum of the headset channels using the *pyloudnorm* toolkit (Steinmetz & Reiss, 2021) to minimize the possibility of signal energy having an impact on the evaluation as the WHAMR mixture loudness is more similar to the targets than the array channels are to the headsets in MC-WSJ-AV. The preprocessing script has been made available on GitHub[1]. The resulting evaluation set comprises 312 real speech mixtures with aligned headset references for each speaker.

9.2.2 Training Configuration

The models use a similar training configuration to the TD-Conformer (Ravenscroft et al., 2023c) with learning rate of 10^{-5} that is fixed for 90 epochs and then reduced if their is no performance improvement after 3 epochs. TSLs are limited to 4s and randomly sampled from the original training example (Ravenscroft et al., 2023b). The feature dimension of the Conformers layers are the same as the TD-Conformer-XL model in Ravenscroft et al. (2023c), i.e. $B = 1024$. The feature dimension of the DP Transformer layers are the same as that in Subakan et al. (2021) $Z = 1024$. For the DP Transformer layers, the number of network cycles $V = 2$. For the Conformer layers, the number of attention heads $G_{\text{ched}} = 4$ as in (Ravenscroft et al., 2023c). For the DP Transformer layers $G_{\text{inter}} = G_{\text{intra}} = 8$ as in (Subakan et al., 2021). The constraint $R_{\text{DPT}} + R_{\text{conf}} = 8$ is used but the specific R values are experimented with in the results section. The value 8 is used as it corresponds to the number of Conformer layers in (Ravenscroft et al., 2023c) and the number of DP Transformer layers in (Subakan et al., 2021).

[1] https://github.com/jwr1995/mc-wsj-aligned

9.2.3 Evaluation Metrics

The main evaluation metric used to assess separation performance is the SISDR improvement over the original mixture signal, denoted Δ SISDR. Improvement in ESTOI, a speech intelligibility metric (Jensen & Taal, 2016), and PESQ, a speech quality metric (Rix et al., 2001b), are also reported for some results. Improvement in SRMR (Falk et al., 2010) is used to assess the residual energy of reverberant effects in the estimated signals. For more information on evaluation metrics, please refer to Section 2.7. The computational complexity of models is assessed using MACs. MACs are computed on a signal length of 5.79s, equal to the mean signal length in the WHAMR and WSJ0-2Mix corpora (Ravenscroft et al., 2023b). Model size is reported in the number of parameters.

9.3 Results

9.3.1 Evaluations on In-Domain Data

The first evaluation looks at the ratio of Conformer layer repeats R_{conf} to DP Transformer repeats R_{DPT} Transformers for the standard configuration with $R_{\text{DPT}} + R_{\text{conf}} = 8$ on the WSJ0-2Mix and WHAMR datasets. R_{conf} is varied from 0 to 8. The results are shown for both with and without using DM in Figure 9.2. For both the WSJ0-2Mix evaluation and the WHAMR evaluation with DM the SISDR performance improves as the number of Conformer layers increases. This corresponds to an increase in number of parameters and a relatively minor decrease in computational complexity. For the WHAMR evaluation without DM, SISDR performance remains fairly consistent for all R_{conf}. This possibly suggests that without DM there is no benefit to having a larger model size as the model is as generalized as is possible without providing the network new training examples. The biggest performance gains with DM are seen on the more challenging WHAMR dataset which demonstrates the benefit of larger model sizes for noisy and reverberant data.

9.3.2 Evaluations on Out-Of-Domain Data

Table 9.1: *Full results for best performing ConSepT model trained on WHAMR using DM in terms of SISDR.*

Eval. set	R_{conf}	R_{DPT}	Params.	PESQ	ESTOI	SRMR	SDR	SISDR	Δ SDR	Δ SISDR
WHAMR	7	1	93.84M	2.29	0.75	9.36	10 dB	8.6 dB	13.6 dB	14.7 dB
WHAMR	8	0	102.30M	2.25	0.75	9.2	10.1 dB	8.6 dB	13.6 dB	14.7 dB
MC-WSJ-AV	7	1	93.84M	2.20	0.53	8.84	5.8 dB	-15.8 dB	8.3 dB	4.3 dB

The models trained on WHAMR are re-evaluated using the out-of-domain MC-WSJ-AV corpus. The results are shown in Figure 9.3. On the MS-WSJ-AV evaluations for models

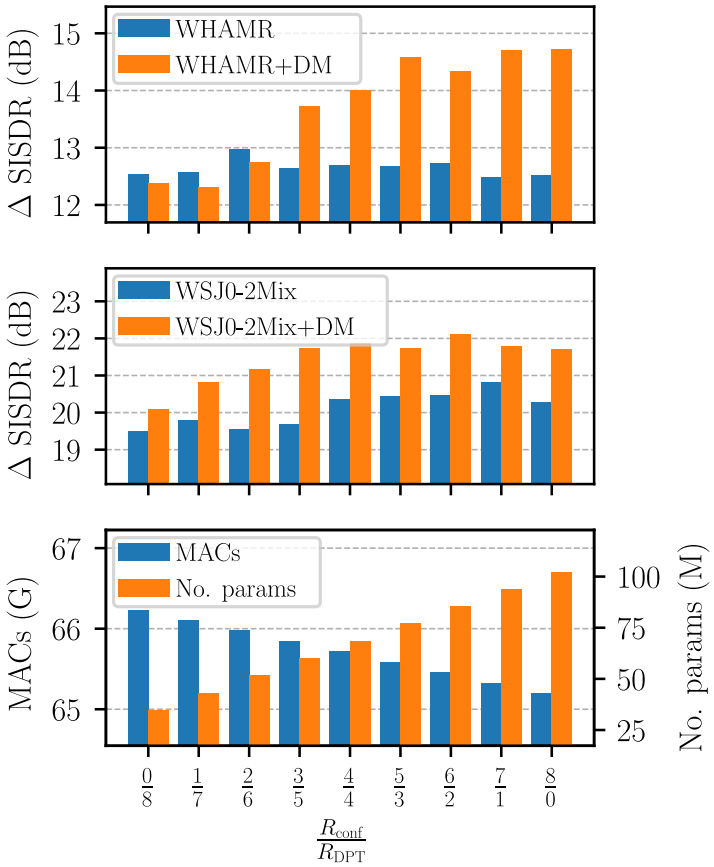


Figure 9.2: *Top and middle: separation performance against model configuration for WHAMR (top) and WSJ0-2Mix (middle). Bottom: corresponding computational complexity (in MACs) and model size for each configuration.*

trained using DM, a similar increasing trend in performance is observed with the increase in model size (i.e. more Conformer layers than DP Transformer layers) for SISDR, PESQ and ESTOI. This implies the models trained using WHAMR and DM are not just generalizing better towards the specific acoustic conditions of the WHAMR noise and reverberation but noise and reverberation in general to some extent. It should be noted the Δ SISDR values between the WHAMR and MC-WSJ-AV evaluations differ by ≈ 10dB, see Table 9.1 for more detailed numbers on the best performing DM models. This is partly explained by the fact that MC-WSJ-AV is both real data and out-of-domain but also it should be noted that the headset references of the MC-WSJ-AV evaluation set are not as "clean" as the WHAMR references, partly due to imperfect alignment but also as there is sometimes some small audio bleed from the other speaker in the room and some minimal noise interference as well. This can be seen in Figure 9.4 where the estimated speech signal $\hat{s}_c$ appears more denoised than the "clean" reference s_c. Interestingly there appears to be no trend in SRMR improvement as the value of R_{conf} increases. The SRMR values are quite high relative to the WHAMR evaluations in

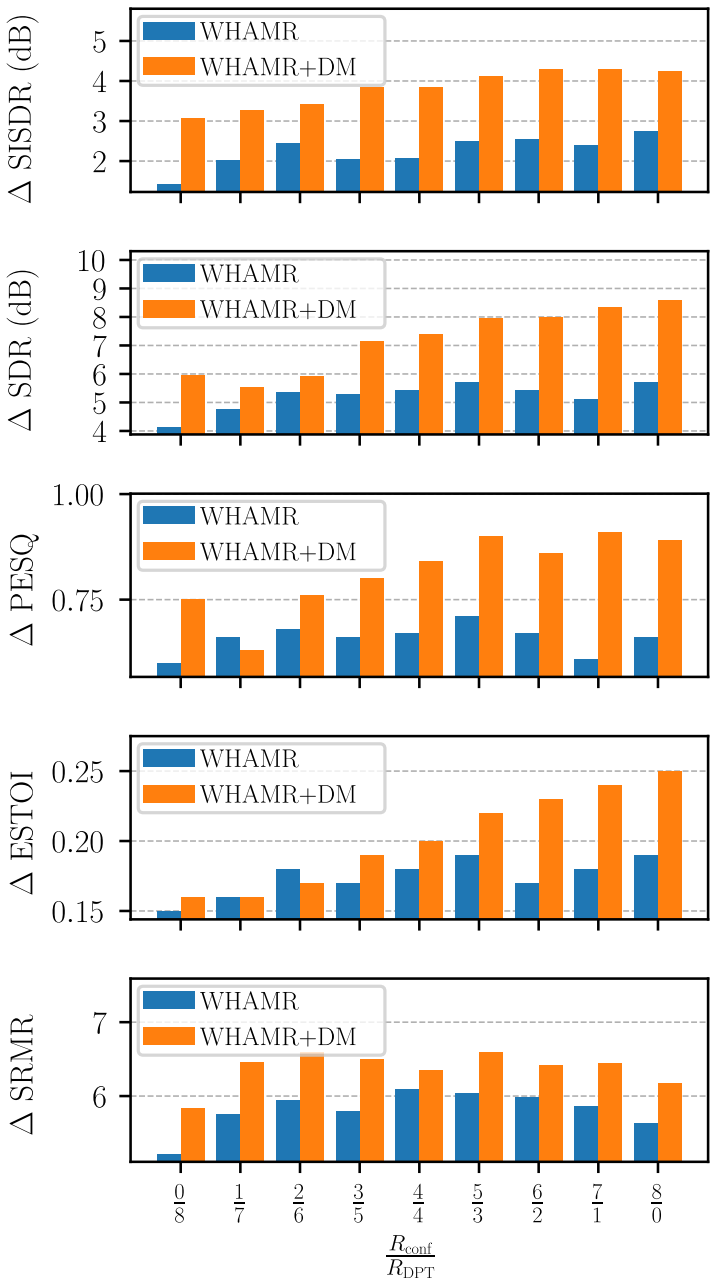


Figure 9.3: *Re-evaluation on MC-WSJ-AV of models trained using WHAMR with and without DM for Δ SISDR, Δ PESQ and Δ ESTOI*

Table 9.1, indicating good dereverberation performance. This was subjectively confirmed by listening through evaluation outputs. All models generally exhibited good dereverberation and noise suppression for WHAMR and MC-WSJ-AV. The estimated speech however, contained obvious distortions and intelligibility was lacking, this is reflected in the various results across all metrics in Table 9.1.

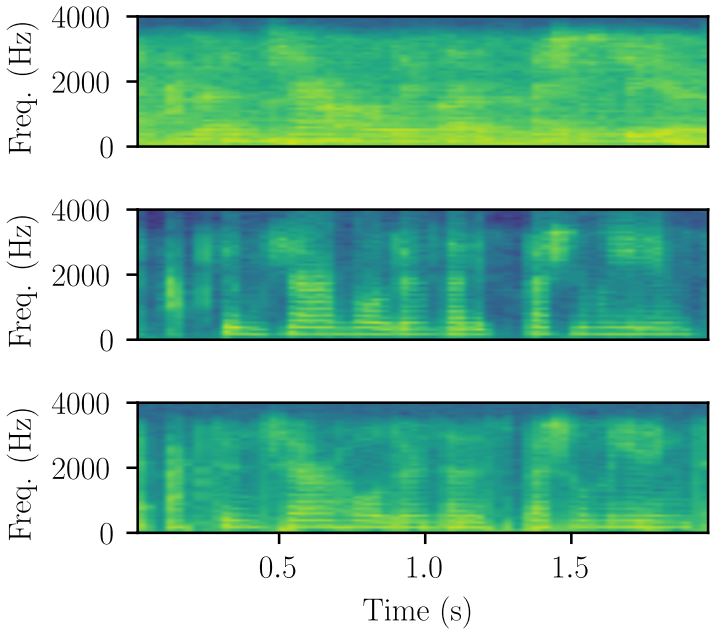


Figure 9.4: *Example spectrograms from the MC-WSJ-AV dataset for $R_{\text{conf}} = 7$ of the far-field mixture x (top), estimated speaker signal $\hat{s}_c$ (middle) and reference "clean" speaker signal s_c (bottom).*

9.4 Conclusions

In this chapter, a novel architecture combining DP Transformer and Conformer layers was proposed for modelling local and global context differently in speech separation networks. It was shown that for the purpose of generalisation in the case of the Conformer layers, having a larger model size was beneficial particularly when DM was being used for training. This generalisation finding was shown to extend to out-of-domain real evaluation data using an aligned version of the MC-WSJ-AV corpus.

Part IV

Applications to Multi-Speaker Speech Recognition

Chapter 10

Transcription-Free Fine-Tuning for ASR

In previous chapters, the focus has been on more theoretical speech separation experiments primarily evaluated in terms of SISDR performance. While SISDR has meaningfulness in terms of evaluating the raw signal quality compared to the reference and it also typically correlates reasonably to objective intrusive perceptual measures such as PESQ (Rix et al., 2001b) and ESTOI (Jensen & Taal, 2016), it is non-optimal when fine-tuning separation for specific tasks such as multi-speaker ASR (Shi et al., 2022; Settle et al., 2018).

In this chapter, the focus is on applying speech separation technology to a realistic use case, namely multi-speaker ASR. As will be shown in this chapter, while speech separation technology can improve the separation of speakers while reducing background noise and reverberation, this doesn't necessarily translate into improved ASR performance often due to model artefacts and distortions introduced in the separation process. To address these issues, a *transcription-free* method is proposed in this chapter to fine-tune speech separation models for improved ASR WER performance.

In (Close et al., 2023, 2024) it has been demonstrated that computing embedding differences between self-supervised speech representations (SSSRs) can be used as loss terms for training speech enhancement networks. Furthermore, in (Bagchi et al., 2018) it was shown that computing the difference between *senone* representations can be used to adapt speech enhancement networks for robust ASR. A similar concept is applied in this chapter for fine-tuning speech separation models for the purpose of multi-speaker ASR. Embeddings from different ASR models, namely a Wav2Vec2 CTC model (Baevski et al., 2020) and a Whisper model (Radford et al., 2023), are used to compute differences between reference speech and separated speech. This is formulated as a loss function such that the averaged difference between the embeddings can be used in back-propagation for fine-tuning the network.

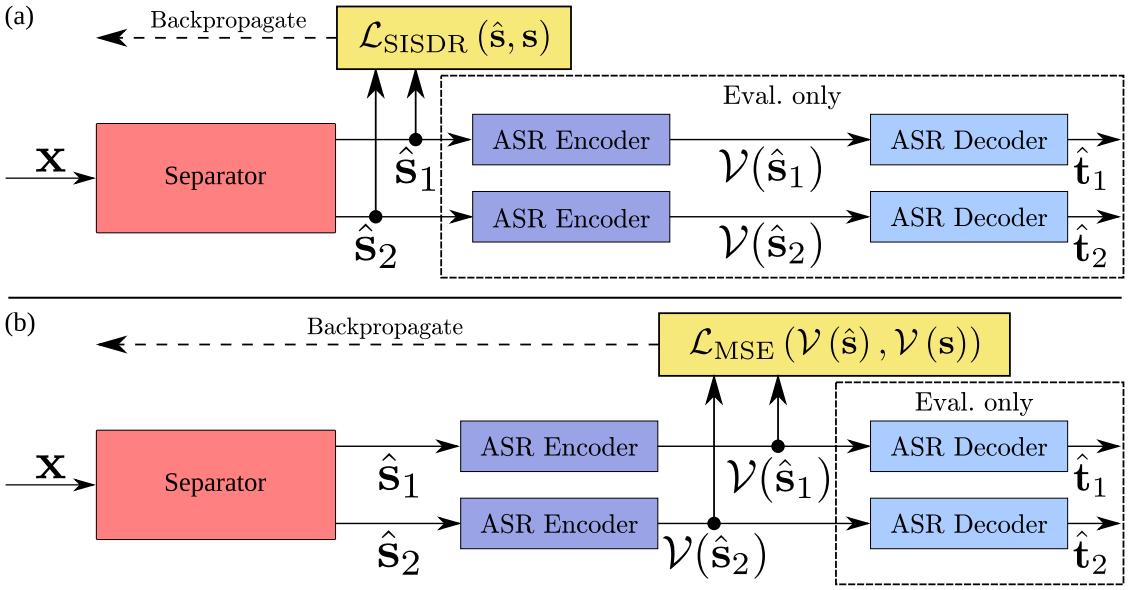


Figure 10.1: *(a) Baseline modular multi-speaker ASR pipeline using a DNN separation network. (b) The proposed transcription-free loss function for fine-tuning DNN-based speech separators using the MSE of ASR encoder outputs. All ASR encoders and decoders are the same. Solid lines indicate information flow. Dashed lines indicate the propagation of gradients.*

10.1 Modular Multi-Speaker Speech Recognition

In this section, a modular (or *separate and recognize*) approach to Multi-Speaker ASR is described, along with a baseline implementation using a TasNet separator and CTC recognizer. The term *modular* refers to the fact that the multi-speaker ASR pipeline is composed of more than one network, cf. Figure 2.10. This differs from non-modular approaches such as SOT where there is only one network that is trained to perform multi-speaker ASR explicitly.

A schematic for the baseline modular approach can be seen in Figure 10.1 (a). The schematic is very similar to Figure 2.6 where the separator is trained using an SISDR loss function, however, each output from the separator is fed separately to an ASR model which produces C predicted transcripts, one for each speaker in the mixture.

In the baseline pipeline used in this section, the TD-Conformer-S model is used for separation with no alterations to the model itself. The ASR model used for evaluation is the base version of Wav2Vec2 model available publicly via the torchaudio library[1].

[1]Link to Wav2Vec2 Base model from torchaudio: `https://pytorch.org/audio/0.10.0/pipelines.html#torchaudio.pipelines.WAV2VEC2_BASE`

10.2 Transcription-Free Fine-Tuning Method

This section describes the proposed transcription-free method for fine-tuning a separation network. The method is visualised in Figure 10.1 (b). The first stage of the approach is to pre-train the separation network using a standard SISDR loss function and a PIT wrapper, cf. Section 2.6. This chapter uses real far-field single-speaker utterances to generate artificial mixtures for pre-training the separation network. The proposed method then uses feature representations from a pre-trained ASR model as opposed to reference grapheme or phoneme (or similar) representations in an attempt to fine-tune a separator to give better performance in terms of WER for each speaker.

10.2.1 Pre-Training on Imperfect Targets

In the first stage, the separation network is pre-trained using an SISDR function on artificial mixtures of real single-speaker utterances. This follows the standard method described in Chapter 8. For each training example, a new speech mixture is created, and then the separator attempts to separate the time-domain mixture into two time-domain speech signals. Finally, the SISDR losses are computed between the estimated signals and the reference signals with a PIT wrapper and then averaged before being backpropagated.

10.2.2 Logit Difference (LD) Loss Function

This section describes the logit difference (LD) loss function used for fine-tuning the speech separation model. The phrase *logit difference* is used to refer to this loss as it computes the difference between the log probabilities (*logits*) of a CTC ASR encoder. Hypothetically, any CTC ASR model could be used in a similar fashion but in this chapter, the Wav2Vec2 base model (as used in the baseline cf. Section 10.1) fine-tuned on LibriSpeech (Panayotov et al., 2015) data is selected. This version was chosen due to computational constraints, but larger models trained on different corpora may yield better results. The CTC Wav2Vec2 Encoder is defined as a function

$$\mathcal{V}_{\text{CTC}} : \mathbb{R}^{L_x} \mapsto \mathbb{R}^{L_{\text{CTC}} \times N_{\text{CTC}}} \tag{10.1}$$

where L_{CTC} is the output sequence length and N_{CTC} is the number of possible labels, i.e. the number of characters that can be interpreted by a decoder or decoding function (minimum of 26 plus word boundary token for the Latin alphabet). In the Wav2Vec2 base model used in the following experiments, the number of labels is $N_{\text{CTC}} = 30$ (29 characters that are interpreted verbatim plus a word boundary token"|").

As previously alluded to, the output layer of the model of dimension N_{CTC} represents log probabilities (i.e. $\log(\mathcal{P}(\cdot))$) for each of the labels referred to as logits. The LD loss function computes the difference between the output logits of the inputs $\hat{s}_c$ and s_c $\forall c \in \{1, \ldots, C\}$.

The loss function is the MSE of the two logit embedding sequences defined as

$$\mathcal{L}_{\text{LD}}\left(\mathbf{s}_c, \hat{\mathbf{s}}_c\right) = \frac{1}{L_{\text{CTC}} N_{\text{CTC}}} \sum_{\ell=1}^{L_{\text{CTC}}} \sum_{w=1}^{N_{\text{CTC}}} \left(\mathcal{V}_{\text{CTC}}(\hat{\mathbf{s}}_c)_{\ell,n} - \mathcal{V}_{\text{CTC}}(\mathbf{s}_c)_{\ell,n}\right)^2 \tag{10.2}$$

10.2.3 Whisper Encoder Loss Function

The Whisper encoder (WE) loss function is proposed as an alternate to the LD loss function. The Whisper model (Radford et al., 2023) was proposed primarily for robust speech recognition. The model relies on large-scale weak supervision to train an encoder-decoder Transformer model. Large-scale weak supervision involves leveraging large amounts of training data with poor-quality transcriptions. This contrasts the Wav2Vec2 CTC model in Section 10.2.2. The Whisper Encoder is defined as a function

$$\mathcal{V}_{\text{WE}} : \mathbb{R}^{L_x} \mapsto \mathbb{R}^{L_{\text{WE}} \times N_{\text{WE}}} \tag{10.3}$$

where L_{WE} is the output sequence length and N_{WE} is the number of output features ($N_{\text{WE}} = 512$ for the Whisper Tiny model used later (Radford et al., 2023)).

The WE loss function computes the difference between the output logits of the inputs $\hat{s}_c$ and s_c $\forall c \in \{1, \ldots, C\}$. The loss function is the MSE of the two Whisper encoder output feature sequences defined as

$$\mathcal{L}_{\text{WE}}\left(\mathbf{s}_c, \hat{\mathbf{s}}_c\right) = \frac{1}{L_{\text{WE}} N_{\text{WE}}} \sum_{\ell=1}^{L_{\text{WE}}} \sum_{n=1}^{N_{\text{WE}}} \left(\mathcal{V}_{\text{WE}}(\hat{\mathbf{s}}_c)_{\ell,n} - \mathcal{V}_{\text{WE}}(\mathbf{s}_c)_{\ell,n}\right)^2 \tag{10.4}$$

10.2.4 Guided Permutation Invariant Training (GPIT)

Empirically, it was found that using a standard PIT wrapper around the LD loss function resulted in model divergence. It was discovered that the permutation-solving algorithm was notably less robust to noise when trying to minimise the LD loss than the SISDR loss. This is likely due to the more fine-grained information in the feature space of the ASR encoder. As such, in this section, a new approach is proposed, which uses an alternate criterion to the actual loss function to guide the permutation solving. The actual loss function is then computed once the minimum permutation has been "solved".

10.3 Experimental Setup

10.3.1 Data and Training Configuration

A corpus of doctor-patient conversations (referred to as DoPaCos) was used for the following experiments[2]. The data mostly consists of conversations recorded in a far-field setting using a mobile phone microphone. The data also includes a small amount of reasonably clear dictation recordings.

For pre-training and fine-tuning, $20,000$ speaker utterance pairs were generated. A *mix-on-the-fly* approach is used for simulating speech mixtures. A TSL scheduler varied the TSLs linearly from 3s to 7s in the pre-training stage and 5 to 7 in the fine-tuning stage. Following from the findings in Chapter 7, the idea behind using a TSL scheduler was to maximise the number of unique examples the network observes but also vary the the amount of context the ASR encoder is able to observe. This was to deal with the fact that it was unknown how the amount of context affects the performance of the proposed losses. An overlap scheduler was also introduced to linearly vary the minimum overlap of speaker 2 to speaker 1 from 90% to 10% for the pre-training stage. Neither of these techniques is validated in this chapter but remains consistent across all experiments, so they are unlikely to impact the overall findings. For pre-training, the TD-Conformer-S model (cf. Chapter 8) is trained using a SISDR loss function over 100 epochs with a learning rate of $\eta = 2 \times 10^{-4}$ that is fixed for 70 epochs and then halved if there is no improvement after 3 epochs. In the fine-tuning stage the pre-trained TD-Conformer-S is trained using the proposed ASR encoder loss functions for an additional 30 epochs with a learning rate of 2×10^{-7}.

10.3.2 Evaluation Measures

The prior evaluation metrics, such as SISDR and PESQ, are less relevant as they do not tell us enough about ASR performance. The most widely used measure for ASR performance is WER. This is a fairly trivial measure for single-input single-output (SISO) ASR, but is non-trivial for multiple-input multiple-output (MIMO) pipelines, as in the case of Multi-Speaker ASR. A number of options are available for multi-speaker WER evaluation (von Neumann et al., 2023b). In this chapter, two such definitions are chosen: concatenated minimum-power word error rate (CP-WER) (Watanabe et al., 2020) and optimal reference combination word error rate (ORC-WER) (Sklyar et al., 2022). The key difference between the two measures is that CP-WER penalizes output speaker channel switches and ORC-WER is unconcerned with whether a given speaker is output on a single channel or multiple channels, so long as the ASR model is still able to estimate the word accurately. CP-WER is the more important

[2]Please note, this corpus is not publicly available and was only made available to the author via an internship with 3M UK & 3M Health Information System Inc.

measure as, ideally, in the proposed system, the goal is to have one speaker for each output channel. However, the addition of ORC-WER provides additional insight into the overall intelligibility of the speech regardless of which channel(s) a given speaker gets output to.

10.4 Results

The results are shown comparing the LD loss with and without SISDR regularization to the WE Loss with regularization in Table 10.1. In addition the WER measures for the oracles, i.e. the output of the ASR model with the clean references $s_c[i]$ as input, and for the original speech mixtures $x[i]$. The third row also shows the result of the TD-Conformer-S after pre-training. These results show that the pre-trained separation network, while it improves CP-WER, actually results in a degradation in performance in ORC-WER compared to the original mixture. The LD loss on its own gives the best results overall in both CP-WER and ORC-WER. The inclusion of SISDR as a regularization term results in worse performance interestingly. Similarly, the WE loss also leads to even worse performance in both WER measures. One should note, however, that these WER gains are very minor though they do indicated the methods works. Further work will explore training on cleaner speech references compared to those in the DoPaCos corpus, assuming that further gains can be made with less distorted ASR encoder representations.

Table 10.1: *Comparisons of LD Loss, WE Loss and baseline performance. The LD and WE loss terms are computed using GPIT with SISDR as the guiding function. The SISDR loss terms are computed using the standard PIT methodology.*

Configuration	Signal	Fine-tune	CP-WER	Δ CP-WER	ORC-WER	Δ ORC-WER
Oracles	$s_c[i]$	-	53.70%	-	53.60%	-
Simulated Mix	$x[i]$	-	80.20%	-	66.10%	-
$\mathcal{L}_{\text{SISDR}}$	$\hat{s}_c[i]$	-	67.40%	-12.8%	66.50%	0.4%
$\mathcal{L}_{\text{LD}}$	$\hat{s}_c[i]$	30 epochs	**65.90%**	**-14.3%**	**65.20%**	**-0.9%**
$\mathcal{L}_{\text{LD}} + \mathcal{L}_{\text{SISDR}}$	$\hat{s}_c[i]$	30 epochs	66.30%	-13.9%	65.70%	-0.4%
$\mathcal{L}_{\text{WE}} + \mathcal{L}_{\text{SISDR}}$	$\hat{s}_c[i]$	30 epochs	66.60%	-13.6%	65.80%	-0.3%

10.5 Conclusion

In this section, the TD-Conformer model proposed in Chapter 8 was applied to multi-speaker ASR. It was shown that in a standard configuration, the TD-Conformer leads to improved CP-WER but a loss in ORC-WER. A transcription-free method for fine-tuning the speech separation model was proposed using ASR encoder representations from the Whisper and Wav2Vec2 models. The Wav2Vec2 model, in particular, led to notable reductions in both

ORC-WER and CP-WER. This Whisper model also led to some improvement but to a lesser extent.

Part V

Conclusions and Future Research Directions

Chapter 11

Discussion and Future Research

A number of different areas have been researched throughout this thesis. While most chapters reached a reasonable point of conclusion, there is still further work that can be done for all to add greater insights and help move speech separation research forward in the right direction.

In Chapter 3 progress was made in understanding how receptive fields of fully convolutional networks impact performance for speech dereverberation tasks. This study could first be expanded to other dimensions of reverberation such as the spectral qualities and reverberation energy. Furthermore, although reverberation was treated as a correlated noise source in this chapter, the study could be repeated for uncorrelated additive noise by replacing reverberation time or T60 with measures of noise such as SNR. These studies could inform the initialisation of TCNs but also the later proposed WD-TCN and DTCN models. Furthermore, as the DTCN and WD-TCN models, though lightweight, do not give SOTA in the proposed configurations compared to more modern models (Wang et al., 2023b; Rixen & Renz, 2022; Zhao & Ma, 2023), research ought to be done to investigate if they can be incorporated into SOTA such as the QDPN model (Rixen & Renz, 2022) which already uses a TCN for modelling local context or the Mossformer model (Zhao & Ma, 2023) which uses gated convolutions to model local context. The combination of the weight-dilation method in Chapter 4 and deformable depthwise layers in Chapter 5 may also yield meaningful performance improvements worth investigating. Additionally combining this with the gating mechanisms in the Mossformer model Zhao & Ma (2023) may yet lead to further improvements. The DTCN model proposed in Chapter 5, while theoretically computationally inexpensive, led to OOM errors with a batch size greater than 2 on a 32 GB GPU. Compare this to the TCN, which it was possible to train on the same GPU with a batch size as large as 8 and it highlights a current flaw that requires further investigation. Lots of effort was put into optimising the deformable algorithm used in the implementation, e.g. using the floor functions in Eq. (5.4), and further improvements can still be made. The main issues are likely due to a twofold issue in the implementation being used: the first is it was written entirely in Python and the second is the irregular memory

addressing often found in deformable convolution as highlighted in Li et al. (2022b); Chu et al. (2023). In the later stages of this project, it was found that a more recent deformable toolkit *tvdcn*[1] enabled a further reduction in memory usage and consequently a batch size as large as 4. One of the benefits of this toolkit is the fact that the kernels are written in low-level CUDA C++ code. This toolkit however came with a reduction in performance, the cause of which is as yet unconfirmed but likely due to the fact that these kernels do not constrain the deformable convolution to a maximum receptive field of the offset estimation sub-network as is the case in Section 5.1.2.1. There has also been research focussed on improving the efficiency of deformable convolution, particularly with a focus on 2D convolutions within the computer vision (CV) community. However, open-source implementations of these for 1D deformable convolution, are extremely uncommon which limits the ease of research in this direction.

The work done on improving the signal encoder in Chapter 6 would benefit from being evaluated with different mask estimation network models. In this work, we focus primarily on the fully convolutional TCN mask estimation network in the evaluations. It would be interesting to gain further insights on whether Transformer-based mask estimation networks might mitigate some of the performance improvements observed with the SA encoder. Furthermore, while we presented a novel way of analysing these representations visually to provide a more "spectrogram-like" visualisation, there has been more in-depth work in speech processing research on what these time-domain encoded representations actually show. For example, in Pasad et al. (2023), it has been possible to show that the convolutional encoder layers in the SSSR models typically learn a representation that correlates quite highly to mel filterbank features by using singular vector canonical correlation analysis (SVCCA) (Raghu et al., 2017) on the encoded features.

In Chapter 8 and Chapter 9, the strengths of Conformer models over the previously popular DP Transformer approaches were demonstrated. This work, combined with other more recent work such as the gated convolutions in Zhao & Ma (2023) and TCN modules in Sinha et al. (2022); Rixen & Renz (2022) used for processing local context, suggests these computationally lighter-weight approaches (although often less parameter-efficient) result in better separation performances for noisy signals with lower memory consumption than the constrained-view Transformer layers of the DP Transformer. To take this study further, it would be useful to contrast the use of various types of layers used for processing local context, including vanilla convolutional layers, gated convolutions, TCNs as well as the proposed WD-TCNs and DTCNs and possibly even RNN layer variants.

The work done in Chapter 10 showed that it was possible to fine-tune separation networks for multi-speaker ASR tasks without requiring access to transcriptions. While the method was shown to lead to improved WER, there is a lot more work that can be done in this

[1]tvdcn on GitHub: `https://github.com/inspiros/tvdcn`

direction. First, it would be useful to repeat the evaluations on an alternate dataset with comparisons to the fine-tuning done with transcription-based loss functions, e.g. the CTC and PIT method used in Chang et al. (2019). In addition, due to time and resource constraints, the method was only evaluated using relatively small models, TD-Conformer-S, Whisper Tiny and Wav2Vec2 Base. In particular, using a larger more powerful ASR encoder, such as Wav2Vec2 Large, for fine-tuning, the model would be able to make more powerful inferences, leading to better training and consequently reduced WER. The models were only evaluated using the Wav2Vec2 base model and the best results were found on the model fine-tuned and evaluated using the Wav2Vec2 base encoder. It is quite possible that the WE Loss might outperform the LD Loss if one were to evaluate using the corresponding Whisper model.

The overall scope of speech separation research as it stands has depended heavily on supervised methods. The drawback of this approach is the dependence on simulated speech mixtures due to the lack of reference audio, particularly with respect to the noisy reverberant scenario heavily investigated in this thesis. The main issue with this is the domain mismatch that usually occurs between simulated mixtures and real mixtures. A greater emphasis on unsupervised methods Wang & Watanabe (2023) and domain adaptation techniques Leglaive et al. (2023) is the most obvious path for speech separation researchers to overcome this hurdle. For task-specific separation models, such as those used in Chapter 10 for multi-speaker ASR, knowledge transfer from the respective domain will likely yield improved results as some of the work in Chapter 10 suggests. Again the purely supervised methods are limited by a reliance on high-quality reference audio and objective functions that may be suboptimal for a given task and thus greater emphasis should be put on how the models specifically use data to optimise for a given downstream task.

Chapter 12

Concluding Remarks

In this thesis, a number of advances were made with respect to DNN techniques for speech separation and enhancement.

12.1 Conclusions from Part II

A novel investigation was done on various TCN configurations for speech derverberation. Evidence was found that the optimal receptive field for TCN configuration has a relationship to the reverberation time of the data, i.e. a larger receptive field is more suitable for data with larger RT60s. With this knowledge two alternate TCN inspired networks were proposed, the WD-TCN and the DTCN. The WD-TCN used a multi-dilation depthwise separable layer with SE attention to allow the network to adapt to different reverberation conditions in the data by applying greater attention in the receptive field to more or less highly localised context. This approach was shown to give consistent performance improvements across all evaluated dereverberation configurations. It was then evaluated for the combined speech enhancement and separation task. Again, this model showed notable performance improvements over the standard TCN model with a negligible increase in model size. The DTCN model employed a method known as deformable depthwise convolution Dai et al. (2017) to replace the depthwise convolutions in the standard TCN model. Deformable convolution allows the network to have a completely dynamic receptive field. The DTCN again showed consistent performance improvements over the TCN model across various model configurations, but in particular, for noisy and reverberant data, validating the original motivation for proposing the technique.

12.2 Conclusions from Part III

The third part of this thesis began by focussing more on the encoder representations of TasNet models. In Chapter 6 it was shown that MHSA (Lin et al., 2017; Vaswani et al., 2017) is

particularly useful for improving the encoded time-domain representations. The self-attention mechanism was analogised to a depthwise cross-correlation operation, i.e. similar features are emphasised more so than uncorrelated features, e.g. additive noise, resulting in a "cleaner" representation of the speech. MHA decoder models were also proposed but these results in less notable performance improvements. In Chapter 7 a novel study was performed into the effect of TSL limits on model performance for the SepFormer and Conv-TasNet models across the WSJ0-2Mix, WHAMR and Libri2mix benchmark datasets. It was found that for specific datasets, WSJ0-2Mix and WHAMR, better performance could actually be achieved on shorter TSLs (3-4s) when random sampling was used. It was also shown that with more informed knowledge about TSLs it was possible to reduce the training time of the SepFormer model in Subakan et al. (2021) by 44% without *any* reduction in separation performance. Following this study, a novel conformed-based model was proposed as an alternative to the SepFormer model. It was shown that Conformers have a reduced computational cost on shorter signal lengths than the dual-path Transformer technique used in SepFormer (Subakan et al., 2021). In Chapter 9 it was further shown that improvements in the Conformer model on noisy reverberant data were likely due to the increased model size enabling better generalization qualities of the model.

12.3 Conclusions from Part IV

The application of speech separation models to multi-speaker ASR was explored in Chapter 10. A novel *transcription-free* method for fine-tuning speech separators for downstream ASR tasks was proposed. This method computed the embedding difference between ASR encoder representations of clean and separated speech as a loss function, which could then be backpropagated through the network. Two ASR encoders, the Wav2Vec2 Base model and the Whisper Tiny model, were analysed. Both resulted in a reduction of ORC-WER and CP-WER with the WAv2Vec2 model yielding the best results in the experiments conducted.

Bibliography

Abdali, S. & NaserSharif, B. (2017). Non-negative matrix factorization for speech/music separation using source dependent decomposition rank, temporal continuity term and filtering. *Biomedical Signal Processing and Control*, 36, 168–175. https://doi.org/https://doi.org/10.1016/j.bspc.2017.03.010 (Cited on p. 18)

Azaria, M. & Hertz, D. (1984). Time delay estimation by generalized cross correlation methods. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2), 280–285. https://doi.org/10.1109/TASSP.1984.1164314 (Cited on p. 135)

Ba, L. J., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. arXiv:2106.04624. (Cited on pp. 27 and 122)

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, volume 33, 12449–12460. https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf (Cited on p. 141)

Bagchi, D., Plantinga, P., Stiff, A., & Fosler-Lussier, E. (2018). Spectral feature mapping with mimic loss for robust speech recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5609–5613. https://doi.org/10.1109/ICASSP.2018.8462622 (Cited on p. 141)

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.0473 (Cited on pp. 35 and 95)

Barker, J., Marxer, R., Vincent, E., & Watanabe, S. (2015). The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 504–511. https://doi.org/10.1109/ASRU.2015.7404837 (Cited on pp. 13 and 14)

Barker, J., Watanabe, S., Vincent, E., & Trmal, J. (2018). The fifth 'CHiME' Speech Separation and Recognition Challenge: Dataset, task and baselines. *Proc. Interspeech*

2018 - 19th Annual Conference of the International Speech Communication Association. https://hal.inria.fr/hal-01744021 (Cited on pp. 13 and 14)

Barry, D., Fitzgerald, D., Coyle, E., & Lawlor, B. (2005). Single channel source separation using short-time independent component analysis. *Audio Engineering Society Convention*. https://mural.maynoothuniversity.ie/723/ (Cited on p. 18)

Bell, A. J. & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129–1159. https://doi.org/10.1162/neco.1995.7.6.1129 (Cited on p. 8)

Benesty, J. (2000). *An Introduction to Blind Source Separation of Speech Signals*, 321–330. Kluwer Academic Publishers. https://doi.org/10.1109/SLT48900.2021.9383580 (Cited on pp. 8, 17, and 18)

Bhagya, D. & Suchetha, M. (2021). A 1-D Deformable Convolutional Neural Network for the Quantitative Analysis of Capnographic Sensor. *IEEE Sensors Journal*, 21(5), 6672–6678. https://doi.org/10.1109/JSEN.2020.3042989 (Cited on pp. 83 and 84)

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2017). Quasi-recurrent neural networks. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=H1zJ-v5xl (Cited on p. 34)

Bradley, J. & Sato, H. (2002). The importance of early reflections for speech intelligibility in rooms. *The Journal of the Acoustical Society of America*, 111, 2411–2411. https://doi.org/10.1121/1.4778223 (Cited on p. 12)

Bradley, J. S., Sato, H., & Picard, M. (2003). On the importance of early reflections for speech in rooms. *The Journal of the Acoustical Society of America*, 113(6), 3233–3244. https://doi.org/10.1121/1.1570439 (Cited on p. 12)

Braun, S. & Tashev, I. (2021). A consolidated view of loss functions for supervised deep learning-based speech enhancement. *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 72–76. https://doi.org/10.1109/TSP52935.2021.9522648 (Cited on p. 39)

Burel, G. & Rondel, N. (1993). Neural networks for array processing: from DOA estimation to blind separation of sources. *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC*, volume 2, 601–606 vol.2. https://doi.org/10.1109/ICSMC.1993.384940 (Cited on p. 8)

Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., & Wellner, P. (2006). The ami meeting corpus: A pre-announcement. *Machine Learning for Multimodal Interaction*, 28–39. (Cited on p. 13)

Cauchi, B., Gerkmann, T., Doclo, S., Naylor, P., & Goetze, S. (2016). Spectrally and spatially informed noise suppression using beamforming and convolutive NMF. *Proc. AES 60th Conference on Dereverberation and Reverberation of Audio, Music, and Speech.* (Cited on p. 18)

Cauchi, B., Kodrasi, I., Rehr, R., Gerlach, S., Jukic, A., Gerkmann, T., Doclo, S., & Goetze, S. (2015). Combination of MVDR beamforming and single-channel spectral processing for enhancing noisy and reverberant speech. *EURASIP Journal on Advances in Signal Processing*, 2015(1). https://doi.org/10.1186/s13634-015-0242-x (Cited on p. 60)

Chang, X., Zhang, W., Qian, Y., Roux, J. L., & Watanabe, S. (2019). MIMO-Speech: End-to-End Multi-Channel Multi-Speaker Speech Recognition. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 237–244. https://doi.org/10.1109/ASRU46091.2019.9003986 (Cited on pp. 2, 13, 59, and 151)

Chang, X., Zhang, W., Qian, Y., Roux, J. L., & Watanabe, S. (2020). End-to-end multi-speaker speech recognition with transformer. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6134–6138. https://doi.org/10.1109/ICASSP40776.2020.9054029 (Cited on p. 60)

Chen, J., Mao, Q., & Liu, D. (2020a). Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation. *Proc. Interspeech 2020*. https://doi.org/10.21437/Interspeech.2020-2205 (Cited on pp. 9, 40, 54, 61, 93, 121, and 124)

Chen, L.-Z., Lin, Z., Wang, Z., Yang, Y.-L., & Cheng, M.-M. (2021a). Spatial information guided convolution for real-time rgbd semantic segmentation. *IEEE Transactions on Image Processing*, 30, 2313–2324. https://doi.org/10.1109/TIP.2021.3049332 (Cited on p. 28)

Chen, S., Wu, Y., Chen, Z., Wu, J., Li, J., Yoshioka, T., Wang, C., Liu, S., & Zhou, M. (2021b). Continuous speech separation with conformer. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5749–5753. https://doi.org/10.1109/ICASSP39728.2021.9413423 (Cited on pp. 120, 121, 127, 129, and 130)

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020b). Dynamic convolution: Attention over convolution kernels. *2020 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR), 11027–11036. https://doi.org/10.1109/CVPR42600.2020.01104 (Cited on pp. 33, 73, 75, and 76)

Chen, Z., Luo, Y., & Mesgarani, N. (2017). Deep attractor network for single-microphone speaker separation. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 246–250. https://doi.org/10.1109/ICASSP.2017.7952155 (Cited on p. 22)

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179 (Cited on p. 34)

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195 (Cited on pp. 31 and 83)

Chu, C., Liu, C., Xu, D., Wang, Y., Luo, T., Li, H., & Li, X. (2023). Accelerating deformable convolution networks with dynamic and irregular memory accesses. *ACM Trans. Des. Autom. Electron. Syst.*, 28(4). https://doi.org/10.1145/3597431 (Cited on p. 150)

Close, G., Hollands, S., Goetze, S., & Hain, T. (2022). Non-intrusive Speech Intelligibility Metric Prediction for Hearing Impaired Individuals. *Proc. Interspeech 2022*, 3483–3487. https://doi.org/10.21437/Interspeech.2022-10182 (Cited on p. 41)

Close, G., Ravenscroft, W., Hain, T., & Goetze, S. (2023). Perceive and predict: Self-supervised speech representation based loss functions for speech enhancement. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10095666 (Cited on pp. 39, 41, and 141)

Close, G., Ravenscroft, W., Hain, T., & Goetze, S. (2024). Multi-cmgan+/+: Leveraging multi-objective speech quality metric prediction for speech enhancement. *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Accepted. (Cited on pp. 39 and 141)

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, 36(3), 287–314. https://doi.org/https://doi.org/10.1016/0165-1684(94)90029-9. Higher Order Statistics (Cited on p. 18)

Cord-Landwehr, T., Boeddeker, C., Von Neumann, T., Zorilă, C., Doddipatla, R., & Haeb-Umbach, R. (2022). Monaural source separation: From anechoic to reverberant environments.

2022 International Workshop on Acoustic Signal Enhancement (IWAENC), 1–5. https://doi.org/10.1109/IWAENC53105.2022.9914794 (Cited on pp. 2, 8, 9, 10, 12, 41, 59, 61, 120, and 124)

Cornell, S., Wiesner, M. S., Watanabe, S., Raj, D., Chang, X., Garcia, P., Masuyam, Y., Wang, Z.-Q., Squartini, S., & Khudanpur, S. (2023). The CHiME-7 DASR Challenge: Distant Meeting Transcription with Multiple Devices in Diverse Scenarios. *Proc. 7th International Workshop on Speech Processing in Everyday Environments (CHiME 2023)*, 1–6. https://doi.org/10.21437/CHiME.2023-1 (Cited on pp. 14 and 62)

Cosentino, J., Pariente, M., Cornell, S., Deleforge, A., & Vincent, E. (2020). LibriMix: An open-source dataset for generalizable speech separation. working paper or preprint. https://inria.hal.science/hal-03354695. working paper or preprint (Cited on pp. 14, 15, 62, 112, and 113)

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 764–773. https://doi.org/10.1109/ICCV.2017.89 (Cited on pp. 3, 33, 83, 84, and 152)

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language modeling with gated convolutional networks. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 933–941. (Cited on pp. 25 and 26)

Davies, M. & James, C. (2007). Source separation using single channel ica. *Signal Processing*, 87(8), 1819–1832. https://doi.org/https://doi.org/10.1016/j.sigpro.2007.01.011. Independent Component Analysis and Blind Source Separation (Cited on p. 18)

Défossez, A., Usunier, N., Bottou, L., & Bach, F. (2021). Music Source Separation in the Waveform Domain. working paper or preprint. https://hal.science/hal-02379796. working paper or preprint (Cited on p. 39)

Delcroix, M., Ochiai, T., Zmolikova, K., Kinoshita, K., Tawara, N., Nakatani, T., & Araki, S. (2020). Improving speaker discrimination of target speech extraction with time-domain speakerbeam. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 691–695. https://doi.org/10.1109/ICASSP40776.2020.9054683 (Cited on pp. 8, 9, and 44)

Deng, C., Zhang, Y., Ma, S., Sha, Y., Song, H., & Li, X. (2020). Conv-TasSAN: Separative Adversarial Network Based on Conv-TasNet. *Proc. Interspeech 2020*, 2647–2651. https://doi.org/10.21437/Interspeech.2020-2371 (Cited on pp. 41 and 42)

Ditter, D. & Gerkmann, T. (2020). A multi-phase gammatone filterbank for speech separation via TasNet. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 36–40. https://doi.org/10.1109/ICASSP40776.2020.9053602 (Cited on pp. 22, 46, 93, 104, 109, and 121)

Drude, L., Heitkaemper, J., Boeddeker, C., & Haeb-Umbach, R. (2019). *Sms-wsj: Database, performance measures, and baseline recipe for multi-channel source separation and recognition.* https://doi.org/10.48550/ARXIV.1910.13934 (Cited on p. 12)

Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107, 3–11. https://doi.org/https://doi.org/10.1016/j.neunet.2017.12.012 (Cited on pp. 25, 26, and 123)

Eriksson, J. & Koivunen, V. (2004). Identifiability, separability, and uniqueness of linear ica models. *IEEE Signal Processing Letters*, 11(7), 601–604. https://doi.org/10.1109/LSP.2004.830118 (Cited on p. 18)

Ernst, O., Chazan, S. E., Gannot, S., & Goldberger, J. (2018). Speech dereverberation using fully convolutional networks. *2018 26th European Signal Processing Conference (EUSIPCO)*, 390–394. https://doi.org/10.23919/EUSIPCO.2018.8553141 (Cited on pp. 42 and 61)

Falk, T. H., Zheng, C., & Chan, W.-Y. (2010). A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7), 1766–1774. https://doi.org/10.1109/TASL.2010.2052247 (Cited on pp. 42, 66, 77, 87, and 136)

Fu, S.-W., Yu, C., Hung, K.-H., Ravanelli, M., & Tsao, Y. (2022). Metricgan-u: Unsupervised speech enhancement/ dereverberation based only on noisy/ reverberated speech. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7412–7416. https://doi.org/10.1109/ICASSP43922.2022.9747180 (Cited on p. 42)

Gaoxiong, Y. & Wei, Z. (2012). The perceptual objective listening quality assessment algorithm in telecommunication: Introduction of itu-t new metrics polqa. *2012 1st IEEE International Conference on Communications in China (ICCC)*, 351–355. https://doi.org/10.1109/ICCChina.2012.6356906 (Cited on p. 42)

Goetze, S., Warzybok, A., Kodrasi, I., Jungmann, J. O., Cauchi, B., Rennies, J., Habets, E. A. P., Mertins, A., Gerkmann, T., Doclo, S., & Kollmeier, B. (2014). A study on speech quality and speech intelligibility measures for quality assessment of single-channel dereverberation algorithms. *2014 14th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 233–237. https://doi.org/10.1109/IWAENC.2014.6954293 (Cited on p. 41)

Graetzer, S., Barker, J., Cox, T. J., Akeroyd, M., Culling, J. F., Naylor, G., Porter, E., & Muñoz, R. V. (2021). Clarity-2021 Challenges: Machine Learning Challenges for Advancing Hearing Aid Processing. *Proc. Interspeech 2021*, 686–690. https://doi.org/10.21437/Interspeech.2021-1574 (Cited on pp. 2 and 14)

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850. http://arxiv.org/abs/1308.0850 (Cited on p. 20)

Gu, A. & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*. https://doi.org/10.48550/arXiv.2312.00752 (Cited on p. 34)

Gu, R., Wu, J., Zhang, S.-X., Chen, L., Xu, Y., Yu, M., Su, D., Zou, Y., & Yu, D. (2019). End-to-End Multi-Channel Speech Separation. *arXiv*. https://doi.org/10.48550/arXiv.1905.06286 (Cited on p. 45)

Guiraud, P., Hafezi, S., Naylor, P. A., Moore, A. H., Donley, J., Tourbabin, V., & Lunner, T. (2022). An introduction to the speech enhancement for augmented reality (spear) challenge. *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*, 1–5. https://doi.org/10.1109/IWAENC53105.2022.9914721 (Cited on p. 14)

Gul, S., Khan, M. S., Yoma, N. B., Shah, S. W., & Sheheryar (2022). Enhancing the correlation between the quality and intelligibility objective metrics with the subjective scores by shallow feed forward neural network for time–frequency masking speech separation algorithms. *Applied Acoustics*, 188, 108539. https://doi.org/https://doi.org/10.1016/j.apacoust.2021.108539 (Cited on p. 41)

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., & Pang, R. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition. *Proc. Interspeech 2020*. https://doi.org/10.21437/Interspeech.2020-3015 (Cited on pp. 26, 120, 122, and 127)

Habets, E. A. P. (2007). *Single- and multi-microphone speech dereverberation using spectral enhancement*. Electrical Engineering. https://doi.org/10.6100/IR627677 (Cited on p. 60)

Haeb-Umbach, R., Heymann, J., Drude, L., Watanabe, S., Delcroix, M., & Nakatani, T. (2021). Far-field automatic speech recognition. *Proceedings of the IEEE*, 109(2), 124–148. https://doi.org/10.1109/JPROC.2020.3018668 (Cited on pp. 2, 3, 10, and 60)

Hain, T., Burget, L., Dines, J., Garau, G., Karafiat, M., Lincoln, M., Vepa, J., & Wan, V. (2006). The AMI Meeting Transcription System: Progress and Performance. *Proceedings*

of the Third International Conference on Machine Learning for Multimodal Interaction, MLMI'06, 419–431. https://doi.org/10.1007/11965152_37 (Cited on p. 13)

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., & Wang, Y. (2021). Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3117837 (Cited on p. 73)

Haykin, S. (2009). *Neural Networks and Learning Machines* (10 ed.). Prentice Hall. (Cited on pp. 20, 23, 24, 25, and 111)

Haykin, S. & Chen, Z. (2005). The Cocktail Party Problem. *Neural Computation*, 17(9), 1875–1902. https://doi.org/10.1162/0899766054322964 (Cited on p. 2)

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034. https://doi.org/10.1109/ICCV.2015.123 (Cited on pp. 25 and 74)

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90 (Cited on p. 37)

Hendrycks, D. & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv.org*. https://doi.org/10.48550/arXiv.1606.08415 (Cited on p. 25)

Herault, J., Jutten, C., & Ans, B. (1985). Detection de grandeurs primitives dans un message composite par une architeture de calcul neuromimetique en apprentissage non supervise. *10° Colloque sur le traitement du signal et des images, 1985 ; p. 1017-1022.* (Cited on pp. 9 and 21)

Hershey, J. R., Chen, Z., Le Roux, J., & Watanabe, S. (2016). Deep clustering: Discriminative embeddings for segmentation and separation. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 31–35. https://doi.org/10.1109/ICASSP.2016.7471631 (Cited on pp. 15, 22, and 87)

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597 (Cited on p. 20)

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012b). *Improving neural networks by preventing co-adaptation of feature detectors*. https://doi.org/10.48550/ARXIV.1207.0580 (Cited on pp. 123 and 124)

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02), 107–116. https://doi.org/10.1142/S0218488598000094 (Cited on pp. 21 and 26)

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (Cited on pp. 34 and 62)

Hongyan, L. & Guanglong, R. (2010). Blind separation of noisy mixed speech signals based independent component analysis. *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, 586–589. https://doi.org/10.1109/PCSPA.2010.147 (Cited on p. 16)

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–7141. https://doi.org/10.1109/CVPR.2018.00745 (Cited on pp. 35 and 75)

Huang, L., Cheng, G., Zhang, P., Yang, Y., Xu, S., & Sun, J. (2019). Utterance-level permutation invariant training with latency-controlled blstm for single-channel multi-talker speech separation. *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 1256–1261. https://doi.org/10.1109/APSIPAASC47483.2019.9023163 (Cited on pp. 40 and 41)

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3), 626–634. https://doi.org/10.1109/72.761722 (Cited on p. 18)

Hyvärinen, A. (2013). Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984), 20110534. https://doi.org/10.1098/rsta.2011.0534 (Cited on pp. 3, 16, and 17)

Hyvärinen, A. & Oja, E. (1999). *Independent component analysis: A tutorial*. https://www.cs.jhu.edu/~ayuille/courses/Stat161-261-Spring14/HyvO00-icatut.pdf. Accessed: 29-10-2023 (Cited on p. 17)

Hyvärinen, A. & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4), 411–430. https://doi.org/https://doi.org/10.1016/S0893-6080(00)00026-5 (Cited on p. 18)

Ingle, V., Kogon, S., & Manolakis, D. (2005) (Cited on p. 17)

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167. http://arxiv.org/abs/1502.03167 (Cited on p. 24)

Isik, Y., Roux, J. L., Chen, Z., Watanabe, S., & Hershey, J. R. (2016). Single-Channel Multi-Speaker Separation Using Deep Clustering. *Proc. Interspeech 2016*, 545–549. https://doi.org/10.21437/Interspeech.2016-1176 (Cited on pp. xxix, 14, 15, 58, 87, 126, 129, and 135)

ITU (2003). *Itu-t recommendation p.800.1, mean opinion score (mos) terminology*. https://www.itu.int/rec/T-REC-P.800.1 (Cited on p. 42)

Jensen, J. & Taal, C. H. (2016). An algorithm for predicting the intelligibility of speech masked by modulated noise maskers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11), 2009–2022. https://doi.org/10.1109/TASLP.2016.2585878 (Cited on pp. 42, 66, 87, 136, and 141)

Jutten, C. & Herault, J. (1991). Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1), 1–10. https://doi.org/https://doi.org/10.1016/0165-1684(91)90079-X (Cited on pp. 8, 9, 16, and 21)

Kadıoğlu, B., Horgan, M., Liu, X., Pons, J., Darcy, D., & Kumar, V. (2020). An empirical study of conv-tasnet. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7264–7268. https://doi.org/10.1109/ICASSP40776.2020.9054721 (Cited on pp. xxx, 101, 103, 104, 106, 107, 108, and 109)

Kanda, N., Gaur, Y., Wang, X., Meng, Z., & Yoshioka, T. (2020). Serialized Output Training for End-to-End Overlapped Speech Recognition. *Proc. Interspeech 2020*, 2797–2801. https://doi.org/10.21437/Interspeech.2020-999 (Cited on p. 58)

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. (Cited on pp. 101, 109, and 111)

Kim, E. & Seo, H. (2021). SE-Conformer: Time-Domain Speech Enhancement Using Conformer. *Proc. Interspeech 2021*. https://doi.org/10.21437/Interspeech.2021-2207 (Cited on pp. 121, 122, and 127)

Kinoshita, K., Delcroix, M., Gannot, S., Habets, E. A. P., Haeb-Umbach, R., Kellermann, W., Leutnant, V., Maas, R., Nakatani, T., Raj, B., Sehr, A., & Yoshioka, T. (2011). A

summary of the REVERB challenge: state-of-the-art and remaining challenges in reverberant speech processing research. *EURASIP Journal on Advances in Signal Processing*, 2016, 7. https://doi.org/10.1186/s13634-016-0306-6 (Cited on p. 13)

Kinoshita, K., Delcroix, M., Kwon, H., Mori, T., & Nakatani, T. (2017). Neural Network-Based Spectrum Estimation for Online WPE Dereverberation. *Proc. Interspeech 2017*. https://doi.org/10.21437/Interspeech.2017-733 (Cited on p. 61)

Kinoshita, K., Ochiai, T., Delcroix, M., & Nakatani, T. (2020). Improving noise robust automatic speech recognition with single-channel time-domain enhancement network. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7009–7013. https://doi.org/10.1109/ICASSP40776.2020.9053266 (Cited on p. 45)

Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., & Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5220–5224. https://doi.org/10.1109/ICASSP.2017.7953152 (Cited on pp. xxiv and 12)

Koizumi, Y., Karita, S., Wisdom, S., Erdogan, H., Hershey, J. R., Jones, L., & Bacchiani, M. (2021). Df-conformer: Integrated architecture of conv-tasnet and conformer using linear complexity self-attention for speech enhancement. *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 161–165. https://doi.org/10.1109/WASPAA52581.2021.9632794 (Cited on p. 121)

Kolbaek, M., Yu, D., Tan, Z.-H., Jensen, J., Kolbaek, M., Yu, D., Tan, Z.-H., & Jensen, J. (2017). Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(10), 1901–1913. https://doi.org/10.1109/TASLP.2017.2726762 (Cited on pp. 40, 43, 61, 112, 125, and 134)

Kolbæk, M., Tan, Z.-H., Jensen, S. H., & Jensen, J. (2020). On loss functions for supervised monaural time-domain speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 825–838. https://doi.org/10.1109/TASLP.2020.2968738 (Cited on pp. 39 and 41)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, volume 25. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (Cited on pp. 20 and 21)

Le Roux, J., Hershey, J. R., & Weninger, F. (2015). Deep nmf for speech separation. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 66–70. https://doi.org/10.1109/ICASSP.2015.7177933 (Cited on pp. 3, 18, and 43)

Leglaive, S., Borne, L., Tzinis, E., Sadeghi, M., Fraticelli, M., Wisdom, S., Pariente, M., Pressnitzer, D., & Hershey, J. (2023). The CHiME-7 UDASE task: Unsupervised domain adaptation for conversational speech enhancement. *Proc. 7th International Workshop on Speech Processing in Everyday Environments (CHiME 2023)*, 7–12. https://doi.org/10.21437/CHiME.2023-2 (Cited on pp. 14, 62, and 151)

Li, C., Shi, J., Zhang, W., Subramanian, A. S., Chang, X., Kamo, N., Hira, M., Hayashi, T., Boeddeker, C., Chen, Z., & Watanabe, S. (2021). ESPnet-SE: End-to-end speech enhancement and separation toolkit designed for ASR integration. *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, 785–792. (Cited on pp. 27, 45, 49, and 53)

Li, C., Yang, L., Wang, W., & Qian, Y. (2022a). Skim: Skipping memory lstm for low-latency real-time continuous speech separation. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 681–685. https://doi.org/10.1109/ICASSP43922.2022.9746372 (Cited on pp. 9, 62, 89, 129, and 130)

Li, H., Fu, S.-W., Tsao, Y., & Yamagishi, J. (2020). iMetricGAN: Intelligibility Enhancement for Speech-in-Noise Using Generative Adversarial Network-Based Metric Learning. *Proc. Interspeech 2020*, 1336–1340. https://doi.org/10.21437/Interspeech.2020-1016 (Cited on p. 42)

Li, K., Yang, R., Sun, F., & Hu, X. (2024). IIANet: An intra- and inter-modality attention network for audio-visual speech separation. *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=FM61SQzF3N (Cited on p. 112)

Li, S., Cao, S., Hui, L., Jiang, Z., Sun, Y., & Xu, S. (2022b). A computational-efficient deformable convolution network accelerator via hardware and algorithm co-optimization. *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, 1–6. https://doi.org/10.1109/SiPS55645.2022.9919242 (Cited on p. 150)

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A Structured Self-Attentive Sentence Embedding. *International Conference on Learning Representations*. https://openreview.net/forum?id=BJC_jUqxe (Cited on pp. 34, 96, and 152)

Lincoln, M., McCowan, I., Vepa, J., & Maganti, H. (2005). The multi-channel wall street journal audio visual corpus (mc-wsj-av): specification and initial experiments. *IEEE*

Workshop on Automatic Speech Recognition and Understanding, 2005., 357–362. https://doi.org/10.1109/ASRU.2005.1566470 (Cited on p. 13)

Lincoln, M., McCowan, I., Vepa, J., & Maganti, H. K. (2005). The multi-channel wall street journal audio visual corpus (MC-WSJ-AV): specification and initial experiments. *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, 357–362. https://doi.org/10.1109/ASRU.2005.1566470 (Cited on pp. 133 and 135)

Liu, Q., Wang, W., Jackson, P. J. B., & Cox, T. J. (2015). A source separation evaluation method in object-based spatial audio. *2015 23rd European Signal Processing Conference (EUSIPCO)*, 1088–1092. https://doi.org/10.1109/EUSIPCO.2015.7362551 (Cited on p. 41)

Luo, J., Wang, J., Cheng, N., Xiao, E., Zhang, X., & Xiao, J. (2022). Tiny-sepformer: A tiny time-domain transformer network for speech separation. *Proc. Interspeech 2022*. (Cited on pp. 10, 62, and 89)

Luo, Y., Chen, Z., Hershey, J. R., Le Roux, J., & Mesgarani, N. (2017). Deep clustering and conventional networks for music separation: Stronger together. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 61–65. https://doi.org/10.1109/ICASSP.2017.7952118 (Cited on p. 22)

Luo, Y., Chen, Z., & Yoshioka, T. (2020). Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 46–50. https://doi.org/10.1109/ICASSP40776.2020.9054266 (Cited on pp. 40, 54, 55, 86, 93, 111, 119, 121, and 122)

Luo, Y., Han, C., Mesgarani, N., Ceolini, E., & Liu, S.-C. (2019). Fasnet: Low-latency adaptive beamforming for multi-microphone audio processing. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 260–267. https://doi.org/10.1109/ASRU46091.2019.9003849 (Cited on p. 45)

Luo, Y. & Mesgarani, N. (2018a). Real-time single-channel dereverberation and separation with time-domain audio separation network. *Proc. Interspeech 2018*. https://doi.org/10.21437/Interspeech.2018-2290 (Cited on pp. 61 and 64)

Luo, Y. & Mesgarani, N. (2018b). Tasnet: Time-domain audio separation network for real-time, single-channel speech separation. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 696–700. https://doi.org/10.1109/ICASSP.2018.8462116 (Cited on pp. 3, 9, 18, 22, 40, 41, 43, 44, 46, 52, 53, 55, 61, 93, 121, and 125)

Luo, Y. & Mesgarani, N. (2019). Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8), 1256–1266. https://doi.org/10.1109/TASLP.2019.2915167 (Cited on pp. xxiv, xxx, 2, 3, 9, 14, 22, 23, 26, 27, 32, 40, 41, 43, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 65, 73, 74, 75, 77, 83, 84, 85, 86, 87, 89, 93, 101, 102, 103, 104, 106, 108, 111, 112, 119, 122, 129, and 130)

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. https://doi.org/10.18653/v1/D15-1166 (Cited on p. 34)

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proc. 30th International Conference on Machine Learning (ICML 2013)*, 3. (Cited on p. 25)

Maciejewski, M., Wichern, G., McQuinn, E., & Roux, J. L. (2020). WHAMR!: Noisy and Reverberant Single-Channel Speech Separation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 696–700. https://doi.org/10.1109/ICASSP40776.2020.9053327 (Cited on pp. xxvi, xxix, xxx, 2, 8, 9, 10, 11, 13, 14, 15, 41, 58, 62, 87, 94, 97, 102, 126, and 135)

McCulloch, W. & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 127–147. (Cited on p. 20)

Mikolov, T. (2012). *Statistical Language Models Based on Neural Networks*. Brno University of Technology, Faculty of Information Technology. https://www.fit.vut.cz/study/phd-thesis/283/ (Cited on p. 52)

Möller, S. & Köster, F. (2017). Review of recent standardization activities in speech quality of experience. *Quality and User Experience*, 2(1), 9. https://doi.org/10.1007/s41233-017-0012-7 (Cited on p. 42)

Morrone, G., Cornell, S., Raj, D., Serafini, L., Zovato, E., Brutti, A., & Squartini, S. (2023). Low-latency speech separation guided diarization for telephone conversations. *2022 IEEE Spoken Language Technology Workshop (SLT)*, 641–646. https://doi.org/10.1109/SLT54892.2023.10023280 (Cited on p. 13)

Mošner, L., Plchot, O., Burget, L., & Černocký, J. H. (2022). Multi-channel speaker verification with conv-tasnet based beamformer. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7982–7986. https://doi.org/10.1109/ICASSP43922.2022.9747771 (Cited on p. 44)

Naithani, G., Nikunen, J., Bramslow, L., & Virtanen, T. (2018). Deep neural network based speech separation optimizing an objective estimator of intelligibility for low latency applications. *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 386–390. https://doi.org/10.1109/IWAENC.2018.8521379 (Cited on p. 42)

Nakatani, T., Yoshioka, T., Kinoshita, K., Miyoshi, M., & Juang, B.-H. (2010). Speech dereverberation based on variance-normalized delayed linear prediction. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7), 1717–1731. https://doi.org/10.1109/TASL.2010.2052251 (Cited on p. 61)

Naylor, P. A. & Gaubitch, N. D. (2010). *Speech Dereverberation* (1st ed.). Springer Publishing. (Cited on pp. 2 and 60)

NIST (2009). *The NIST rich transcription 2009 (RT'09) evaluation*. http://www.itl.nist.gov/iad/mig/tests/rt/2009/docs/rt09-meeting-eval-plan-v2.pdf. (Cited on p. 13)

NIST (2020). *Rich Transcription Evaluation*. https://www.nist.gov/itl/iad/mig/rich-transcription-evaluation. (Cited on p. 13)

Nossier, S. A., Wall, J., Moniri, M., Glackin, C., & Cannings, N. (2020). Mapping and masking targets comparison using different deep learning based speech enhancement architectures. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206623 (Cited on p. 23)

Ochiai, T., Delcroix, M., Ikeshita, R., Kinoshita, K., Nakatani, T., & Araki, S. (2020). Beam-tasnet: Time-domain audio separation network meets frequency-domain beamformer. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6384–6388. https://doi.org/10.1109/ICASSP40776.2020.9053575 (Cited on pp. 2, 9, 44, and 45)

Oppenheim, A., Schafer, R., & Stockham, T. (1968). Nonlinear filtering of multiplied and convolved signals. *IEEE Transactions on Audio and Electroacoustics*, 16(3), 437–466. https://doi.org/10.1109/TAU.1968.1161990 (Cited on p. 60)

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An asr corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964 (Cited on pp. 14, 15, 16, and 143)

Pariente, M., Cornell, S., Deleforge, A., & Vincent, E. (2020). Filterbank design for end-to-end speech separation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6364–6368. https://doi.org/10.1109/ICASSP40776.2020.9053038 (Cited on pp. 46 and 93)

Park, S., Jeong, Y., Kim, M. S., & Kim, H. S. (2018). Linear prediction-based dereverberation with very deep convolutional neural networks for reverberant speech recognition. *ICEIC 2018*, volume 2018-January, 1–2. https://doi.org/10.23919/ELINFOCOM.2018.8330593 (Cited on p. 61)

Pasad, A., Shi, B., & Livescu, K. (2023). Comparative layer-wise analysis of self-supervised speech models. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096149 (Cited on p. 150)

Paul, D. B. & Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*. https://aclanthology.org/H92-1073 (Cited on p. 14)

Peharz, R. & Pernkopf, F. (2012). Sparse nonnegative matrix factorization with $\ell 0$-constraints. *Neurocomputing (Amst)*, 80(1), 38–46. (Cited on p. 19)

Pham, D. T. & Garat, P. (1997). Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7), 1712–1725. https://doi.org/10.1109/78.599941 (Cited on p. 18)

Proakis, J. G. & Manolakis, D. K. (1996). *Digital Signal Processing* (3rd ed.). Prentice Hall. (Cited on p. 29)

Purushothaman, A., Sreeram, A., Kumar, R., & Ganapathy, S. (2020). Deep Learning Based Dereverberation of Temporal Envelopes for Robust Speech Recognition. *Proc. Interspeech 2020*. https://doi.org/10.21437/Interspeech.2020-2283 (Cited on p. 61)

Qian, Y., Chang, X., & Yu, D. (2018). Single-channel multi-talker speech recognition with permutation invariant training. *Speech Communication*, 104, 1–11. https://doi.org/https://doi.org/10.1016/j.specom.2018.09.003 (Cited on p. 59)

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. (Cited on pp. 13, 34, 35, 59, 141, and 144)

Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6078–6087. (Cited on p. 150)

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada,*

April 30 - May 3, 2018, Workshop Track Proceedings. https://openreview.net/forum?id=Hkuq2EkPf (Cited on p. 26)

Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2), 92–102. https://doi.org/10.1109/TETCI.2017.2762739 (Cited on p. 34)

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., De Mori, R., & Bengio, Y. (2021). *Speechbrain: A general-purpose speech toolkit*. https://doi.org/10.48550/ARXIV.2106.04624 (Cited on pp. 27, 45, 49, 53, 74, 77, 83, 87, 102, and 126)

Ravenscroft, W., Goetze, S., & Hain, T. (2022a). Att-TasNet: Attending to Encodings in Time-Domain Audio Speech Separation of Noisy, Reverberant Speech Mixtures. *Frontiers in Signal Processing*, 2. https://doi.org/10.3389/frsip.2022.856968 (Cited on pp. 5, 6, 8, 47, 65, 74, 75, 77, 86, and 93)

Ravenscroft, W., Goetze, S., & Hain, T. (2022b). Receptive field analysis of temporal convolutional networks for monaural speech dereverberation. *2022 30th European Signal Processing Conference (EUSIPCO)*, 80–84. https://doi.org/10.23919/EUSIPCO55093.2022.9909855 (Cited on pp. 3, 5, 64, 73, 75, 76, 77, 83, 116, and 121)

Ravenscroft, W., Goetze, S., & Hain, T. (2022c). Utterance weighted multi-dilation temporal convolutional networks for monaural speech dereverberation. *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*, 1–5. https://doi.org/10.1109/IWAENC53105.2022.9914752 (Cited on pp. 5, 6, 76, and 83)

Ravenscroft, W., Goetze, S., & Hain, T. (2023a). Deformable temporal convolutional networks for monaural noisy reverberant speech separation. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10095230 (Cited on pp. 5, 6, 8, 73, 83, 118, and 129)

Ravenscroft, W., Goetze, S., & Hain, T. (2023b). On Data Sampling Strategies for Training Neural Network Speech Separation Models. *31st European Signal Processing Conference (EUSIPCO 2023)*. (Cited on pp. 5, 6, 111, 118, 124, 126, 135, and 136)

Ravenscroft, W., Goetze, S., & Hain, T. (2023c). On time domain conformer models for monaural speech separation in noisy reverberant acoustic environments. *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1–7. https://doi.org/10.1109/ASRU57964.2023.10389669 (Cited on pp. 5, 6, 120, 131, 132, 134, and 135)

Ravenscroft, W., Goetze, S., & Hain, T. (2024). Combining Conformer and Dual-Path-Transformer Networks for Single Channel Noisy Reverberant Speech Separation. *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Accepted. (Cited on pp. 5, 7, and 132)

Richard, G., Smaragdis, P., Gannot, S., Naylor, P. A., Makino, S., Kellermann, W., & Sugiyama, A. (2023). Audio signal processing in the 21st century: The important outcomes of the past 25 years. *IEEE Signal Processing Magazine*, 40(5), 12–26. https://doi.org/10.1109/MSP.2023.3276171 (Cited on pp. 9 and 16)

Rix, A., Beerends, J., Hollier, M., & Hekstra, A. (2001a). Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, 749–752 vol.2. https://doi.org/10.1109/ICASSP.2001.941023 (Cited on p. 42)

Rix, A., Beerends, J., Hollier, M., & Hekstra, A. (2001b). Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. *ICASSP 2001*. https://doi.org/10.1109/ICASSP.2001.941023 (Cited on pp. 66, 77, 87, 136, and 141)

Rixen, J. & Renz, M. (2022). QDPN - Quasi-dual-path Network for single-channel Speech Separation. *Proc. Interspeech 2022*. https://doi.org/10.21437/Interspeech.2022-700 (Cited on pp. 8, 9, 22, 41, 43, 53, 89, 111, 119, 121, 125, 130, 149, and 150)

Robinson, T., Fransen, J., Pye, D., Foote, J., & Renals, S. (1995). WSJCAMO: a British English speech corpus for large vocabulary continuous speech recognition. *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, 81–84 vol.1. https://doi.org/10.1109/ICASSP.1995.479278 (Cited on p. 13)

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 234–241. (Cited on p. 62)

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408. https://doi.org/10.1037/h0042519 (Cited on p. 23)

Roux, J. L., Wisdom, S., Erdogan, H., & Hershey, J. R. (2019). SDR – Half-baked or Well Done? *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 626–630. https://doi.org/10.1109/ICASSP.2019.8683855 (Cited on pp. 18, 22, 53, 76, 87, 96, 112, and 125)

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. https://doi.org/10.1038/323533a0 (Cited on pp. 20 and 59)

Santos, J. F., Senoussaoui, M., & Falk, T. H. (2014). An improved non-intrusive intelligibility metric for noisy and reverberant speech. *2014 14th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 55–59. https://doi.org/10.1109/IWAENC.2014.6953337 (Cited on p. 42)

Scheibler, R., Bezzam, E., & Dokmanić, I. (2018). Pyroomacoustics: A python package for audio room simulation and array processing algorithms. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 351–355. https://doi.org/10.1109/ICASSP.2018.8461310 (Cited on pp. 15 and 66)

Schmidt, M. (2009). *Single-channel source separation using non-negative matrix factorization*. Technical University of Denmark, DTU Informatics. (Cited on p. 19)

Schmidt, M. N. & Olsson, R. K. (2006). Single-Channel Speech Separation using Sparse Non-Negative Matrix Factorization. *Proc. Interspeech 2006*. (Cited on pp. 10 and 19)

Segbroeck, M. V., Ahmed, Z., Kutsenko, K., Huerta, C., Nguyen, T., Hoffmeister, B., Trmal, J., Omologo, M., & Maas, R. (2019). Dipco - dinner party corpus. *Proc. Interspeech 2020*. https://www.amazon.science/publications/dipco-dinner-party-corpus (Cited on p. 13)

Settle, S., Roux, J. L., Hori, T., Watanabe, S., & Hershey, J. R. (2018). End-to-end multi-speaker speech recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4819–4823. https://doi.org/10.1109/ICASSP.2018.8461893 (Cited on p. 141)

Shi, J., Chang, X., Watanabe, S., & Xu, B. (2022). Train from scratch: Single-stage joint training of speech separation and recognition. *Computer Speech & Language*, 76, 101387. https://doi.org/https://doi.org/10.1016/j.csl.2022.101387 (Cited on p. 141)

Sinha, R., Tammen, M., Rollwage, C., & Doclo, S. (2022). Speaker-conditioning single-channel target speaker extraction using conformer-based architectures. *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*, 1–5. https://doi.org/10.1109/IWAENC53105.2022.9914691 (Cited on pp. 121 and 150)

Sklyar, I., Piunova, A., Zheng, X., & Liu, Y. (2022). Multi-turn rnn-t for streaming recognition of multi-party speech. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8402–8406. https://doi.org/10.1109/ICASSP43922.2022.9746074 (Cited on pp. 42 and 145)

Steinmetz, C. J. & Reiss, J. D. (2021). pyloudnorm: A simple yet flexible loudness meter in python. *150th AES Convention*. (Cited on p. 135)

Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 334–340. http://arxiv.org/abs/1806.03185 (Cited on p. 41)

Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., & Zhong, J. (2021). Attention is all you need in speech separation. *Proc. Interspeech 2021*. https://doi.org/10.1109/ICASSP39728.2021.9413901 (Cited on pp. xxvii, xxix, 2, 8, 9, 10, 11, 14, 22, 23, 40, 43, 53, 54, 55, 56, 57, 58, 61, 62, 86, 89, 93, 111, 112, 119, 120, 121, 122, 124, 125, 126, 127, 129, 130, 131, 132, 133, 134, 135, and 153)

Subakan, C., Ravanelli, M., Cornell, S., Grondin, F., & Bronzi, M. (2023). Exploring self-attention mechanisms for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31, 2169–2180. https://doi.org/10.1109/TASLP.2023.3282097 (Cited on pp. xxix, 54, 55, 58, and 118)

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2010). A short-time objective intelligibility measure for time-frequency weighted noisy speech. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4214–4217. https://doi.org/10.1109/ICASSP.2010.5495701 (Cited on p. 42)

Takase, S. & Kiyono, S. (2023). Lessons on parameter sharing across layers in transformers. *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, 78–90. https://doi.org/10.18653/v1/2023.sustainlp-1.5 (Cited on p. 34)

Tharwat, A. (2021). Independent component analysis: An introduction. *Applied Computing and Informatics*, 17(2), 222–249. https://doi.org/10.1016/j.aci.2018.08.006 (Cited on p. 17)

Tzinis, E., Adi, Y., Ithapu, V. K., Xu, B., Smaragdis, P., & Kumar, A. (2022a). Remixit: Continual self-training of speech enhancement models via bootstrapped remixing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6), 1329–1341. https://doi.org/10.1109/JSTSP.2022.3200911 (Cited on pp. 11 and 62)

Tzinis, E., Wang, Z., Jiang, X., & Smaragdis, P. (2022b). Compute and memory efficient universal sound source separation. *Journal of Signal Processing Systems*, 94(2), 245–259. https://doi.org/10.1007/s11265-021-01683-x (Cited on pp. 9, 41, 42, 43, 53, 61, 62, 89, 124, 129, and 130)

Tzinis, E., Wang, Z., & Smaragdis, P. (2020). Sudo rm-rf: Efficient networks for universal audio source separation. *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. (Cited on p. 62)

Ueda, T., Nakatani, T., Ikeshita, R., Kinoshita, K., Araki, S., & Makino, S. (2021). Low latency online source separation and noise reduction based on joint optimization with dereverberation. *2021 29th European Signal Processing Conference (EUSIPCO)*, 1000–1004. https://doi.org/10.23919/EUSIPCO54536.2021.9616119 (Cited on p. 11)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, volume 30. (Cited on pp. 4, 34, 35, 36, 37, 42, 54, 94, 95, 97, 99, 101, 109, 124, and 152)

Vincent, E., Watanabe, S., Barker, J., & Marxer, R. (2016). The 4th CHiME speech separation and recognition challenge. *URL: http://spandh. dcs. shef. ac. uk/chime challenge {Last Accessed on 3 November, 2023}*. (Cited on pp. 13 and 14)

Virtanen, T. (2007). Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3), 1066–1074. https://doi.org/10.1109/TASL.2006.885253 (Cited on p. 8)

von Neumann, T., Boeddeker, C., Cord-Landwehr, T., Delcroix, M., & Haeb-Umbach, R. (2023a). *Meeting recognition with continuous speech separation and transcription-supported diarization*. https://doi.org/10.48550/arXiv.2309.16482 (Cited on p. 59)

von Neumann, T., Boeddeker, C., Kinoshita, K., Delcroix, M., & Haeb-Umbach, R. (2023b). On word error rate definitions and their efficient computation for multi-speaker speech recognition systems. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094784 (Cited on p. 145)

Wang, H., Wu, B., Chen, L., Yu, M., Yu, J., Xu, Y., Zhang, S., Weng, C., Su, D., & Yu, D. (2021). TeCANet: Temporal-Contextual Attention Network for Environment-Aware Speech Dereverberation. *Proc. Interspeech 2021*. (Cited on p. 61)

Wang, Y., Narayanan, A., & Wang, D. (2014). On training targets for supervised speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12), 1849–1858. https://doi.org/10.1109/TASLP.2014.2352935 (Cited on p. 42)

Wang, Z. & Wang, D. (2020). Deep Learning Based Target Cancellation for Speech Dereverberation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 941–950. (Cited on p. 61)

Wang, Z.-Q., Cornell, S., Choi, S., Lee, Y., Kim, B.-Y., & Watanabe, S. (2023a). TF-GridNet: Integrating Full- and Sub-Band Modeling for Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31, 3221–3236. https://doi.org/10.1109/TASLP.2023.3304482 (Cited on pp. 2, 22, 23, 41, 61, 129, and 130)

Wang, Z.-Q., Cornell, S., Choi, S., Lee, Y., Kim, B.-Y., & Watanabe, S. (2023b). TF-GRIDNET: Making Time-Frequency Domain Models Great Again for Monaural Speaker Separation. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094992 (Cited on pp. 3, 9, 14, 41, 53, 59, 93, and 149)

Wang, Z.-Q., Roux, J. L., & Hershey, J. R. (2018). Alternative objective functions for deep clustering. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 686–690. https://doi.org/10.1109/ICASSP.2018.8462507 (Cited on pp. 2, 3, 14, 22, and 43)

Wang, Z.-Q. & Watanabe, S. (2023). UNSSOR: Unsupervised Neural Speech Separation by Leveraging Over-determined Training Mixtures. *Advances in Neural Information Processing Systems*. (Cited on pp. 62 and 151)

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Enrique Yalta Soplin, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). ESPnet: End-to-end speech processing toolkit. *Proc. Interspeech 2018*, 2207–2211. https://doi.org/10.21437/Interspeech.2018-1456 (Cited on p. 49)

Watanabe, S., Mandel, M., Barker, J., Vincent, E., Arora, A., Chang, X., Khudanpur, S., Manohar, V., Povey, D., Raj, D., Snyder, D., Subramanian, A. S., Trmal, J., Yair, B. B., Boeddeker, C., Ni, Z., Fujita, Y., Horiguchi, S., Kanda, N., Yoshioka, T., & Ryant, N. (2020). CHiME-6 Challenge: Tackling Multispeaker Speech Recognition for Unsegmented Recordings. *Proc. 6th International Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 1–7. https://doi.org/10.21437/CHiME.2020-1 (Cited on pp. 14, 42, 58, and 145)

Wichern, G., Antognini, J., Flynn, M., Zhu, L. R., McQuinn, E., Crow, D., Manilow, E., & Le Roux, J. (2019). WHAM!: Extending speech separation to noisy environments. *Proc. Interspeech 2019*. (Cited on pp. 11, 13, 14, and 15)

Wisdom, S., Tzinis, E., Erdogan, H., Weiss, R., Wilson, K., & Hershey, J. (2020). Unsupervised sound separation using mixture invariant training. *Advances in Neural Information Processing Systems*, volume 33, 3846–3857. https://proceedings.neurips.cc/paper_files/paper/2020/file/28538c394c36e4d5ea8ff5ad60562a93-Paper.pdf (Cited on p. 62)

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1), 37–52. https://doi.org/https://doi.org/10.1016/0169-7439(87)80084-9. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists (Cited on p. 17)

Wu, Y. & He, K. (2018). Group normalization. *Proceedings of the European Conference on Computer Vision (ECCV)*. (Cited on pp. 27, 28, and 124)

Xu, Y., Weng, C., Hui, L., Liu, J., Yu, M., Su, D., & Yu, D. (2019). Joint training of complex ratio mask based beamformer and acoustic model for noise robust asr. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6745–6749. https://doi.org/10.1109/ICASSP.2019.8682576 (Cited on p. 60)

Yang, G.-P., Tuan, C.-I., Lee, H.-Y., & shan Lee, L. (2019). Improved Speech Separation with Time-and-Frequency Cross-Domain Joint Embedding and Clustering. *Proc. Interspeech 2019*, 1363–1367. https://doi.org/10.21437/Interspeech.2019-2181 (Cited on p. 93)

Yilmaz, O. & Rickard, S. (2004). Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7), 1830–1847. https://doi.org/10.1109/TSP.2004.828896 (Cited on p. 8)

Yu, D., Kolbæk, M., Tan, Z.-H., & Jensen, J. (2017). Permutation invariant training of deep models for speaker-independent multi-talker speech separation. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 241–245. https://doi.org/10.1109/ICASSP.2017.7952154 (Cited on p. 22)

Zeghidour, N. & Grangier, D. (2021). Wavesplit: End-to-end speech separation by speaker clustering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 2840–2849. https://doi.org/10.1109/TASLP.2021.3099291 (Cited on pp. 23, 83, 112, 117, 118, and 129)

Zhang, J., Zorilă, C., Doddipatla, R., & Barker, J. (2020a). On end-to-end multi-channel time domain speech separation in reverberant environments. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6389–6393. https://doi.org/10.1109/ICASSP40776.2020.9053833 (Cited on pp. 11 and 61)

Zhang, L., Shi, Z., Han, J., Shi, A., & Ma, D. (2020b). Furcanext: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks. *MultiMedia Modeling*, 653–665. (Cited on pp. 33, 45, 83, and 86)

Zhao, S. & Ma, B. (2023). Mossformer: Pushing the performance limit of monaural speech separation using gated single-head transformer with convolution-augmented joint self-attentions. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096646 (Cited on pp. 43, 129, 130, 149, and 150)

Zhao, T., Zhao, Y., Wang, S., & Han, M. (2021). Unet++-based multi-channel speech dereverberation and distant speech recognition. *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 1–5. https://doi.org/10.1109/ISCSLP49672.2021.9362064 (Cited on p. 61)

Zhao, Y., Wang, D., Xu, B., & Zhang, T. (2020). Monaural Speech Dereverberation Using Temporal Convolutional Networks With Self Attention. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 1598–1607. https://doi.org/10.1109/TASLP.2020.2995273 (Cited on p. 61)

Zhu, L. (2022). *Thop: Pytorch-opcounter*. https://pypi.org/project/thop/. Accessed: 19-10-2022. (Cited on pp. xxx, 89, and 127)

Žmolíková, K., Delcroix, M., Kinoshita, K., Ochiai, T., Nakatani, T., Burget, L., & Černocký, J. (2019). Speakerbeam: Speaker aware neural network for target speaker extraction in speech mixtures. *IEEE Journal of Selected Topics in Signal Processing*, 13(4), 800–814. https://doi.org/10.1109/JSTSP.2019.2922820 (Cited on p. 8)