Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

Hybrid and End-to-End Approaches for Noise Robust Automatic Speech Recognition

Lujun Li

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades einer

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Thomas Eibert

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Gerhard Rigoll
2. Prof. Dr.-Ing. Werner Hemmert

Die Dissertation wurde am 30.09.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 25.02.2022 angenommen.

To my parents.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Gerhard Rigoll for offering me the opportunity to work at the Institute for Human-Machine Communication at TUM and for his excellent supervision, invaluable supports, and insightful discussions during my time at the institute. He was never short of ideas for exciting new research directions and comments to improve my scientific thinking and writing. His fruitful academic achievements and rich working experience have always motivated and propelled me. It has been a privilege and remarkable experience to work with Prof. Rigoll.

Secondly, I am grateful to Prof. Hemmert for doing the second review of this thesis and providing constructive comments and suggestions, and Prof. Eibert for chairing the examination board.

Next, I would like to thank my colleagues at the Institute for Human-Machine Communication. Special thanks go to Ludwig Kürzinger and Tobias Watzel for their selfless help of my research and their sincere congratulations, comfort, and encouragement. Special thanks also to Stefan Hörmann, Patrick Lindemann, Okan Köpüklü, Martin Knoche, and Maryam Babaee for our inspirational discussions, for countless support, help, and advice, and for all the amazing moments we have spent together! Thanks also to Fabian Herzog, Maximilian Rettinger, Torben Teepe, Johannes Gilg, Daniel Merget, and Mohammadreza Babaee, their commitments to excellence have always impelled and invigorated me.

Besides, I would like to thank Mr. Peter Brand and Mr. Heiner Hundhammer for their software and hardware supports. For the administrative help and advice, I would like to thank Ms. Birgit Feicht, Ms. Agata Gorska, Ms. Martina Römpp, and Ms. Marion Bächle.

Thanks again to all the aforementioned colleagues and my beloved ones in Germany, who make my working experience in the Institute for Human-Machine Communication and the life experience in Germany impressive and unforgettable for my whole life! I will cherish my stay here in memory as long as I live!

Last but not least, no word is strong enough to express my great gratitude to my parents, to whom I owe everything. For doing the Ph.D. overseas, neither the decision nor any step therein is easy, massive thanks to their unconditional support and love, and to always being there for me in the past four years, whatever success or failure! Without

them, I cannot be the person I am today!

Munich, August 2021 Lujun Li

Abstract

Automatic speech recognition (ASR) is becoming increasingly more integral in our day-to-day lives, enabling more applications to adopt speech-based human-machine interaction, e.g., Siri, Cortana, Amazon Echo, and many more. While recent breakthroughs have tremendously improved ASR performance, these models still suffer considerable degradation from ambient noise. With the increasing use of ASR systems in everyday life, ASR robustness under adverse conditions becomes more important than ever. According to the processing stages of an ASR system, approaches for increasing ASR robustness can be classified into three groups: (i) Back-end techniques, i.e., model adaptation and multi-condition training; (ii) Front-end techniques, like noise mask estimation, model denoising, and robust acoustic feature extraction; (iii) Joint training of front- and back-ends. In this respect, this thesis follows these three axes of research.

Along the first axis, this thesis discusses an approach to optimize the Hidden Markov Model (HMM) structure, namely Deep Neural Fenonic Baseform Growing. This method, which is data-driven, concisely designed, and computationally cheap, customizes the HMM structure for each phone precisely without external assumptions concerning the state number or the transition pattern. Experimental results on both TIMIT and TEDliumv2 corpora indicate that the proposed HMM structure improves both the monophone and the triphone systems substantially and increases system robustness in noisy environments. Besides, Deep Neural Fenonic Baseform Growing further improves state-of-the-art speech recognition systems with remarkably reduced parameters.

The second axis presents two optimized architectures of speech enhancement generative adversarial networks (GANs). For one thing, this study investigates speech enhancement GANs equipped with the self-attention mechanism in three manners. For another, this thesis usefully merges the Sinc convolution and the speech enhancement GAN, resulting in a customizable, lightweight, and interpretable system. Moreover, a set of data augmentation techniques in the time domain is also employed to further improve system performance and generalization abilities. Experimental results show that the proposed system outperforms a set of competitive models, especially on higher-level perceptual quality and speech intelligibility.

The third axis introduces a self-attention based joint training framework, concatenating a speech enhancement front-end and a downstream ASR module to be jointly trained

as an extensive network to boost the noise robustness of the ASR system. Moreover, a discriminant component plays the role of the global guide in the adversarial joint training, which guides the enhancement front-end to output more desirable features for the subsequent ASR module and thereby offsets the limitation of the separate training and handcrafted loss functions. Systematic experiments reveal that the proposed framework significantly outperforms other competitive solutions, especially in challenging environments.

Zusammenfassung

Automatische Spracherkennung (Automatic Speech Recognition, ASR) wird immer mehr zu einem integralen Bestandteil unseres täglichen Lebens und ermöglicht nutzerspezifische Anwendungen zur sprachbasierten Mensch-Maschine-Kommunikation, z.B. Siri, Cortana, Amazon Echo und viele mehr. Während die letzten Durchbrüche die ASR-Leistung enorm verbessert haben, sind diese Modelle immer noch anfällig für Umgebungsgeräusche. Mit dem zunehmenden Einsatz von ASR-Systemen im Alltag wird die Robustheit der ASR wichtiger denn je. Anhand der Verarbeitungsstufen eines ASR-Systems lassen sich die Ansätze zur Erhöhung der ASR-Robustheit in drei Gruppen einteilen: (i) Back-End-Techniken, d.h. Anpassung der Netzwerkarchitektur und Multi-Condition-Training; (ii) Front-End-Techniken, wie z.B. die Schätzung von Rauschmasken, Rauschentfernung und Extraktion robuster akustischer Merkmale; (iii) Gemeinsames Training von Front- und Back-End. Diese Arbeit folgt dementsprechend diesen drei Forschungsschwerpunkten.

Im ersten Teil dieser Arbeit wird ein Ansatz zur Optimierung von Hidden Markov Modell (HMM)-Strukturen diskutiert, der als Deep Neural Fenonic Baseform Growing bezeichnet wird. Dieses Verfahren mit geringer Rechenkomplexität optimiert die HMM-Struktur für jedes Phone ohne externe Annahmen über die Anzahl der Zustände und mögliche Übergänge. Experimentelle Ergebnisse mit den Sprachkorpora TIMIT und TEDliumv2 zeigen, dass sich die Genauigkeit bei sowohl Monophon- als auch Triphon-basierten HMM-Strukturen erheblich verbessert. Zusätzlich erhöht sich die Robustheit in verrauschten Umgebungen. Darüber hinaus lassen sich durch die Anwendung der vorgestellten HMM-Strukturen modernste Spracherkennungssysteme verbessern, bei gleichzeitiger Reduzierung der Parameteranzahl.

Der zweite Teil präsentiert zwei optimierte Architekturen von Generativen Neuronalen Netzwerken (Generative Adversarial Networks, GANs) zur Sprachverbesserung. Zunächst werden in dieser Arbeit GANs mit drei verschiedenen Selbstaufmerksamkeitsmechanismen untersucht. Die vorgeschlagenen Systeme verbessern konsistent die Genauigkeit. Außerdem werden in dieser Studie die Sinc-Faltung und die Sprachverbesserungs-GANs sinnvoll miteinander kombiniert, was zu einem anpassbaren und interpretierbaren System mit einer kleinen Anzahl von Gewichten führt. Darüber hinaus werden eine Reihe von Techniken zur Datenerweiterung im Zeitbereich eingesetzt, um die Leistung des Systems und seine Generalisierungsfähigkeiten weiter zu verbessern. Die experimentellen

Ergebnisse zeigen, dass das vorgeschlagene System verschiedene konkurrierende Modelle übertrifft, insbesondere bei der Wahrnehmungsqualität und der Sprachverständlichkeit.

Der dritte Teil konzentriert sich auf einen auf Selbstaufmerksamkeit basierenden gemeinsamen Trainingsansatz. Dieser verknüpft ein Sprachverbesserung Front-End und ein ASR Back-End miteinander, welche dann zusammen als Netzwerk trainiert werden, um die Störgeräuschrobustheit des ASR-Systems zu erhöhen. Darüber hinaus übernimmt ein diskriminatives Netzwerk die globale Steuerung des gegnerischen gemeinsamen Trainings, was zur Erzeugung geeigneterer Merkmale für die Spracherkennungskomponente führt und die Einschränkungen des getrennten Trainings der Front-End und Back-End Komponenten kompensiert. Mithilfe systematischer Experimente wird demonstriert, dass die vorgeschlagene Lösung andere konkurrierende Lösungen deutlich übertrifft, insbesondere in anspruchsvollen Umgebungen.

Contents

List of Acronyms

AM	Acoustic Model
ANN	Artificial Neural Network
AR	Auto-Regressive
ASR	Automatic Speech Recognition
BLSTM	Bidirection Long Short-Term Memory
CBAK	MOS Prediction of the Intrusiveness of Background Noises
CER	Character Error Rate
CNN	Convolution Neural Network
COVL	MOS Prediction of the Overall Effect
CSIG	MOS Prediction of the Signal Distortion Attending only to the Speech Signal
CTC	Connectionist Temporal Classification
D	Discriminator
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
DNNVQ	Deep Neural Network Vector Quantizer
DOA	Direction-of-Arrival
DS	Delay-and-Sum
DSEGAN	Deep Speech Enhancement Generative Adversarial Network
EM	Expectation-Maximization
FFN	Feed-Forward Network

G	Generator
GAN	Generative Adversarial Network
GLU	Gated Linear Unit
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IF	Instantaneous Frequency
ISEGAN	Iterated Speech Enhancement Generative Adversarial Network
LM	Language Model
LP	Linear Prediction
LSGAN	Least-Squares Generative Adversarial Network
LSTM	Long Short-Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MCT	Multi-Conditional Training
MF-PLP	Mel Frequency Perceptual Linear Prediction Cepstral Coefficient
MFCC	Mel Frequency Cepstral Coefficient
MHSA	Multi-Head Self-Attention Module
MLP	Multilayer Perceptron
MMI	Maximum Mutual Information
MOS	Mean Opinion Score
NFBG	Neural Fenonic Baseform Growing
NN	Neural Network
PASE	Problem-Agnostic Speech Encoder
PDF	Probability Density Function
PESQ	Perceptual Evaluation of Speech Quality
PLDA	Probabilistic Linear Discriminant Analysis
PLP	Perceptual Linear Prediction Cepstral Coefficient
PReLUs	Parametric Rectified Linear Units
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SA_ASR	Self-Attention Automatic Speech Recognition
SASEGAN	Self-Attention Speech Enhancement Generative Adversarial Network
SE	Speech Enhancement
SEGAN	Speech Enhancement Generative Adversarial Network

SNR	Signal-to-Noise Ratio
SOTA	state-of-the-art
SSNR	Segmental SNR
STFT	Short-Time Fourier Transform
STOI	Short-Time Objective Intelligibility
t-SNE	t-distributed Stochastic Neighbor Embedding
TDNN	Time-Delay Neural Network
TTUR	Two-Timescale Update Rule
WER	Word Error Rate

List of Symbols

General Notations

$\approx$	approximately equal to
$\propto$	proportional to
x	scalar quantity (lower plain letter)
$\boldsymbol{x}$	vector quantity (lowercase bold letter)
$\boldsymbol{X}$	matrix (uppercase bold letter)
$\mathcal{X}$	distributions or sets (calligraphic captial leters)
$\boldsymbol{X}^T$	transpose of matrix $\boldsymbol{X}$
$(\cdot)^{-1}$	inverse of a square matrix

Functions

$f_\theta(\cdot)$	a objective function or a mapping function
θ	parameters of the objective function or the mapping function
$\arg\max_x f(x)$	value of x that maximises $f(x)$
$\delta(\cdot)$	Kronecker delta

Probability Distributions

$p(\cdot)$	probability density function
$p(\cdot\|\cdot)$	conditional probability density
$P(\cdot)$	probability mass function

$P(\cdot\|\cdot)$	conditional probability mass distribution
$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$	Guassian multivariate distributions of $\boldsymbol{x}$
$\mathcal{L}(\cdot)$	(log-) likelihood functions
$H(X)$	the entropy of X
$H(X\|Y)$	the entropy of X conditioned on Y
$I(X;Y)$	the mutual information of X and Y

HMM Parameters

$\mathcal{W}$	hypothesis, or word sequence
$\hat{\mathcal{W}}$	the most probable word sequence
$\boldsymbol{o}_t$	observation vector at time t
$\mathcal{O}$	observation sequence $\mathcal{O} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$
$\boldsymbol{A}$	state transition probability matrix $\boldsymbol{A}$
a_{ij}	discrete state transition probability from state i to state j
$\boldsymbol{B}$	state output probability distribution $\boldsymbol{B}$
$b_j(\boldsymbol{o})$	output probability distribution at state j
s_t	state at time t
$\boldsymbol{\mu}$	the mean vector of a Guassian distribution
$\boldsymbol{\Sigma}$	the covariance matrix of a Guassian distribution

ANN Parameters

$\boldsymbol{x}$	the input of a neural network
$\boldsymbol{y}$	the output of a neural network
$\boldsymbol{W}$	the weight matrix of an ANN layer
$\boldsymbol{b}$	the bias vector of an ANN layer
$\boldsymbol{h}_t$	hidden sequence at time t
$\overrightarrow{\boldsymbol{h}_t}$	forward hidden sequence at time t
$\overleftarrow{\boldsymbol{h}_t}$	backward hidden sequence at time t
ϕ	the partial best-path score
α_{LM}	the LM scaling factor
β_{pro}	the pronunciation probability scalar
γ_{in}	the insertion penalty

End-to-End Model Parameters

$\mathcal{V}$	the vocabulary
$\mathcal{V}^T$	the collection of all sequences of length T defined on the vocabulary
$\boldsymbol{\nu}$	a single path in the vocabulary
$\boldsymbol{a}_n$	the attention weight vector for output location n
$\boldsymbol{c}_n$	the context vector for output location n
$\boldsymbol{s}_n$	the state vector for output location n

Back-End Techniques

y^*	the ground truth of the input vector $\boldsymbol{x}$
N_{K}	the dimension of the ground-truth label space
$\boldsymbol{m}$	the output vector of DNNVQ
$\hat{m}$	the index of the maximum value in the DNNVQ output layer
N_{clu}	the dimension of the output layer of DNNVQ
ε	a small constant
N_{b}	batch size
$\mathcal{F}$	the alphabet of fenones
$\mathcal{F}^*$	the set of all finite-length strings constructed by concatenating elements of $\mathcal{F}$
$\mathcal{G}$	the set of N_{g} monophones
$\mathcal{K}_i$	the set of all segments affiliated to the corresponding phone g_i
f	fenone
$\mathcal{A}$	the output matrix of the same frames of all padded segments of a monophone

Front-End Techniques

$\boldsymbol{Q}$	query inputs of the self-attention layer
$\boldsymbol{K}$	key inputs of the self-attention layer
$\hat{\boldsymbol{K}}$	key inputs with locality modelling

$\boldsymbol{V}$	value inputs of the self-attention layer
$\hat{\boldsymbol{V}}$	value inputs with locality modelling
$\boldsymbol{x}^*$	clean raw inputs of the speech enhancement network
$\tilde{\boldsymbol{x}}$	noisy raw inputs of the speech enhancement network
$\hat{\boldsymbol{x}}$	enhanced speech signals by the speech enhancement network
$\boldsymbol{x}_c$	the extra input of conditioned GANs
$\boldsymbol{z}$	a latent variable
$\boldsymbol{c}$	the encoding vector
$\boldsymbol{F}$	the input feature map of the self-attention layer
$\boldsymbol{F'}$	the output feature map of the self-attention layer
$\bar{\boldsymbol{A}}$	the attention map
$\hat{\boldsymbol{A}}$	the attention map with locality modeling
$\bar{\boldsymbol{O}}$	the output of the attention layer
$\hat{\boldsymbol{O}}$	the output of the attention layer with locality modeling
β	the weight of $\bar{\boldsymbol{O}}$
L	the time dimension
C	the number of channels
b, p	factors for memory efficiency
κ, γ	factors for attention augmented convolutional SEGAN
$\bar{g}(\cdot)$	a filter-bank consisting of rectangular bandpass filters
$\bar{G}[\cdot]$	a generic bandpass filter in the frequency domain
$\bar{f}_1, \bar{f}_2$	the learnable low and high cutoff frequencies
$rect(\cdot)$	the rectangular function in the magnitude frequency domain
$\bar{\boldsymbol{h}}_{\bar{l}}$	the input of the $\bar{l}$th decoder layer
$\boldsymbol{\vartheta}_{\bar{l}}$	a learnable vector of the $\bar{l}$th decoder layer
$\odot$	an element-wise product along channels

Joint Training Techniques

$\hat{\boldsymbol{u}}$	the normalized log FBank features
h	the head numbers
pos	the position of the input
N_{e}	the number of encoder blocks of Transformer
N_{d}	the number of decoder blocks of Transformer
$\mathcal{L}_{asr}$	the loss of the ASR module
$\mathcal{L}_{enh}$	the loss of the generator of SEGAN
$\mathcal{L}_{gan}$	the loss of the discriminator of SEGAN
ζ, ρ	two hyper-parameters weighting the magnitude of the enhancement loss and adversarial loss
lr	the learning rate
τ	the step number of Adam optimizer
ϖ	a tunable scalar of Adam optimizer
η	the length penalty

List of Figures

List of Tables

"We must have perseverance and above all confidence in ourselves. We must believe that we are gifted for something and that this thing must be attained."

– Marie Salomea Skłodowska Curie

1 Introduction

1.1 Motivation

Over the past decades, the debut of Deep Neural Networks (DNNs) [49] triggers a breakthrough in the field of automatic speech recognition (ASR). Various hybrid systems, e.g., DNN-Hidden Markov Model (HMM) systems [99], Convolution Neural Network (CNN)-HMM systems [238], and Recurrent Neural Network (RNN)-HMM systems [80], occupy the predominant status and even achieve human parity performance [243, 305]. Hybrid systems consist of two main components: the acoustic model (AM) and the language model (LM), which are built and trained separately. Albeit hybrid systems achieve satisfactory performance, their process on speech signals is complex. In this light, another round of revolution in deep learning triggers ASR architectures' diversification into a completely new approach, specifically end-to-end models, where AMs and LMs are jointly optimized. Currently, two prominent modeling techniques are widely applied to the end-to-end speech recognition systems: connectionist temporal classification (CTC) technique [78] and attention mechanism [12, 38]. Both techniques map speech feature sequences to text label sequences directly, namely sequence-to-sequence tasks. So far, end-to-end models have achieved competitive performance [79, 43, 278, 42].

Hybrid systems have been demonstrated to process remarkable robustness to environment distortions. Seltzer et al. [253] investigate the noise robustness of DNN-based AMs and find that they can match state-of-the-art (SOTA) performance on the Aurora-4 [313] task without any explicit noise compensation or model adaption. Weng et al. [299] investigate the RNNs with deep architecture in hybrid systems for robust ASR and obtain the SOTA performance on the second CHiME challenge [281] without front-end preprocessing, speaker adaptive training or multiple decoding passes. Geiger et al. [70] utilize long short-term memory (LSTM) [239] as an AM for a robust speech recognition system and experimental results show that the bidirection-LSTM (BLSTM) in the hybrid module outperforms the best entry to the original CHiME challenge. In addition, a further improvement is obtained by combining different LSTM AMs. Consequently, hybrid systems still draw considerable attention [91, 203, 146, 130, 330, 292].

End-to-end modeling techniques significantly simplify the model building procedure and have made considerable breakthroughs in speech recognition tasks, and thus embrace

the bright prospect and the fast development. However, they still suffer drastic performance degradation in adverse environments. Since end-to-end models learn the acoustic and linguistic information simultaneously, the perturbation in the acoustic module could be easily transmitted to the linguistic module. Furthermore, the self-attention system predicts the next output symbol conditioned on the full sequence of the previous predictions. Once a mistake occurs in one estimation step due to the noise interference, all the subsequent steps will be disturbed. From these perspectives, end-to-end frameworks are more vulnerable to ambient noises, and thus improving the robustness of end-to-end systems remains to be a big challenge for their broader application in realistic situations. So far, a lot of research efforts have been made in this field [120, 157, 135, 115, 102, 263, 63, 325].

1.2 Related Work

According to the addressed number of channels, robust speech recognition can be categorized into single-channel [6] and multiple-channel [97, 271] techniques. They adopt different approaches to suppress ambient noises. For single-channel techniques, the widely adopted method is adding the speech enhancement component at the front-end of ASR, including spectral subtraction [28], Wiener filtering [245], and deep learning based speech enhancement methods [283, 173, 293, 193, 62, 65, 314]. Pascual et al. [193] apply generative adversarial networks into speech enhancement for the first time. Enhanced samples that are evaluated on both objective and subjective metrics confirm the viability and effectiveness of the innovative application. Yin et al. [314] propose a phase-and-harmonics-aware DNN. Unlike previous methods which directly use a complex ideal ratio mask to supervise the DNN learning, they design a two-stream network, where amplitude stream and phase stream are dedicated to the amplitude and phase prediction. In addition, they propose frequency transformation blocks to catch long-range correlations in the frequency field. Their proposed system outperforms previous methods substantially on four metrics on Voice Bank + DEMAND dataset [273]. For multiple-channel techniques, acoustical beamforming is the main approach, which transfers the outputs of microphone arrays to a single-channel signal. Thereafter, the converted outputs can be operated by back-end techniques for the single-channel speech. During converting, the speech from the target direction is accentuated and the audio signals from other arrays are attenuated. Moreover, the amplified target speech is futher enhanced by a microphone array post-filter [154]. Medennikov et al. [156] present the Speech Technology Center systems for the CHiME-6 challenge aimed at multi-microphone multi-speaker speech recognition and diarization in a dinner party scenario. They utilize data augmentation approaches, multi-stream/multi-stride self-attention layers in AM, and a novel Target-Speaker Voice Activity Detection approach to achieve state-of-the-art results in the complex multi-channel dinner party scenario. Arora et al. [7] explore multi-array processing techniques at each stage of the pipeline, such as multi-array-guided source separation for enhancement and AM training data, posterior fusion for speech activity detection, probabilistic linear discriminant analysis (PLDA) score fusion for diarization,

and lattice combination for ASR. As the result, they achieve an improvement of 10.8 % and 10.4 % absolute, over the challenge baselines for the respective tracks.

According to the spectral distribution, the noises can be basically classified as stationary noises (constant with respect to time, i.e., the white noise and the Sinusoid noise) and non-stationary noises (inconstant with time, i.e., transient sound events, babble, and music). Comparatively speaking, detecting and tackling non-stationary noises are more challenging in practice [17, 16, 123, 282, 311, 315], sparkling plenty of research efforts, as showcased in various challenges themed on robust speech recognition (e.g., REVERB [123] and CHiME [18, 297]).

According to the processing stage of the whole ASR framework, previous techniques aiming at robust ASR can be grouped into front-end, back-end, or joint-training (joint front- and back-ends training) approaches.

Typical application scenarios of the front-end techniques can be speech enhancement, source separation, and feature enhancement. Based on the output of the network, front-end techniques can be further categorised into (i) mapping-based methods [184, 228, 324], where network outputs are the representation, straightforwardly extracted from clean speech, or (ii) masking-based methods [46], where outputs are a mask calculated between clean and noisy speech. Rethage et al. [228] propose an end-to-end learning method for speech denoising based on Wavenet [184]. The proposed model adaptation retains Wavenet's powerful acoustic modeling capabilities, while significantly reducing its time-complexity by eliminating its auto-regressive nature. Specifically, the model makes use of non-causal, dilated convolutions and predicts target fields instead of a single target sample. The discriminative adaptation of the their proposed model learns in a supervised fashion via minimizing a regression loss. These modifications make the model highly parallelizable during both training and inference. Both quantitative and qualitative evaluations indicate that the proposed method is preferable to Wiener filtering. Cui and Bao [46] propose a novel weighted mean square error to improve the DNN-based mask approximation method for speech enhancement, in which the weighting is closely related to the power exponent about noisy spectrum amplitude base. Also, the experimental results show that the outstanding weighting is the noisy spectrum base with the power exponent 1 for the phase-unaware masking and results in better harmonic structure restoration. The objective function with the weighted mean square error on the noisy spectrum amplitude can averagely improve 0.1 on the test of perceptual evaluation of speech quality (PESQ) and 1.7 % on the test of short-time objective intelligibility (STOI) compared with the mean square error based mask approximation methods.

The back-end techniques let the neural networks autonomously detect the relationship between the noisy observations and the phonetic targets, in stead of operating on the observed noisy speech. Typical algorithms of back-end techniques include multi-condition training [150], model adaptation [267], noise-aware training [161], and multi-task training [224]. Malek and Zdansky [150] aim for careful selection of a limited number of acoustic conditions that are highly relevant to the target environment. In this manner, they keep the computational requirements feasible, while retaining the improved accuracy of the augmented models. For the experiment, they analyze two augmentation scenarios and draw conclusions regarding suitable setup choices. Tan et al. [267] propose a more

advanced model referred to as the very deep convolutional residual network. To alleviate the mismatch between the training and testing conditions, model adaptation and adaptive training are developed. Their proposed network achieves a new milestone of 5.67 % word error rate (WER) on Aurora-4 [313]. Mirsamadi and Hansen [161] propose a novel strategy for training neural network AMs based on adversarial training which makes use of environment labels during training. By adjusting the parameters of the initial layers of the network adversarially with respect to a domain classifier trained to recognize the recording environments, they enforce better invariance to the diversity of recording conditions. The proposed multi-domain adversarial training achieves a relative character error rate (CER) reduction of 25.4 % with respect to a clean-trained baseline. Ravanelli et al. [224] design a problem-agnostic speech encoder for robust speech recognition in noisy and reverberant environments, which contains a convolutional encoder and subsequent multiple neural networks. Shortly after, an optimised encoder equipped with an efficient combination of recurrent and convolutional networks are proposed, for better learning short- and long-term speech dynamics. Experimental results indicate that the proposed system learns transferable representations efficiently, and thus suitable for highly mismatched acoustic conditions.

The key idea of joint training techniques is integrating a speech enhancement front-end and an acoustic model into a larger neural network, and jointly adjusting the weights of each module. Wang and Wang [295] design a joint training framework for speech separation and recognition. To further improve the robustness, they add more noise- and reverberation-robust features for acoustic modeling. The resultant jointly-trained multi-stream represents the best performance on the test set of the reverberant and noisy CHiME-2 dataset (task-2) and a 22.75 % error reduction over the best existing method. Gao et al. [67] contribute to this line by employing a hybrid DNN architecture to jointly train DNNs for both feature mapping and acoustic modeling. The input of the hybrid DNN is the original noisy speech feature vectors. The proposed system reports the best published result on the Aurora-4 task without using any adaptation techniques. Ravanelli et al. [223] present an joint-training approach, containing a speech enhancement and a speech recognition DNN, coupled with batch normalization in order to make the whole framework less sensitive to exterior changes. With batch normalization, the proposed joint architecture can be effectively trained independently of any pre-training steps.

1.3 Contributions

Since multi-channel robust speech recognition falls down to single-channel problems eventually (as described in Section 1.2), this thesis concentrates on the single-channel robust speech recognition. It takes stationary and non-stationary noises into its research scope as both are ubiquitous in our daily life. Furthermore, this thesis targets the aforementioned three processing techniques (front-end, back-end, and joint-training techniques) of the whole ASR framework by trying to achieve the following objectives:

1. HMM structure construction with deep neural fenonic baseform growing [13†, 9†, 7†].

This thesis proposes an innovative, concise, data-driven, and deep-learning-based method to customize the HMM topology for every phone. The proposed algorithm allows the data to reveal their dynamic structure according to the training data without external assumptions and with a low computational cost. Different from previous studies, the proposed method discovers the potential information of the data and contains the model complexity simultaneously, avoiding excessive growth of the number of states. The derived data-driven state tying leads to optimized robustness of the HMM structure in adverse environments, and improved recognition rates for existing SOTA systems and their parameter scales are shrunk considerably.

2. Lightweight speech enhancement GANs with self-attention mechanism and Sinc convolution [$6^{\dagger}$, $8^{\dagger}$].

   This thesis presents two optimized architectures of speech enhancement GANs (SEGANs) as the front-end of ASR systems. Firstly, this study integrates the self-attention mechanism with SEGAN to improve its flexibility of both long-range and local dependency modeling for speech enhancement in three methods, namely applying the stand-alone self-attention layer, modeling locality on the stand-alone self-attention layer, and coupling the self-attention layer with the (de)convolutional layer. Systematic experiment results reveal that equipped with self-attention mechanism, the proposed systems deliver consistent performance improvements. Sinc convolution for speech enhancement is still an under-explored research direction. Therefore, this study proposes to bridge this gap by usefully merging the Sinc convolution and SEGAN. In this light, this study transfers the success achieved by the Sinc convolution in the field of speech and speaker recognition to the field of end-to-end speech enhancement. In addition, it optimizes the SEGAN architecture and enhance the original Sinc convolution layer to fit the advanced SEGAN architecture. Besides, this study applies a series of data augmentation techniques on raw speech waveforms directly to further improve the system performance, and the learnt filters of the Sinc convolution layer is also analyzed.

3. Adversarial joint training with self-attention mechanism [$4^{\dagger}$, $5^{\dagger}$].

   This thesis presents an adversarial joint training framework with the self-attention mechanism to boost the noise robustness of the ASR system. Generally, it consists of a self-attention speech enhancement generative adversarial networks and a self-attention end-to-end ASR model. To the best of the author's knowledge, this is the first joint training scheme that benefits from the advantages of both the self-attention mechanism and adversarial training. There are two advantages which are worth noting in this proposed framework. One is that it benefits from the advancement of both self-attention mechanism and GANs; while the other is that the discriminator of GAN plays the role of the global discriminant network for the adversarial joint training, which guides the enhancement front-end to capture more compatible structures for the subsequent ASR module and thereby offsets

the limitation of the separate training and handcrafted loss functions. With the adversarial joint optimization, the proposed framework is expected to learn more robust representations suitable for the ASR task. As achievement, remarkable results have been obtained.

1.4 Structure of the Thesis

The rest of this thesis is organised as follows: An overview of the general framework of automatic speech recognition is given in Chapter 2. Thereafter, Chapter 3 elaborates on the optimization on the ASR back-end, Chapter 4 presents two novel architectures of the speech enhancement front-end, and Chapter 5 discusses techniques of jointly training the whole framework. Eventually, Chapter 6 gives a summary of the current work and an outlook for the future work.

Chapter 2 presents a review of the general framework of automatic speech recognition. It starts with the front-end processing, followed by HMM-based and NN-based acoustic modeling. Afterwards, the lexicon, the language modeling, decoding, and the evaluation criteria are introduced. In the end, the structure of end-to-end models are discussed, including CTC-based end-to-end models, RNN-transducer end-to-end models, and attention-based end-to-end models.

Chapter 3 proposes a novel approach for constructing HMM structure for robust speech recognition, named deep neural fenonic baseform growing. To begin with, the deep neural network vector quantizer is introduced. Next, steps of deep neural fenonic baseform growing are introduced, containing segment lengths padding, dynamic baseform generation, and elementary Markov model for fenones. In the end, systematic experiments are exerted to validate the efficacy of the proposed approach in monophone systems, systems with context-dependent inputs, triphone systems, and in adverse environments. Additionally, the performance comparison between the proposed approach and other advanced models is also presented.

Chapter 4 presents two lightweight architectures of the speech enhancement front-end. One is the speech enhancement GAN augmented with self-attention mechanism; and the other is the speech enhancement GAN using Sinc convolution. Detailed implementations of these two proposed architectures are demonstrated in Section 4.1 and Section 4.2, separately. A brief summary is also provided at the end of each application.

Chapter 5 discusses the joint training strategy of the whole ASR scheme. At the beginning, it presents an overview of the whole joint training framework, including a self-attention based speech enhancement front-end, a self-attention based ASR back-end (realized by Transformer [54] or Conformer [82] architecture), and the joint training strategy. Then it emphasizes the role of the discriminant component of the framework, which acts as the local guide of the speech enhancement front-end, and the global guide of the joint training, simultaneously. Lastly, methodically designed experiments validate the superiority of the proposed joint training strategy over other competitive baselines.

Chapter 6 summarizes all works introduced in this thesis followed by various potential directions for future research.

"Science demands from a man all his life. If you had two lives that would not be enough for you. Be passionate in your work and in your searching."

— Ivan Petrovich Pavlov

2

General Framework of Automatic Speech Recognition

This chapter will introduce the general framework of ASR systems, including hybrid systems and end-to-end systems. Various speech signal processing techniques are discussed, including speech signal pre-processing, feature extraction, acoustic modeling, language modeling, and decoding approaches.

2.1 Overview

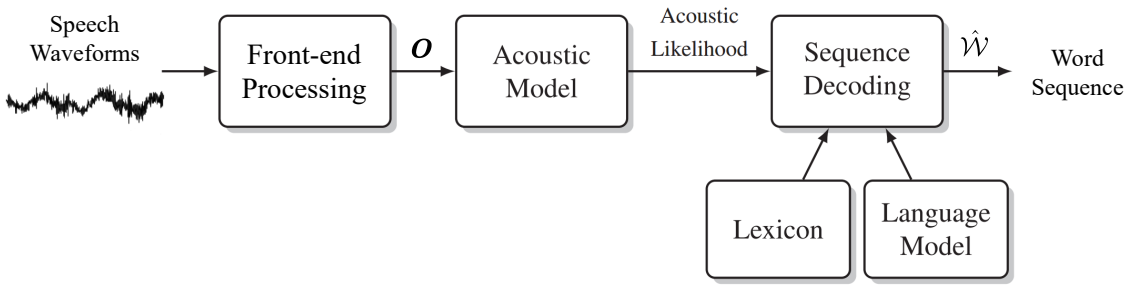


Figure 2.1: General framework of an ASR system [186].

The goal of an ASR system is to output the word sequence with the highest probability given the input audio signal [321]. To achieve this goal, a general speech recognition framework usually involves three steps: front-end processing, acoustic modeling and sequence decoding [186], as illustrated in Fig. 2.1. Front-end processing usually starts with feature extraction, where acoustic feature observations, or vectors, are extracted from incoming speech waveforms. Extracted features should be representative and compact, conveying sufficient information to the subsequent recognition module. Thereafter, AMs model a statistical relationship, i.e., the acoustic likelihood, between the feature vector sequence and the linguistic units, e.g., phonemes or phones. Lastly, the sequence modeling

takes the acoustic likelihood as inputs and outputs the most likely word sequence, where lexicons and language models are used. The lexicon, namely the dictionary, is commonly employed in large vocabulary continuous speech recognition (LVCSR) systems to bridge the sub-word units and the words existed in the LM. LMs include the prior knowledge about the syntactic and semantic information of the output word sequences. Broadly, sequence modeling can be expressed as:

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}} P(\mathcal{W}|O), \tag{2.1}$$

where $\mathcal{W}$ represents all probable word sequences given the observation sequence $\mathcal{O} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$, and $\hat{\mathcal{W}}$ denotes the most probable word sequence. Following Bayes' theorem, the posterior probability in Eq. 2.1 can be expressed as a conditional probability of the acoustic observations given the word sequence $p(O|\mathcal{W})$, multiplied by a prior probability of the word sequence $P(\mathcal{W})$, and normalized by the marginal likelihood of observation sequences $p(O)$ as

$$\begin{aligned} \hat{\mathcal{W}} &= \arg\max_{\mathcal{W}} \frac{p(O|\mathcal{W})P(\mathcal{W})}{p(O)} \\ &= \arg\max_{\mathcal{W}} p(O|\mathcal{W})P(\mathcal{W}). \end{aligned} \tag{2.2}$$

$p(O|\mathcal{W})$ is calculated by the acoustic model and $P(\mathcal{W})$ is modeled by the language model. It is worth noting that the marginal probability $p(O)$ (the denominator) can be neglected for being an constant.

2.2 Front-End Processing

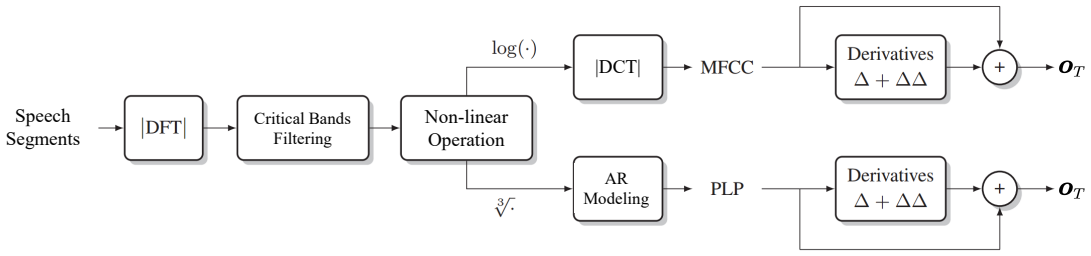


Figure 2.2: The diagram of Mel Frequency Cepstral Coefficient (MFCC) and Perceptual Linear Prediction Cepstral Coefficient (PLP) feature extraction [186]. |DFT| denotes the magnitude of the discrete Fourier transform, |DCT| denotes the magnitude of the discrete cosine transform, AR modeling stands for auto-regressive modeling, and Δ and $\Delta\Delta$ denote the first and second order derivatives across time, respectively.

Front-end processing is always the first operation on incoming waveforms in a common speech recognition system. This stage mainly consists of two steps: pre-processing (e.g., detecting the target speech waveforms, pre-emphasis, and framing) and feature extraction.

Speech signals are generally non-stationary signals as the statistical characteristics of speech signals vary over time for a series of reasons such as the individual difference of the vocal tract, different emotional state of even the same speaker, the homophony, etc. Speech coding techniques in telephony have shown that speech can be processed as short segments, where speech waveforms can be considered as short-term stationary signals. One explanation for this hypothesis is that studies have shown that short-term speech signals can be considered as output of a linear time-invariant vocal tract filter excited by periodic or aperiodic vibration of vocal cords [216]. Moreover, the short-term speech can be transformed and transmitted in the channel, and reconstructed at the end of the receiver while keeping the intelligibility or message intact [216]. Particularly, speech intelligibility can be preserved by preserving the envelope structure of the short-term spectrum of speech signals, which characterizes the vocal tract system [247]. Based on these aspects, extracting a representative and compact feature, which differentiates the speech sounds, for the short-term speech signal is the next step. In this light, Mel Frequency Cepstral Coefficient (MFCC) [51] and Perceptual Linear Prediction Cepstral Coefficient (PLP) [95] are designed.

For the feature extraction of both MFCC and PLP, the first stage is frame division. A window function, which is usually 25-30 ms long, is applied to the incoming audio stream with a typical 10 ms frame rate. As the result, each frame usually lasts for 25-30 ms with a 10 ms frame shift. Framed speech can be considered as stationary signals as the vocal tract is assumed to be relatively constant in such short terms. The high frequencies of the windowed speech is amplified by a first-order high-pass filter in the formant structure. And the magnitude spectrum is obtained by applying the short time Fourier transform on the overlapping speech frames. A perceptional evidence indicates that human listeners tend to judge larger intervals to produce equal pitch increments [51] at a higher frequency, so the normal frequency needs to be warped to a perceptually motivated frequency scale. In this light, the Mel-frequency scale is designed. In the Mel-frequency scale, the normal frequency scale f_{Hz} is warped to a perceived pitch f_{mel} by:

$$f_{\text{mel}} = 1127 \log(1 + \frac{f_{\text{Hz}}}{700}). \tag{2.3}$$

After warping, a series of triangular band-passed filters, which are spaced in the Mel-frequency scale linearly, are deployed to convolve the linear spectrum. Resultantly, a set of linear filter bank coefficients are obtained. Then the dynamic range of these coefficients are compressed by a logarithmic transform, and thus log Mel-spectral filterbank coefficients are acquired. However, these filterbank coefficients are highly correlated. As the commonly-used ASR model, Gaussian Mixture Model (GMM)-HMM systems contain diagonal covariance matrices, which cannot be well represented by highly-correlated coefficients. To solve this problem, a truncated Discrete Cosine Transform (DCT) is applied to decorrelate the feature, and produce the cepstral coefficients. Importantly, the high-order cepstral coefficients are normally discarded, and the 0th cepstral coefficient is replaced by a normalized log energy.

PLP is another popular feature representation based on the short-time spectrum analysis. Different from MFCC, the linear frequency of the power or magnitude spectrum

of PLP is wrapped into another perceptually motivated scale, the Bark frequency scale, via

$$f_{\text{bark}} = 6\log([\frac{f_{\text{Hz}}}{600} + 1]^{0.5} + \frac{f_{\text{Hz}}}{600}). \tag{2.4}$$

In the Bark frequency field, the power or magnitude spectrum is filtered out by equally-deployed band filters. Following the equal-loudness and intensity-loudness power law, the output of these band filters are nonlinearly transformed. Linear prediction (LP) analysis is applied to the transformed outputs and the resultant LP coefficients are further converted to cepstral coefficients. Woodland et al. [303] shows a modified form of PLP features, where an equal-loudness curve is applied on the Mel filterbank outputs and then a cubic root is utilized to further compress the scaled spectrum. Likewise, the LP analysis is applied to the compressed spectrum, and the resultant LP coefficients are further converted to cepstral coefficients. Eventually, the MF-PLP features are obtained. Woodland et al. [303] indicates that the MF-PLP features is superior to the standard PLP features.

Fig. 2.2 illustrates the diagram of MFCC or PLP feature extraction, including (i) discrete Fourier transform; (ii) filtering the magnitude spectrum based on the speech perception knowledge via critical bands analysis; (iii) a nonlinear operation; and (iiii) decorrelating features via DCT. These first four steps only model the frame-level spectrum information, but speech signals possess intrinsic temporal characteristics. Therefore, the first- and second-order derivatives of the static features are employed to model the dynamic information in the speech waveforms eventually [66].

2.3 Hidden Markov Model based Acoustic Modeling

Section 2.2 concentrates on the feature extraction of speech waveforms. This section will emphasize the acoustic modeling for the speech recognition task, where AMs are utilized to calculate the acoustic likelihood of a sequence of feature observations. HMMs are the most classic AM in the field of ASR [212]. This section will introduce basic concepts of HMMs and their applications to ASR.

It is assumed that in the HMM-based AMs, the feature observation of each acoustic unit (e.g., phones, sub-phones, or words) is generated by a finite state machine. There is a hidden state for each time t, and following a certain transition probability, the current hidden state can jump to another state at time $t + 1$. Furthermore, according to a state-dependent output distribution, the current state also emits an observation vector at time t. When employing the finite state machine, two fundamental assumptions are adhered to:

- Quasi-stationary. This assumes that after segmenting the non-stationary observation vectors within certain acoustic units into frames, the frame-level vectors can be deemed as stationary observations.

- Conditional independence. This assumes that the generated feature vectors only depend on the current hidden state, where it is derived from, and is conditionally independent of the other states or feature vectors.

It is worth noting that for realistic speech signals, neither of these foregoing assumptions is true. However, HMMs have been demonstrated to be suitable for speech signal processing and have achieved a great success [243, 91, 203, 130, 330, 146, 125, 198, 40, 244].

2.3.1 Left-to-Right HMMs

The main parameters of an N-state HMM are:

- State transition probability matrix $\boldsymbol{A}$
  Each element a_{ij} of matrix $\boldsymbol{A}$ represents the transition probability jumping from state i at time t to state j at time $t+1$ as

$$(\boldsymbol{A})_{ji} = a_{ij} = P(s_{t+1} = j | s_t = i). \tag{2.5}$$

  Note that the transition probability a_{ij} is independent of the time index t. To be a valid probability distribution, the sum of each column of the state transition probability matrix $\boldsymbol{A}$ must satisfy

$$\sum_{j=1}^{N_j} a_{ij} = 1 \quad \forall i = 1, \cdots, N_i. \tag{2.6}$$

- State output probability distribution $\boldsymbol{B}$
  Each element $b_j(\boldsymbol{o})$ denotes the emitting probability given state j:

$$b_j(\boldsymbol{o}) = p(\boldsymbol{o} | s = j). \tag{2.7}$$

Since speech signals possess natural temporality, left-to-right HMMs are the most suitable topology for modeling this characteristic. Fig. 2.3 shows an example of a left-to-right HMM with 3 emitting states. Let $\mathcal{O} = (\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t, \cdots, \boldsymbol{o}_T)$ be an observation sequence generated by this 3-state left-to-right HMM. $\boldsymbol{o}_t$ denotes the observation vector at time t and T is the length of the vector sequence. Note that only emitting states (e.g., state 2-4 in Fig. 2.3) generate observation vectors. The entry state (state 1) and exit state (state 5) are non-emitting, and the utility of non-emitting states is to facilitate the construction of composite HMMs. At each time t, the state can jump to the next state or stay at the current state by the transition probability a_{ij}. Once an emitting state is reached, the observation vector is generated complying with the probability density $b_j(\boldsymbol{o})$. For each observation vector sequence $\boldsymbol{o}$ with the length T, there is a state sequence $\boldsymbol{s} = (s_1, \cdots, s_t, \cdots, s_T)$ with the same length, where s_t is the state at time t. However, only $\boldsymbol{o}_t$ can be observed, while s_t is hidden and needs to be inferred from the observations.

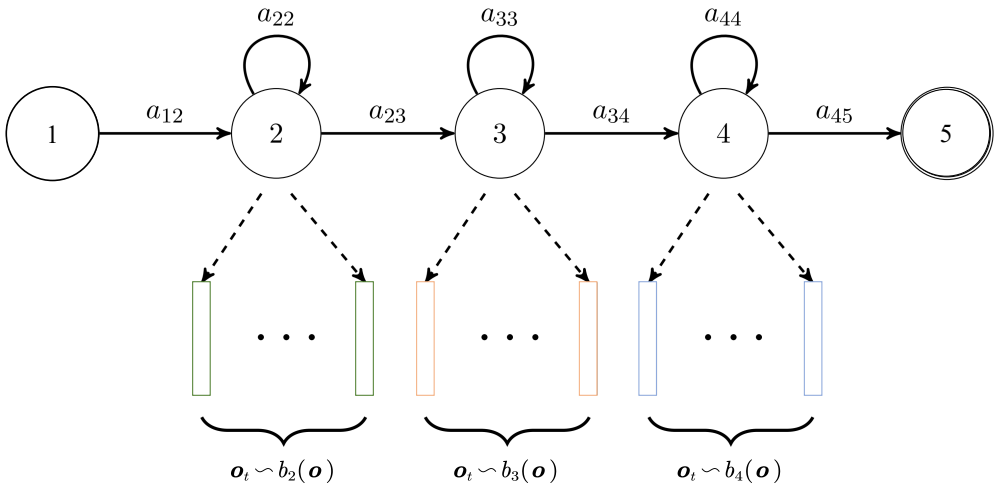


Figure 2.3: An example of the left-to-right HMM with 3 emitting states.

2.3.2 Acoustic Units

For some speech recognition tasks, e.g., the isolated word recognition, or keyword spotting, the amount of words in the vocabulary is normally less than 1 K. In this case, HMMs can be built for each word. However, for LVCSR, it is impossible to model every word with an individual HMM as the amount of words is usually over 10 K. Then HMMs are built upon sub-words. For example, the sub-word unit [th] affiliates with the word *thin*. When the recognition stage finishes, all sub-words are composed to words according to the mapping rules specified by a dictionary. The reason why words have to be decomposed to sub-words initially is that the quantity of sub-words is considerably less than that of the words. Let English be an example. In English, phones are often basic acoustic units for HMM modeling. There are around 40 to 60 phones compared to 20 K to 64 K words in typical state-of-the-art speech recognition systems.

Basically, there are two sorts of phone systems. One is the *monophone* system, where HMMs are built on context-independent phones. The other is the *triphone* system. *Triphones* are the most popular context-dependent acoustic units, which combine the left- and right-adjacent phones with the center phone. For instance, the triphone *f-iy+l* is [iy]-centric, where "-" stands for the left context and "+" denotes the right one. Since the pronunciation of the center phone is highly influenced by the adjacent phones, acoustic units based on context-dependent phones are normally employed in LVCSR [134].

However, employing context-dependent triphones in LVCSR brings back the quantity problem. For instance, a monophone system containing 40 phones can derive up to 100 K triphones. It is impossible to collect adequate training data for each individual triphone. Moreover, some triphones may occur rarely or even never exist in the training data. In this light, parameter tying techniques are necessary. The most popular parameter tying technique is state clustering [317], where all states are classified into different clusters and the state output distributions of the same cluster are shared. Fig. 2.4 demonstrates the state clustering of three HMMs. The upper part of Fig. 2.4 is the untied system,

each state possesses an individual output distribution, resulting in 9 output distributions to be estimated. The state clustering technique [317, 109] groups these 9 distributions into 4 clusters. As the result, the amount of model needed to build is reduced to be 4, which averts the problem of insufficient data for each state.

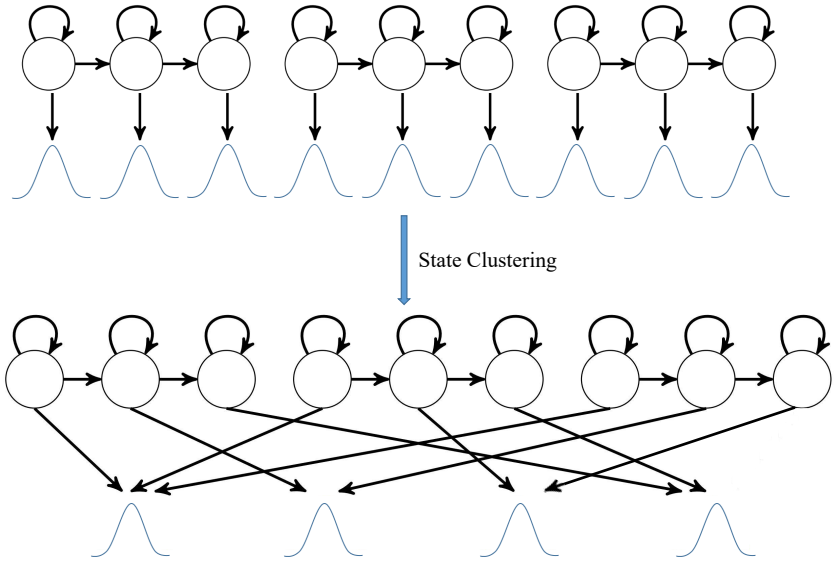


Figure 2.4: An example of state clustering within a three-HMM system. Before state clustering, each state has a unique state output distribution. Afterwards, grouped states share the same distribution.

Generally, there are two approaches that can be used for state clustering: bottom-up and top-down. The bottom-up approach merges the most similar distributions, and as the result, a new distribution emerges. The number of final clusters can be set previously, so when the amount of new distributions reach the set number, the state clustering stops. The main demerit of this approach is that it neglects unseen contexts with no occurrence in the training data. This issue can be solved by the top-down approach, e.g., the phonetic decision tree clustering approach. The top-down approach groups states into root nodes, and at each node, a phonetic question about contexts is inquired to split states affiliated to the current root into two subsidiary roots. Likewise, this process repeats until a certain number of nodes is reached. When an unseen triphone appears, the top-down approach can always find a proper leaf node for each state of this new triphone by selected phonetic questions. Resultantly, a new HMM is synthesized for this triphone. The top-down approach is the most widely-used state clustering approach in most SOTA ASR systems [107].

2.3.3 Acoustic Likelihood Estimation

Formally, Eq. 2.2 separates the sequence modeling into two independent models: the acoustic likelihood probability $p(O|\mathcal{W})$ and a prior probability of the word sequence $P(\mathcal{W})$. In the HMM framework, the acoustic likelihood $p(O|\mathcal{W})$ is estimated by

$$\begin{aligned} p(O|W) &= \sum_{S \in \mathcal{S}} p(O, S|W) \\ &= \sum_{S \in \mathcal{S}} p(O|S, W) P(S|W) \\ &= \sum_{S \in \mathcal{S}} p(O|S) P(S|W) \\ &\approx \max_{S \in \mathcal{S}} p(O|S) P(S|W) \\ &\approx \max_{S \in \mathcal{S}} \prod_{t=1}^{T} b_j(\boldsymbol{o}_t) a_{ij} \end{aligned} \quad . \tag{2.8}$$

Each element of the feature sequence $O = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t, \cdots, \boldsymbol{o}_T\}$ is assumed to be emitted by a state of the hidden state sequence $\boldsymbol{s} = \{s_1, \cdots, s_t, \cdots, s_T\} \in \mathcal{S}$. Each hidden state emits an observation following the state output probability distribution $b_j(\boldsymbol{o}) = p(\boldsymbol{o}|s = j)$ (Eq. 2.7). After applying Bayes' theorem as described in Eq. 2.2, Viterbi approximation is implemented in the forth step, where the sum over all possible state sequence is replaced by the state sequence with the highest probability. The fifth step arises from the HMM assumption (as described at the beginning of Section 2.3), which is that the acoustic observation $\boldsymbol{o}_t$ at time t depends only on the current state s_t, and the first order Markovian assumption, which is that the current state s_t depends only on the previous state s_{t-1}. a_{ij} is the transition probability from state i to state j.

The emission probabilities $b_j(\boldsymbol{o})$ are usually estimated by GMMs or Artificial Neural Networks (ANNs) [186].

In the GMM-HMM system, emission probabilities are estimated by a mixture of Gaussian distributions as

$$b_j(\boldsymbol{o}) = \sum_{n=1}^{N_{\text{GMM}}} c_{ij} \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \sum), \tag{2.9}$$

where N_{GMM} denotes the number of Gaussians, c_{ij} denotes the weight of a Gaussian distribution, which is

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \sum) = \frac{1}{(2\pi)^{d_o/2} |\sum|^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{o} - \boldsymbol{\mu})^T \sum^{-1} (\boldsymbol{o} - \boldsymbol{\mu})). \tag{2.10}$$

d_o denotes the dimension of $\boldsymbol{o}$. $\boldsymbol{\mu}$ and $\sum$ represent the mean vector and the covariance matrix, respectively.

In hybrid ANN-HMM systems [32], emission probabilities are estimated by ANNs, which will be discussed later in Section 2.4. ANNs estimate the class condition probabilities $P(s_t = j|\boldsymbol{o}_t)$ of each acoustic unit cluster $j \in \{1, \cdots, N\}$ given the feature

$\boldsymbol{o}_t$. Emission probabilities $b_j(\boldsymbol{o})$ of HMM states are obtained through Bayes' theorem: dividing ANN outputs by the state prior probability $P(s_t = j)$. The mathematical description is

$$b_j(\boldsymbol{o}) \propto \frac{p(\boldsymbol{o}_t|s_t = j)}{p(\boldsymbol{o}_t)} = \frac{P(s_t = j|\boldsymbol{o}_t)}{P(s_t = j)} \quad \forall j \in \{1, \cdots, N\}. \tag{2.11}$$

The state prior probability $P(s_t = j)$ is usually estimated by counting the occurrence of the state in the training data.

In both GMM-HMM systems and ANN-HMM systems, parameters are learnt by optimizing the expectation-maximization (EM) algorithm [215, 212], which is a maximum likelihood-based cost function. The parameters of HMMs are often estimated by Baum-Welch algorithm [20] (also termed as the forward-backward algorithm [215]) or the Viterbi EM algorithm [114]. In most cases, Viterbi EM is utilized to train ANNs. Viterbi EM algorithem maily comprises two iterative steps:

- Expectation (E-step): Search for the state sequence with the highest probability given the current parameters.
- Maximization (M-step): Based on the current classification error, train a new ANN according to the cost function.

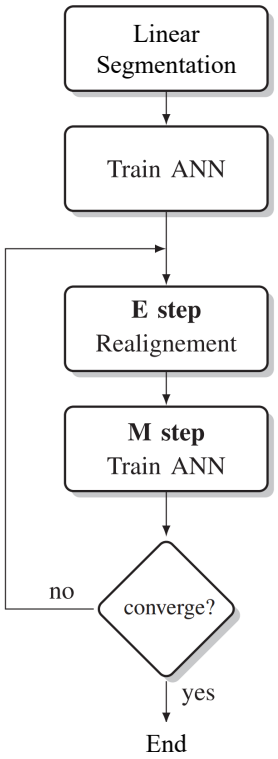


Figure 2.5: Flowchart of the Viterbi EM algorithm [186].

Fig. 2.5 illustrates the flowchart of the Vertibi EM algorithm. As we can see, a new ANN is trained at each M-step. A GMM-HMM system is trained to obtain segmentations and alignments for the subsequent ANN.

2.4 Neural Network based Acoustic Modeling

2.4.1 Overview

This section will formally discuss ANNs and then present popular ANN-based acoustic models in speech recognition.

ANNs are non-linear adaptive models that model the relation between an input vector $\boldsymbol{x}$ of dimension d_x and the corresponding output vector $\boldsymbol{y}$ of dimension d_y. ANNs are inspired by biological brain systems [155] and the first attempt was the perceptron, a linear model for information storage and organization [232]. The perceptron was then extended to the non-linear classifier as an universal approximator [47, 104].

An ANN comprises several layers. A typical ANN layer contains a linear transformation, a weight matrix, and a non-linear activation function:

$$\boldsymbol{y} = f(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}). \tag{2.12}$$

$\boldsymbol{W} \in \mathbb{R}^{d_x \times d_y}$ is the weight matrix, $\boldsymbol{b}$ is the bias vector of dimension d_x. $f(\cdot)$ is an activation function, e.g., the sigmoid [87] function or the rectified linear unit (ReLU) [172].

A typical ANN architecture consists of several hidden layers, which is also referred to as multilayer perceptron (MLP). The learnt parameters of ANNs are the weight matrix and bias vector of each hidden layer, which are updated during training. The number of units of each layer is set empirically.

ANNs can be used for both classification (e.g., speech recognition task [49]) and regression tasks (e.g., speech enhancement task [308]). In speech recognition, a typical classification task, ANNs model the relation between the label of the input $\boldsymbol{x}$ and its hypothesized class. The network outputs a score for each class in the form of the posterior probability. To compute the posterior probability, the softmax non-linearity is employed [34]. The posterior probability of label $i \quad 1 \leq i \leq N_{\mathrm{K}}$ is calculated as

$$P(i|\boldsymbol{x}) = \frac{e^{f_i(\boldsymbol{x})}}{\sum_i e^{f_i(\boldsymbol{x})}}, \tag{2.13}$$

where $f_i(\boldsymbol{x})$ is the score of a class.

Cross-entropy has been demonstrated to be effective for training ANNs [229, 171]. The cross-entropy criterion is a proximity measure between the network outputs and its target. The target is usually an *one hot* representation of class $i \quad 1 \leq i \leq N_{\mathrm{K}}$, i.e., a vector of size N_{K} with 1 for the ith component and 0 elsewhere, indicating that class i is the target of the current input. Given a training set of N_{K} labels $(\boldsymbol{x}_n, i_n) 1 \leq n \leq N_{\mathrm{K}}$,

the cost function $\mathcal{L}$ can be calculated by

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \log(P(i_n)|\boldsymbol{x}_n, \theta), \tag{2.14}$$

where the logarithm is calculated as

$$\log(P(i_n)|\boldsymbol{x}_n, \theta) = f_i(\boldsymbol{x}, \theta) - \text{logadd}_j(f_j(\boldsymbol{x}, \theta)), \tag{2.15}$$

and the logadd$(\cdot)$ function is calculated as

$$\text{loggadd}_j(z_j) = \log(\sum_j e^{z_j}). \tag{2.16}$$

Normally, the model parameters are initialized randomly, and updated by back-propagation algorithm [234, 132] following the chain derivative rule:

$$\theta \longleftarrow \theta + \lambda \frac{\partial \log(P(i|\boldsymbol{x}, \theta))}{\partial \theta}, \tag{2.17}$$

where θ denotes the model parameters and the λ is the learning rate. The back-propagation algorithm propagates the error backward either by accumulating the cost gradient from several examples within a batch, or by stochastic gradient descent technique [29], which randomly iterates over the training set and chooses one example for the likelihood gradient calculation. Alternatively, a compromise is accumulating the cost gradient from examples within a mini-batch. The main issue when training ANNs is overfitting. Overfitting is a phenomenon that the model performance on the training set keeps increasing while it decreases on the test set, namely its generalization capabilities decreasing. To avoid this problem, a validation set is often separated from the training dataset and utilized for evaluating the model performance at each iteration. The training stops automatically when classification accuracy on the validation set starts decline, which is referred to as early stopping [170]. Normally the model that performs best on the validation dataset is selected as the best model.

ANN-based AMs have drawn a great amount of research efforts since mid 1980s in speech recognition. The first successful attempts have been obtained in phoneme recognition [129, 286]. Thereafter, it was extended to isolated word recognition [30]. For continuous speech recognition, successful applications were first obtained on small vocabularies [85]. Meanwhile, the hybrid ANN-HMM acoustic modeling was developed [171, 31, 22, 227]. In this section, several popular ANN architectures will be presented.

2.4.2 Deep Neural Networks

The increment of computational power encourages the development from MLPs to DNNs. Different from MLPs, DNNs possess more hidden layers (usually more than two hidden layers). Therefore, Eq. 2.12 should be extended to be

$$\boldsymbol{y} = f(f(f(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1) \cdots \boldsymbol{W}_{n-1} + \boldsymbol{b}_{n-1})\boldsymbol{W}_n + \boldsymbol{b}_n). \tag{2.18}$$

$\boldsymbol{W}_n$ and $\boldsymbol{b}_n$ represent the weight matrix and the bias of the nth layer, respectively. $f(\cdot)$ denotes the activation function. Fig. 2.6 exhibits an example of DNN architecture with three hidden layers.

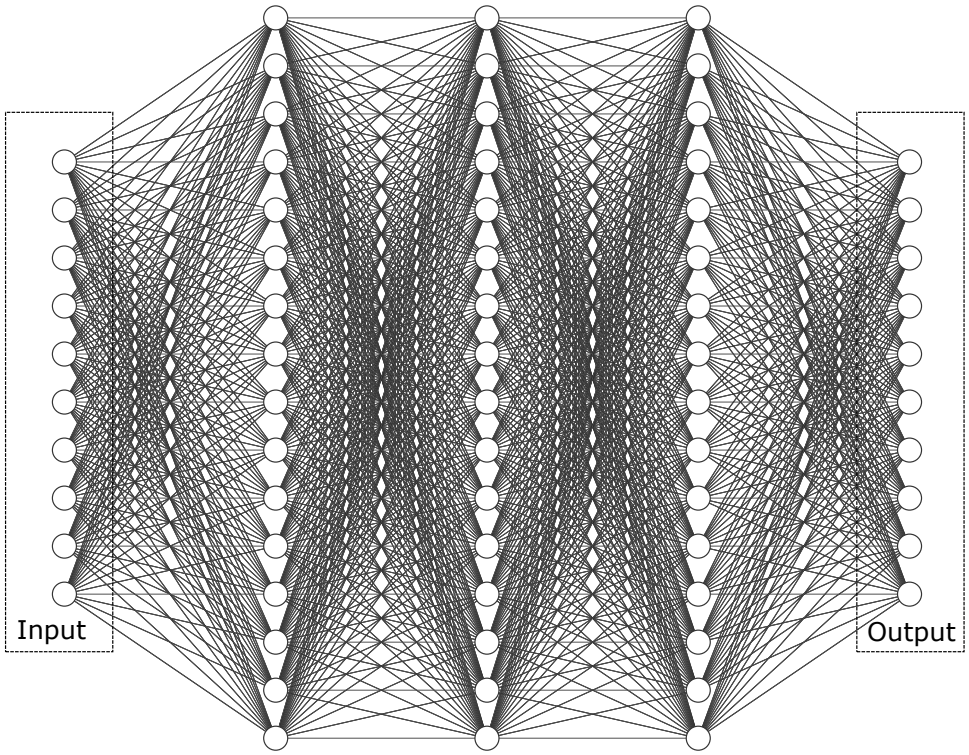


Figure 2.6: An example of DNN architecture with three hidden layers.

DNNs have achieved lots of performance improvements compared to MLPs. However, DNNs are difficult to be trained, especially when the training data is limited, because of its large scale of parameters. To tackle this problem, pre-training techniques are introduced to initialize the DNN parameters. Deep Belief Networks is a popular pre-training approach, which is based on the restricted Boltzmann machines [100]. The goal of Deep Belief Networks is to maximize the likelihood of the joint probability of data and their corresponding labels. Besides, regularization techniques are also proposed, such as dropout. Dropout is a technique for improving the network generalization. During training, dropout sets a certain amount of weights of each hidden layer to zero randomly to reduce the mutual dependency of the neurons [260].

There are different inputs for hybrid DNN-HMM systems. One is the cepstral-based feature for phone recognition or continuous speech recognition [49, 166, 250, 168]. Besides, bottleneck features have been investigated for continuous speech recognition [320, 237, 48]. Afterwards, there has been a growing interest in utilizing *intermediate* representations (between raw waveforms and cepstral-based features) as inputs. Another classification is Spectral-based features [133, 167, 27, 326], and self-learnt features from spectrum have also been proposed in [236]. More recently, using raw speech signals as inputs is drawing more and more attention [61, 183].

2.4.3 Convolutional Neural Networks

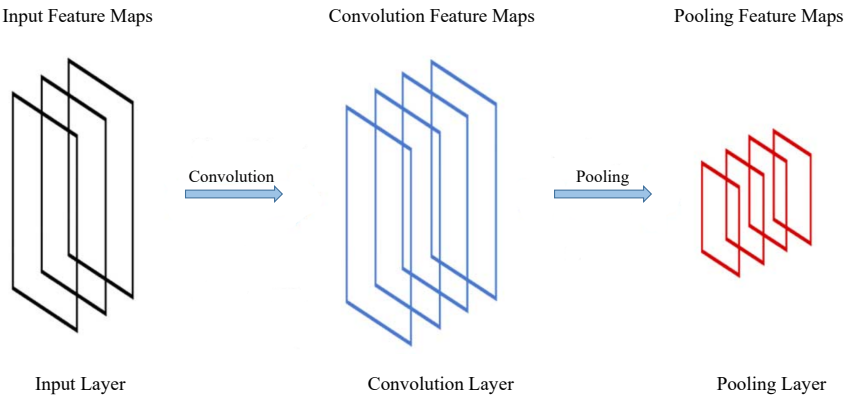


Figure 2.7: An example of one CNN layer comprising a pair of convolution operations and a pooling operation in succession.

Inspired by studies on computer vision, CNNs possess a natural priority of dealing with sequential data [131]. Fig. 2.7 shows the basic architecture of a CNN layer. Different from DNNs, CNNs take a sequence of vectors as the input and a convolution operation is applied to the input by a kernel. Lower layers extract low-level representations, and the higher the layer, the more abstract the representation it extracts. CNNs suit for speech signal processing as adjacent frames of the center frame (the context) always carry important information.

Time-Delay Neural Networks (TDNNs) are inspired by one-dimensional CNNs. They were initially studied on phoneme recognition [286] and isolated word recognition [30], and are implemented on LVCSR recently. [198] presents the performance of TDNN on several LVCSR tasks with training data ranging from 3 to 1800 hours, and TDNNs are demonstrated to be efficient in learning wider temporal dependencies in both small and large data scenarios. [203] improves TDNNs to a factored form which is structurally the same as a TDNN whose layers have been compressed via singular value decomposition, and two factors of each matrix are constrained to be semi-orthogonal. Factored TDNNs give substantial improvements over TDNNs.

2.4.4 Recurrent Neural Networks

RNNs [57] are a class of neural networks, which access the predictions of the earlier examples to classify the current example at a time instance. Connections between neurons of the RNN form a directed graph. Bi-directional RNNs [248] consist of two RNNs in different directions. Hence, the prediction at time t is based on predictions in both directions. Fig. 2.8 shows an example of a Bi-directional RNN.

There are a series of RNN-based variants. The alpha-net [33] is a RNN-based neural network utilized in the HMM framework. Robinson [231] proposes a recurrent nets for phone probability estimation, where the current output of the network is predicted by the

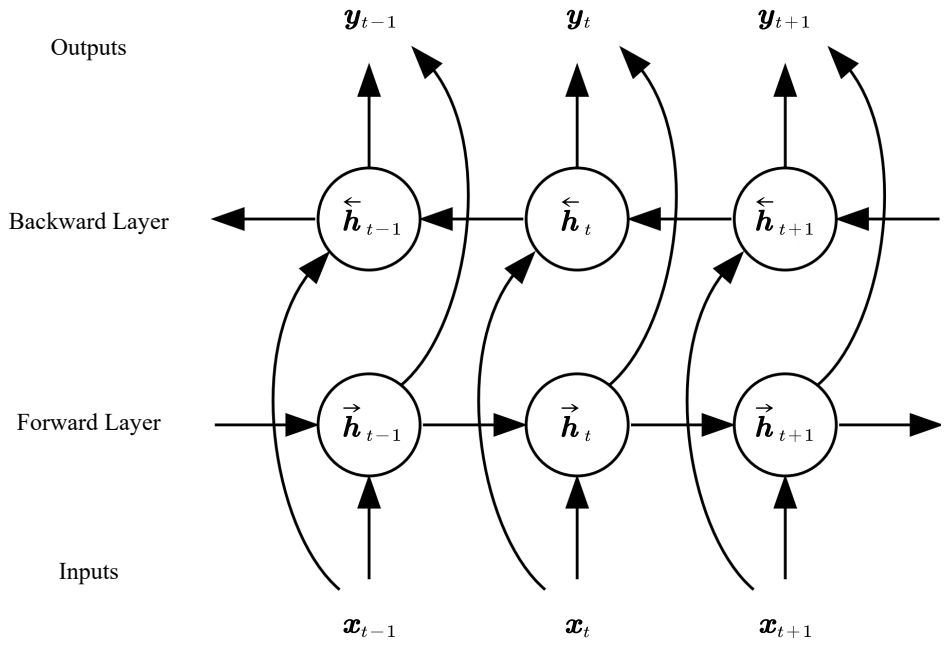


Figure 2.8: An example of the Bi-directional RNN. $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ represent the input and output at the time instance t, respectively. $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ denote the forward hidden sequence and the backward hidden sequence, respectively. [80]

current input and a hidden state variable. In the mean while, this hidden state is related to all previous inputs. However, there is a common problem of these RNNs, which is the gradient vanishment. This problem constrains their ability to access long-range contexts. In this light, the Long Short Term Memory network (LSTM) [101] is proposed.

LSTM [101] is a particular type of RNNs, which comprises LSTM gates. An LSTM gate can choose to store or delete the input information, and it can replace the RNN neuron units or be deployed as an auxiliary module. Therefore, LSTM gates allow the network to handle very long contexts at a given time. Fig. 2.9 shows the interior architecture of an LSTM gate. The Bi-directional LSTM (BLSTM) approach contains bi-directional RNNs, enabling the network to access to contexts of the input in both directions. BLSTM is originally proposed for framewise phoneme classification [81]. A preliminary study on continuous speech recognition is presented in [79]. LSTM layers can also be integrated to other types of ANNs for tackling speech signals, e.g., CNNs [36] and DNNs [94].

2.5 Lexicon

The lexicon factorizes each word into a sequence of acoustic units according to its pronunciation. The most popular acoustic unit is the phone [185], and as mentioned earlier, phones can be monophones or triphones. Each monophone or triphone can be considered as an independent acoustic unit. For example, the word *able* can be factorized

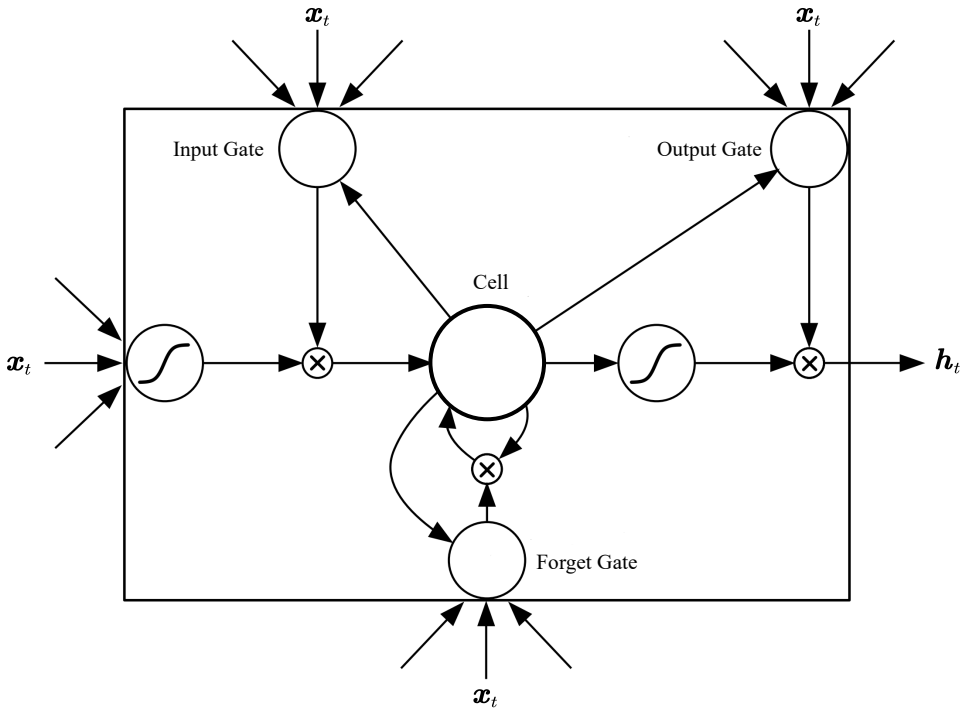


Figure 2.9: An example of the LSTM cell. $\boldsymbol{x}_t$ represents the input sequence at time instance t. $\boldsymbol{h}_t$ denotes the corresponding hidden sequence. [80]

as {[ey] [b] [l]} in the case of monophones, and as {[ey-b] [ey-b+l] [b+l]} in the case of triphones. However, there always exist unseen acoustic units during training, and this problem is usually addressed by utilizing top-down state clustering techniques [249] (as described in Section 2.3.2).

2.6 Language Modeling

Let $\mathcal{W} = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_i, \cdots, \boldsymbol{w}_{N_w}\}$ $1 \leq i \leq N_w$, and the probability of $\mathcal{W}$ can be factorized into a product of conditional probabilities:

$$P(\mathcal{W}) = \prod_{i=1}^{N_w} P(\mathcal{W}_i | \mathcal{W}_{i-1}, \cdots, \mathcal{W}_1). \tag{2.19}$$

Assuming the word sequences possess the n-order Markovian property [151], each conditional probability of Eq. 2.19 can be approximated by

$$P(\mathcal{W}_i | \mathcal{W}_{i-1}, \cdots, \mathcal{W}_1) \approx P(\mathcal{W}_i | \mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1}). \tag{2.20}$$

As the result, the probability of $\mathcal{W}$ can be expressed as

$$P(\mathcal{W}) = \prod_{i=1}^{N_w} P(\mathcal{W}_i | \mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1}), \tag{2.21}$$

which is referred to as n-gram LM.

The maximum-likelihood estimation of the word sequence $\mathcal{W}_i$ given its n-order Markovian property is

$$P(\mathcal{W}_i|\mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1}) = \frac{f(\mathcal{W}_i, \mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1})}{\sum_{\mathcal{W}} f(\mathcal{W}_i, \mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1})}. \tag{2.22}$$

$f(\mathcal{W}_i, \mathcal{W}_{i-1}, \cdots, \mathcal{W}_{i-n+1})$ represents the occurrence of the n-gram word sequence in the training transcripts. Note that the premise of an precise estimation is that all possible n-gram word sequences are well covered. However, this hypothesis is not always met in the large vocabulary task as some rare word sequences probably never occur. Thus, the smoothing method of the conditional probabilities is necessary. Generally, there are three categories of smoothing schemes [321]:

- Discounting. If some n-gram word sequences never occur in the training data, the conditional probabilities of those n-grams are 0 in the maximum-likelihood estimation. To avert this problem, a certain amount of overall probability mass is allocated to these unseen n-grams. The ratio of re-allocated probabilities to the overall probabilities is controlled by a discounting scalar, e.g., Good-Turing discounting [119], Kneser-Ney smoothing [124], and absolute discounting [177].

- Back-off. In the Back-off strategy, shorter histories are used instead of assigning probability mass to the unseen n-grams in the above discounting approaches. The back-off strategy can be applied recursively. For example, a tri-gram, bi-gram or uni-gram distribution can serve as the back-off of a 4-gram distribution.

- Interpolation. To construct a more robust estimation, high-order n-gram LMs can be interpolated with low-order n'-gram LMs. Besides, different LMs generated from different text data can also be interpolated. Note that the interpolation weights are often fine-tuned on a different dataset.

The foregoing LMs are generative LMs. There is another type of LMs, referred to as discriminative LMs, which has drawn more attention, e.g., neural network LMs [23]. In a neural network LM, each word $\{\boldsymbol{w}_{i-j} : j = 1, \cdots, n-1\}$ is mapped to a vector $\boldsymbol{w'}$ in a continuous space. Given a sequence of word embedding vectors $\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{n-1}\}$, a feed-forward neural network is used to predict the posterior probability of $\boldsymbol{w}_i$. A special example is RNN LM [160], where the input layer and the hidden layer are connected recurrently. With the help of this recurrent connection, the RNN LM is able to model long-term dependency.

2.7 Decoding

Decoding aims to search for the most probable path. As demonstrated in Fig. 2.1, the decoding algorithm requires three inputs: the acoustic likelihood modelled by the AM, the probability of a word sequence modelled by the LM, and the lexicon which specifies

the composition of a word sequence by individual acoustic units (e.g., phones). However, a word may have different phone compositions due to allophones, and thus, HMM phone models may result in different state sequences. Therefore, Eq. 2.2 can be expanded to be a multiple-level marginalization as

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}}\{P(\mathcal{W})\sum_{\boldsymbol{\phi}\in\mathcal{W}} P(\boldsymbol{\phi}|\mathcal{W})\sum_{\boldsymbol{s}\in\boldsymbol{\phi}} p(O,\boldsymbol{s}|\boldsymbol{\phi})\}. \tag{2.23}$$

$\boldsymbol{\phi}$ denotes a possible phone sequence from the hypothesis set $\mathcal{W}$, and $\boldsymbol{s}$ is one of the possible state sequences of $\boldsymbol{\phi}$. $P(\boldsymbol{\phi}|\mathcal{W})$ is the pronunciation probability provided by the lexicon [86], and $p(O,\boldsymbol{s}|\boldsymbol{\phi})$ and $P(\mathcal{W})$ are the acoustic and language scores respectively. The direct evaluation over all possible paths could result in prohibitive computational cost, so the Viterbi algorithm [285] is applied. The Viterbi algorithm converts the summation over possible phone and state sequences to a maximum:

$$\hat{\mathcal{W}} \approx \arg\max_{\mathcal{W}}\{P(\mathcal{W})\max_{\boldsymbol{\phi}\in\mathcal{W}} P(\boldsymbol{\phi}|\mathcal{W})\max_{\boldsymbol{s}\in\boldsymbol{\phi}} p(O,\boldsymbol{s}|\boldsymbol{\phi})\}. \tag{2.24}$$

To search for the best state sequence $\hat{\boldsymbol{s}}$ for an observation sequence $O = \{\boldsymbol{o}_1,\cdots,\boldsymbol{o}_t,\cdots,\boldsymbol{o}_T\}$, a partial best-path score $\phi_j(t)$ is defined as

$$\phi_j(t) = \max_{\boldsymbol{s}_1,\cdots,\boldsymbol{s}_{t-1}} p(\boldsymbol{o}_1,\cdots,\boldsymbol{o}_t,\boldsymbol{s}_1,\cdots,\boldsymbol{s}_{t-1},\boldsymbol{s}_t = j). \tag{2.25}$$

The partial best-path score can be recursively calculated by

$$\phi_j(t) = \max_i\{\phi_i(t-1)a_{ij}\}b_j(\boldsymbol{o}_t) \quad 1 < j < N_s, 1 \le t < T, \tag{2.26}$$

and

$$\varphi_j(t) = \arg\max_i\{\phi_i(t-1)a_{ij}\}, \tag{2.27}$$

with the initialization

$$\phi_1(0) = 1 \tag{2.28}$$

and

$$\phi_j(1) = a_{1j}b_j(\boldsymbol{o}_t). \tag{2.29}$$

N_s denotes the state number in the given HMM sequence. The best previous state in a partial path, ending at time instance t and state j, is stored in $\varphi_j(t)$. $\phi_{N_s}(T)$ gives the likelihood of the best path, and the best state sequence $\{s_1,\cdots,s_T\}$ can be retrieved by the recursion:

$$s_T = N_s, \tag{2.30}$$

and

$$s_t = \varphi_{s_{t+1}}(t+1), \quad t = T-1,\cdots,1. \tag{2.31}$$

The Viterbi algorithm can be also applied in the decoding of the continuous speech recognition [178, 316], where each state has one or more tokens at each time step. Each token contains two elements: a word-end link and the value of the partial path it

represents. At each time instance, only the token with the highest likelihood is proceeded. When a token is transmitted through an arc connecting two words, the word-end link is updated with the last word. In this way, the most likely sequence of words can be traced back using the word-end link. There is another decoding method called n-best score or word lattices [182], where at a time instance, n tokens with top scores are propagated forward instead of only the best one. Since the starting time of each word is hypothesized to be independent [257], the n-best score method cannot assure the exact n-best paths. Nevertheless, it can be implemented efficiently and is proven to be feasible.

The decoding complexity increases considerably when a high-order n-gram LM is used, because tokens can only be merged when its $n-1$ previous words are identical. To keep the decoding complexity in a certain range, token paths with the score below a threshold need to be removed. This is referred to as pruning [178]. Note that the beam-width needs to balance the computational cost and the search errors, since if it is too wide, the decoding complexity cannot be contained; if it is too tight, the most likely path may be pruned at an early step.

Another issue existing in the practical implementation is that the acoustic score and the language score have different dynamic ranges [321]. The language score is often much smaller than the acoustic score. To solve this problem, the language scores are often scaled up by a scaling factor, which is often empirically set for a particular task. Similarly, the pronunciation probability can be also scaled using a separate scalar. Another issue is the use of the word insertion penalty to avoid recognition errors from a number of short words. With these techniques, Eq. 2.24 can be rewritten as

$$\hat{\mathcal{W}} \approx \arg\max_{\mathcal{W}} \{\alpha_{\text{LM}} \log P(\mathcal{W}) + \beta_{\text{pro}} \log \max_{\boldsymbol{\phi} \in \mathcal{W}} P(\boldsymbol{\phi}|\mathcal{W}) + \log \max_{\boldsymbol{s} \in \boldsymbol{\phi}} p(O, \boldsymbol{s}|\boldsymbol{\phi}) + \gamma_{\text{in}}\, l_{\mathcal{W}}\}, \quad (2.32)$$

where α_{LM} is the LM scaling factor, β_{pro} is the pronunciation probability scalar, γ_{in} is the insertion penalty, and $l_{\mathcal{W}}$ is the length of the word sequence $\mathcal{W}$.

2.8 Evaluation Criteria

The performance of an ASR system is mainly evaluated on WER by comparing the hypothesised transcription and the reference transcription [153]. The deletions (E_d), substitutions (E_s), and insertions (E_i) are all considered as errors, and the WER stands for the percentage of all errors to the total number of words (N_w), i.e.,

$$WER = \frac{E_d + E_s + E_i}{N_w} \times 100\,\%. \quad (2.33)$$

Importantly, the calculated WER can be beyond 100 %.

Instead of WER, CER is employed in experiments utilizing Mandarin corpora, since a single character often represents a word in the Mandarin writing system. CER follows the same formula of WER, but characters are utilized as the computational unit instead of words.

2.9 End-to-End Models

In HMM-based framworks, different modules are trained independently. HMMs are mainly for frame-level dynamic time warping, and GMMs or DNNs are employed to estimate HMM hidden states' emission probabilities [158]. This construction determines that HMM-based models have intrinsic weaknesses:

- Local optimality. AMs and LMs are trained independently and differently with their own optimization objective functions. Hence the optimality of each module cannot ensure the global optimality [327, 79].
- Quasi-stationary and conditional independence assumptions. Although these two assumptions simplify the model's construction considerably, as described in Section 2.3, neither of these two assumptions is true for practical usage.

Due to the foregoing weaknesses of HMM-based models, end-to-end LVCSR has attracted more and more research efforts. The end-to-end model directly maps the input speech sequence to the word sequence or other graphemes. Basically, end-to-end speech recognition models are composed of an encoder, an aligner, and a decoder. The encoder maps the input audio sequence to the feature sequence, the aligner generates the alignment between the feature sequence and the output labels, and the decoder decodes the output word sequence. Fig. 2.10 illustrates the structure of a typical end-to-end model.

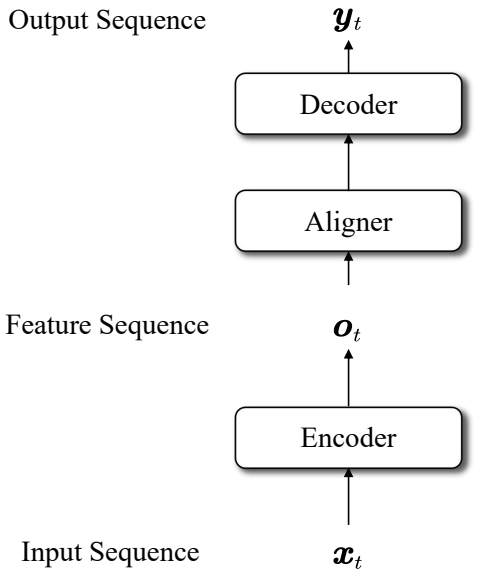


Figure 2.10: The structure of a typical end-to-end model. $\boldsymbol{x}_t$, $\boldsymbol{o}_t$, and $\boldsymbol{y}_t$ represent the input sequence, the feature sequence, and the output sequence at time instance t, respectively.

Different from HMM-based models that consist of multiple independent modules, end-to-end models possess a deep architecture [103]. Therefore, there is no need to

execute posterior estimation on the output. However, alignments between the speech data and the label sequence is still an inevitable problem for end-to-end models. Different from the forced alignment, which HMM-based models utilize, end-to-end models adopt the soft alignment, where each audio frame is aligned to all possible states with a certain probability distribution. Based on the category of soft alignment approaches, end-to-end models can be grouped into three classifications: CTC-based end-to-end models, RNN-based transducers, and attention-based end-to-end models.

2.9.1 CTC based End-to-End Models

Before training HMM-based model, a necessary step is roughly estimating the forced alignments of the training data. Without HMM, this estimation is accomplished by the CTC algorithm [78]. The CTC algorithm make it possible to achieve the real end-to-end framework. It solves the problem of forced alignment by acting as the loss function essentially. And for the CTC algorithm, the output of the end-to-end model can be the target transcription directly [289].

2.9.1.1 Path Searching

Given an input signal sequence $\boldsymbol{x} = \{x_1, \cdots, x_t, \cdots, x_T\}$, where T is the length of the input sequence. Firstly, it is encoded into a feature sequence $\boldsymbol{o} = \{o_1, \cdots, o_t, \cdots, o_T\}$ by the encoder. Then CTC performs on the feature sequence $\boldsymbol{o} = \{o_1, \cdots, o_t, \cdots, o_T\}$, and converts it into a probability distribution sequence $\boldsymbol{q} = \{q_1, \cdots, q_t, \cdots, q_T\}$. Please note that for CTC-based models, their vocabularies always include a special token *blank* "-". Through the distribution sequence, every frame $x_t \in \boldsymbol{x}$ is mapped to a meaningful label in the vocabulary or *blank* "-", and as the result, the input sequence $\boldsymbol{x}_t$ is mapped to a path also with the length T. From this respect, CTC process can be considered as a forced alignment process.

Please note that in the path searching process, there is an implicit independence assumption on the output sequence. Namely, the selected label at a time instance is totally independent on labels of other time steps. By contrast, context information is included in the encoding process. In other words, CTC postulates conditional independence in LMs solely, not in AMs. Therefore, the CTC-trained encoder is essentially an AM, not being able to model languages.

2.9.1.2 Path Aggregation

From the path searching process, the length of the output sequence is equal to that of the input sequence, which does not match the practical situation. To aggregate output labels to be readable utterances, two operations are mainly included: merging the contiguous labels, and then deleting the *blank* label "-" in the path. Let the word *sun* be an example. *sun* is a label sequence of length 3, but it could contain 7 different output paths of length 5-7, as shown in Fig. 2.11(a). Fig. 2.11(b) shows the lattice example of CTC, where *1, 2, 3, 4* represent time steps and *s*, *u*, *n*, and - denote the labels in the vocabulary. In the

lattice, each passable path starting at time step 1 and ending at time step 4 stands for a possible path of *sun*.

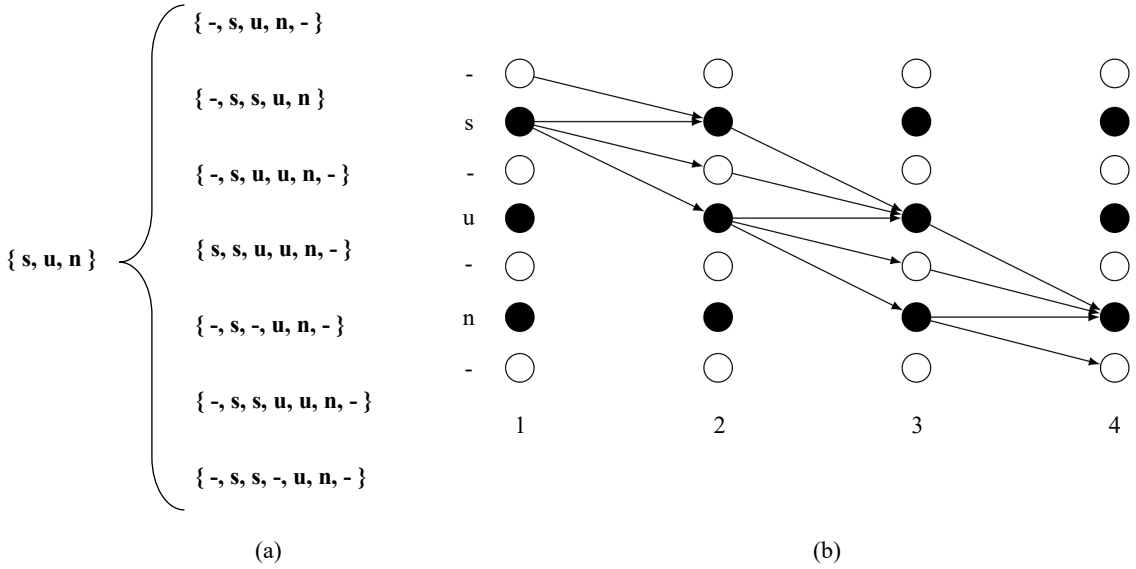


Figure 2.11: (a) Possible paths for label sequence *sun*. (b) Lattice examples in CTC.

During path searching and path aggregation, probabilities of label sequences are also calculated. Let $\mathcal{V}^T$ be the set of all T-length sequences defined on the vocabulary $\mathcal{V}$ and $\mathcal{V'}^T$ be the subset of $\mathcal{V}^T$ of the label sequence $\boldsymbol{x'}$. $\boldsymbol{\nu} \in \mathcal{V}^T$ denotes a single path in the set $\mathcal{V}^T$. Then the conditional probability of the label sequence $\boldsymbol{x'}$ given the input sequence $\boldsymbol{x}$ can be calculated as

$$\begin{aligned} p(\boldsymbol{x'}|\boldsymbol{x}) &= \sum_{\boldsymbol{\nu} \in \mathcal{V'}^T} p(\boldsymbol{\nu}|\boldsymbol{x}), \\ p(\boldsymbol{\nu}|\boldsymbol{x}) &= \prod_{t=1}^{T} q_t^{\nu_t}, \quad \forall \boldsymbol{\nu} \in \mathcal{V'}^T. \end{aligned} \tag{2.34}$$

ν_t represents the label at time instance t of sequence $\boldsymbol{\nu}$. Obviously $p(\boldsymbol{x'}|\boldsymbol{x})$ is differentiable. Hence the model can be trained with the gradient back propagation method. Nevertheless, since the number of paths in $\mathcal{V'}^T$ is unknown, it is still difficult to follow Eq. 2.34. Therefore, the calculation method that is utilized in the practical situation is the forward-backward algorithm [78].

Although, CTC method can achieve the frame-to-frame forced alignment, it does not require the output sequence to be the same length as the input sequence. Therefore, CTC adopts the soft alignment eventually, which is an essential difference from HMM-based systems [79].

With the advent of the CTC algorithm, the construction of LVCSR models eliminates the need of the forced alignment, and is thus considerably simplified. Besides, CTC also makes it feasible to build an real end-to-end model, which is able to map audio waveforms to texts [60] directly.

2.9.1.3 CTC Language Model

Eq. 2.34 indicates that the CTC algorithm hypothesizes all output labels are independent of each other, which determines that CTC is incapable of modeling languages properly. Therefore, many CTC-based works [137, 90, 147, 108, 294, 5] employ an auxiliary LM to CTC. Their results show that the auxiliary LM improves the performance of CTC-based end-to-end systems substantially.

Basically, there are two sorts of LMs in CTC-based ASR systems. One is the re-score method (as know as re-ranking method). The re-score method starts with CTC calculating the probability distribution, based on which the beam-search algorithm screens the first path candidates. Then these paths are re-scored by an auxiliary LM. Finally, both scores are taken into consideration to determine the optimal decoding path [79]. The other sort of LM is often called first-pass method. As the term implies, the optimal decoding path is gained in one scoring process, where the score of the auxiliary LM is combined with the CTC calculation at each extending node during beam-searching. Since the first-pass method integrates the LM into the beam-search process, it generally achieves better results and is more widespread.

However, introducing LM diverts CTC-based systems from the end-to-end principle, and the original goal of joint training an individual framework is also destroyed. Besides, LM can be very complex. The parameter scale of the LM used in [147] is up to 21 GB, which causes great real-time delay.

2.9.2 RNN-Transducer End-to-End Models

As mentioned above, two restrictions limit the effectiveness of CTC. One is the independence hypothesis, and the other is the premise that the output sequence can only be shorter than its corresponding input sequence. To overcome these limitations, the RNN-transducer is proposed in [76]. Similar to the CTC algorithm, RNN-transducer also aims to solve the forced alignment problem in end-to-end speech recognition, and it also introduces the *blank* label. However, their path possibility calculation and path aggregation methods are completely different. As the result, the RNN-transducer can map an input to any finite output sequence, and the interdependency within output elements is also jointly modeled. Many works on the RNN-transducer show improvements compared to CTC-based algorithms [80, 240, 55].

As shown in Fig. 2.12, a RNN-transducer consists of three modules:

- The transcription module. The transcription module is an encoder as well as an AM. For an acoustic input sequence $\boldsymbol{x} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ of length T, the transcription module maps it to a feature sequence $\boldsymbol{o} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$.

- The prediction module. The prediction network is acting as the LM in the decoder. It models the interdependencies among the output labels. Namely, to determine the output at time t, outputs at first $t-1$ time instances are considered.

- The joint module. It generates the alignment between the input and the output sequences based on the outputs of the transcription module and the prediction

module. The joint module plays the role of calculating the label distribution $p(y_n|\boldsymbol{y}_{1:n-1}, \boldsymbol{x}_t)$ at the output location n, given the history output sequence $\boldsymbol{y}_{1:n-1}$ and the input $\boldsymbol{x}_t$ at time t. The label distribution information is required by the decoding process.

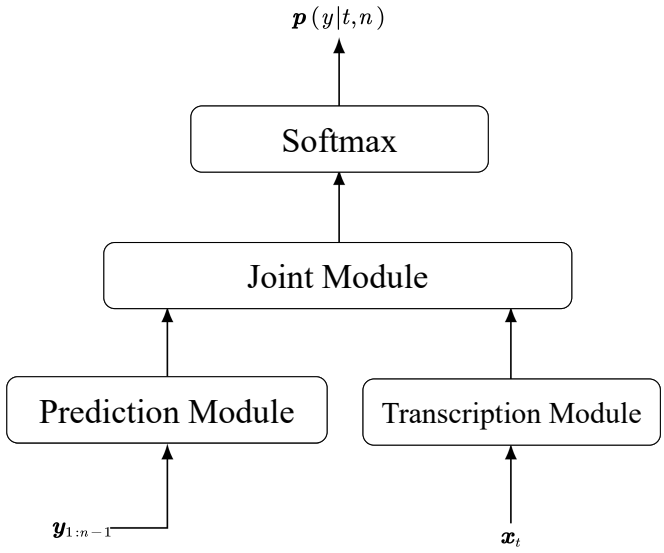


Figure 2.12: Illustration of the structure of the RNN-transducer [206].

The RNN-transducer adopts the forward-backward algorithm to calculate the probability of a given label sequence. Becasue of the path aggregation process, the RNN-transducer also produces soft alignments.

For the decoding process, the RNN-transducer reads an input $\boldsymbol{x}_t$, and keeps outputting labels until a *blank* appears. Then RNN-transducer reads in the next input $\boldsymbol{x}_{t+1}$ and repeats the decoding procedure until the whole input sequence has been processed. Since an input $\boldsymbol{x}_t$ may be mapped to an output of any length, the RNN-transducer can deal with situations where the length of the output sequence is longer than that of the input.

To sum up, there are three main characteristics of the design of the RNN-transducer [289]:

- Theoretically, the RNN-transducer allows the output sequence to be an arbitrary length.

- Each state is determined by the current input, previous states, and previous outputs. In this way, the RNN-transducer models the interdependence within the output sequence.

- The joint module models the relation between the acoustic model and the language model.

2.9.3 Attention based End-to-End Models

2.9.3.1 Overview

In 2014, Bahdanau et al. [11] propose to extend the using of a fixed-length vector by allowing a model to automatically (soft-)search for positions of a source sentence that are relevant to predicting the current label, without having to form these positions as a hard segment explicitly. The most distinguishing feature of this proposed approach is that the encoder does not attempt to encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and assigns a weight to each vector adaptively while decoding the translation. This feature frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector. This method enhances the model's ability to tackle long input sequences, and it also improves the speech recognition performance substantially, because it fits well with some natural characters of the speech recognition task:

- Similar to the machine translation task, speech recognition is also a sequence-to-sequence process that converts the input audio waveforms to the output text sentences.
- The encoder–decoder structure based on the attention mechanism does not require the alignment information between the input audio and the transcript, since the attention mechanism adopts the soft alignment instead of the forced alignment.
- As the encoder encodes the input into a vector sequence, the model is able to handle inputs of various lengths, which is in line with the realistic situations.

Fig. 2.13 illustrates an overview of the attention-based end-to-end model. The encoder maps the input sequence to a high-level representation, then the attention-based decoder decodes the current input based on all previous predictions. Given the input sequence $\boldsymbol{x}_t$ at time instance t, the network predicts the posterior probabilities of the current input $y_n \in \mathcal{V}$ as follows:

$$P(y_n|\boldsymbol{x}_t) = \prod_n P(y_n|\boldsymbol{x}_t, \boldsymbol{y}_{1:n-1}), \tag{2.35}$$

$$\boldsymbol{h}_t = \text{Encoder}(\boldsymbol{x}_t), \tag{2.36}$$

$$\boldsymbol{c}_n = \text{Attention}(\boldsymbol{a}_{n-1}, \boldsymbol{s}_n, \boldsymbol{h}_t), \tag{2.37}$$

$$y_n = \text{Decoder}(\boldsymbol{c}_n, \boldsymbol{y}_{1:n-1}). \tag{2.38}$$

$\boldsymbol{y}_{1:n-1}$ is a label sequence from y_1 to y_{n-1}. The encoder transforms the input sequence $\boldsymbol{x}_t$ to the l-length representation $\boldsymbol{h}_t = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l\}$. Next the attention-based aligner calculates the l-dimensional attention weight vector $\boldsymbol{a}_n \in [0, 1]^l$ to integrate all encoding outputs $\boldsymbol{h}_t$ into a context vector $\boldsymbol{c}_n \in \mathbb{R}^{D_h}$. $\boldsymbol{s}_n$ is the state vector at output step n. Then the decoder estimates the posteriori for output label y_n at output step n conditioned on the previous predictions $\boldsymbol{y}_{1:n-1}$ and the context vector $\boldsymbol{c}_n$.

According to the utilized information in attention calculation, the attention mechanism can be classified into three categories: content-based, location-based, and hybrid [289]. Eq. 2.39 shows the calculation of their attention weights a_n at the output position n, respectively:

$$a_n = \begin{cases} Attention(\boldsymbol{s}_{n-1}, \boldsymbol{h}_t) & \text{content-based} \\ Attention(\boldsymbol{s}_{n-1}, \boldsymbol{a}_{n-1}) & \text{location-based} \\ Attention(\boldsymbol{s}_{n-1}, \boldsymbol{a}_{n-1}, \boldsymbol{h}_t) & \text{hybrid} \end{cases} . \tag{2.39}$$

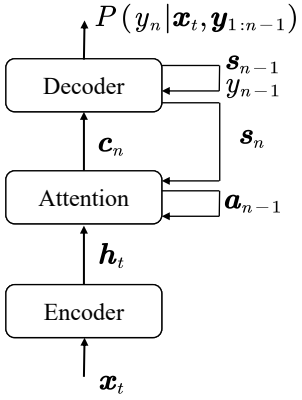


Figure 2.13: Overview of an attention-based end-to-end model.

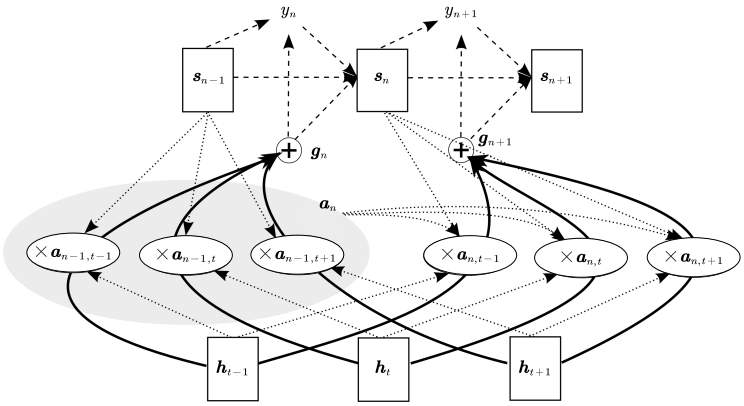


Figure 2.14: Illustration of the hybrid attention mechanism [43]. $\boldsymbol{g}_i$ is the so-called *glimpse* [165] in the terminology.

- Context-based [11, 306]. The context-based attention mechanism takes only the encoding representation $\boldsymbol{h}_t$ and previous hidden state $\boldsymbol{s}_{n-1}$ for the attention weight calculation at each position. However, the lack of the position information results

in the problem that the attention mechanism yields the same weights for the feature's occurrences in different sequences, leading to the so-called "similarity speech fragment problem".

- Location-based [77]. In the location-based attention mechanism, each previous attention weight $\boldsymbol{a}_{n-1}$ is employed as the location information at that step to calculate the current weight $\boldsymbol{a}_n$. However, it still lacks the output of the encoder.
- Hybrid [43]. As the name suggests, the hybrid attention mechanism calculates the current attention weight based on the encoding representation $\boldsymbol{h}_t$, the previous attention weight $\boldsymbol{a}_{n-1}$, and the previous hidden state $\boldsymbol{s}_{n-1}$, which enables the system to take advantages of both context-based and location-based attention systems. Fig. 2.14 illustrates the hybrid attention mechanism graphically.

2.9.3.2 Self-Attention Mechanism

An attention function maps a query and a set of key-value pairs to an output. The relation among the query, keys, values, and the output can be described as: One query's output is computed as a weighted sum of the values, where each weight of the value is computed by a designated function of the query with the homologous key. Self-attention [278] relates the information over different positions of the entire input sequence. The attention distribution is computed using scaled dot-product attention:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}. \tag{2.40}$$

$\boldsymbol{Q} \in \mathbb{R}^{t_q \times d_q}$, $\boldsymbol{K} \in \mathbb{R}^{t_k \times d_k}$, and $\boldsymbol{V} \in \mathbb{R}^{t_v \times d_v}$ are three inputs of the self-attention layer: queries, keys, and values, where t_q, t_k, and t_v are the element numbers in different inputs and d_q, d_k, and d_v denote the corresponding element dimensions. A softmax function is applied to obtain the weights on the values. Vaswani et al. [278] indicate that the two most popular attention functions are the additive attention [11], and the multiplicative (dot-product) attention. The dot-product attention is identical to the foregoing algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. The additive attention adopts a feed-forward network with a single hidden layer to compute the compatibility function. Although the dot-product attention and the additive attention are similar in terms of complexity theoretically, the former is much faster and more space-efficient than the later in practice because of its implementation by highly-optimized matrix multiplication functions. As similarly these two mechanisms perform for small values of d_k, the dot products grow large in magnitude for large values of d_k. Therefore, the scalar $\frac{1}{\sqrt{d_k}}$ is applied to prevent the softmax function from falling into regions with tiny gradients.

"You never fail until you stop trying."

– Albert Einstein

3

Back-End Techniques for Robust Automatic Speech Recognition

3.1 Overview

As described in Section 1.2, according to the processing stage of the whole ASR framework, previous techniques aiming at robust ASR can be basically grouped into three types: front-end, back-end, and joint-training techniques. This chapter focuses on the back-end technique. The back-end techniques are known as model-based techniques, which process the ASR model directly instead of the noisy inputs. The goal of back-end techniques is to let the ASR model learn the relationship between the observed noisy speech and the phonetic targets by itself. This chapter studies the HMM structure adaption for two reasons. One is that HMM-based systems have been occupying a significant position in the field of ASR for decades and are still drawing substantial attention. The other is that HMMs possess innate robustness against noise interference.

HMMs [19] are a stochastic process for modeling time-series data. Since speech signals possess intrinsic temporality, GMM-HMMs are the most classic acoustic model of the ASR system for decades. With the advent of DNNs, hybrid systems occupy the predominant position, including DNN-HMM systems [99, 49], CNN-HMM systems [238], and RNN-HMM systems [80], etc. In recent years, another round of revolution in machine learning triggers ASR architectures' diversification into a completely new approach, specifically end-to-end models, where HMM is abandoned [78, 38, 79, 43, 278]. However, the straightforward and challenging problems derived from end-to-end models are the tremendous growth of the model size, the increasing computational complexity, and the weak robustness to the input variations [26]. This drawback is proved by Lüscher et al. in [146], where on the LibriSpeech 960h task [188], the hybrid DNN-HMM system outperforms the attention-based system by 15 % relative on the clean and 40 % relative on the other test sets in terms of WER. Moreover, experiments on a reduced 100h-subset of the LibriSpeech training corpus show a more pronounced margin between the hybrid and attention-based architectures. Another argument is [292], where Wang et al. demonstrate that their transformer-based hybrid system outperforms the attention-based system by

16.4 % relative. Consequently, statistical approaches remain to be essential and still draw considerable attention [243, 91, 203, 146, 130, 330, 292, 125, 198, 40, 244].

The commonly-employed HMM structure is the left-to-right structure, with three states which model the beginning, the middle, and the end of a phone. Nevertheless, there is no vindicated evidence for its suitability and superiority. Since the structure affects the modeling capability considerably, there are multiple studies on optimizing the HMM structure.

Bakis-type HMMs [15] are word-based models, which are derived from sample utterances of the word. The number of states in the model is equal to the average duration of the word in frames. The frame size in Bakis's system is 10 milliseconds, and the average number of states for a word is about 30. Rabiner and Levinson [213] describe another word-based model in which the number of states is reduced to approximately 5. This model results in a substantial reduction in the number of parameters without much deteriorating the accuracy. This is because neighboring states in the Bakis model tend to be quite similar, and reducing several similar consecutive states into a single state does not degrade the model very much. Biem et al. [25] replace Bayesian Information Criterion with Discriminative Information Criterion, where discriminative power among models is maximized together with the likelihood. It achieves a slightly higher recognition rate at the expense of more complicated models. Geiger et al. [69] present a method to determine the number of states in HMMs. They propose a modification to the Bakis method [15] and a technique to improve the topology with few iterations.

However, there are three general disadvantages existing in the previous works [15, 213, 25, 69, 331, 246, 113, 3, 44]. Firstly, as they are all based on statistical methods, HMM topologies are constructed from limited data. Secondly, the statistical methods adopted in these works are computationally-expensive heuristic algorithms (e.g., the tree search algorithms), and not easy to employ. Thirdly, they balance between the state length and the model complexity poorly, leading to either high model complexity or limited performance improvements.

To address these problems, this study proposes a novel approach to optimize the HMM structure, leveraging deep learning, specifically, a Deep Neural Network Vector Quantizer (DNNVQ). First, this study introduces the concept *fenone* for representing sub-phones as the basic acoustic unit [169] (as described in Section 2.3). Fenone is the building block of phones and is modeled as one state of the HMM, which can be obtained automatically through a vector quantizer [14]. Next, all data are classified against different phones and then the vector quantization is applied on each phone's data. Finally, DNNVQ generates the fenonic baseforms for every phone, and accordingly, the HMM structure is decided. This algorithm is referred to as Neural Fenonic Baseform Growing (NFBG).

3.2 Related Work

The notion of *fenone* is inspired by [13] and [14], where the authors describe a new technique for constructing HMMs for the acoustic representation of words. They create

the notion of *fenone* to represent sub-word units, and it is derived automatically from one or more utterances of that word. Then the word model is constructed from fenonic forms. Since the word models are all composed of a small inventory of sub-word models, training for large-vocabulary speech recognition systems can be accomplished with a small training script by this technique. A method for combining phonetic and fenonic models is also presented in [14], and impressive improvements are achieved with speaker-dependent and speaker-independent models on several isolated-word recognition tasks.

The Neural Network Vector Quantizer was first proposed by Rigoll and Neukirchen in [230], which is a shallow neural network and is trained with the mutual information criterion. The index of the neuron in the output layer with the highest activation returns the label for the training sample, and thereby, the network performs the quantization to assign the input feature to a specific cluster. This model outperforms a K-means system and nearly matches the performance of a system with continuous (non-quantized) models in terms of word recognition accuracy rate.

Watzel et al. [298] extend the neural network vector quantizer to a deep neural network quantizer and introduce a novel approach, a mapping function, to train it in a supervised fashion with an arbitrary output layer size even though suitable target values are not available. The experiments demonstrate that the deep neural network quantizer reduces the WER by 17.6 % on monophones and by 2.2 % on triphones, respectively, compared to a continuous GMM-HMM system. Inspired by the success in [298], the deep neural network vector quantizer is taken over as the vector quantizer for this study.

Furthermore, this study extends the concept from *word* to *phone*, substitutes the sophisticated tree search algorithm in [14] with a concise neural network, and confirms its viability on the task of large vocabulary automatic speech recognition.

Besides, there also have been previous attempts of HMM adaption for increasing the ASR system robustness. Li et al. [136] present a new approach to HMM adaptation that jointly compensates for additive and convolutive acoustic distortion in environment-robust speech recognition. The hallmark of their new approach is the usage of a nonlinear, phase-sensitive model of acoustic distortion that captures phase asynchronous between clean speech and the mixing noise. Seltzer et al. [252] propose an algorithm to perform HMM adaptation to noisy environments called Linear Spline Interpolation, where the nonlinear relationship between clean and noisy speech features is modeled using linear spline regression. Linear spline parameters that minimize the error between the predicted noisy features and the actual noisy features are learned from training data. A variance associated with each spline segment captures the uncertainty in the assumed model.

3.3 Deep Neural Network Vector Quantizer

Let $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{x}_i, y_i^*)\}_{i=1}^N$ be a dataset comprising feature vectors $\boldsymbol{x_i} \in \mathbb{R}^D$ and their corresponding ground-truth labels $y_i^* \in \mathbb{N}$. The goal of the training is to find a function $f : \boldsymbol{x}_i \rightarrow y_i^*$. In [298], this goal is converted to approximate $g_\theta : \boldsymbol{x}_i \rightarrow \hat{m}_i$, where θ represents the parameters of the network and $\hat{m}_i \in \mathbb{N}$ defines the index of the maximum

value in the DNNVQ output layer $\boldsymbol{m}_i \in \mathbb{R}^{N_{\text{clu}}}$ by

$$\hat{m}_i = \underset{1 \leq j \leq N_{\text{clu}}}{\arg\max}\, m_i^j. \tag{3.1}$$

N_{clu} is the dimension of the output layer, and j describes the jth neuron in the layer. In contrast, the ground-truth label y_i^* is in the range $[1, N_{\text{K}}] = \{y_i^* \in \mathbb{N} | 1 \leq y_i^* \leq N_{\text{K}}\}$, where N_{K} denotes the dimension of the ground-truth label space.

Watzel et al. [298] employ maximum mutual information (MMI) as the criterion of the training. The mutual information $I(Y;M)$ is defined as

$$I(Y;M) = H(Y) - H(Y|M). \tag{3.2}$$

Y denotes a ground-truth label sequence, and M is a firing neuron sequence. $H(Y)$ defines the entropy of Y, and $H(Y|M)$ denotes the entropy of Y conditioned on M. Their probability mass functions are

$$P(M = \hat{m}^j) = \frac{1}{N} \sum_{i=1}^{N} \delta(\hat{m}_i, j) \quad \forall 1 \leq j \leq N_{\text{clu}} \tag{3.3}$$

and

$$P(Y = y^{*k}) = \frac{1}{N} \sum_{i=1}^{N} \delta(y_i^*, k) \quad \forall 1 \leq k \leq N_{\text{K}}. \tag{3.4}$$

The probability mass functions are created by counting occurrence numbers of $\hat{m}^j$ and y^{*k} based on all samples $\hat{m}_i$ and y_i^*, where the index k denotes the kth label in the ground truth label space and $\delta(\cdot)$ refers to Kronecker delta.

As the entropy $H(Y)$ is independent of the DNNVQ's parameters, $H(Y|M)$ needs to be minimized in order to maximize $I(Y;M)$. For this purpose, increasing the dimension of emitted labels $\hat{\boldsymbol{m}}_i$ could be a straightforward solution. However, it causes a new problem for training, where the dimension of the output layer and that of the ground-truth label space are unequal, i.e., $N_{\text{clu}} \neq N_{\text{K}}$. To tackle this problem, Watzel et al. [298] introduce the conditional probability $P_{\text{b}}(Y|M)$ of the ground-truth labels y_i^* conditioned on the DNNVQ outputs $\boldsymbol{m}_i$ as

$$\begin{aligned} P_{\text{b}}(Y|M) &= P(y_{b,k}^* | m_{b,j}) \\ &\approx \frac{\varepsilon + \sum_{i=1}^{N_b} \delta(y_i^*, k) m_i^j}{\varepsilon N_{clu} + \sum_{i=1}^{N_b} m_i^j}, \\ &\forall 1 \leq k \leq N_{\text{K}}, 1 \leq j \leq N_{\text{clu}} \end{aligned} \tag{3.5}$$

where ε is a small constant and the conditional probability $P_{\text{b}}(Y|M) \in \mathbb{R}^{N_K \times N_{\text{clu}}}$. This study takes minibatches with a sufficient batch size N_{b} to approximate $P_{\text{b}}(y|m) \approx P(Y|M)$. Then, the output $\boldsymbol{m}_i$ is mapped from dimension N_{clu} to dimension N_{K} with $P_{\text{b}}(Y|M)$ as

$$\boldsymbol{m}_{tra,i} = P_{\text{b}}(Y|M)\boldsymbol{m}_i \quad \forall 1 \leq i \leq N_{\text{b}}, \tag{3.6}$$

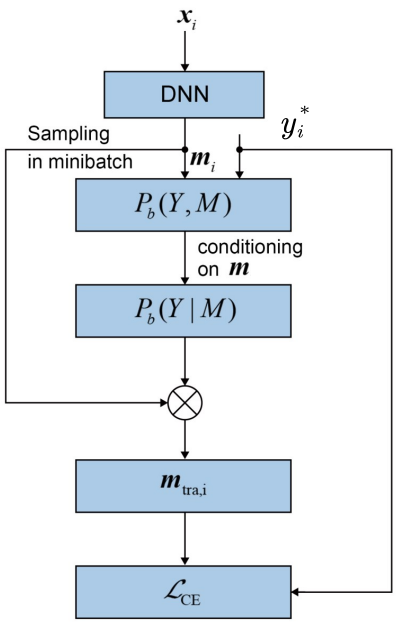


Figure 3.1: Diagram of DNNVQ training.

with $\boldsymbol{m}_{tra,i}$ denoting the transformed outputs of $\boldsymbol{m}_i$. In this way, the prototype size of the vector quantizer can be arbitrary even though the dimension of the ground-truth labels y_i^* is determinate. During training, the mutual information $I(Y;M)$ is implicitly maximized by minimizing the loss function $\mathcal{L}_{CE}(\boldsymbol{m}_{tra,i}; y_i^*)$ [280], where

$$\mathcal{L}_{CE} = -\frac{1}{N_\text{b}} \sum_{i=1}^{N_\text{b}} \sum_{k=1}^{N_\text{K}} \delta(y_i^*, k) \log(m_{tra,i}^k). \tag{3.7}$$

The diagram of DNNVQ training is depicted in Fig. 3.1.

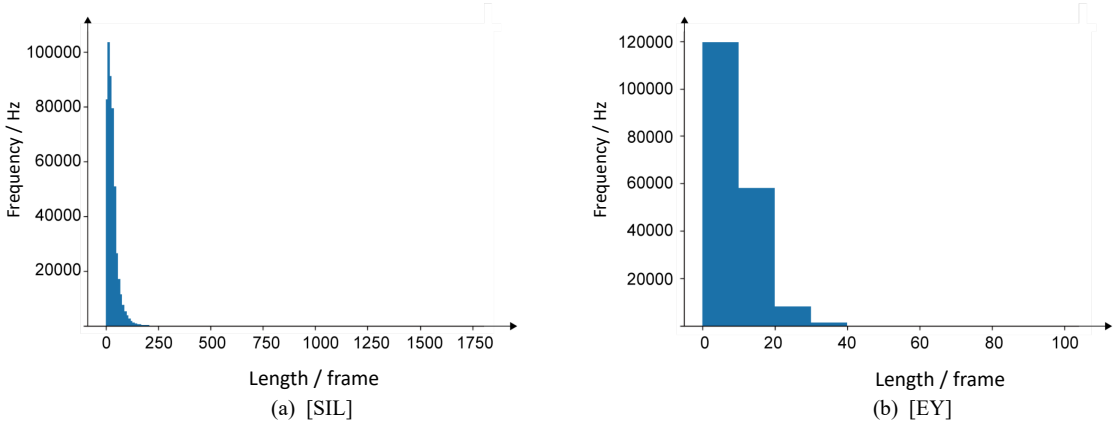
3.4 Neural Fenonic Baseform Growing

3.4.1 Overview

The motivation for introducing fenonic baseform growing supported by a DNN is as follows: If a hybrid system is created in the usual way, a GMM-HMM system is initially employed to generate ground-truth targets, which are the HMM states assigned from Viterbi alignments, for the subsequent DNN training. If the mutual information between the phoneme sequence and the corresponding Viterbi-state sequence is computed, it will have the "perfect maximum value", as each state will only occur for one specific phoneme. Likewise, in this approach, the feature vectors of the training data are processed by the DNNVQ described in Section 3.3. DNNVQ will attempt to maximize the mutual information between the phoneme sequence and the vector quantization

labels. Consequently, the vector quantization labels will converge to the corresponding "virtual states", which can then be considered as "fenones" as introduced in [14]. However, in this case, they should have even higher quality, due to the MMI principle fulfilled. The construction of the new fenonic baseforms using these "neural fenones" is in the following way:

- Step 1: Train a vanilla GMM-HMM model from flat-start and obtain forced alignments as the ground truth for the subsequent DNNVQ training.
- Step 2: Train a DNNVQ, as described in Section 3.3, using the forced alignments obtained from Step 1 to maximize the mutual information between the ground truth labels and the output units. For each training, the number of the prototypes, namely the dimension of the output layer, is specified and fixed.
- Step 3: Extract segments that contains feature vector sequences corresponding to each of the monophone in the system. Exclude extreme cases in the segment set of each monophone.
- Step 4: Pad the remained segments to the same length.
- Step 5: For the same frames of each phone's segments, calculate the products of all posteriors on the same output unit, namely the same prototype. The prototype with the highest product value is the fenone of the current frame. Consequently, the fenone sequence of this phone is acquired.
- Step 6: Compact the fenone sequence to the fenonic baseform by eliminating all successive duplicated fenones.



(a) [SIL] (b) [EY]

Figure 3.2: Histograms of the segment length distribution of phones (a) [SIL] and (b) [EY]. Every frame lasts for 25 ms.

3.4.2 Segment Lengths Padding

Intuitively, utilizing all training data must deliver the most accurate result. However, for one thing, computation complexity increases exponentially with the increment of training data; for another, the extreme cases, e.g., the longest ones or the shortest ones, which occur quite rare in the realistic scene, mislead the final decision. For this purpose, histograms for each phone are plotted, exhibiting the distribution range of lengths of all segments affiliated to a phone. Fig. 3.2 demonstrates the histograms of phones [SIL] and [EY]. As shown in Fig. 3.2, lengths of phone [SIL] differ in the range [0, 1750], while that of phone [EY] is [0, 100]. According to these histograms, the extreme cases of every monophone are discarded, but keep at least 80 % data for each monophone eventually. For instance, this study keeps segments ranging in [4, 120] for [SIL] and that for phone [EY] is [4, 30]. This study also conducted experiments with 70 % or 90 % data for each phone, but results indicated no difference, which also proves the robustness of the proposed approach to noise. Afterwards, all segments affiliated to one phone need to be justified to the identical length for the purpose that all frames representing for the beginning, the middle, or the end of the phoneme are aligned together, i.e., this study needs to justify all lengths of the segments to the maximum one. Instead of zero padding, a simple division is used as the alternative. Assume $l_{max} \div l_n = l_q \cdots l_r$, where l_{max} is the maximum length of all segments, l_n represents the length of any segment in the same set, l_q denotes the quotient, and l_r refers to the remainder. Then each frame of the current segment is duplicated l_q times and the last frame l_r times more. Taking the shortest and the longest segments of phone [EY] as the example, since $30 \div 4 = 7 \cdots 2$, each frame of the 4-frame segment should be duplicated 7 times, while the last frame 2 times more. As the result, the 4-frame segment is padded to be the maximum length.

3.4.3 Dynamic Baseform Generation

Fenones represent short speech events and are obtained automatically through the employment of a vector quantizer. Different from [14], where fenones represent sub-word units, this study extends its application to sub-phone units, a finer level of details. DNNVQ is deployed for generating the fenone sequence for each phone. Since the fenone sequence of a phone is derived from its utterances, this study realigns all training utterances against 40 monophones on TEDliumv2 [233] and 48 monophones on TIMIT [68], respectively. In order to distinguish from utterances, the sub-units of utterances are henceforth named as segments. Let $\boldsymbol{\mathcal{F}} = \{f_1, f_2, \cdots, f_{N_F}\} \quad 1 \leq N_{\mathrm{F}} \leq N_{\mathrm{clu}}$ be the alphabet of fenones and $\boldsymbol{\mathcal{F}}^*$ be the set of all finite length strings constructed by concatenating elements of $\boldsymbol{\mathcal{F}}$, namely fenone sequence. $\boldsymbol{\mathcal{G}} = \{g_1, g_2, \cdots, g_{N_{\mathrm{g}}}\}$ is the set of N_{g} monophones while $\boldsymbol{\mathcal{K}_i} \in \boldsymbol{\mathcal{K}}^* = \{\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2, \cdots, \boldsymbol{\mathcal{K}}_{N_{\mathrm{g}}}\}$ denotes the set of all segments affiliated to the corresponding phone g_i $1 \leq i \leq N_{\mathrm{g}}$. The goal here is to generate a fenone sequence for phone g_i based on its segments set $\boldsymbol{\mathcal{K}}_i$. The generated fenone sequence $\boldsymbol{f}_{1 \to l_{g_i}} = \{f_1, f_2, \cdots, f_{l_{g_i}}\} \in \boldsymbol{\mathcal{F}}^*$ is spanned up on $f \in \boldsymbol{\mathcal{F}}$, leveraging DNNVQ.

Initially, all segments affiliated to one phone g_i need to be padded to the identical length. For instance, this study extracts n segments for phone [AW] from all utterances

and their lengths vary from l_{min} to l_{max} because of different pronunciation habits or allophones. l_{min} is the minimum length, while l_{max} is the maximum. Consequently, $\boldsymbol{\mathcal{K}}_i = \{\boldsymbol{k}^{(1)}_{1\to l_1}, \boldsymbol{k}^{(2)}_{1\to l_2}, \cdots, \boldsymbol{k}^{(n)}_{1\to l_n}\}$ is converted to $\boldsymbol{\mathcal{K}}^{'}_i = \{\boldsymbol{k}^{\prime(1)}_{1\to l_{max}}, \boldsymbol{k}^{\prime(2)}_{1\to l_{max}}, \cdots, \boldsymbol{k}^{\prime(n)}_{1\to l_{max}}\}$. Afterwards, the tth first frame of $\boldsymbol{k}^{\prime(t)} \in \boldsymbol{\mathcal{K}}^{'}_i 1 \leq t \leq n$ is fed into the DNNVQ in turn (There are n first frames from n padded segments in total.). As the output, the output vector $\boldsymbol{m}_t 1 \leq t \leq n$ is obtained in turn

$$\boldsymbol{m}_t = [\, p(m^1_1|k^{\prime}_1{}^{(1)}) p(m^2_1|k^{\prime}_1{}^{(1)}) \cdots p(m^{N_{\text{clu}}}_1|k^{\prime}_1{}^{(1)}) \,]. \tag{3.8}$$

After getting all the outputs of the first frames of $\boldsymbol{k}^{\prime(t)} \in \boldsymbol{\mathcal{K}}^{'}_i 1 \leq t \leq n$, $\boldsymbol{\mathcal{A}}^{n\times N_{\text{clu}}}$ is acquired as

$$\begin{aligned}
\boldsymbol{\mathcal{A}} &= [\boldsymbol{m}_1 \quad \boldsymbol{m}_2 \quad \cdots \quad \boldsymbol{m}_n]^T \\
&= \begin{bmatrix} p(m^1_1|k^{\prime}_1{}^{(1)}) & p(m^2_1|k^{\prime}_1{}^{(1)}) & \cdots & p(m^{N_{\text{clu}}}_1|k^{\prime}_1{}^{(1)}) \\ p(m^1_2|k^{\prime}_1{}^{(2)}) & p(m^2_2|k^{\prime}_1{}^{(2)}) & \cdots & p(m^{N_{\text{clu}}}_2|k^{\prime}_1{}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ p(m^1_n|k^{\prime}_1{}^{(n)}) & p(m^2_n|k^{\prime}_1{}^{(n)}) & \cdots & p(m^{N_{\text{clu}}}_n|k^{\prime}_1{}^{(n)}) \end{bmatrix}, \\
&= [\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \cdots \quad \boldsymbol{q}_{N\text{clu}}]
\end{aligned} \tag{3.9}$$

where

$$\boldsymbol{q}_{\text{j}} = [q_{1\text{j}} \quad q_{2\text{j}} \quad \cdots \quad q_{n\text{j}}]^T, 1 \leq j \leq N_{clu}. \tag{3.10}$$

$\boldsymbol{m}_t 1 \leq t \leq n$ is the row vector of matrix $\boldsymbol{\mathcal{A}}^{n\times N_{\text{clu}}}$ and it denotes the output vector of the first frame of the tth segment. $\boldsymbol{q}_j$ is the column vector of matrix $\boldsymbol{\mathcal{A}}^{n\times N_{\text{clu}}}$, and it represents the posterior probabilities of all first frames on fenone j. By the element-wise product of $\boldsymbol{q}_j$, the product of the jth column $P_{\boldsymbol{q}_j}$ is obtained as

$$P_{\boldsymbol{q}_j} = \prod_{t=1}^{n} p(m^{(j)}_t|k^{\prime}_1{}^{(t)}) \quad \forall 1 \leq j \leq N_{\text{clu}}. \tag{3.11}$$

Let $\hat{j}$ be the index of the maximum value of $P_{\boldsymbol{q}_{\text{j}}}$, i.e.,

$$\hat{j} = \underset{1\leq j\leq N_{\text{clu}}}{\arg\max} P_{\boldsymbol{q}_j}, \tag{3.12}$$

then the first frame of $k^{\prime(t)} 1 \leq t \leq n$ is quantized to the $\hat{j}$th neuron and the corresponding fenone is $f_{\hat{j}}$. Due to the risk of underflow, the logarithm is employed, then

$$\hat{j} = \underset{1\leq j\leq N_{\text{clu}}}{\arg\max} \sum_{t=1}^{n} \log(p(m^j_t|k^{\prime}_1{}^{(t)})). \tag{3.13}$$

Similarly, this study repeats the same steps for the remaining $l_{max} - 1$ frames chronologically, and the whole fenone sequence $\{f_{\hat{j}_1}\ f_{\hat{j}_2}\ \ldots\ f_{\hat{j}_{l_{max}}}\}$ for phone $g_i 1 \leq i \leq N_g$ is acquired. Importantly, the fenone sequence $\{f_{\hat{j}_1}\ f_{\hat{j}_2}\ \ldots\ f_{\hat{j}_{l_{max}}}\}$ could contain several

identical fenones. Subsequently, this study eliminates all successive duplicated fenones in the obtained fenone sequence to generate the final fenonic baseform for a phone. Hence the fenonic baseform is refined from the corresponding fenone sequence, without any duplicated fenone. The whole procedure of NFBG is illustrated in Alg. 1. Taking phone [AW] as an example, the length of padded segments of the phone [AW] is 20 and the generated fenone sequence is {27 27 27 27 27 27 27 27 27 27 27 27 27 92 92 92 92 92 5 5 5}. Thereby, this study merges these adjacent identical fenones, and in consequence, the fenonic baseform of phone [AW] appears to be {27 92 5}. Fig. 3.3 demonstrates the NFBG process of phone [AW].

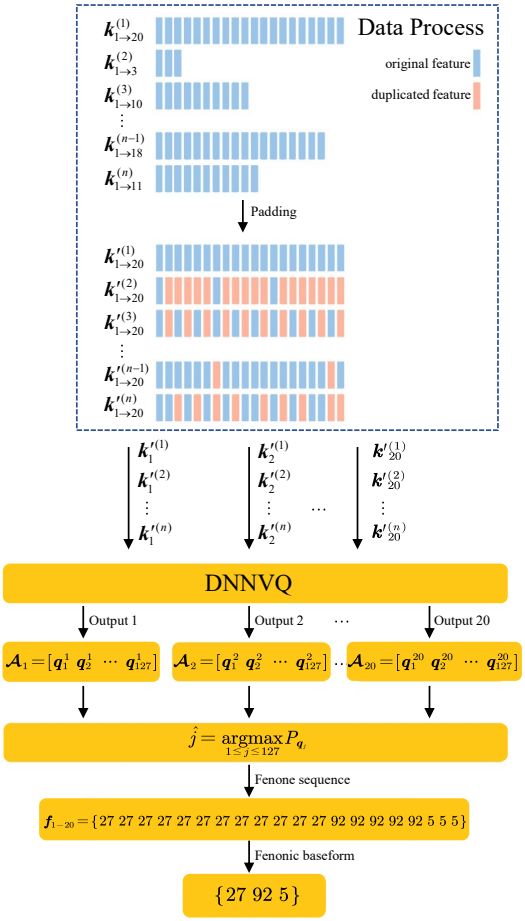


Figure 3.3: Illustration of the NFBG process of phone [AW]. The upper half is the padding process, where the segment lengths of phone [AW] vary from 3 to 20, and the pink frames are the duplicates of the blue original frame before. The lower half is the process of NFBG, which starts with the 1st - 20th frames being fed into DNNVQ in turn and ends up with compacting the 20-frame fenone sequence into the fenonic baseform.

Algorithm 1: Pseudo-code for dynamic baseform growing.

1.25

Input: padded segments $\boldsymbol{\mathcal{K}}_i^{'} = \{\boldsymbol{k}_{1\to l_{max}}^{\prime(1)}, \boldsymbol{k}_{1\to l_{max}}^{\prime(2)}, \cdots, \boldsymbol{k}_{1\to l_{max}}^{\prime(n)}\}$ of phone $g_i 1 \leq i \leq N_{\mathrm{g}}$;

Output: the corresponding fenone of frame l;

Training DNNVQ with $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{x}_i, y_i^*)\}_{i=1}^N$;

while $1 \leq i \leq l_{max}$ **do**

  for $t = 1; t \leq n; t++$ **do**

    the output $\boldsymbol{m}_t$:

    $\boldsymbol{m}_t = [\, p_t^1 p_t^2 \cdots p_t^{N_{\mathrm{clu}}} \,]$

  end

  the score of the tth frame on N_{clu} clusters:

  $P_{\boldsymbol{q}_j} = \prod_{t=1}^{n} p_t^j \quad 1 \leq j \leq N_{\mathrm{clu}}$;

  if $\hat{j} = \underset{1\leq j\leq N_{clu}}{\arg\max}\, P_{\boldsymbol{q}_j}$ **then**

    the tth frame of $g_i \xleftarrow{fenone} \mathrm{f}_{\hat{j}}$

  end

  $i = i + 1$;

end

the fenone sequence of g_i:

$\{f_{\hat{j}_1}, f_{\hat{j}_2}, \cdots, f_{\hat{j}_{l_{max}}}\}$;

merge the successive duplicated fenones;

the fenonic baseform of g_i:

$\{f_{\hat{j}_1}{}', f_{\hat{j}_2}{}', \cdots, f_{\hat{j}_l}{}'\} 1 \leq \hat{j}_l^{\,\prime} \leq N_{\mathrm{clu}}$;

return fenonic baseform

3.4.4 Elementary Markov Model for Fenones

The HMM of a phone is constructed by concatenating the elementary Markov model of the fenones in its fenonic baseform. The fenonic baseform merely indicates the number of states of an HMM, but the topology remains undetermined. This study investigates three sorts of topology for the elementary Markov model: ergodic, Bakis-type [15], and Vintsyuk-type [284], as depicted (a), (b), and (c) in Fig. 3.4. Each state in the ergodic topology can transit to every other state in a single step. Thus the ergodic topology possesses the highest flexibility as well as the highest complexity. By contrast, every state in the Bakis-type [15] topology can only transit to itself or the next one, but the Bakis-type topology possesses the advantage of simplicity. The Vintsyuk-type [284] topology is a compromise of the former two, as it allows a maximum shortening by a factor of two. It contains model parameters and preserves the flexibility by a skip arc simultaneously. This study executes ablation experiments on the efficacy of each topology.

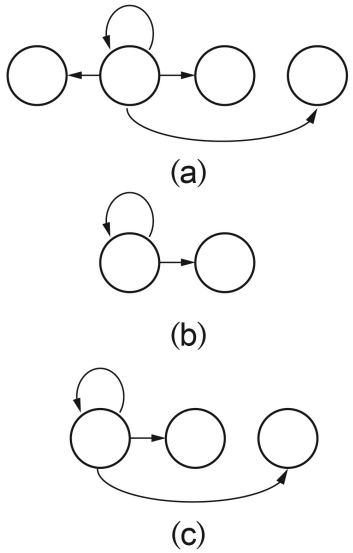


Figure 3.4: Examples of Markov models for fenones. (a) ergodic, (b) Bakis-type [15], (c) Vintsyuk-type [284].

3.5 Experimental Setups

3.5.1 Corpora and Features

The proposed approach is tested on TIMIT [68] and TEDliumv2 [233]. TIMIT contains a total of 6300 sentences (5.4 hours), consisting of 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. The 462-speaker training set is used. All SA records (i.e., identical sentences for all speakers in the database) are removed as they could bias the results. After realigning the training utterances against 48

monophones, 220535 segments are obtained as the training dataset. Results are reported using the 24-speaker core test set. All of experiments apply a bigram language model over phones, estimated from the training set.

TEDliumv2 contains 207-hour training data from TED talks, consisting of male and female speakers, native and non-native speakers, and speakers from all age ranges. The contents of the data cover various fields. After the realignment of training utterances against 40 monophones, 8439059 segments are obtained as the training dataset. All recognition results are reported on the heavily pruned 4-gram language model and the dictionary with roughly 152 K words and 160 K pronunciations released by [233].

As for features, this study utilizes 12-dim MFCC along with the additional energy feature and their first and second temporal derivatives, hence 39-dim MFCC feature vectors in total. Cepstral mean normalization is employed.

3.5.2 DNNVQ Setups

The DNNVQ system is trained on the Tensorflow library [1]. The network is composed of four fully-connected hidden layers with 512 neurons and the ReLU activation function followed by a batch-normalization [110] layer, respectively. The dropout [261] layer is discarded due to worse results. A subsequent fully-connected layer with the ReLU activation function is deployed as the output layer. This study optimizes the DNNVQ with Adam optimizer [121]. An exponentially decaying schedule starts with an initial learning rate of 0.01 and halves the rate when the improvement of the frame accuracy between two successive epochs on a cross-validation set stops.

3.5.3 Baseline

The training of the GMM-HMM and the DNN-HMM baseline systems is pursuant to the Kaldi example recipe [204]. They are trained on the MFCC features described in Section 3.5.1. The HMM structure adopted on TEDliumv2 is 3-state left-to-right structure for vocal phones while 5-state structure for "silence" and "noise", leading to 127 Probability Density Functions (PDFs). In contrast, the HMM structure of all 48 monophones is the identical 3-state left-to-right structure on TIMIT, resulting in 144 PDFs. GMM-HMM system is trained from scratch, and 1 K Gaussian models are deployed in total; in the DNN-HMM system, the DNN has four hidden layers, each of which has 512 neurons. The number of nodes of the final layer is determined by the number of PDFs, which is also the number of states of the original HMMs. The DNN is initialized randomly with weights drawn from $\mathcal{N}(0, 0.01)$ and the uniform bias drawn randomly from $\mu(-4.1, -3.9)$. Stochastic gradient descent is utilized to minimize the cross-entropy, with the minibatch size of 512 frames. The learning rate is set at 0.0015 initially and decays to 0.00015 progressively. All baseline systems are implemented in the Kaldi toolkit [204].

Primary structures are chosen for baseline systems to force focus on the impact of HMM structures. Besides, the effectiveness of the proposed method in advanced systems will also be presented later.

Table 3.1: Fenonic baseforms for every monophone in Tedliumv2 corpus.

Phone	Fenonic baseform	Phone	Fenonic baseform	Phone	Fenonic baseform	Phone	Fenonic baseform
SIL	17 124 13 17	AW	27 92 5	DH	43 79 28	G	97 52
AA	57 102 24	AY	20 47	EH	111 12 105	HH	86 49
AE	27 82 23	B	125 45 79	ER	84 80 48	IH	4 18 44
AH	6 107 2	CH	38 94 1 40 28	EY	3 64 39 81	IY	16 67 32
AO	87 93 24	D	51 97 42	F	88 0 21	JH	94 40
Phone	**Fenonic baseform**	**Phone**	**Fenonic baseform**	**Phone**	**Fenonic baseform**	**Phone**	**Fenonic baseform**
K	69 108	OW	114 112 46	SH	29 1 40	V	25 68 36
L	106 7 34	OY	87 93 46 81	T	38 121 53	W	122 26 33
M	72 90 104	P	65 115 49	TH	76 0 21 40	Y	101 75
N	77 31 8	R	91 80 63	UH	6 107 2 21	Z	83 60 55
NG	10 56 89	S	73 59 70 11	UW	99 71	ZH	29 1 40

Table 3.2: Fenonic baseforms for every monophone in TIMIT corpus.

Phone	Fenonic baseform	Phone	Fenonic baseform	Phone	Fenonic baseform	Phone	Fenonic baseform
AA	108 26 44 86	AY	100 10 112	DX	36 67	EY	42 50 126 66
AE	136 23 77	B	62 3	EH	109 90 101 32	F	89 11 87 122
AH	142 113 80	CH	37 127 133 25	EL	2 124	G	56 95
AO	103 93 69	CL	28 107	EN	17 96 88	HH	76 46
AW	136 6 116	D	105 56	EPI	79 60 12	ICH	92 35 106
AX	141 58 1	DH	128 18 51	ER	50 5 49	IX	135 57 73
Phone	**Fenonic baseform**	**Phone**	**Fenonic baseform**	**Phone**	**Fenonic baseform**	**Phone**	**Fenonic baseform**
IY	114 24 65	NG	102 83 95	SH	16 133 25	V	98 72
JH	94 25 143	OW	137 97 20	SIL	13 104 110 13	VCL	68 117 70
K	138 46	OY	103 22 112	T	8 123	W	38 3
L	4 140	P	132 60 9	TH	34 53 87 55	Y	19 48
M	115 118	R	7 5	UH	142 113 80 139	Z	29 75
N	82 88	S	33 91 71	UW	120 39 74	ZH	16 133 25

3.6 Fenonic Baseform Results

This section exhibits the obtained fenonic baseforms for phones included in Tedliumv2 (Table 3.1) and TIMIT (Table 3.2) corpora when there are 127 VQ prototypes in the case of Tedliumv2 while 144 VQ prototypes in the case of TIMIT, respectively. There are 40 phones in Tedliumv2 corpus and 48 phones included in TIMIT corpus. In TEDliumv2 corpus, 1 phone gains five states, 6 phones gain four states, 7 phones gain two states, and the remained phones gain three states. In contrast, 8 phones gain four states, 17 phones gain two states, and the rest gain three states, in TIMIT corpus. Consequently, there are 94/127 active fenones in Tedliumve, while 104/144 active fenones in TIMIT. Hence the average numbers of states are 3.025 in Tedliumv2, while 2.8125 in TIMIT. Then it is safe to draw the conclusion that even if the average number of states does not change much, the state sharing relations among phones reduce parameters.

After analysing the fenonic baseform results, two conclusions can be drawn. First, the shared states tend to appear in the same or similar location of phones. For instance, in TEDliumv2, state /40/ is shared by phones [CH], [JH], [SH], [TH], and [ZH]. It appears

as the last state in cases of [JH], [SH], [TH], and [ZH], while as the penultimate state in the case of [CH]. The same phenomenon also appears in TIMIT, where [CH], [JH], [SH], and [ZH] share the last state /25/, and [CH], [SH], and [ZH] even share the last two states, /133/ and /25/. This pattern reveals that a specific state always tends to express a specific part of the phone (the beginning, the middle, or the end). Second, phones with the same suffix tend to share the state in the ending, while phones with the same prefix tend to share the state at the beginning or in the middle. For instance, in TEDliumv2, [EY] and [OY] share the last state /81/, while in TIMIT, [AE] and [AW] share the first state /136/. Nevertheless, the latter half of the pattern is not as common as the former half, and thus the suffix is more decisive than the prefix for the pronunciation of a specific phone.

Besides, there are some interesting phenomenons in the resultant fenonic baseforms which are worth highlighting here for any possible inspiration to the readers. Firstly, some phones seemingly irrelative share states. For example, [TH] and [F] share two states (/0/ and /21/) in TEDliumv2, and they share one state (/87/) in TIMIT; [B] and [DH] share the state /79/ in Tedliumv2; and [K] and [HH] share the state of /46/ in TIMIT. Secondly, some phones possess a high similarity between them in terms of the state. For instance, it surprised the author that the states of [SH] are identical to those of [ZH]. This phenomenon appears in both corpora, so the author believes that it is not a coincidence. A similar pattern also falls on the case of [AH] and [UH], which share three states in both corpora. For these counter-intuitive sharing relationships, the author believes that it reveals a sort of interior relevance between those phones.

Additionally, there are two phenomenons emerging in the process of compacting the fenonic sequence to the fenonic baseform which are worth noting. For one thing, [SIL] processes the repetitive fenone in its fenonic baseform. For example, the fenonic baseform of [SIL] in TEDliumv2 is derived from its original fenone sequence {17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 124, 124, 124, 124, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17}. The author tried to simplify its fenonic baseform as {17, 124, 13} and added a skipping-back arc from state 13 to state 17. However, the result turned to be slightly worse than keeping it as {17, 124, 13, 17} with the uniform Bakis-type topology for every state. For another, some phones process a singleton state occurrence. For instance, the fenonic sequence of phone [ZH] in TEDliumv2 is {29, 29, 29, 29, 29, 29, 29, 29, 1, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40}. The author recognized the single occurrence of state 1 as a fortuity and believed that it should have been 29 or 40 in that location. However, the comparison experiments between setting the fenonic baseform of [ZH] as {29, 40} and {29, 1, 40} indicate that every remained state should be respected even though it is a singleton.

3.7 Experimental Results

For the experiments, this study first conducts ablation tests on different elementary HMM topologies for the fenones and the dimension of DNNVQ prototypes. Thereafter, this study validates the proposed NFBG-based HMM structure in both monophone and triphone systems, with context-independent and context-dependent inputs, and in already advanced systems.

3.7.1 Effects of the Elementary HMM Topology of the Fenones

To execute the test in a fair comparison, this study controls the nodes of the output layer to stay the same as their respective Kaldi recipe (i.e., 127 nodes for Tedliumv2 while 144 nodes for TIMIT). This rules out the possibility that any performance improvement would come from different dimensions of the output layer. Table 3.3 shows the effect of different HMM topologies for the fenone. Three sorts of HMM topologies in the table correspond to three HMM model examples in Fig. 3.4. It is apparently shown that the basic left-to-right topology outperforms the ergodic topology considerably, consistent with the observation in [2]. The full ergodic model tends to overfit the training data since it has large amounts of parameters and the resultant high model complexity, resulting in a poor generalization. The margin between the Bakis topology [15] and Vintsyuk topology [284] is more pronounced on TIMIT than that on TEDlium. The author believes that it is owing to less training data on TIMIT. As the Bakis-type topology outperforms both the ergodic topology and the Vintsyuk topology, all the subsequent results in this paper are obtained using it as fenone topology.

Table 3.3: WER[%] on TIMIT and TEDliumv2 for different elementary HMM topologies in monophone systems.

Corpus	HMM topology	GMM-HMM	DNN-HMM
TEDliumv2	ergodic	59.6	45.9
	Bakis [15]	**54.5**	**36.9**
	Vintsyuk [284]	54.7	37.2
TIMIT	ergodic	35.6	31.1
	Bakis [15]	**30.8**	**23.1**
	Vintsyuk [284]	31.3	24.3

3.7.2 Effects of the Number of DNNVQ Prototypes

As NFBG introduces state tying, different numbers of DNNVQ prototypes lead to a different number of HMM states in the NFBG-based model. Accordingly, the model complexity and model strength vary. Fig. 3.5 highlights the effect of setting different numbers of DNNVQ prototypes. The emerging HMM states of $N_{\text{clu}} \in$

$\{127, 250, 350, 450, 700, 1000\}$ are $\{94, 110, 116, 120, 132, 199\}$ on Tedlium, while the HMM states of $N_{\text{clu}} \in \{144, 250, 350, 450, 700, 1000\}$ are $\{110, 121, 134, 139, 144, 220\}$ on TIMIT. The recognition performance is gradually improved before $N_{\text{clu}} = 250$ on both TEDliumv2 and TIMIT; thereafter, it exposes a downward trend on both corpora. Especially on TIMIT, the curve soars from $N_{\text{clu}} = 450$, where the model underfits due to large amounts of parameters and insufficient data. Therefore, $N_{\text{clu}} = 250$ is the default number of DNNVQ prototypes for the subsequent experiments.

3.7.3 NFBG Validation in Monophone Systems

NFBG introduces a natural state tying in the monophone system. Consequently, the NFBG-based HMM structure reduces the number of PDFs from 127 to 110 on TEDliumv2 while 144 to 121 on TIMIT, leading to ∼15 % fewer HMM structure's parameters compared to the baseline. Additionally, fewer parameters also make the HMM topology more resistant to overfitting. After constructing the NFBG-based HMM, the GMM-HMM system is trained from scratch using linear alignments. For the training of the DNN-HMM system, the Viterbi algorithm is applied upon the outputs of the DNNVQ and the transition probabilities are calculated manually. Then the realigned outputs of DNNVQ are used as the gold-standard labels for the subsequent DNN training. In addition, the weights of DNNVQ are transferred as initialized weights for the DNN. As presented in Table 3.4, on TEDliumv2, NFBG-based HMM delivers 2.5 % relative improvements in the GMM-HMM system, while 13.8 % in the DNN-HMM system with a 15 % smaller parameter scale. Comparatively, improvements are more distinct on TIMIT, which are 5.8 % and 14.8 % relative in the GMM-HMM system and DNN-HMM system, respectively, with more than 15 % fewer parameters.

Table 3.4: Impacts of the NFBG-based HMM structure in monophone systems. Results are in WER[%].

Corpus	HMM structure	GMM-HMM	DNN-HMM
TEDliumv2	baseline	55.9	42.8
	this study	**54.5**	**36.9**
TIMIT	baseline	32.7	27.1
	this study	**30.8**	**23.1**

3.7.4 NFBG Validation with Context-Dependent Inputs

This study also examines the effectiveness of NFBG-based HMM in the system with context-dependent inputs. By setting $N_{\text{spl}} = m$, inputs are spliced over $(2m + 1)$ frames. It is worth noting that the setups of the context-dependent system stay in accordance with the monophone system (Section 3.7.3) except for the inputs. This study only displays results in the DNN-HMM system. Table 3.5 displays that the NFBG-based

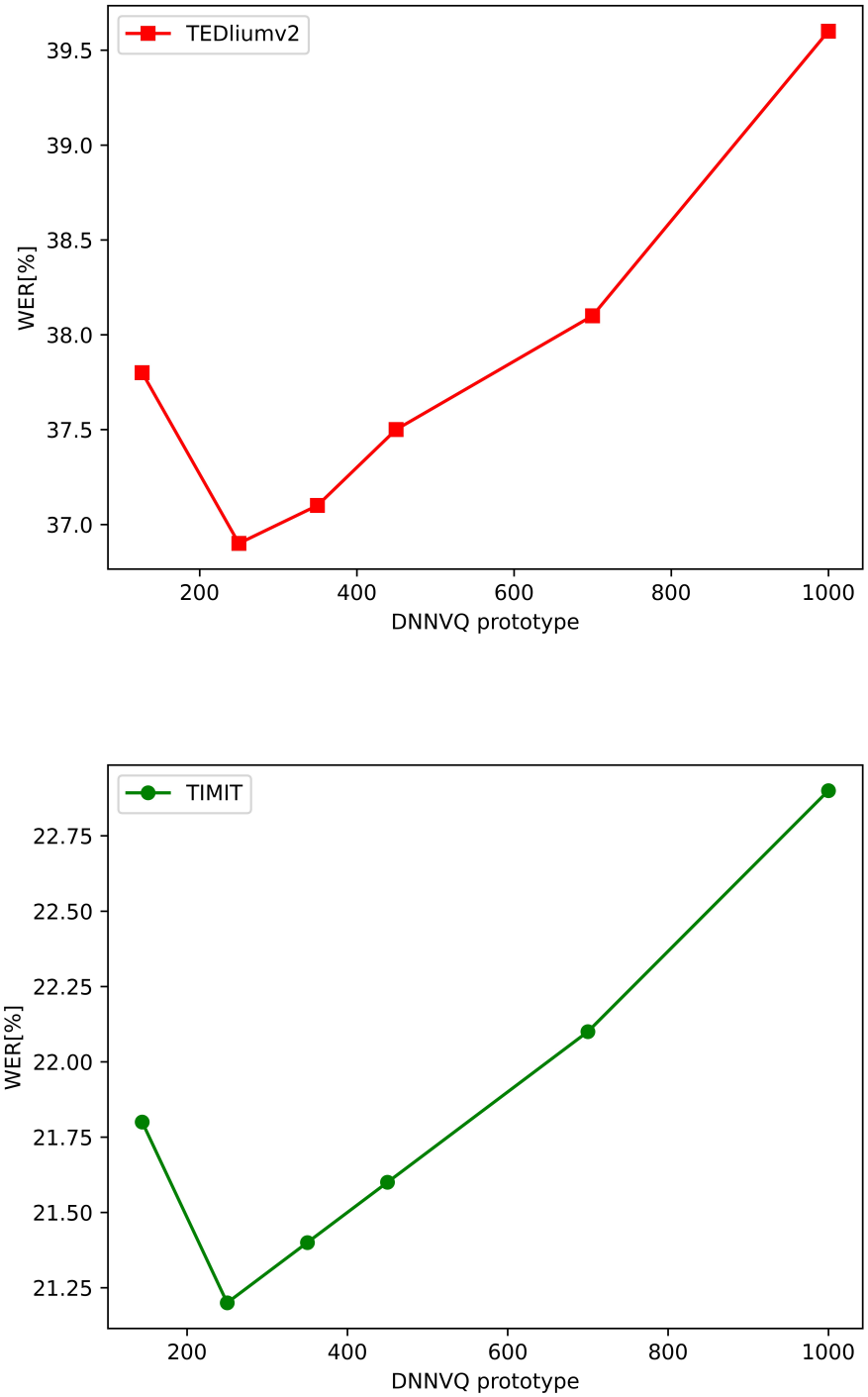


Figure 3.5: Evolution of WER[%] along the number of DNNVQ prototypes on TEDliumv2 and TIMIT, respectively.

HMM outperforms the corresponding baseline system in all $N_{\text{spl}} \in \{0, 1, 2, 3, 4\}$ settings on both corpora. Especially on TEDliumv2, the NFBG-based HMM yields 17.1 % relative improvements when $N_{\text{spl}} = 1$. However, constant improvements cannot be obtained by increasing the input dimension. When $N_{\text{spl}} > 2$, there is no further improvement.

Table 3.5: Impacts of the NFBG-based HMM structure with context-dependent inputs in monophone systems. Results are in WER[%].

Corpus	HMM structure	N_{spl}				
		0	1	2	3	4
TEDliumv2	baseline	42.8	36.3	35.3	35.1	35.1
	this study	**36.9**	**30.1**	**29.3**	**29.3**	**29.3**
TIMIT	baseline	27.1	23.6	23.2	23.1	23.0
	this study	**23.1**	**21.7**	**21.1**	**21.1**	**21.2**

3.7.5 NFBG Validation in Triphone Systems

The triphone generation leverages the Kaldi recipe[1]. Table 3.6 shows that on Tedliumv2, the relative improvement attained by the NFBG-based HMM is 1.8 % in the GMM-HMM system. Similarly, in the DNN-HMM system, the NFBG-based HMM improves WERs in all splice conditions while the most significant improvement appears in the $N_{\text{spl}} = 1$ case, which is 3.3 %. As for TIMIT, overall improvements are more distinct compared to TEDliumv2, and the most predominant improvement also appears in $N_{\text{spl}} = 1$, which is 4.5 %. Similar to Section 3.7.4, there is no further improvement when $N_{\text{spl}} > 2$.

3.7.6 NFBG Validation in Adverse Environments

An intrinsic advantage of HMM-based hybrid systems is their robustness in noisy environments, and thus, this study also empirically investigates whether NFBG-based HMMs perform better than standard HMMs from this respect. This study executes experiments in the triphone DNN-HMM system, and since the overall improvements on TIMIT are more distinct than TEDliumv2 in triphone systems, this study chooses TIMIT as the corpus for clean utterances. For the context-dependent information, N_{spl} is set to 2, because there is no further improvement when $N_{\text{spl}} > 2$. For the noisy data, this study artificially mixes clean utterances in TIMIT with intrusions from the NOISEX-92 dataset [276] to generate noisy training, validation, and test datasets in the same manner. This study starts with training both the baseline system and the proposed system with the clean training set of TIMIT. Thereafter, both systems are

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/wsj/s5/steps

trained with the generated noisy training set. Table 3.7 displays the performance of the standard HMM and NFBG-based HMM when interfered with experimental noises. Generally speaking, the performances of clean-trained systems plunge in the noisy test set, and noise training enhances the system robustness to a great degree. Nevertheless, it is worth noting that NFBG-based HMM performs always better than the standard HMM when being trained either with clean or noisy utterances, and the performance improvement is higher in adverse environments. For instance, when trained with noisy utterances, the performance improvement is 2.5 % in clean test set and 7.1 % in noisy test set. This phenomenon corroborates that the NFBG-based HMM possesses stronger robustness.

Table 3.6: Impacts of the NFBG-based HMM structure in triphone systems. Results are in WER[%].

TEDliumv2

GMM-HMM		DNN-HMM				
N_{spl}	0	0	1	2	3	4
baseline	27.3	22.1	21.5	20.2	20.0	19.9
this study	**26.8**	**21.6**	**20.8**	**19.7**	**19.7**	**19.7**

TIMIT

GMM-HMM		DNN-HMM				
N_{spl}	0	0	1	2	3	4
baseline	25.6	20.4	20.2	19.7	19.6	19.6
this study	**24.9**	**19.8**	**19.3**	**19.2**	**19.2**	**19.3**

3.7.7 Comparisons with Advanced Models

This section configures the best HMM structure for published state-of-the-art systems on both TEDliumv2 and TIMIT. This study chooses three representative systems for Tedliumv2: the time delay neural network (TDNN) [198], SincNet architecture [222], and the improved RWTH ASR system with SpecAugment [330]. The TDNN [198] models long term temporal dependencies with training times comparable to standard feed-forward DNNs. The network uses sub-sampling to reduce computation during training. It shows a

Table 3.7: Impact of the NFBG-based HMM structure in adverse environments. Results (WER[%]) are reported in triphone DNN-HMM systems on TIMIT. $N_{\text{spl}} = 2$.

ASR	Training	WER	
		clean	noisy
baseline	clean	19.7	72.1
	noisy	19.8	25.5
this study	clean	**19.2**	**69.2**
	noisy	**19.3**	**23.7**

Table 3.8: The impact of the NFBG-based HMM structures in different advanced models. Results are in WER[%].

Model	Corpus	Model Structure	Baseline	This Study
TDNN [198]	TEDliumv2	HMM-TDNN+iVectors+4-gram LM	17.9	**17.6**
SincNet [222]	TEDliumv2	CNN+layer Norm+Dropout+DNN+4-gram LM	21.8	**21.5**
RWTHv[330]	TEDliumv2	HMM-BLSTM+iVectors+SpecAugment+sMBR+Transformer LM	5.6	**5.5**
Regularization [275]	TIMIT	DNN-HMM with last layer regularization	18.3	**17.6**
IF feature [175]	TIMIT	DNN-HMM with MFCC + IF features	17.7	**17.2**

relative WER improvement of 6 % on both Switchboard and TEDlumv2 corpora. SincNet [222] is a novel CNN architecture that encourages the first convolutional layer to discover more meaningful filters. In contrast to standard CNNs, which learn all elements of each filter, only low and high cutoff frequencies are directly learned from data with SincNet. Experimental results show that SincNet converges faster and performs better than a standard CNN on raw waveforms. The improved RWTH ASR system with SpecAugment [330] is a complete training pipeline to build a state-of-the-art hybrid HMM-based ASR system on the TEDliumv2 corpus. Data augmentation using SpecAugment [191] is successfully applied therein. Their best system achieves a 5.6 % WER on the test set, which outperforms the previous state-of-the-art by 27 % relative.

Besides, this study also chooses two systems on TIMIT: DNN with a regularization post-layer [275] and DNN with instantaneous frequency features [175]. Vaněk et al. [275] propose a regularization post-layer that can be combined with prior techniques, and it brings additional robustness to the DNN. On the TIMIT benchmark task, the adoption

of the regularization post layer gives better results than DNNs with DBN pre-training. Nayak et al. [175] extract features from its time derivative, referred to as instantaneous frequency (IF), to solve the inevitable phase wrapping problem. The combination of IF and MFCC features based systems, using minimum Bayes risk decoding, provides a relative improvement of 8.7 % over the baseline system.

Table 3.8 presents the effectiveness of the NFBG-based HMM in the aforementioned state-of-the-art models. As we observe, these advanced systems are hard to be further improved since they are already exceedingly-optimized. On TEDliumv2, the NFBG-based HMM achieves a 0.3 % absolute improvement in both the SincNet system and TDNN. Furthermore, the absolute improvement in the improved RWTH ASR system is 0.1 %. In comparison, on TIMIT, the NFBG-based HMM performs better. It delivers 3.8 % and 2.8 % relative improvements in the regularization post-layer and IF feature systems, respectively.

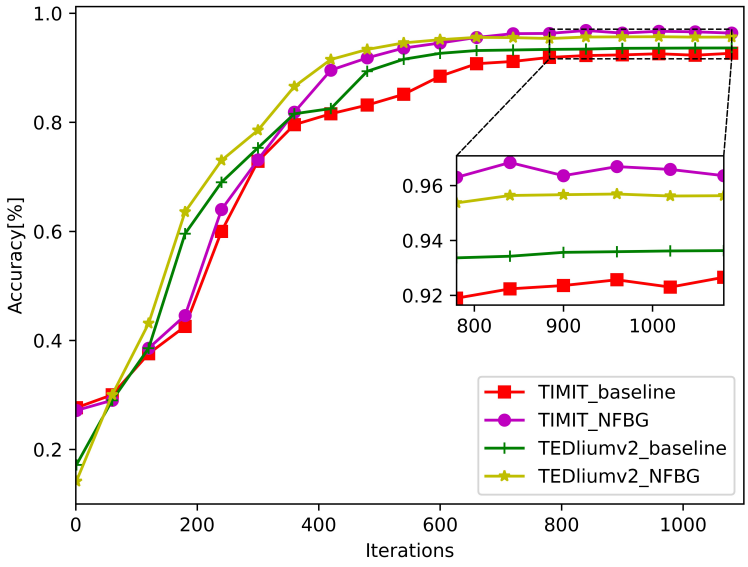


Figure 3.6: Training accuracy and converge speed on both TEDliumv2 and TIMIT.

3.8 Discussion

From the above results, it is seemingly that the NFBG-generated HMM structure yields improvements in both monophone and triphone systems on both TEDliumv2 and TIMIT. Overall, the improvement on TIMIT is more predominant than that on TEDliumv2. For instance, in the monophone hybrid system, the NFBG-based HMM achieves 13.8 % relative improvement on TEDliumv2 while the counterpart of TIMIT is 14.8 %. Besides, in the triphone hybrid system, the NFBG-based HMM outperforms the baseline by 3.3 % on TEDliumv2 while that is 4.5 % on TIMIT. The author assumes that since the

amount of training data of TIMIT is fewer than that of TEDliumv2, TIMIT benefits more from the reduction of parameters. This section provides evidence from the aspect of the convergence speed and the classification performance of the network with and without NFBG-based HMM.

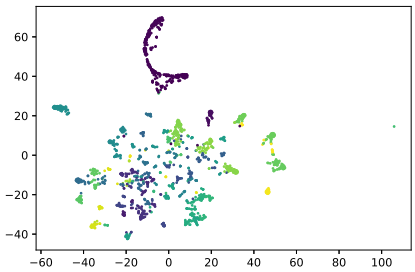
As displayed in Fig. 3.6, introducing NFBG for the HMM construction leads to the accuracy increment on both TEDliumv2 and TIMIT. Additionally, the effectiveness is more distinctive on TIMIT. Besides, the system with the proposed HMM converges faster on both corpora. Moreover, TEDliumv2 is even faster than TIMIT, since there are 110 HMM states on TEDliumv2 while 121 HMM states on TIMIT.

Besides, this study also chooses t-distributed stochastic neighbor embedding (t-SNE) [149] to visualize the classification ability of the network with and without the proposed HMM structure. To begin with, this study extracts one utterance from the test sets of both TEDliumv2 and TIMIT corpora. Furthermore, a comparison on more test utterances from both corpora is also made. This study sets the number of prototypes of the DNNVQ the same as the number of PDFs in the baseline model. The perplexity is set to be 30. From every pair of comparisons as depicted in Fig. 3.7, it is apparent that the employment of NFBG-based HMM reinforces networks' classification ability.

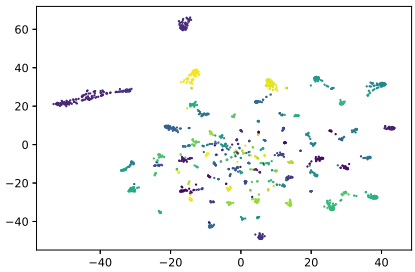
3.9 Summary

This chapter presents a concise and data-driven model adaption approach for HMM-based ASR systems. The proposed algorithm allows the data to reveal its dynamic structure without external assumptions and with a low computational cost. This study conducts ablation tests on different HMM topologies and the number of DNNVQ prototypes. Besides, this study validates the proposed algorithm on TEDliumv2 and TIMIT in both monophone and triphone systems. Importantly, its robustness in adverse environments is also studied. Empirical results indicate that the proposed approach improves both the monophone system's and the triphone system's performances. The margin on TIMIT, a corpus with a small amount of training data, is more remarkable. Furthermore, it not only improves the ASR performance of the model, but also enforces the model robustness when interfered by ambient noises. Despite the limited improvements in the already highly-optimized systems, it reduces the parameters of those systems by 15 %. It is safe to conclude that this robust and lightweight HMM structure possesses considerable potentials in various realistic situations, e.g., the keyword spotting task in the always-on and battery-powered application scenarios for smart devices, with severe constraints on hardware resources and power consumption; the task of low-resource speech recognition; the classification task on portable devices.

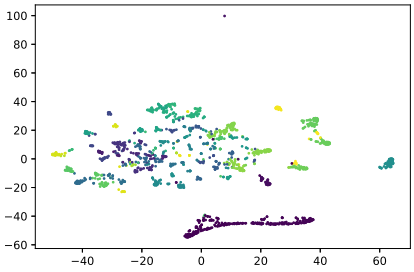
Besides the model adaption, as the back-end technique, front-end techniques also play an important role in robust ASR, which will be discussed in the next chapter.



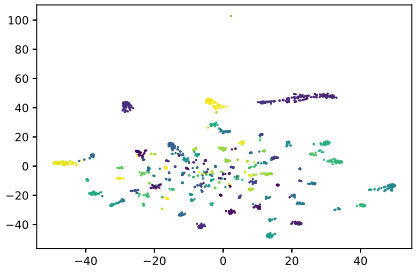
(a) 2D t-SNE visualisation of one utterance of TIMIT from the baseline model.



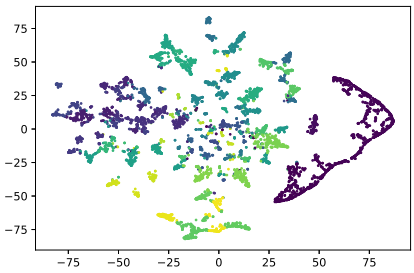
(b) 2D t-SNE visualisation of one utterance of TIMIT from this study.



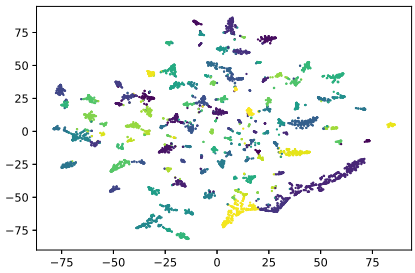
(c) 2D t-SNE visualisation of one utterance of TEDliumv2 from the baseline model.



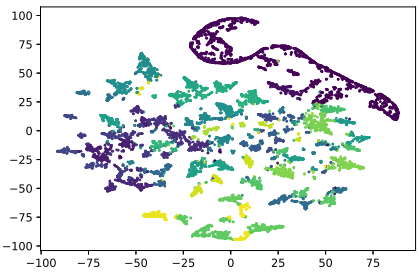
(d) 2D t-SNE visualisation of one utterance of TEDliumv2 from this study.



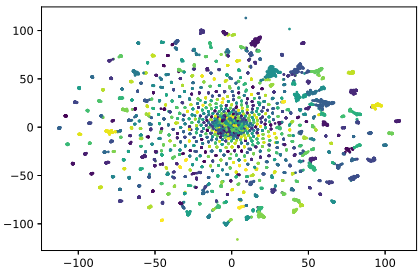
(e) 2D t-SNE visualisation of five utterances of TEDliumv2 from the baseline model.



(f) 2D t-SNE visualisation of five utterances of TEDliumv2 from this study.



(g) 2D t-SNE visualisation of fifteen utterances of TIMIT from the baseline model.



(h) 2D t-SNE visualisation of fifteen utterances of TIMIT from this study.

Figure 3.7: 2D t-SNE visualisation from the baseline model and the proposed model. Horizontal axis: the 1st dimension of t-SNE; vertical axis: the 2nd dimension of t-SNE.

"Humanity needs practical men, who get the most out of their work, and, without forgetting the general good, safeguard their own interests. But humanity also needs dreamers, for whom the disinterested development of an enterprise is so captivating that it becomes impossible for them to devote their care to their own material profit."

— Marie Salomea Skłodowska Curie

4

Front-End Techniques for Robust Automatic Speech Recognition

This chapter will discuss the front-end techniques of robust ASR systems. Common front-end techniques include speech enhancement, speech separation, robust feature extraction, etc. This thesis focuses on the speech enhancement approaches. Speech enhancement aims to improve speech intelligibility and quality in adverse environments by transforming the interfered speech to its original clean version [142]. Speech enhancement can serve as a front-end for downstream speech-related tasks, e.g., robust speech recognition [300], speaker identification [266], speech emotion recognition [8], etc. In addition, it is also applied successfully in communication systems, e.g., speech coding [329], hearing aids [226, 128] and cochlear implants [73, 287]. Classic speech enhancement methods are the Wiener filter [140], time-frequency masking [173, 293, 180], signal approximation [302, 59], spectral mapping [179], etc. Recently, significant improvements in speech enhancement performance have been reported for discriminative deep learning algorithms, e.g., DNNs [308], CNNs [192], and RNNs [300].

A proper speech enhancement front-end should first be lightweight; otherwise, its real-life application is limited. For instance, it cannot be applied on portable devices. Furthermore, it could result in unacceptable delay for the downstream task. Besides being lightweight, the front-end should also be efficient; otherwise, it is not conducive to the downstream task and could introduce unseen distortion to speech signals. This thesis principally discusses two front-end approaches, and both of them are based on the seminal work [193], speech enhancement GAN (SEGAN). One delves into the optimized combination of SEGAN and self-attention mechanism, and the other aims to further ameliorate SEGAN performance by the popular Sinc convolution.

4.1 Lightweight Self-Attention Augmented Generative Adversarial Networks for Speech Enhancement

4.1.1 Overview

GANs [74] have been demonstrated to be efficient for speech enhancement [193, 196, 256, 159, 138, 98], where the generative training results in fewer artifacts than discriminative models. Conforming to GAN's principles, the generator is designated for learning an enhancement mapping that can imitate the clean data distribution to generate enhanced samples. In contrast, the discriminator plays the role of a classifier that distinguishes real samples, coming from the dataset that the generator is imitating, from fake samples made up by the generator. Simultaneously, the discriminator guides the parameter updating of the generator towards the distribution of clean speech signals. Most previous attempts utilize convolutional layers as the backbone, limiting the network's ability in capturing long-range dependencies due to the convolution operator's local receptive field. To alleviate this issue, one popular solution is substituting RNNs for CNNs, but RNNs are computationally inefficient, caused by the unparallelization of their temporal iterations.

In 2017, Vaswani et al. [278] proposed the self-attention mechanism, dispensing with RNNs and CNNs entirely. Compared to discriminative deep learning models, self-attention is computationally efficient. Compared to DNNs, it possesses much fewer parameters. Compared to CNNs, it is flexible in modeling both long-range and local dependencies. Compared to RNNs, it is based on matrix multiplication, which is highly parallelizable and easily accelerated. The self-attention mechanism has been successfully used for different human–machine communication tasks [54, 200, 258, 116, 118, 117], including the speech enhancement tasks [41, 126]. Nevertheless, there are still two problems in the previous works. Firstly, some of them [126, 189] did not adopt adversarial training, which suffers from unseen distortion derived from handcrafted loss functions. Secondly, some works used discriminative models as the architecture backbone (e.g., DNN [256], CNN [202] or LSTM [98]). However, DNNs are computationally inefficient due to the huge parameter scale. CNNs command the extraordinary ability to model local information, but they experience difficulties in capturing long-range dependencies. RNNs are computationally inefficient, caused by the unparallelization of its temporal iterations.

To combine the adversarial training and the self-attention mechanism, Zhang et al. [323] propose the self-attention generative adversarial network for image synthesis, which introduces the self-attention mechanism into convolutional GANs. In their work, the self-attention module is complementary to convolutional layers and helps with modeling long-range and multi-level dependencies across image regions. In the same year, Ramachandran et al. [219] provide the theoretical basis for substituting the self-attention mechanism for discriminative models. They verify that self-attention layers can completely replace convolutional layers and achieve state-of-the-art performance on

vision tasks. Afterwards, Cordonnier et al. [45] present evidence that self-attention layers can perform convolution and attend to pixel-grid patterns similarly to convolutional layers.

Nonetheless, Yang et al. [310] suggest that the self-attention mechanism might fully attend to all elements, dispersing the attention distribution, and thus overlook the relation of neighboring elements and phrasal patterns. Guo et al. [84] indicate that the generalization ability of the self-attention mechanism is weaker than CNNs or RNNs, especially on moderate-sized datasets, and the reason can be attributed to its unsuitable inductive bias of the self-attention structure. To this end, Yang et al. [310] propose a parameter-free convolutional self-attention model to enhance the feature extraction of neighboring elements and validate its effectiveness and universality. Guo et al. [84] regard self-attention as a matrix decomposition problem and propose an improved self-attention module by introducing locality linguistic constraints. Xu et al. [307] propose a hybrid attention mechanism via a gating scalar for leveraging both the local and global information and verified that these two types of contexts are complementary to each other.

Inspired by prior works, this section presents a series of SEGANs equipped with a self-attention mechanism in three ways: first, this study deploys the stand-alone self-attention layer in a SEGAN. Next, this work employs locality modeling on the stand-alone self-attention layer. Finally, this study investigates the functionality of the self-attention augmented convolutional SEGAN. This section aims to probe the performance of a SEGAN equipped (i) with stand-alone standard self-attention layers, (ii) with stand-alone hybrid (global and local) self-attention layers, and (iii) with self-attention augmented convolutional layers. In addition, the parameter scales of these proposed models are also calculated.

Please note that there are four highlights of these works. Firstly, the adversarial training is deployed to alleviate the distortion introduced by handcrafted loss functions, and hence the enhancement module is supposed to capture more underlying structural characteristics. Secondly, self-attention layers are employed to obtain a more flexible ability to capture both long-range or local interactions. Thirdly, the locality modeling of the self-attention layer is a parameter-free method. Lastly, raw speech waveforms are utilized as inputs of the system to avoid any distortion introduced by handcrafted features.

The proposed systems are evaluated in terms of various objective evaluation criteria. Systematic experiment results reveal that equipped with the stand-alone self-attention layer, the proposed system outperforms baseline systems in terms of various objective evaluation criteria with up to 95 % fewer parameters. In addition, the locality modeling on the stand-alone self-attention layer delivers further performance improvements without increasing any parameters. Moreover, the self-attention augmented SEGAN outperforms all baseline systems and achieves the best results on SSNR and STOI of this work with acceptable increased parameters.

4.1.2 Related Work

Pascual et al. [193] open the exploration of generative architectures for speech enhancement, leveraging the ability of deep learning to learn complex functions from large example sets. The enhancement mapping is accomplished by the generator, whereas the discriminator, by discriminating between real and fake signals, transmits information to the generator so that the generator can learn to produce outputs that resemble the realistic distribution of the clean signals. The proposed system learns from different speakers and noise types, and incorporates them together into the same shared parametrization, which makes the system simple and generalizable in those dimensions.

On the basis of [193], Phan et al. [201] indicate that all existing SEGAN systems execute the enhancement mapping via a single stage by a single generator, which may not be optimal. In this light, they hypothesize that it would be better to carry out multi-stage enhancement mapping rather than a single-stage one. To this end, they divide the enhancement process into multiple stages, with each stage containing an enhancement mapping. Each mapping is conducted by a generator, and each generator is tasked to further correct the output produced by its predecessor. All these generators are cascaded to enhance a noisy input signal gradually to yield a refined enhanced signal. They propose two improved SEGAN frameworks, namely iterated SEGAN (ISEGAN) and deep SEGAN (DSEGAN). In the ISEGAN system, parameters of its generator are fixed, constraining ISEGAN's generators to apply the same mapping iteratively, as its name implies. DSEGAN's generators have their own independent parameters, allowing them to learn different mappings flexibly. However, the parameters of DSEGAN's generators are N_{G} times more numerous than ISEGAN's generators, where N_{G} is the number of generators.

Afterwards, Phan et al. [202] reveal that the existing class of GANs for speech enhancement solely relies on the convolution operation, which may obscure temporal dependencies across the sequence input. To remedy this issue, they propose a self-attention layer adapted from non-local attention, coupled with the convolutional and deconvolutional layers of the SEGAN, referred to as SASEGAN. Furthermore, they empirically study the effect of placing the self-attention layer at the (de)convolutional layers with varying layer indices, including all layers as long as memory allows.

As Pascual et al. [193] state, they open the exploration of generative architectures for speech enhancement to progressively incorporate further speech-centric design choices for performance improvement. This study aims to further optimize SEGAN, especially its variant with a self-attention mechanism. Unlike [201], this work preserves the single generator architecture to maintain the lightweight parameter scale. Phan et al. [202] focus on coupling only one self-attention layer to one convolutional layer in the encoder. Namely, maximum three layers of SEGAN are equipped with the self-attention mechanism each time: one convolutional layer of the encoder, one deconvolutional layer of the decoder, and one convolutional layer of the discriminator. Although they also experiment with the performance of *SASEGAN-all*, i.e., simply coupling self-attention layers to all (de)convolutional layers, this work queries whether there are more optimized

and efficient coupling combinations [1]. For example, can coupling the self-attention mechanism to the 10th and 11th (de)convolutional layers outperform *SASEGAN-all* with even less parameters? In addition, inspired by [219] and [45], this study explores the feasibility of substituting self-attention layers with (de)convolutional layers completely, namely SEGAN with stand-alone self-attention layers. Moreover, to take full advantage of the self-attention layer's flexibility of modeling of both long-range and local dependencies, this study introduces the parameter-free locality modeling [310] of the self-attention mechanism in SEGAN. To the best knowledge of the author, these three following explorations: stand-alone self-attention layers, the locality modeling on self-attention layers, and optimized combination of coupling self-attention layers with (de)convolutional layers, were never executed by previous works in the SEGAN class.

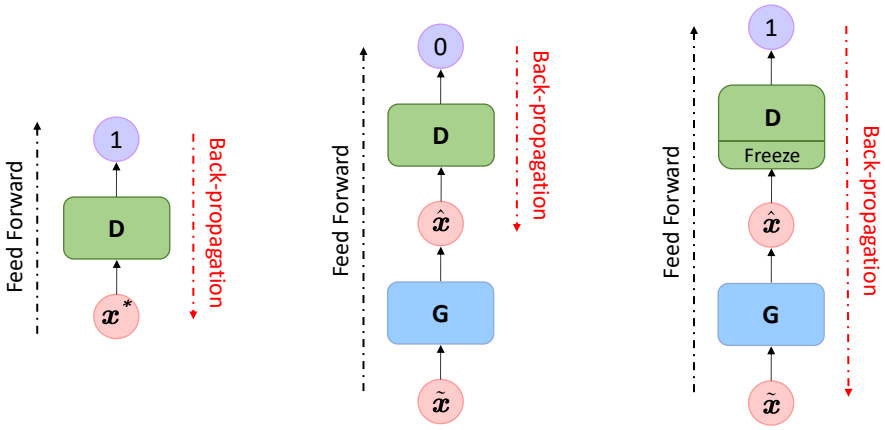


Figure 4.1: Illustration of the GAN training process. Firstly, the discriminator (D) is trained by a batch of real examples and classifies them as true. Next, the discriminator updates according to a batch of fake examples generated by the generator (G), and classifies them as false. Lastly, the discriminator's parameters are frozen and the generator adjusts to make the discriminator misclassify [193].

4.1.3 Self-Attention Speech Enhancement GANs

4.1.3.1 Generative Adversarial Networks

GANs [74] are generative networks that learn to map samples $\tilde{\boldsymbol{x}}$ (e.g., images, audio, etc.) from the original distribution $\tilde{\mathcal{X}}$ to samples $\boldsymbol{x}^*$ from a different distribution $\mathcal{X}^*$. The mapping component within the GAN structure is called the generator, and it is designated to learn an effective mapping that can imitate the real data distribution to generate novel samples similar to the training set. Notably, the generator accomplishes the task by mapping the data distribution characteristics to the manifold defined by $\mathcal{X}^*$

[1] Actually, [202] only couples the self-attention layer to the 3rd–11th layers in the encoder, decoder, and the discriminator because of the memory limitation, although they refer it to *SASEGAN-all*.

instead of memorizing input-output pairs [193]. Therefore, the generated samples are new and unseen samples, not the selected data from the training set.

The way that the generator learns the mapping is by means of an adversarial training, where a discriminator component also participates. The discriminator is basically a binary classifier, and its inputs are either real samples, coming from the dataset $\mathcal{X}^*$, or fake samples, made up by the generator. Then the task of the discriminator is to classify the samples coming from $\mathcal{X}^*$ as real, whereas classify the samples made up by the generator as fake. Namely, the generator needs to learn to fool the discriminator. To this end, the generator adapts its parameters when the discriminator classifies the generator's output as real. As the result, the discriminator appears to be better and better at distinguishing between real and fake samples and, in turn, the generator adjusts its parameters to move towards the manifold defined by the real samples. Fig. 4.1 illustrates the progress of the adversarial training, and it can be mathematically formulated as a minimax game between the generator (G) and the discriminator (D), with the objective

$$\begin{aligned}\min_{\mathrm{G}}\max_{\mathrm{D}}\mathcal{L}(\mathrm{D},\mathrm{G}) =&\mathbb{E}_{\boldsymbol{x}^*\sim p_{data}(\boldsymbol{x}^*)}[\log \mathrm{D}(\boldsymbol{x}^*)]+\\ &\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\tilde{\boldsymbol{x}}}(\tilde{\boldsymbol{x}})}[\log(1-\mathrm{D}(\mathrm{G}(\tilde{\boldsymbol{x}})))].\end{aligned} \tag{4.1}$$

There is also a conditioned version of GANs, where an extra input $\boldsymbol{x}_c$ can be added as the extra information in the generator and the discriminator to perform mapping and classification [111]. In this case, the objective function is changed to

$$\begin{aligned}\min_{\mathrm{G}}\max_{\mathrm{D}}\mathcal{L}(\mathrm{D},\mathrm{G}) =&\mathbb{E}_{\boldsymbol{x}^*,\boldsymbol{x_c}\sim p_{data}(\boldsymbol{x}^*,\boldsymbol{x_c})}[\log \mathrm{D}(\boldsymbol{x}^*,\boldsymbol{x_c})]+\\ &\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\tilde{\boldsymbol{x}}}(\tilde{\boldsymbol{x}}),\boldsymbol{x_c}\sim p_{data}(\boldsymbol{x}_c)}[\log(1-\mathrm{D}(\mathrm{G}(\tilde{\boldsymbol{x}},\boldsymbol{x_c}),\boldsymbol{x_c}))].\end{aligned} \tag{4.2}$$

4.1.3.2 Speech Enhancement GANs (SEGANs)

Given a dataset $\mathcal{X} = \{(\boldsymbol{x}_1^*, \tilde{\boldsymbol{x}}_1), (\boldsymbol{x}_2^*, \tilde{\boldsymbol{x}}_2), \cdots, (\boldsymbol{x}_N^*, \tilde{\boldsymbol{x}}_N)\}$ consisting of N pairs of raw signals: clean speech signal $\boldsymbol{x}^*$ and noisy speech signal $\tilde{\boldsymbol{x}}$. Speech enhancement aims to find a mapping $f_\theta(\tilde{\boldsymbol{x}}) : \tilde{\boldsymbol{x}} \rightarrow \hat{\boldsymbol{x}}$ to transform the raw noisy signal $\tilde{\boldsymbol{x}}$ to the enhanced signal $\hat{\boldsymbol{x}}$. θ contains the parameters of the enhancement network.

Conforming to GAN's principle (cf. Section 4.1.3.1), the generator learns an effective mapping that can imitate the real data distribution to generate novel samples related to those of the training set. Hence the generator acts as the enhancement function. In contrast, the discriminator plays the role of a classifier which distinguishes the real sample, coming from the dataset that the generator is imitating, from the fake samples, made up by the generator. The discriminator guides θ towards the distribution of clean speech signals. To sum up, SEGAN designates the generator for the enhancement mapping, i.e., $\hat{\boldsymbol{x}} = \mathrm{G}(\tilde{\boldsymbol{x}})$, while designates the discriminator to guide the training of the generator by classifying $(\boldsymbol{x}^*, \tilde{\boldsymbol{x}})$ as real and $(\hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})$ as fake. Eventually, the generator learns to produce enhanced signals $\hat{\boldsymbol{x}}$ good enough to fool the discriminator such that the discriminator classifies $(\hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})$ as real.

Various loss functions have been proposed to improve the training of GANs, e.g., Wasserstein loss [328], relativistic loss [10], metric loss [328], and least-squares loss [152]. In this study, the least-squares loss [152] with binary coding is utilized instead of the cross-entropy loss. Due to the effectiveness of the L_1 norm in the image manipulation domain [111], it is deployed to encourage the generator to gain more fine-grained and realistic results. The scalar λ controls the magnitude of the L_1 norm. Consequently, the loss functions of the generator (G) and the discriminator (D) are

$$\begin{aligned}\min_{\mathrm{D}} \mathcal{L}(\mathrm{D}) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}^*,\tilde{\boldsymbol{x}}\sim p_{data}(\boldsymbol{x}^*,\tilde{\boldsymbol{x}})}[\mathrm{D}(\boldsymbol{x}^*,\tilde{\boldsymbol{x}})-1]^2 + \\ & \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z}),\tilde{\boldsymbol{x}}\sim p_{data}(\tilde{\boldsymbol{x}})}[\mathrm{D}(\mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}),\tilde{\boldsymbol{x}})]^2,\end{aligned} \tag{4.3}$$

$$\begin{aligned}\min_{\mathrm{G}} \mathcal{L}(\mathrm{G}) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z}),\tilde{\boldsymbol{x}}\sim p_{data}(\tilde{\boldsymbol{x}})}[\mathrm{D}(\mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}),\tilde{\boldsymbol{x}})-1]^2 \\ & + \lambda \parallel \mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}) - \boldsymbol{x}^* \parallel_1 .\end{aligned} \tag{4.4}$$

$\mathrm{D}(\cdot)$ is the discriminator module, $\mathrm{G}(\cdot)$ is the generator module, and $\boldsymbol{z}$ denotes a latent variable.

4.1.3.3 Stand-Alone Self-Attention Speech Enhancement GANs

This section demonstrates the self-attention layer adapted in GANs [323], which enables both the generator and the discriminator to efficiently model relations between widely separated spatial regions. Given the feature map $\boldsymbol{F} \in \mathbb{R}^{L\times C}$ as the input of the self-attention layer, where L is the time dimension and C is the number of channels, the query matrix $\boldsymbol{Q}$, the key matrix $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$ are obtained via transformations:

$$\boldsymbol{Q} = \boldsymbol{F}\boldsymbol{W}^Q, \boldsymbol{K} = \boldsymbol{F}\boldsymbol{W}^K, \boldsymbol{V} = \boldsymbol{F}\boldsymbol{W}^V, \tag{4.5}$$

where $\boldsymbol{W}^Q \in \mathbb{R}^{C\times\frac{C}{b}}$, $\boldsymbol{W}^K \in \mathbb{R}^{C\times\frac{C}{b}}$, and $\boldsymbol{W}^V \in \mathbb{R}^{C\times\frac{C}{b}}$ denote the learnt weight matrices of the 1×1 convolutional layer of $\frac{C}{b}$ filters. b is a factor for reducing the channel numbers. Additionally, a max pooling layer with filter width and stride size of p is deployed to reduce the number of keys and values for memory efficiency. Therefore, the dimensions of the matrices are $\boldsymbol{Q} \in \mathbb{R}^{L\times\frac{C}{b}}$, $\boldsymbol{K} \in \mathbb{R}^{\frac{L}{p}\times\frac{C}{b}}$, and $\boldsymbol{V} \in \mathbb{R}^{\frac{L}{p}\times\frac{C}{b}}$. The attention map $\bar{\boldsymbol{A}}$ is then computed as

$$\bar{\boldsymbol{A}} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^T), \quad \bar{\boldsymbol{A}} \in \mathbb{R}^{L\times\frac{L}{p}}, \tag{4.6}$$

$$\bar{a}_{j,i} = \frac{\exp(\bar{s}_{ij})}{\sum_{i=1}^{L}\exp(\bar{s}_{ij})}, \quad \text{where } \bar{s}_{ij} = \boldsymbol{Q}(\bar{\boldsymbol{q}}_{\boldsymbol{i}})\boldsymbol{K}(\bar{\boldsymbol{k}}_{\boldsymbol{j}})^T. \tag{4.7}$$

Each element $\bar{a}_{ij} \in \bar{\boldsymbol{A}}$ indicates the extent to which the model attends to the jth column $\boldsymbol{v}_j$ of $\boldsymbol{V}$ when producing the ith output $\bar{\boldsymbol{o}}_i$ of $\bar{\boldsymbol{O}}$. The output of the attention layer $\bar{\boldsymbol{O}}$ is then computed as

$$\bar{\boldsymbol{O}} = (\bar{\boldsymbol{A}}\boldsymbol{V})\boldsymbol{W}^{\bar{O}}, \quad \boldsymbol{W}^{\bar{O}} \in \mathbb{R}^{\frac{C}{b}\times C}. \tag{4.8}$$

With the weight matrix $\boldsymbol{W}^{\bar{O}}$ realized by a 1×1 convolution layer of C filters, the shape of $\bar{\boldsymbol{O}}$ is restored to the original shape $L \times C$. Eventually, SASEGAN contains a shortcut connection to facilitate information propagation, and a learnable parameter β is employed to balance the weight between the output $\bar{\boldsymbol{O}}$ and the input feature map $\boldsymbol{F}$ as

$$\boldsymbol{F'} = \beta\bar{\boldsymbol{O}} + \boldsymbol{F}. \tag{4.9}$$

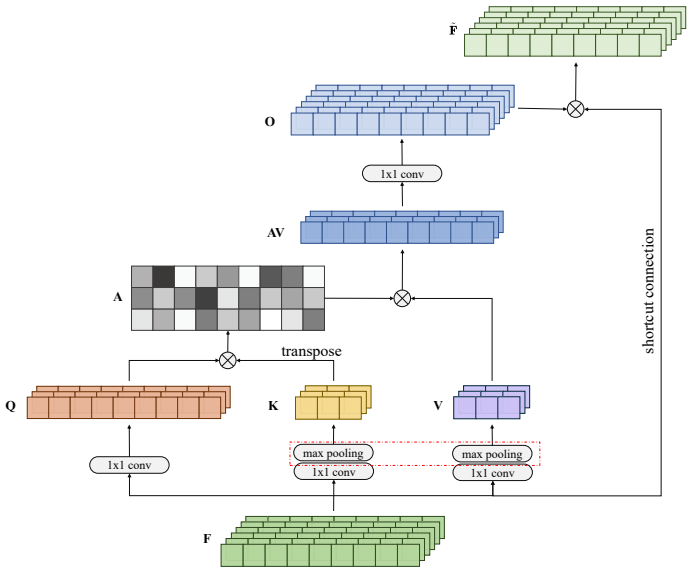


Figure 4.2: Illustration of the stand-alone self-attention layer with $L = 9$, $C = 6$, $p = 3$, and $b = 2$. The max pooling layers in the red frame are discarded for matrix $\boldsymbol{K}$ and $\boldsymbol{V}$ when modeling locality of the stand-alone self-attention layers [202].

Fig. 4.2 illustrates the diagram of a simplified self-attention layer with $L = 9$, $C = 6$, $p = 3$, and $b = 2$.

4.1.3.4 Locality Modeling for Stand-Alone Self-Attention Layers

As illustrated in Fig. 4.3, for the query $\boldsymbol{Q}$, this study restricts its attention region (e.g., $\boldsymbol{K} = \{\bar{\boldsymbol{k}}_1, \cdots, \bar{\boldsymbol{k}}_i, \cdots, \bar{\boldsymbol{k}}_L\}$, $1 \leq i \leq L$) to a local scope with a fixed window size $M_{\text{win}} + 1$ ($M_{\text{win}} \leq L$) centered at the position i as

$$\hat{\boldsymbol{K}} = \{\bar{\boldsymbol{k}}_{i-\frac{M_{\text{win}}}{2}}, \cdots, \bar{\boldsymbol{k}}_i, \cdots, \bar{\boldsymbol{k}}_{i+\frac{M_{\text{win}}}{2}}\}, \quad \hat{\boldsymbol{K}} \in \mathbb{R}^{(M_{\text{win}}+1)\times\frac{C}{b}}, \tag{4.10}$$

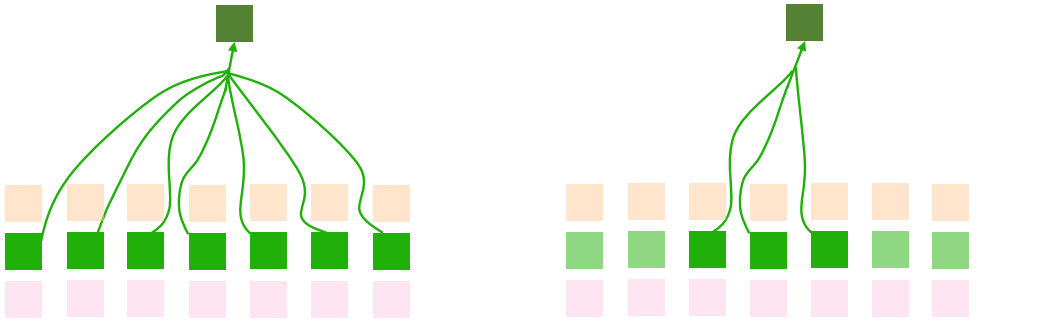
$$\hat{\boldsymbol{V}} = \{\bar{\boldsymbol{v}}_{i-\frac{M_{\text{win}}}{2}}, \cdots, \bar{\boldsymbol{v}}_i, \cdots, \bar{\boldsymbol{v}}_{i+\frac{M_{\text{win}}}{2}}\}, \quad \hat{\boldsymbol{V}} \in \mathbb{R}^{(M_{\text{win}}+1)\times\frac{C}{b}}. \tag{4.11}$$

When applying the locality modeling to the self-attention layer, the factor p should be discarded to help preserve the original neighborhood for the centered position (as

red-framed in Fig. 4.2). Accordingly, the local attention map and the output of the attention layer are modified as

$$\hat{\boldsymbol{A}} = \text{softmax}(\boldsymbol{Q}\hat{\boldsymbol{K}}^T), \quad \hat{\boldsymbol{A}} \in \mathbb{R}^{L\times(M+1)}, \tag{4.12}$$

$$\hat{\boldsymbol{O}} = (\hat{\boldsymbol{A}}\hat{\boldsymbol{V}})\boldsymbol{W}^{\bar{O}}, \quad \boldsymbol{W}^{\bar{O}} \in \mathbb{R}^{\frac{C}{b}\times C}. \tag{4.13}$$



(a) Vanilla self-attention layer (b) Locality modeling with the window size = 3

Figure 4.3: Illustration of (a) vanilla self-attention layer; (b) locality modeling with the window size = 3. Semi-transparent colors represent masked tokens that are invisible to the self-attention layer [310]

.

4.1.3.5 Attention Augmented Convolutional SEGAN

This study implements an attention augmented convolutional SEGAN by coupling the self-attention layer with the (de)convolutional layer(s). This proposed architecture possesses an advantage, where the distance-aware information from the convolutional layer and the distance-agnostic dependencies modeled by the self-attention layer are supplementary and complementary to each other.

To this end, this study introduces two learnable parameters, κ and γ, to weigh the input feature map $\boldsymbol{F}$ and the output feature map $\bar{\boldsymbol{O}}$ of the coupled layer (with self-attention mechanism and convolution) in the augmented output $\boldsymbol{F'}$ as

$$\boldsymbol{F'} = \kappa\bar{\boldsymbol{O}} + \gamma\boldsymbol{F}. \tag{4.14}$$

4.1.4 Experimental Setups

This study systematically evaluates the effectiveness of (i) the stand-alone self-attention layer on SEGAN, (ii) the locality modeling on the stand-alone self-attention layer, and (iii) the attention augmented convolutional SEGAN.

4.1.4.1 Dataset

All experiments are exerted on the publicly available dataset introduced in [273]. This publicly available dataset is originated from the Voice Bank corpus [279], which contains 30 speakers, of which 28 speakers are included in the training set while the remaining 2 are included in the test set. For the noisy training set, 40 different noisy conditions are considered by combining 10 sorts of intrusions (2 artificially generated and 8 derived from the Demand database [268]) with 4 different signal-to-noise ratios (SNRs) each (15, 10, 5, 0 dB). For the test set, 20 different noisy conditions are considered by combining 5 sorts of intrusions (all originated from Demand database) with 4 SNRs (17.5, 12.5, 7.5, and 2.5 dB) each. There are 10 different utterances in each adverse condition per training speaker, while 20 utterances in each condition per test speaker. Notably, the test set is entirely unseen by the training set, namely no overlap existing in either speakers or adverse conditions. All utterances are downsampled to 16 kHz.

4.1.4.2 Evaluation Criteria

All the proposed architectures are evaluated with the following classic objective criteria for speech enhancement (the higher the better):

- SSNR: Segmental SNR [211] (in the range of $[0\,, +\infty)$);
- STOI: Short-time objective intelligibility [265] (in the range of $[0\,, 100]$);
- CBAK: Mean opinion score (MOS) prediction of the intrusiveness of background noises [106] (in the range of $[1\,, 5]$);
- CSIG: MOS prediction of the signal distortion attending only to the speech signal [106] (in the range of $[1\,, 5]$);
- COVL: MOS prediction of the overall effect [106] (in the range of $[1\,, 5]$);
- PESQ: Perceptual evaluation of speech quality, using the wide-band version recommended in ITU-T P.862.2 [225] (in the range of $[-0.5\,, 4.5]$).

All results have been computed using the implementation demonstrated in [142], and is available on the publisher's website.[2]

4.1.4.3 Network Architecture

This section first introduces the architecture of SEGAN, as all three variants, namely (i) SEGAN with stand-alone self-attention layers, (ii) stand-alone self-attention layers with locality modeling, and (iii) attention augmented convolutional SEGAN, are based on it.

The classic enhancement systems are based on the short-time Fourier analysis/synthesis framework [142]. They assume that the short-time phase is not important for speech

[2] `https://www.crcpress.com/downloads/K14513/K14513_CD_Files.zip`

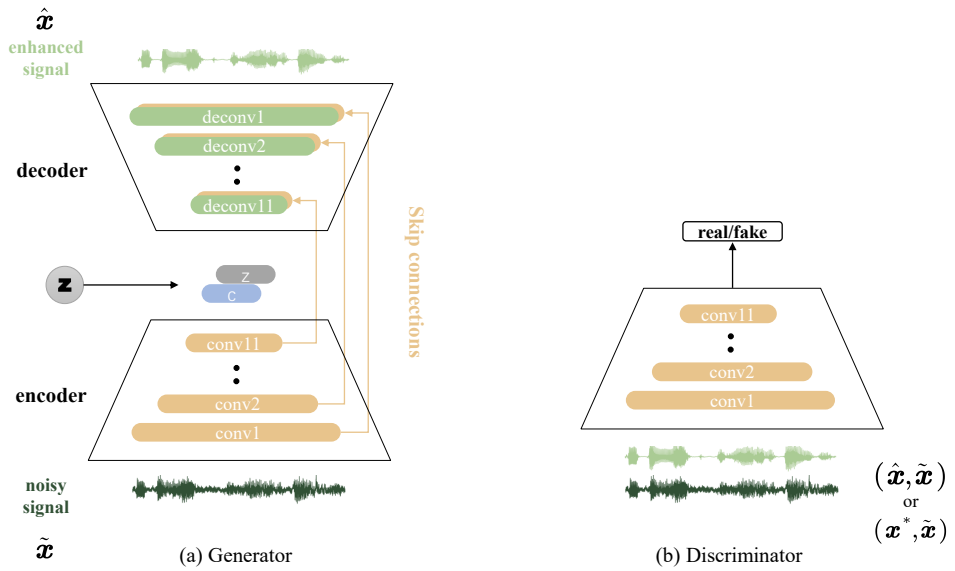


Figure 4.4: Illustration of the architecture of speech enhancement GAN (SEGAN) [193]. (a) The generator component. (b) The discriminator component.

enhancement [288], and they only process the spectrum magnitude. However, further studies [187] show the intensive relation between the clean phase spectrum and the speech quality. Therefore, this study utilizes raw inputs for all experiments. Approximately one-second waveform chunks are extracted (∼16,384 samples) with a sliding window every 500 ms. The generator makes use of an encoder–decoder structure. The encoder is composed of 11 1-dim strided convolutional layers of filter width 31 and stride 2, followed by parametric rectified linear units (PReLUs) [92]. Along with the depth, the number of filters per layer increases while the signal duration shrinks. To compensate for the smaller and smaller convolutional output, the number of filters increases along the encoder's depth $\{16, 32, 32, 64, 64, 128, 128, 256, 256, 512, 1024\}$, resulting in dimensions per layer of $\{8192 \times 16, 4096 \times 32, 2048 \times 32, 1024 \times 64, 512 \times 64, 256 \times 128, 128 \times 128, 64 \times 256, 32 \times 256, 16 \times 512, 8 \times 1024\}$. At the 11th layer of the encoder, the encoding vector $\boldsymbol{c} \in \mathbb{R}^{8\times1024}$ is stacked with the noise $\boldsymbol{z} \in \mathbb{R}^{8\times1024}$, sampled from the distribution $\mathcal{N}(0, I)$, and presented to the decoder. The decoder mirrors the encoder architecture entirely to reverse the encoding process by means of the transposed convolution, termed as deconvolution. There are skip connections connecting each convolutional layer to its homologous deconvolutional layer. They bypass the compression performed in the middle of the model and allow the fine-grained details (e.g., phase, alignment) of speech signals to flow into the decoding stage directly. This is done because if all information is forced to flow through bottleneck structures, much useful low-level information could be lost through the compression. In addition, skip connections offer a better training behavior, as the gradients can flow more deeply through the whole structure [93]. Notably, skip connections and the addition of the latent vector double the number of feature maps in every layer.

The discriminator resembles the encoder's structure with the following differences: (i) it receives a pair of raw audio chunks as the input, i.e., $(\boldsymbol{x}^*, \tilde{\boldsymbol{x}})$ or $(\hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})$; (ii) it utilizes virtual batch-norm before LeakyReLU [148] activation with $\alpha = 0.3$; (iii) in the last activation layer, there is a 1×1 convolution layer to reduce the feature required for the final classification neuron from 8×1024 to 8. The architecture of SEGAN is illustrated in Fig. 4.4.

4.1.4.4 SEGAN with Stand-Alone Self-Attention Layers

This study substitutes the self-attention layer, illustrated in Section 4.1.3.3, with the (de)convolutional layers of both the generator and the discriminator. Fig. 4.5 (a) and (b) demonstrate an example of substituting the self-attention layer with the $\bar{l}$th (de)convolutional layers.

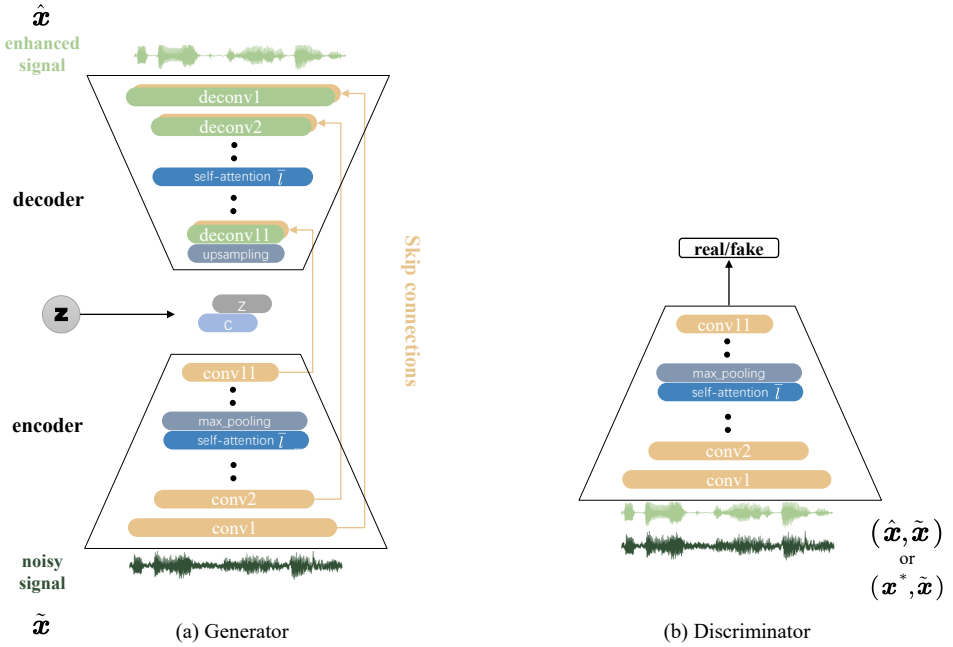


Figure 4.5: Illustration of the architecture of SEGAN with stand-alone self-attention layers. (a) The generator component. (b) The discriminator component.

When the stand-alone self-attention layer is deployed on the $\bar{l}$th convolutional layer of the encoder, the mirroring $\bar{l}$th deconvolutional layer of the decoder and the $\bar{l}$th convolutional layer in the discriminator are also replaced with it. To keep the dimension of the feature map per layer in accordance with that of SEGAN, a max pooling layer with a kernel size of 2 and a stride length of 2 follows every stand-alone self-attention layer in the encoder. Accordingly, the upsampling needs to be deployed before the 11th deconvolutional layer in the decoder to ensure the same feature dimensions flowing in the skip connections. This study experiments with two interpolation methods: nearest and bilinear. The results suggest that the bilinear interpolation outperforms the nearest

one, so this study utilizes the bilinear interpolation for upsampling in all experiments. Theoretically, the stand-alone self-attention layer can be placed in any number, even all, of the (de)convolutional layers. This study empirically explores the effect of placing the stand-alone self-attention layer at (de)convolutional layers with lower or higher layer indices as well as layer combinations, provided that memory allows.

Similarly, the discriminator component follows the same structure as the generator's encoder stage. This study sets $b = 8$, $p = 4$ for the self-attention layer. λ is set to be 100, and β is initialized as 0 for the self-attention layer.

4.1.4.5 Stand-Alone Self-Attention Layer with Locality Modeling

The general architecture remains the same as the case of Section 4.1.4.4. However, the factor p is eliminated to help preserve the original neighborhood for the centered position. Namely, the max pooling layers in Fig. 4.2 are discarded for matrix $\boldsymbol{K}$ and $\boldsymbol{V}$. The factor b remains as 8. Ablation tests are exerted on the impacts of the window size $(M_{win} + 1)$ and the placement of the window on the system performance.

Peters et al. [199] and Raganato and Tiedemann [218] indicate that higher layers of the system tend to learn semantic information while lower layers capture more surface and lexical information. Therefore, this study centrally applies locality modeling to the lower layers, in line with the configuration in [319, 309]. Then, it is expected that the representations are learnt in a hierarchical fashion. Namely, the distance-aware and local information extracted by the lower layers can be complementary to the distance-agnostic and global information captured by the higher layers.

4.1.4.6 Attention Augmented Convolutional SEGAN

Instead of replacing (de)convolutional layers with stand-alone self-attention layers (Section 4.1.4.4), this section couples stand-alone self-attention layers with (de)convolutional layers of both the generator and discriminator. As illustrated in Fig. 4.6, when the stand-alone self-attention layer is coupled with the $\bar{l}$th convolutional layer of the encoder, the mirror $\bar{l}$th deconvolutional layer of the decoder, and the $\bar{l}$th convolutional layer in the discriminator are also coupled with it. Spectral normalization is applied to all the (de)convolutional layers. For scalars γ and κ, this study experiments with three initialization pairs: $\gamma = 0$ and $\kappa = 0$, $\gamma = 0.75$ and $\kappa = 0.25$ (inspired by the results provided by [21]), and $\gamma = 0.25$ and $\kappa = 0.25$, finding that best results are acquired when both γ and κ are initialized as 0.25. Empirical study is exerted for the effect of coupling the self-attention layer with lower and higher layer indices as well as different layer combinations. Two introduced factors (introduced in Section 4.1.3.3) are set to be $p = 4$ and $b = 8$.

Since coupling the self-attention layer with a single convolutional layer has already been studied by [202] in detail, this study focuses on more optimized couple combinations for this topic.

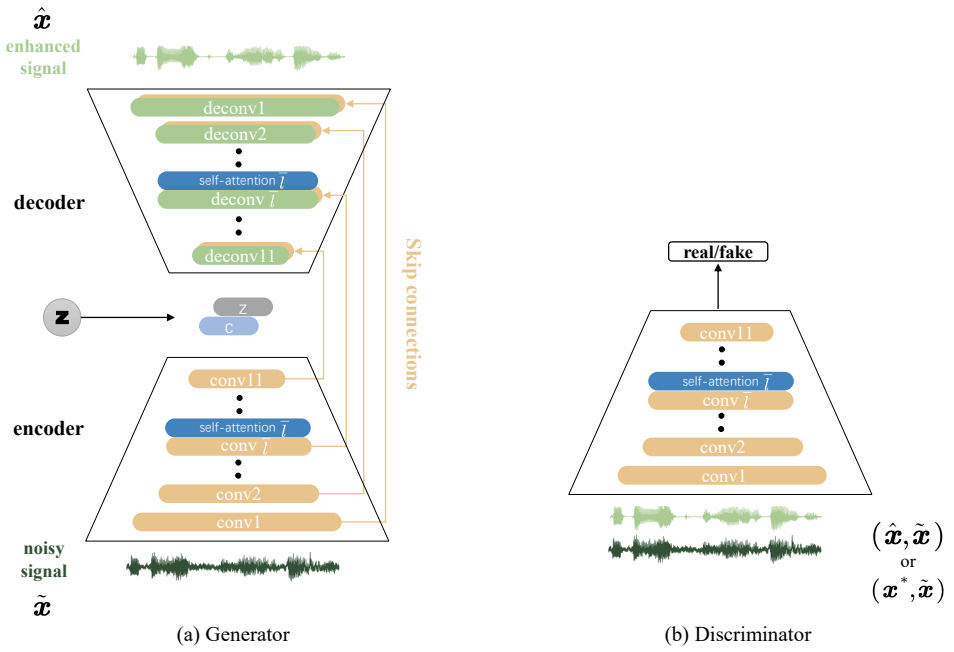


Figure 4.6: Illustration of the attention augmented convolutional SEGAN architecture [202]. (a) The generator component. (b) The discriminator component.

4.1.4.7 Baseline Systems

For comparison, this study takes the seminal work [193], and other SEGAN variants [201, 202] that are introduced in Section 4.1.2 as baseline systems. From [201], this study chooses the results of ISEGAN with two shared generators and DSEGAN with two independent generators as baseline results (the situation of $N_{\mathrm{G}} = 2$) for two reasons. On one hand, the number of generators leads to exponential parameter incrementation. On the other hand, Phan et al. [201] indicate marginal impacts of ISEGAN's number of iterations and DSEGAN's depth larger than $N_{\mathrm{G}} = 2$ for no significant performance improvements are seen. Phan et al. [202] present detailed results of the influence of the self-attention layer placement in the generator and the discriminator. This study chooses the average result of coupling the self-attention layer with a single (de)convolutional layer (referred to as *SASEGAN-avg*), and the result of coupling self-attention layers with all (de)concolutional layers (referred to as *SASEGAN-all*) to ensure a fair comparison. It is worth noting that it is stated in [202] that compared to *SASEGAN-avg*, results of *SASEGAN-all* are slightly further boosted, but these gains are achieved at the cost of increased computation time and memory requirements.

4.1.4.8 Configurations

Networks are trained with RMSprop [270] for 100 epochs with a minibatch size of 50. A high-frequency preemphasis filter of coefficient 0.95 is applied to both training and test

samples. In the test stage, the window is slid through the whole duration of every test utterance without overlap, and the enhanced outputs are deemphasized and concatenated at the end of the stream.

4.1.5 Results

This study systematically evaluates the effectiveness of (i) the stand-alone self-attention layer on SEGAN, (ii) the locality modeling on the stand-alone self-attention layer, and (iii) attention augmented convolutional SEGAN. Table 4.1 exhibits the performance of SEGAN equipped with the stand-alone self-attention layer. The upper part displays the results of baseline systems, while the lower part displays the results of this work.

As shown in Table 4.1, when the 6th and 10th (de)convolutional layers are replaced with stand-alone self-attention layers, the system overtakes all baselines across all metrics and achieves the best SSNR result, with 47 % fewer (compared to DSEGAN) or 12 % fewer (compared to SASEGAN-all) parameters. Furthermore, when the stand-alone self-attention layer is adopted at the 9th to 11th (de)convolutional layers, it still yields comparable or even better (STOI) results, with parameters plunging drastically to merely 5 % (95 % fewer) of DESEGAN or 9 % (91 % fewer) of SASEGAN-all. When $\bar{l} = 4$, the parameter scale of the proposed system is closest to that of the baseline systems. Under such circumstances, it outperforms baseline systems in PESQ, CBAK, COVL, and STOI, and achieves the best results in PESQ, CBAK, and COVL. When this experiment substitutes the self-attention layer with $\bar{l}$th ($6 \leq \bar{l} \leq 11$) (de)concolutional layers, the system performance plummets as the parameters of the whole system are only 1 % of SASEGAN-all [202] and 0.6 % of DSEGAN [201]. The results reveal that the stand-alone layer can be a powerful and lightweight primitive for speech enhancement.

Next, this study investigates the effects of locality modeling and its window size. Prior studies [319, 255] indicate that lower layers usually extract lower-level features, so they should attend to the local field more. Additionally, they [319, 255] prove empirically that modeling locality on lower layers achieves better performances. Therefore, this study only applies the locality modeling on layers not deeper than the 6th one. As plotted in Fig. 4.7, the tiny window size limits the receptive field too much, and hence exacerbates the performance due to the deprivation of the ability of modeling long-range and multi-level dependencies. It appears that a window size of 14 is superior to other settings, approximately consistent with [310] on machine translation tasks. When the window size continues to increase, the performance tends to be the same without windows, which is self-explanatory. Completed results on six criteria are exhibited in Table 4.2. Compared to Table 4.1, employing locality modeling on the 4th layer yields the most significant improvement, in accordance with the conclusion in [319, 255]. It also achieves the best or comparable results across all criteria, which demonstrates the functionality of the locality modeling without further computational cost. An explanation of the undesirable SSNR is that the suboptimal upsampling method introduces speech distortion, which is also manifested on CSIG. Importantly, a fixed-size window is not the state-of-the-art approach in the field of locality modeling of the self-attention mechanism. This study chooses it as it is parameter free, corresponding to the goal of a lightweight system. It is

worth noting that the proposed SEGAN with stand-alone self-attention layers is general enough to combine other more advanced locality modeling approaches [318, 277] in cases where the computation complexity is secondary.

Lastly, this study investigates the functionality of the attention augmented convolutional networks according to Eq. 4.14. This study chooses the combination from lower-to-middle layers (*augmentation-4,6*), middle-to-higher layers (*augmentation-6,10*), and all layer ranges (*augmentation-4,10* and *augmentation-4,6,10*). As displayed in Table 4.3, coupling the self-attention layer on the 4th and 6th layers is more competitive on CSIG, COVL, and STOI, and it achieves the best STOI performance. In contrast, adding the self-attention layer on the 6th and 10th layers overtakes all baseline systems across all metrics, and it gives the best result on SSNR. The combination of the 4th and 10th layers still outperforms baseline systems, except for SSNR. However, the combination of the 4th, 6th, and 10th layers only outperforms baseline systems on PESQ and STOI, although it still yields decent results on other metrics. These results demonstrate the efficiency of the attention augmentation for the convolutional SEGAN. Nevertheless, it is worth noting that system parameters inevitably increase when coupling the self-attention layer to (de)convolutional layers.

Table 4.1: Effects of the stand-alone self-attention layer(s) on speech enhancement GANs (SEGANs). This study denotes the proposed architecture with the stand-alone self-attention layer(s) at the $\bar{l}$th (de)convolutional layer(s) as *standalone-*$\bar{l}$. Values that overtake all baseline systems are in bold. Values with an asterisk are the best ones achieved for each metric.

Architecture	Params (M)	Metric					
		PESQ	CSIG	CBAK	COVL	SSNR	STOI
Noisy	-	1.97	3.35	2.43	2.63	1.69	92.10
SEGAN [193]	294	2.16	3.48	2.94	2.79	7.66	93.12
ISEGAN [201]	294	2.24	3.23	2.93	2.68	8.19	93.29
DSEGAN [201]	513	2.35	3.56	3.10	2.94	8.70	93.25
SASEGAN-avg [202]	295	2.33	3.52	3.05	2.90	8.08	93.33
SASEGAN-all [202]	310	2.35	3.55	3.10	2.91	8.30	93.49
standalone-4	293	**2.49***	3.54	**3.62***	**3.11***	7.70	**93.72**
standalone-6	292	**2.41**	3.54	3.07	**2.96**	8.10	**93.63**
standalone-11	103	**2.43**	**3.74***	3.01	**3.07**	7.11	**93.55**
standalone-6,10	274	**2.39**	**3.59**	**3.12**	**2.98**	**8.71***	**93.66**
standalone-10,11	51	**2.37**	**3.57**	3.00	**2.95**	7.71	**93.54**
standalone-4,6,10	275	**2.45**	**3.61**	**3.10**	**3.01**	8.30	**93.73**
standalone-9,10,11	28	**2.43**	3.50	2.97	2.88	8.51	**93.90***
standalone-6$\sim$11	3	2.01	3.36	2.62	2.64	6.98	93.32

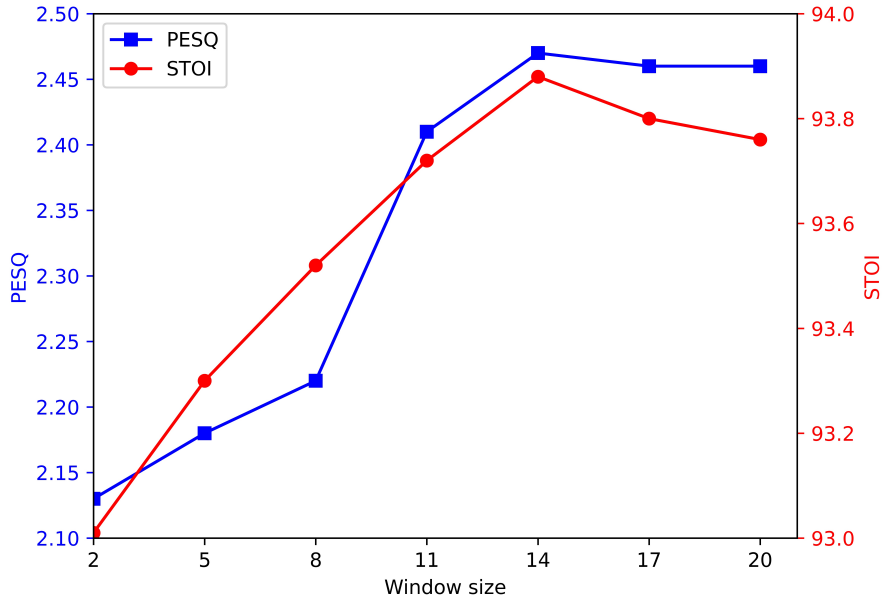


Figure 4.7: Effects of the window size on the self-attention layers.

Table 4.2: Effects of the locality modeling on the stand-alone self-attention layer. The layer numbers with curly braces represent the employment of locality modeling on the current layer. Values that overtake all baseline systems are in bold. Values with an † are the ones that overtake their best counterparts on the same metric in Table 4.1.

Architecture	Params (M)	Metric					
		PESQ	CSIG	CBAK	COVL	SSNR	STOI
standalone-{4}	293	**2.50**†	3.55	**3.64**†	**3.13**†	8.10	**93.83**
standalone-{6}	292	**2.41**	3.54	3.08	**2.96**	7.90	**93.68**
standalone-{6},10	274	**2.40**	**3.59**	**3.12**	**2.99**	**8.70**	**93.76**
standalone-{4},{6},10	275	**2.45**	**3.58**	**3.11**	**3.03**	8.30	**93.88**

Table 4.3: Performance of the attention augmented convolutional SEGAN. The proposed architecture where the self-attention couples the $\bar{l}$th (de)convolutional layer(s) is denoted as *augmentation-$\bar{l}$*. Values that overtake all baseline systems are in bold. Values with an ‡ are the ones that overtake their best counterparts on the same metric in Table 4.1 and Table 4.2.

Architecture	Params (M)	Metric					
		PESQ	CSIG	CBAK	COVL	SSNR	STOI
augmentation-4,6	293	**2.43**	**3.64**	**3.11**	**3.02**	8.20	**93.99**‡
augmentation-6,10	299	**2.47**	**3.58**	**3.15**	**3.01**	**8.87**‡	**93.61**
augmentation-4,10	298	**2.45**	**3.58**	**3.10**	**3.00**	7.86	**93.61**
augmentation-4,6,10	299	**2.39**	3.50	3.08	2.92	8.41	**93.68**

4.1.6 Discussion

In general, the biggest advantage of applying the stand-alone self-attention layer in SEGAN is that it simultaneously outperforms baseline systems, and decreases the model complexity drastically. In particular, when applying the stand-alone self-attention layer as the 6th and 10th layers of the system, the resultant system overtakes all baselines across all metrics and achieves the best SSNR result with only $\sim 50\,\%$ parameters (compared to DSEGAN [201]). In addition, locality modeling can be an effective auxiliary to stand-alone self-attention layers, which further improves their performance without any extra parameter increment. Notably, locality modeling on a lower self-attention layer delivers more perceptible performance improvements, consistent with [310, 319, 309]. For the self-attention augmented SEGAN, it performs modestly better. Although it is less lightweight than those two approaches, it still decreases parameters by $42\,\%$ compared to DSEGAN.

However, different placements of the stand-alone self-attention layer or the coupled self-attention layer lead to different performance improvements, and the compromise between system performance and system complexity is always ineluctable. This study only presents the achieved performance and the homologous model complexity of representative placements, which readers can take for reference according to the desired application.

4.1.7 Summary

This study integrates the self-attention mechanism with SEGAN to improve its flexibility of both long-range and local dependency modeling for speech enhancement in three methods, namely, applying the stand-alone self-attention layer, modeling locality on the stand-alone self-attention layer, and coupling the self-attention layer with the (de)convolutional layer. The proposed systems deliver consistent performance improvements. The main merit of the stand-alone self-attention layer is its low model complexity, and it can perform even better when equipped with the locality modeling. In contrast,

the self-attention augmented convolutional SEGAN delivers more stable improvements, whereas it increases the model complexity.

Importantly, the locality modeling method utilized in this study is basic. This study chooses it to achieve the goal of light weight, but more advanced locality modeling approaches can be applied simply. Moreover, all proposed approaches described in this study are generic enough to be applied to existing SEGAN models for further performance improvements, which can be left for future studies.

The next section will present another lightweight architecture for speech enhancement, Sinc-SEGAN. It usefully merges two powerful paradigms: Sinc convolution and SEGAN. Moreover, a set of data augmentation techniques in the time domain are empirically studied and substantiated to be efficient for further performance improvements.

4.2 Lightweight End-to-End Speech Enhancement Generative Adversarial Network Using Sinc Convolutions

4.2.1 Overview

In the last few years, supervised methods for speech enhancement leveraging deep learning methods have become the mainstream. Well-known models include deep denoising autoencoders [143], CNNs [64], RNNs [301, 262].

There exists a class of generative methods relying on GANs [74], which have CNNs as the backbone and have been verified to be efficient for speech enhancement [193, 196, 98, 208, 139, 53]. GANs designate the generator for enhancement mapping and the discriminator for distinguishing real signals from fake ones. With the transmitted information from the discriminator, the generator learns to produce outputs that resemble the realistic distribution of clean signals. Most past attempts deal with magnitude spectrum inputs as it is often claimed that short-time phase is unimportant for speech enhancement [288]. However, further studies [187] demonstrate that a clean phase spectrum can deliver significant improvements of speech quality.

CNNs are the most popular deep learning architecture for processing raw speech inputs due to weight sharing, local filters, and pooling, extracting robust and invariant representations. The first convolutional layer is a critical part of waveform-based CNNs [221], since it processes high-dimensional inputs and extracts low-level speech representations for deeper layers. However, it is susceptible to vanishing gradient problems. To alleviate this issue, Ravanelli et al. [221] propose the Sinc convolution to learn more meaningful filters in the input layer. Different from a standard CNN, the Sinc-convolution layer convolves the waveform with a set of parametrized Sinc functions that implement band-pass filters, and only need to learn the low and high cutoff frequencies. Consequently, the Sinc convolution is faster-converging and lightweight. It performs extraordinarily in capturing selective speech clues, e.g., the pitch region, the first formant, and the second formant, which are essential for resembling clean speech signals, and

has achieved remarkable success in some fields of speech signal processing, e.g., speech recognition [220, 190], speaker identification [221], keyword spotting [162], etc.

Unfortunately, Sinc convolution for speech enhancement is still an under-explored research direction. There is no available model fusing these two powerful paradigms: the Sinc convolution operating over raw speech waveforms and the generative adversarial models for speech enhancement. Therefore, this study proposes to bridge this gap by usefully merging the Sinc convolution and SEGAN, resulting in a customizable, lightweight, and interpretable system, termed Sinc-SEGAN. Sinc-SEGAN first optimizes the SEGAN architecture from the seminal work [193], and enhances the original Sinc convolution layer to fit the optimized SEGAN. Besides, this study also analyzes the learnt filters of the Sinc convolution layer. Furthermore, a set of data augmentation techniques are applied on raw speech waveforms directly to further improve the system performance.

Experimental results show that the proposed Sinc-SEGAN overtakes a set of competitive baseline models, especially on higher-level perceptual quality and speech intelligibility. Additionally, the system parameters reduce drastically up to merely 17.7 % of the baseline system. In addition, data augmentation techniques further boost the system performance across all classic objective evaluation metrics of speech enhancement. Analysis of the Sinc filters discloses that the learnt filters are tuned precisely to capture critical speech clues. Notably, the proposed Sinc-SEGAN system is generic enough to be applied to existing GAN models of speech enhancement for further performance improvement.

4.2.2 Related Work

The seminal work of SEGAN [193] and two variants [201, 202] based on it have been introduced in Section 4.1.2. As Pascual et al. [193] state, they open the exploration of generative architectures for speech enhancement to progressively incorporate further speech-centric design choices for performance improvement. This study aims to further optimize SEGAN by fusing the powerful diagram: Sinc convolution. To the best knowledge of the author, although Sinc convolution has achieved great success and been widely utilized in some fields of speech signal processing, e.g., speech recognition [220] and speaker verification [221], its implementation on the speech enhancement task remains unexplored.

4.2.3 Sinc Convolution

For details of the SEGAN implementation, please refer to Section 4.1.3.2. This section will introduce the implementation of Sinc convolution.

Different from a standard convolutional layer that performs a set of time-domain convolutions between the input waveform and some Finite Impulse Response filters, Sinc convolution conducts the convolutional operation with a predefined function $\bar{g}$, depending on few learnable parameters θ as

$$y[n] = x[n] * \bar{g}[n, \theta], \tag{4.15}$$

where $x[n]$ is a chunk of speech waveforms, and $y[n]$ is the filtered output. Inspired by standard filtering in digital signal processing, Ravanelli et al. [221] define $\bar{g}$ as a filter-bank consisting of rectangular bandpass filters. In the frequency domain, the magnitude of a generic bandpass filter can be written as

$$\bar{G}[\bar{f}, \bar{f}_1, \bar{f}_2] = rect(\frac{\bar{f}}{2\bar{f}_2}) - rect(\frac{\bar{f}}{2\bar{f}_1}), \tag{4.16}$$

where $\bar{f}_1$ and $\bar{f}_2$ are the learnable low and high cutoff frequencies, and $rect(\cdot)$ is the rectangular function in the magnitude frequency domain. In the time domain, the reference function $\bar{g}$ transforms to

$$\bar{g}[n, \bar{f}_1, \bar{f}_2] = 2\bar{f}_2 sinc(2\pi\bar{f}_2 n) - 2\bar{f}_1 sinc(2\pi\bar{f}_1 n). \tag{4.17}$$

The Sinc function here is the unnormalized Sinc function, i.e., $sinc(x) = \frac{sin(x)}{x}$. Ravanelli et al. [221] initialize filters with cutoff frequencies of the Mel-scale filterbank, which processes the advantage of directly allocating more filters in the lower part of the spectrum. Fainberg et al. [61] execute experiments over different initialisation schemes but no benefit for the downstream task is observed. Please note that there are two constraints in Eq. 4.17 that need to be satisfied: $\bar{f}_1 \geqslant 0$ and $\bar{f}_2 \geqslant f_1$. In addition, the de facto frequencies that are calculated in Eq. 4.17 are $\bar{f}_1'$ and $\bar{f}_2'$, where

$$\begin{aligned} \bar{f}_1' &= |\bar{f}_1|, \\ \bar{f}_2' &= |\bar{f}_1| + |\bar{f}_2 - \bar{f}_1|. \end{aligned} \tag{4.18}$$

To smooth out the abrupt discontinuities at the end of $\bar{g}$, Hamming window $\digamma[n]$ [214] of length L is deployed by

$$\bar{g}_{\digamma}[n, \bar{f}_1', \bar{f}_2'] = \bar{g}[n, \bar{f}_1', \bar{f}_2'] \cdot \digamma[n], \tag{4.19}$$

where

$$\digamma[n] = 0.54 - 0.46 \cdot cos(\frac{2\pi n}{L}). \tag{4.20}$$

It is also suggested that no significant performance difference appears when adopting other window functions [221]. As the filters $\bar{g}$ are symmetric, the filters can be computed efficiently by considering one side of the filter and inheriting the results for the other half. Moreover, the symmetry does not introduce any phase distortion, keeping the essence of processing raw inputs for speech enhancement.

Another remarkable property of the Sinc convolution is its small parameter scale. Unlike a standard convolutional layer, only two parameters are employed for each Sinc filter, regardless of its length. Taking a layer composed of N_f filters of length L_f as an example, a standard convolutional layer employs $N_f \cdot L_f$ parameters, against $2N_f$ in the case of the Sinc convolution. If $N_f = 50$ and $L_f = 100$, there are 5 K parameters in the case of the standard convolutional layer, while only 100 parameters for the Sinc convolution. This property offers the possibility to obtain selective filters with many taps, with negligible parameter increment.

4.2.4 Sinc-SEGAN Architecture

This study investigates into two different deployments of the Sinc convolution illustrated in Section 4.2.3: (i) being deployed before the first layer of the generator's encoder and the discriminator, and behind the last layer of the generator's decoder, referred to as the addition architecture. (ii) acting as the substitute of the first standard convolutional layers of the generator's encoder and the discriminator, and the last standard convolutional layer of the generator's decoder, referred to as the substitution architecture.

In the case of the addition architecture, the generator makes use of an encoder-decoder architecture [194]. The first layer of the encoder is Sinc convolution, using 64 filters of length 251 and stride = 1, followed by a max pooling layer (stride = 2). Thereafter, there are 5 1-dim strided convolutional layers with a common filter width of 31 and a stride of 4, each followed by PReLUs [92]. At the 6th layer of the encoder, the encoding vector $\boldsymbol{c} \in \mathbb{R}^{8\times1024}$ is stacked with the noise sample $\boldsymbol{z} \in \mathbb{R}^{8\times1024}$, sampled from the distribution $\mathcal{N}(0, I)$, and presented to the decoder.

The decoder component mirrors the encoder architecture, with the same number of filters and filter width, to reverse the encoding process through deconvolutions, namely fractional-strided transposed convolution. The last layer of the decoder is also a Sinc convolution (filter number = 64, kernel size = 251, and stride = 1), and there is an un-maxpooling layer (stride = 2) before it for upsampling. Learnable skip connections are deployed to connect the encoding layer with its corresponding decoding layer to allow the information to be summed to the decoder feature maps. The learnable vectors $\boldsymbol{\vartheta}_{\bar{l}}$ multiply every channel of its corresponding shuttle layer $\bar{l}$ by a scalar factor. Hence, for the input of the $\bar{l}$th decoder layer $\bar{\boldsymbol{h}}_{\bar{l}}$, the corresponding $\bar{l}$th encoder layer response is added, following

$$\bar{\boldsymbol{h}}_{\bar{l}} = \bar{\boldsymbol{h}}_{\bar{l}-1} + \boldsymbol{\vartheta}_{\bar{l}} \odot \bar{\boldsymbol{h}}_{\bar{l}\mathrm{e}}, \tag{4.21}$$

where $\odot$ is an element-wise product along channels.

The discriminator is constructed of an architecture similar to the encoder of the generator. However, it receives a two-channel input, i.e., $(\boldsymbol{x}^*, \tilde{\boldsymbol{x}})$ or $(\hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})$, and utilizes virtual batch-norm [242] before LeakyReLU [148] activation with $\alpha = 0.3$. Moreover, the discriminator is topped up with a 1×1 convolutional layer to reduce the dimension of the output of the last convolutional layer for the subsequent classification task by the softmax layer. To sum up, in the addition architecture, the generator consists of 12 layers and the discriminator consists of 6 layers. The addition architecture of Sinc-SEGAN is illustrated in Fig. 4.8.

The substitution architecture is similar to the first case, but the original first layer of the encoder is substituted as Sinc convolution, using 64 filters of length 251. Its stride is 4, in line with the standard convolutional layer, and the max pooling layer is removed. Thereafter, there are 4 1-dim strided convolutional layers with a common filter width of 31 and a stride of 4, each followed by PReLUs [92]. At the 6th layer of the encoder, the encoding vector $\boldsymbol{c} \in \mathbb{R}^{16\times1024}$ is stacked with the noise sample $\boldsymbol{z} \in \mathbb{R}^{16\times1024}$, sampled from the distribution $\mathcal{N}(0, I)$, and presented to the decoder.

The decoder component still processes the mirror architecture to reverse the encoding process through deconvolutions. The last deconvolutional layer of the decoder is also

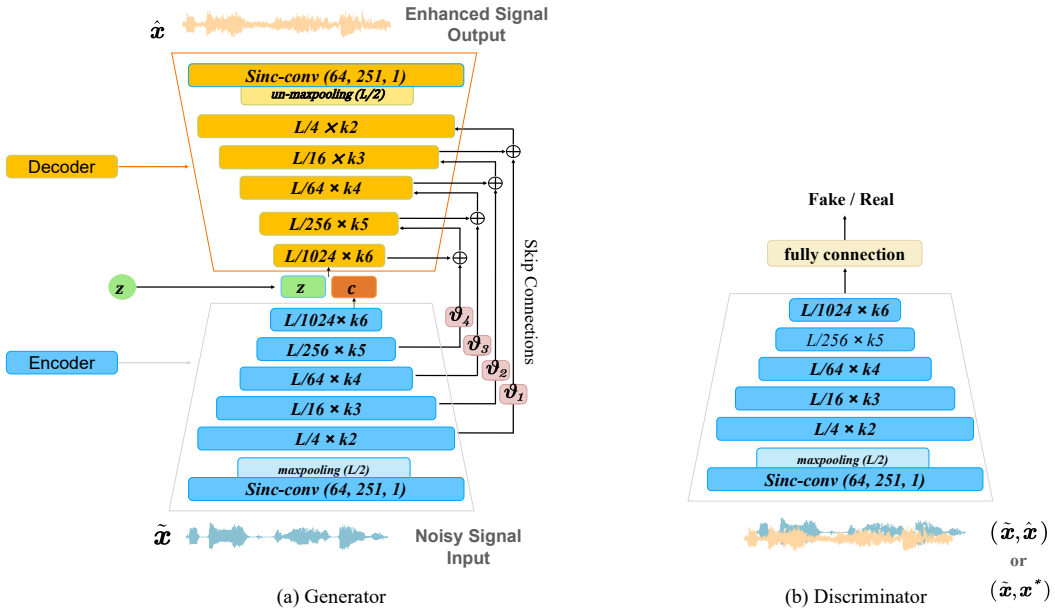


Figure 4.8: Illustration of the addition architecture of (a) the generator and (b) the discriminator, where the Sinc convolution is located before the first layer of the encoder and the discriminator, and behind the last layer of the decoder. Skip connections with learnable ϑ_l are depicted in pink boxes, which are summed to each intermediate activation of the decoder.

replaced with a Sinc convolution (filter number = 64, kernel size = 251, and stride = 4), and the upsampling layer is not needed anymore.

The first layer of the discriminator is also substituted with the Sinc convolution layer with 64 filters of length 251 and stride = 1. In summary, in the substitution architecture, the generator consists of 10 layers and the discriminator consists of 5 layers, as illustrated in Fig. 4.9.

4.2.5 Experimental Setups

To assess the performance of proposed Sinc-SEGANs, this study reports objective measures on the Valentini [272] benchmarks. The details of the employed corpus, evaluation metrics, and baseline systems have been illustrated in Section 4.1.4.1, Section 4.1.4.2, and Section 4.1.4.7, respectively.

4.2.5.1 Implementation Details

The networks are trained for 100 epochs with the batchsize 100. Different from previous works [193, 201, 202], this study utilizes Adam optimizer [121] ($\beta_1 = 0$ and $\beta_2 = 0.9$), with the two-timescale update rule (TTUR) [96]. According to the recent successful approach to training GANs quickly and stably [323], this study sets the learning rate of the discriminator to 0.0004 while that of the generator to 0.0001, i.e., the discriminator

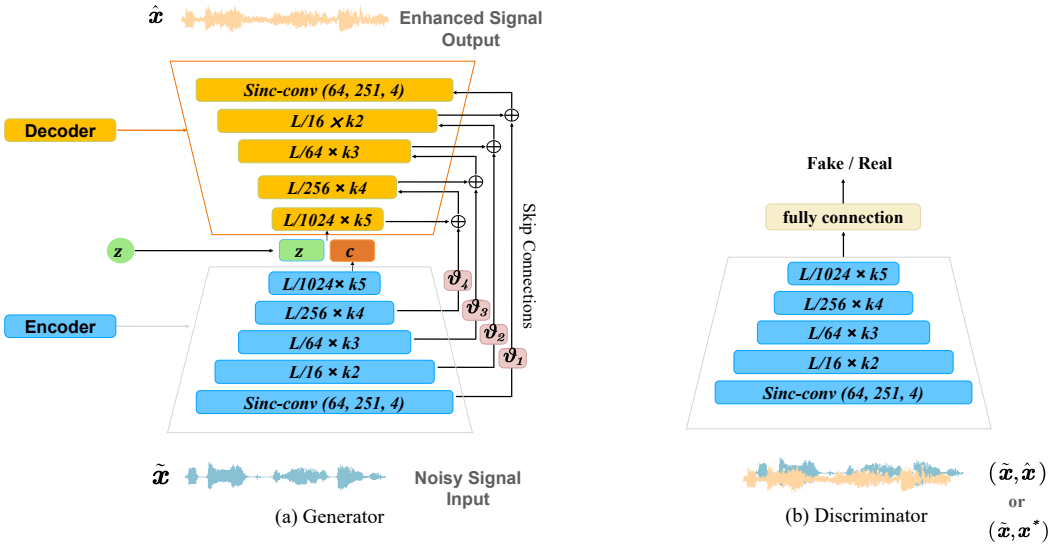


Figure 4.9: Illustration of the substitution architecture of (a) the generator and (b) the discriminator, where the Sinc convolution acts as the substitute of the first standard convolutional layers of the encoder and the discriminator, and the last standard convolutional layer of the decoder. Skip connections with learnable $\vartheta_{\bar{l}}$ are depicted in pink boxes, which are summed to each intermediate activation of the decoder.

has a four times faster learning rate to virtually emulate numerous iterations in the discriminator prior to updating the generator. This study extracts raw speech chunk of length $L = 16,384$ (approximately 1 s) over 50 % overlap as the input, to avoid any speech distortion caused by handcrafted features. A high-frequency preemphasis filter of coefficient 0.95 is applied to each waveform chunk before presenting to the networks as it is proved to help cope with some high-frequency artifacts in the denoising setup. During testing, raw speech chunks are extracted from testing utterances without overlap, and outputs are correspondingly deemphasized and concatenated as the enhanced waveforms. The number of filters varies along with the depth. In the situation of the addition architecture, they are $\{64, 64, 128, 256, 512, 1024\}$, resulting in the output size of the feature map $\{8092 \times 64, 2048 \times 64, 512 \times 128, 128 \times 256, 32 \times 512, 8 \times 1024\}$. In contrast, in the instance of the substitution architecture, they are $\{64, 128, 256, 512, 1024\}$, resulting in the output size of the feature map $\{4096 \times 64, 1024 \times 128, 256 \times 256, 64 \times 512, 16 \times 1024\}$. This study also conducts ablation tests on the influence of the input length, the filter number and the kernel size.

4.2.5.2 Data Augmentation

Three data augmentation methods are employed on the dataset. First, this study applies a random shift between 0 and 4 seconds. Second, the noises with one batch are shuffled to form new noisy mixtures, termed ReMix. Third, this study deploys a band-stop filter with a stop-band between $\bar{f}_1'$ and $\bar{f}_2'$ (termed Band Mask), sampled to remove 20 % of

the frequencies uniformly in the Mel scale, which is equivalent to the SpecAugment [191] used for the automatic speech recognition task in the time domain.

4.2.6 Results

4.2.6.1 Ablation Tests on the Configuration of Sinc Convolution

This work empirically studies the impacts of the placement of Sinc convolution layer, the input length, the number of Sinc filter, the kernel size of Sinc convolution. Table 4.4 demonstrates the configuration of five ablation tests, and Table 4.5 shows their results across criteria introduced in Section 4.1.4.2. After making the comparison among these experiments, the following conclusions can be drawn:

- Increasing the number of Sinc filters degrades the system performance, since more filters introduce more system complexity, and the training becomes more difficult accordingly.
- Decreasing the kernel size of the Sinc convolution deteriorates the performance as smaller kernel size limits its ability to extract representative speech clues.
- Systems benefit from longer input length due to more context information being included.
- The addition architecture outperforms the substitution architecture as the former is deeper.

These experimental results explain why this study chooses the length of input signals as 1 s, the number of Sinc filters as 64, and the kernel size of Sinc convolution as 251.

Table 4.4: The demonstration of different configurations of five ablation tests. *substitution* is shorten to *sub*, and *addition* is shorten to *add*.

Experiment	A	B	C	D	E
Architecture	sub	sub	sub	sub	add
Input length	1 s	1 s	1 s	250 ms	1 s
Number of Sinc filters	64	80	64	64	64
Kernel size of Sinc convolution	251	251	101	251	251

4.2.6.2 Performance and Parameter Comparisons with Baseline Systems

Table 4.6 demonstrates the performance and parameter comparisons between the proposed Sinc-SEGANs (+augment) and the previous SEGAN variants [193, 201, 202] on the Valentini [272] benchmark. Results indicate that the substitution architecture outperforms baseline systems on PESQ, CBAK, and COVL. Considering the designs of

Table 4.5: Ablation test results over different configurations of Sinc convolution: system architecture, input length, number of Sinc filters, and kernel size of Sinc convolution.

Experiment	PESQ	CSIG	CBAK	COVL	SSNR	STOI
A	2.37	3.55	3.13	2.97	8.68	93.40
B	2.32	3.49	2.84	2.91	5.51	92.99
C	2.40	3.46	3.07	2.89	8.66	93.39
D	2.36	3.57	3.07	2.94	8.70	93.37
E	2.39	3.69	3.23	3.00	8.71	93.53

these criteria, the result suggests that for speech signals enhanced by Sinc-SEGAN-sub (*substitution* is shorten to *sub*), the general perceptive quality is higher, and they are reasonably comprehensive for users. Comparable results are achieved on CSIG, SSNR and STOI. Please note that although Sinc-SEGAN-sub underperforms DSEGAN [201] on CSIG and SSNR or SASEGAN-all [202] on STOI, it ourperforms SEGAN [193], ISEGAN [201], and SASEGAN-avg [202] across all criteria. Additionally, the number of Sinc-SEGAN-sub parameters is merely 31.0 % compared to SEGAN, ISEGAN, or SASEGAN-avg, 29.4 % compared to SASEGAN-all, 17.7 % compared to DSEGAN. In contrast, Sinc-SEGAN-add (*addition* is shorten to *add*) outperforms all baseline systems, with the parameter scale that is 71 % compared to SEGAN, ISEGAN, or SASEGAN-avg, 67 % compared to SASEGAN-all, and 41 % compared to DSEGAN. Moreover, data augmentation methods deliver further improvements, leading to the best performance across all evaluation metrics, without additional parameters. These results validate the efficacy of the Sinc convolution on the speech enhancement task.

4.2.6.3 Ablation Tests on Augmentation Methods

In order to better understand the influence of different data augmentation methods on the overall performance, ablation tests are executed. This study reports system performance on all evaluation criteria for each of the methods in Table 4.7. Results suggest that each of these data augmentation methods contributes to overall performance, with the time shift augmentation producing the most significant performance increment, followed by the Band Mask. Surprisingly, ReMix augmentation only shows limited contribution to the overall performance.

4.2.6.4 Interpretation of Sinc Convolution

Inspecting the learnt filters is a valuable practice that provides insights into what the network is actually learning. To this end, this study visualizes the learnt low and high cutoff frequencies of Sinc-convolution filters in Fig. 4.10, and plots four examples of the learnt Sinc filters of the proposed system in Fig. 4.11. Besides, this study exhibits examples of spectrograms of the speech signal enhanced by SEGAN [193], DSEGAN

Table 4.6: Results on objective metrics of the proposed systems (Sinc-SEGANs) against previous SEGAN variants using the Valentini benchmark [272]. The unit of the number of parameters (Params) is million (M).

Architecture	Params (M)	Metric					
		PESQ	CSIG	CBAK	COVL	SSNR	STOI
Noisy	—	1.97	3.35	2.44	2.63	1.69	92.10
SEGAN [193]	294	2.16	3.48	2.94	2.79	7.66	93.12
ISEGAN [201]	294	2.24	3.23	2.93	2.68	8.19	93.29
DSEGAN [201]	513	2.35	3.56	3.10	2.94	8.70	93.25
SASEGAN-avg [202]	295	2.33	3.52	3.05	2.90	8.08	93.33
SASEGAN-all [202]	310	2.35	3.55	3.10	2.91	8.30	93.49
Sinc-SEGAN-sub	91	**2.37**	3.55	**3.13**	**2.97**	8.68	93.40
Sinc-SEGAN-add	210	**2.39**	**3.69**	**3.23**	**3.00**	**8.71**	**93.53**
Sinc-SEGAN-add +augment	210	**2.86**	**3.87**	**3.66**	**3.15**	**8.87**	**94.96**

[201], SASEGAN-all [202], and the proposed Sinc-SEGAN-add, respectively in Fig. 4.12. As observed in Fig. 4.10, the Sinc convolution learns a filter-bank containing more filters with high cut-frequencies. In addition, a tendency towards a higher amplitude is noticeable, indicating an inclination of the Sinc convolution to directly process raw speech waveforms. As we can see from Fig. 4.11 and Fig. 4.12, Sinc convolution is specifically designed to implement rectangular band-pass filters. Considering the speech waveform distribution in the time domain, the specific design makes Sinc convolution suitable for extracting narrow-band speech clues, e.g., the pitch region, the first formant, and the second formant, in accordance with the results of the seminal work [221].

Table 4.7: Ablation study over different data augmentation methods: ReMix, Band Mask (BM), and the time shift (shift).

Sinc-SEGAN-add	PESQ	CSIG	CBAK	COVL	SSNR	STOI
+ BM	2.44	3.55	3.37	3.05	8.79	93.75
+ BM, + ReMix	2.45	3.57	3.40	3.07	8.81	93.80
+ BM, + ReMix, + shift	2.86	3.87	3.66	3.15	8.87	94.96

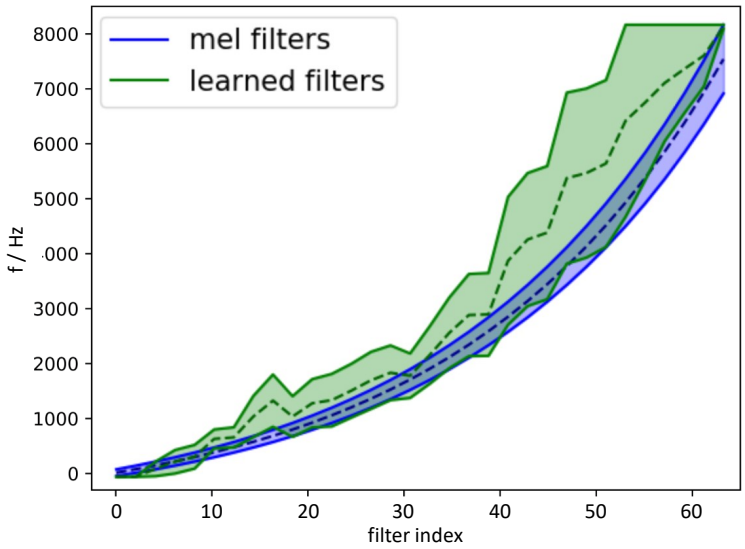


Figure 4.10: Visualization of the learnt upper and lower bounds per Sinc-convolution filter.

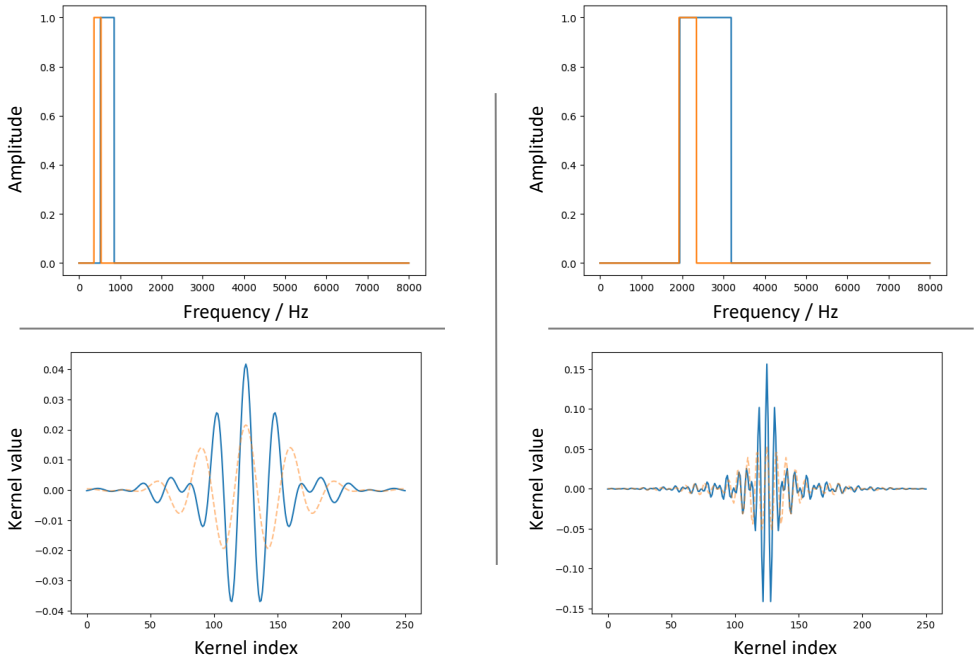
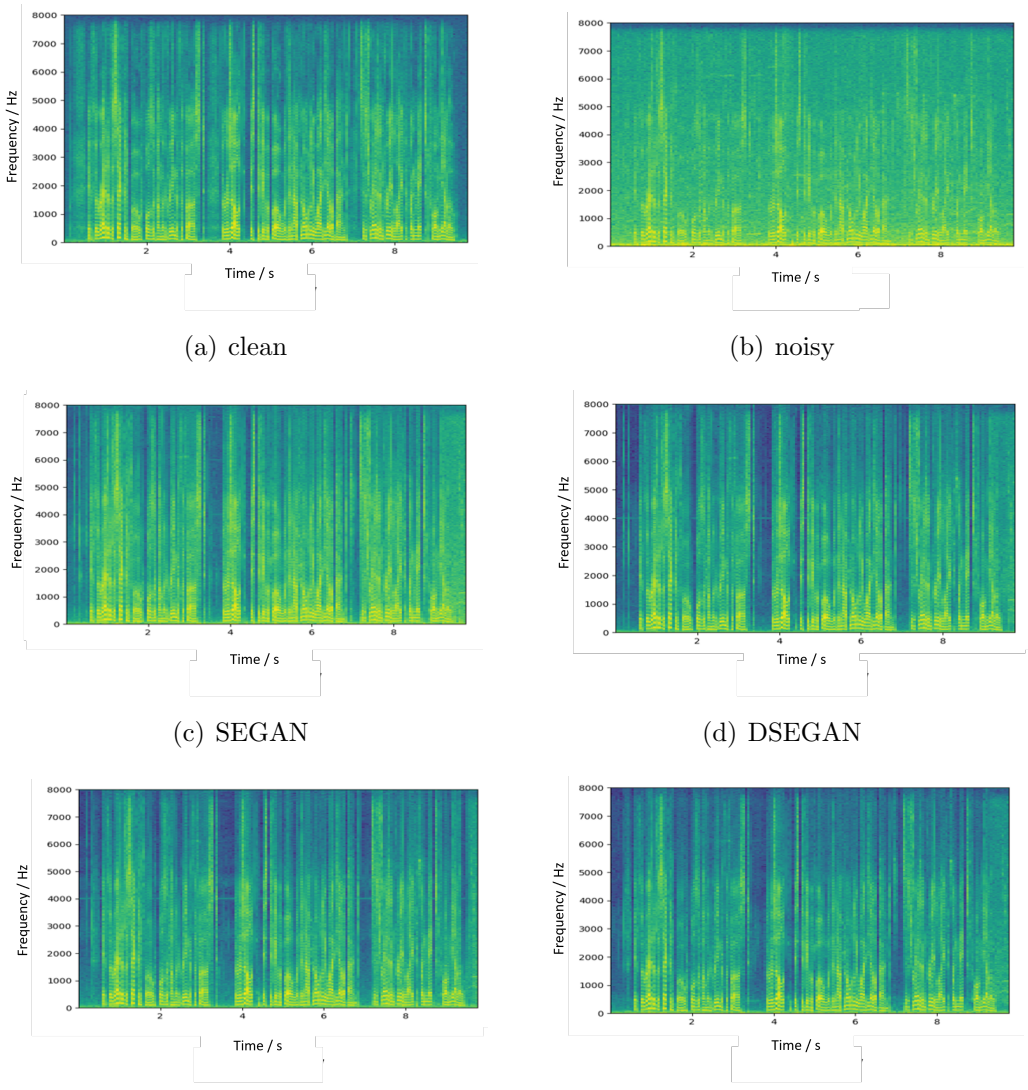


Figure 4.11: Examples of the learnt filters of the Sinc convolution. The upper row reports the frequency response of the filters, while the lower row reports their impulse response. The orange dashed line depicts the corresponding Mel-scale filter.



(a) clean (b) noisy

(c) SEGAN (d) DSEGAN

(e) SASEGAN-all (f) Sinc-SEGAN-add

Figure 4.12: Spectrograms of an example utterance enhanced by (c) SEGAN [193], (d) DSEGAN [201], (e) SASEGAN-all [202], and the proposed (f) Sinc-SEGAN-add. The (a) clean and (b) noisy spectrograms are also exhibited for reference.

4.2.7 Summary

This study proposes Sinc-SEGAN, a system that merges the Sinc convolution layer with the optimized SEGAN to capture more underlying representative speech characteristics. Sinc-SEGAN processes raw speech waveforms directly to prevent distortion caused by imperfect phase estimation. This study investigates into two different deployments of the Sinc convolution: (i) being deployed before the first layer of the encoder and the discriminator, and behind the last layer of the decoder, referred to as Sinc-SEGAN-add. (ii) acting as the substitute of the first standard convolutional layers of the encoder and the discriminator, and the last standard convolutional layer of the decoder, referred to as Sinc-SEGAN-sub. Ablation tests are conducted for the influence of the input length, number of Sinc filters, and kernel size of Sinc convolution. To train the proposed system more efficiently, this study also employs three data augmentation methods in the time domain.

Experimental results show that Sinc-SEGAN-sub yields enhanced signals with higher-level perceptual quality and speech intelligibility, even with drastically reduced system parameters. By contrast, the proposed Sinc-SEGAN-add overtakes all baseline systems across all classic objective evaluation criteria, with up to $\sim$50 % fewer parameters compared to the baseline system. Regarding to data augmentation methods, it is worth nothing that without them, there is no distinct difference between the SEGAN variants from Section 4.1 and the Sinc-SEGAN from Section 4.2 in terms of the system performance or the parameter scale. However, data augmentation methods further boost the system performance and give the Sinc-SEGAN a clear advantage over the SEGAN variants.

Analyses of the Sinc filters reveal that the learnt filter-bank is tuned precisely to select narrow-band speech clues and hence suitable for the speech enhancement task in the time domain.

"I have never met a man so ignorant that I couldn't learn something from him."

– Galileo Galilei

5

Joint Training Techniques for Robust Automatic Speech Recognition

5.1 Overview

After discussing the back- and front-end techniques in Chapter 3 and Chapter 4, respectively, this chapter focuses on the fusion of these two parts, namely the joint training technique. Joint training is a advantageous integration of various components. The core idea is to pipeline a speech enhancement (SE) and a speech recognition network as if they were within a single bigger network to avoid the weak matching and communication between them. The network that will be discussed in this chapter holds two highlights. One is that it benefits from the advancement of both self-attention mechanism and GANs; while the other is that it is an adversarial joint training with a global discriminant guide.

In recent years, attention-based end-to-end neural networks, which subsume the acoustic and language models into a single neural network, have triggered a revolution in the field of ASR [43, 39] and are challenging the dominance of HMM-based hybrid systems [99]. Furthermore, the self-attention mechanism has made another breakthrough in the innovation of the attention architecture, which considers the whole sequence at once to model feature interactions that are arbitrarily distant in time, leading to faster convergence and state-of-the-art results in ASR [42, 205, 269, 241, 88, 89, 200, 312, 145]. The self-attention system predicts the next output symbol conditioned on the full sequence of the previous predictions. Once a mistake occurs in one estimation step due to noise interference, all the subsequent steps will be disturbed. As speech signals are inevitably interfered by various background noises in realistic environments, it is crucial to improve the robustness of the self-attention mechanism for the practical application.

The mainstream solution to the noise robustness problem is adding an independent speech enhancement module as the front-end of ASR. Speech enhancement aims to transform the interfered speech to its original clean version, which is achieved by various approaches, i.e., statistical methods such as Wiener filter [140], time-frequency masking [173, 293, 180], signal approximation [302, 59], spectral mapping [308, 179], etc. No

matter what approach the speech enhancement module adopts to achieve the goal, it is trained separately from the ASR model on different loss functions (i.e., mean squared error [58]) and evaluated by different objective criteria (i.e., CBAK [106], Segmental SNR [211]). This mismatch between the enhancement training and the final ASR task leads to a sub-optimum easily [251]. Moreover, the handcrafted loss functions tend to generate over-smoothed spectra or introduce unseen distortions, which sometimes even degrade the downstream ASR performance [295]. To obtain the optimum and circumvent introducing unnecessary distortion, the idea of a joint training framework is proposed for robust speech recognition [295, 26, 210, 181]. The fundamental concept of the joint training is concatenating the speech enhancement front-end and a downstream ASR model to build an entire neural network and jointly adjust the parameters in each module. The goal here is that the enhancement front-end tends to produce enhanced features desired by the ASR component, and the ASR module can guide the enhancement module to a more discriminative direction. In this way, the joint framework is optimized on the final ASR objectives, i.e., W/CER.

GANs aim at mapping samples $\tilde{\boldsymbol{x}}$ from the distribution $\tilde{\mathcal{X}}$ to samples $\boldsymbol{x}^*$ from another distribution $\mathcal{X}^*$. There are two components within GANs. One is the generator, which performs the mapping; and the other is the discriminator, which guides the training of the generator. GANs have been applied to various speech signal processing tasks, such as speech enhancement [193, 256], robust speaker verification [159], spoken language identification [254], speech emotion recognition [235], data augmentation [105], and robust speech recognition [53].

Inspired by the advancement of self-attention mechanism and various applications of GAN in speech-related tasks, this study proposes an adversarial joint training framework with self-attention mechanism to boost the robustness of the self-attention ASR systems, which consists of a self-attention speech enhancement GAN (SASEGAN) and a self-attention end-to-end ASR model (SA_ASR), which are implemented with Transformer [54] and Conformer [82]. The discriminant component of SASEGAN is first utilized to distinguish the enhanced features from the original clean features, instructing the enhancement module to output the clean distribution. When it comes to the stage of the joint training, the discriminator acts as the global training guide, and it will shift the direction for the generator to produce more congruous features for the ASR task. As the global guide, the discriminator is expected to remedy the limitation of the separate training and handcrafted loss functions, alleviate the distortion, and lead the speech enhancement component to the global optimum. Meanwhile, the enhancement module is supposed to capture more underlying structural characteristics. With this global guide, the whole framework is expected to learn more robust representations compatible with the ASR task automatically. As the result, the proposed framework yields remarkable results, which achieves relative improvements up to 66 % compared to the ASR model trained by clean data solely, 35.1 % compared to the scheme without joint training, and 5.3 % compared to multi-condition training.

5.2 Related Work

GANs have been applied in speech enhancement tasks without attention [193, 201, 9] and with attention [202, 126]. These works validate the functionality of GAN in the enhancement task on diverse objective criteria; however, they lack proofs of the effectiveness of their work for the downstream ASR task.

GANs have also been employed to improve the robustness of the ASR model [53, 259, 290, 141]. A potential limitation lies in the weak matching and communication between the integrated modules. For instance, speech enhancement and speech recognition are often designed independently, and the enhancement system is tuned according to metrics that are not straightly relative to the final ASR performance.

To address this concern, joint training is a promising approach. An early attempt was proposed in [56], where a feature extraction front-end and a GMM-HMM back-end are jointly trained on maximum mutual information. Afterwards, other interesting works are published in this field [295, 67, 223, 210, 207]. Nevertheless, an effective integration between various systems has been difficult for many years, mainly due to the different nature of the technologies involved at different steps. For example, in [295, 67], the joint training is actually performed as a fine-tuning procedure. To tackle this problem, this chapter deploys the discriminant component of GAN as a global guide, leading the enhancement module to match the downstream ASR module.

5.3 Self-Attention based SE-ASR Scheme

5.3.1 Overview

Fig. 5.1 illustrates an overview of the proposed joint training framework for robust end-to-end speech recognition pictorially. The system consists of a self-attention enhancement front-end and a self-attention ASR model. Given the raw noisy speech input $\tilde{\boldsymbol{x}}$ and the raw clean input $\boldsymbol{x}^*$, the entire procedure of the joint training pipeline can be illustrated in the following forms:

$$\hat{\boldsymbol{x}} = \text{Generator}(\tilde{\boldsymbol{x}}), \tag{5.1}$$

$$\hat{\boldsymbol{u}} = \text{FBank}(\hat{\boldsymbol{x}}), \tag{5.2}$$

$$P(\boldsymbol{y}|\hat{\boldsymbol{u}}) = \text{SA\_ASR}(\hat{\boldsymbol{u}}), \tag{5.3}$$

$$P(D|\hat{\boldsymbol{x}}, \boldsymbol{x}^*) = \text{Discriminator}(\hat{\boldsymbol{x}}, \boldsymbol{x}^*). \tag{5.4}$$

Here, Generator$(\cdot)$ acts as a speech enhancement front-end realized by the generator component of SASEGAN [202], which transforms the noisy raw input $\tilde{\boldsymbol{x}}$ to the enhanced $\hat{\boldsymbol{x}}$. FBank$(\cdot)$ is a function for extracting the normalized log FBank features $\hat{\boldsymbol{u}}$ from the enhancement outputs $\hat{\boldsymbol{x}}$. Subsequently, SA_ASR$(\cdot)$ is an ASR system based on self-attention layers realized by Transformer [54] or Conformer [82] architecture. $\boldsymbol{y}$ is the output of the whole scheme. Discriminator$(\cdot)$ is realized by the discriminator component of SASEGAN [202], which distinguishes enhanced outputs from clean data.

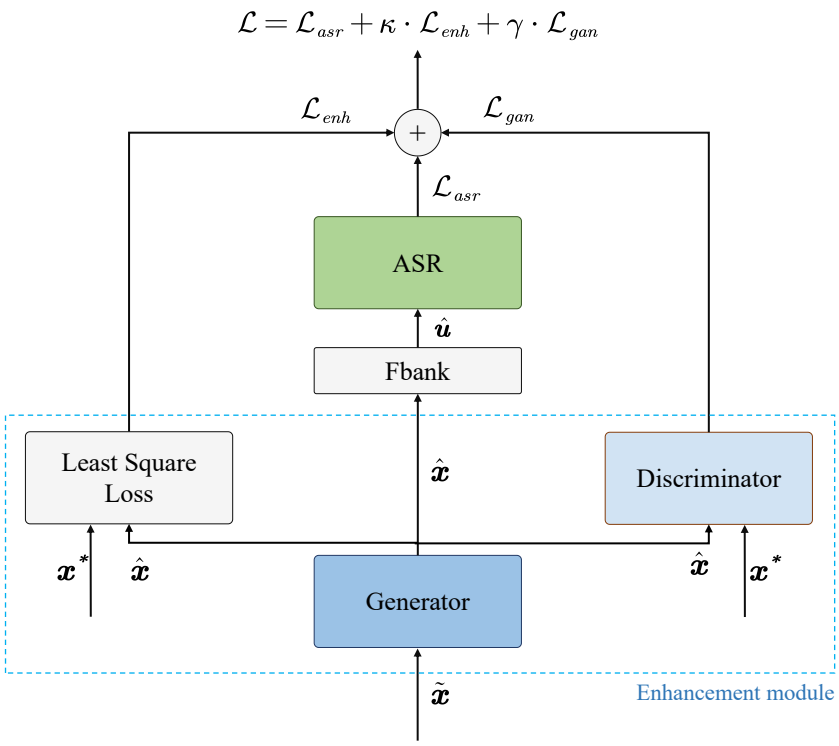


Figure 5.1: Overview of the SE-ASR joint training framework.

5.3.2 Self-Attention Speech Enhancement GANs

The concept of the self-attention mechanism, the training process of GANs, the architecture of SEGAN, and the implementation of the self-attention mechanism in SEGAN have been introduced in Section 2.9.3.2, Section 4.1.3.1, Section 4.1.3.2, and Section 4.1.3.3, respectively. Therefore, this section only clarifies the architecture of SASEGAN [202] in this study.

The architectures of the generator and the discriminator are depicted in Fig. 5.2 (a) and (b). The generator makes use of an encoder-decoder architecture with fully-convolutional layers [217]. The generator's encoder comprises 11 1-dim stridden convolutional layers with a common filter width of 31 and a stride length of 2, followed by PReLUs [92]. The encoder receives a one-second segment of the raw signal sampled at 16 kHz, approximately 16,384 samples as the input. To compensate for the smaller and smaller convolutional output, the number of filters increases along the encoder's depth $\{16, 32, 32, 64, 64, 128, 128, 256, 256, 512, 1024\}$, resulting in output size of the feature map $\{8192 \times 16, 4096 \times 32, 2048 \times 32, 1024 \times 64, 512 \times 64, 256 \times 128, 128 \times 128, 64 \times 256, 32 \times 256, 16 \times 512, 8 \times 1024\}$. At the 11th layer of the encoder, the encoding vector $\boldsymbol{c} \in \mathbb{R}^{8\times1024}$ is stacked with the noise sample $\boldsymbol{z} \in \mathbb{R}^{8\times1024}$, sampled from the distribution $\mathcal{N}(0, I)$, and presented to the decoder.

The decoder component mirrors the encoder architecture with the same number of filters and the filter width to reverse the encoding process through deconvolutions. The same as the encoder, each deconvolutional layer is again followed by PReLUs. The skip

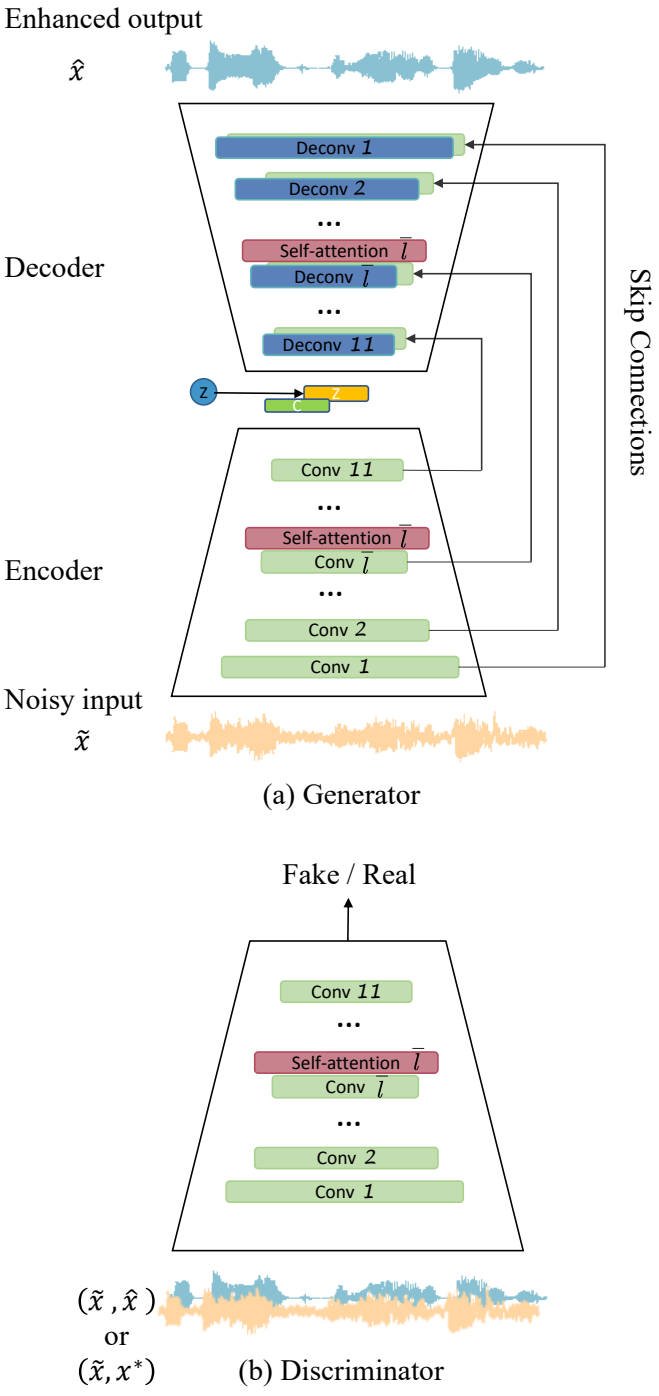


Figure 5.2: Illustration of the SASEGAN architecture [202]. (a) the generator component. (b) the discriminator component.

connections are deployed to connect the encoding layer with its corresponding decoding layer to allow the information to flow between the encoding stage and the decoding stage.

The discriminator is constructed of an architecture similar to the encoder component of the generator. However, it receives the two-channel input and utilizes virtual batch-norm [242] before LeakyReLU [148] activation with $\alpha = 0.3$. Moreover, the discriminator is topped up with a 1×1 convolutional layer to reduce the dimension of the output of the last convolutional layer from 8×1024 to 8 for the subsequent classification task with the softmax layer.

The self-attention layer illustrated in Section 4.1.3.3 couples with the (de)convolutional layer of both the generator and the discriminator. Fig. 5.2 (a) and (b) demonstrate an example of the self-attention layer coupling with the $\bar{l}$th (de)convolutional layers. As we can see, if the self-attention layer is added to the $\bar{l}$th convolutional layer of the encoder, the mirror $\bar{l}$th deconvolutional layer of the decoder and the $\bar{l}$th layer in the discriminator are also coupled a self-attention layer. Theoretically, the self-attention layer can be placed in any number, even all, of the (de)convolutional layers.

5.3.3 FBank Extraction Network

This study extracts the normalized log FBank feature $\hat{\boldsymbol{u}}$ as the input of the subsequent ASR model, which is computed from the enhanced signal $\hat{\boldsymbol{x}}$:

$$\hat{\boldsymbol{u}} = \text{FBank}(\hat{\boldsymbol{x}}) = \text{Norm}(\log(\text{Mel}(\text{STFT}(\hat{\boldsymbol{x}})))), \tag{5.5}$$

where $\text{STFT}(\cdot)$ is the operation of short-time Fourier transform (STFT), $\text{Mel}(\cdot)$ is the operation of Mel matrix multiplication, and $\text{Norm}(\cdot)$ is for normalizing the mean and variance to 0 and 1, respectively. Consequently, the FBank feature extraction layer is differentiable.

5.3.4 Transformer

5.3.4.1 Multi-Head Attention Mechanism

Multi-head attention mechanism [278], as the terminology implies, contains more than one self-attention module (cf. Section 2.9.3.2). As the core module of the Transformer [54], it leverages different attending representations jointly. Before performing each attention, three linear projections transform the queries, keys, and values to more discriminated representations, respectively. Afterwards, each dot-product attention is calculated independently, and their outputs are concatenated and fed into another linear projection to obtain the final d_{model}-dimensional outputs:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(head_1, head_2, \cdots, head_h)\boldsymbol{W}^{\text{OUT}}, \tag{5.6}$$

where

$$head_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \quad 1 \leq i \leq h. \tag{5.7}$$

h refers to the head numbers, and $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ have the same dimensions of d_{model}. Four projection matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_q}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $\boldsymbol{W}^{\text{OUT}} \in \mathbb{R}^{hd_v \times d_{model}}$. Additionally, $d_q = d_k = d_v = d_{\text{model}}/h$.

5.3.4.2 Positional Encoding

One obvious limitation of the Transformer model is that the output is invariant to the input order permutation, i.e., the Transformer does not model the order of the input sequence. Vaswani et al. [278] solve this problem by injecting information about absolute positions into the input sequence via sinusoid positional embeddings:

$$\mathrm{PE}_{(pos,d_{\mathrm{pos}})} = \begin{cases} sin(pos/10000^{d_{\mathrm{pos}}/d_{\mathrm{model}}})) & \text{if } d_{\mathrm{pos}} \text{ is even} \\ cos(pos/10000^{d_{\mathrm{pos}}/d_{\mathrm{model}}})) & \text{if } d_{\mathrm{pos}} \text{ is odd} \end{cases}, \tag{5.8}$$

where *pos* refers to the position and d_{pos} is the dimension. The sinusoidal function allows the model to extrapolate from long sequence lengths.

5.3.4.3 Feed-Forward Network

The feed-forward network (FFN) is another core module of the Transformer [54]. It is composed of two linear transformations with a ReLU activation in between. The dimensionality of the input and output is d_{model}, and the inner layer has the dimensionality d_{ff}. Specifically,

$$\mathrm{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{5.9}$$

where the weights $\boldsymbol{W}_1 \in \mathbb{R}^{d_{\mathrm{model}} \times d_{ff}}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_{ff} \times d_{\mathrm{model}}}$ and the biases $\boldsymbol{b}_1 \in \mathbb{R}^{d_{ff}}$, $\boldsymbol{b}_2 \in \mathbb{R}^{d_{\mathrm{model}}}$. The linear transformations are the same across different positions.

5.3.4.4 Network Architecture

The detailed model architecture of the ASR-Transformer [54] is as follows:

The encoder is shown in Fig. 5.3 (a). The input-embedding is for extracting expressive representations of dimension d_{model}. Thereafter, to enable the model to attend on the auxiliary position information, the d_{model}-dim positional encoding (Section 5.3.4.2) is added to the input encoding. Then the sum of encoded outputs is fed into a stack of N_{e} encoder blocks, each of which has two sub-blocks: one is the multi-head attention (Section 5.3.4.1), receiving queries, keys, and values from the previous block; the other is the feed-forward networks (Section 5.3.4.3). In the meanwhile, layer normalization and residual connection are introduced to each sub-block for effective training. Thus, the pipeline of the sub-block is:

$$\boldsymbol{x} + \mathrm{SubBlock}(\mathrm{Layer\,Norm}(\boldsymbol{x})). \tag{5.10}$$

The decoder is shown in Fig. 5.3 (b). The output-embedding converts the character sequence to dimension d_{model}. Added with the positional encoding, the sum of them is fed into a stack of N_{d} decoder blocks, which consists of three sub-blocks: The first is a masked multi-head attention, which ensures that the predictions for position j depends only on the known outputs at positions less than j. The second is a multi-head attention whose keys and values come from the encoder outputs while queries come from the previous sub-block outputs. The third is also feed-forward networks. Similar to the

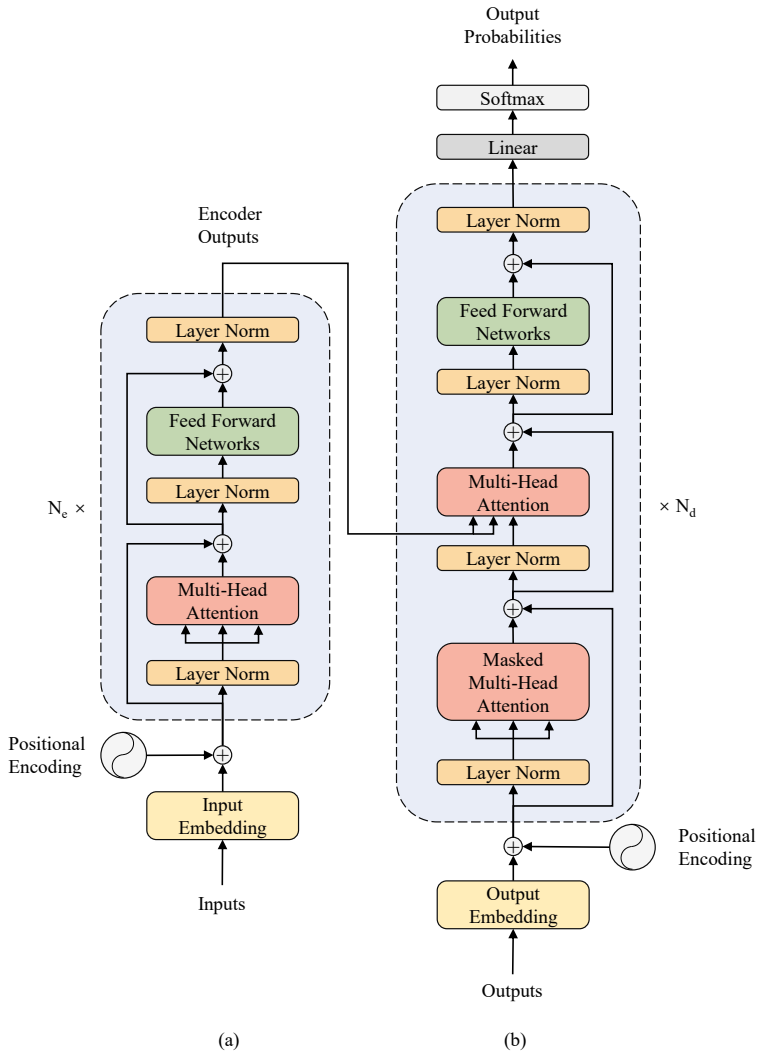


Figure 5.3: Model architecture of the Transformer. (a) Encoder (b) Decoder [54]

encoder, layer normalization and residual connection are also employed to each sub-block of the decoder. Eventually, the output probabilities are acquired by a linear projection and a subsequent softmax function.

5.3.5 Conformer

Conformer [82, 83] is a state-of-the-art ASR encoder architecture. Different from the Transformer block (as described in Section 5.3.4), it is equipped with a convolution layer to increase the local information modeling capability of the Transformer encoder model [278] and a pair of FFN modules sandwiching the multi-head self-attention module and the integrated convolution module. The Conformer model consists of a Conformer encoder proposed in [82] and a Transformer decoder [54]. The encoder first processes the input with a convolution subsampling layer and then with Conformer blocks, as illustrated

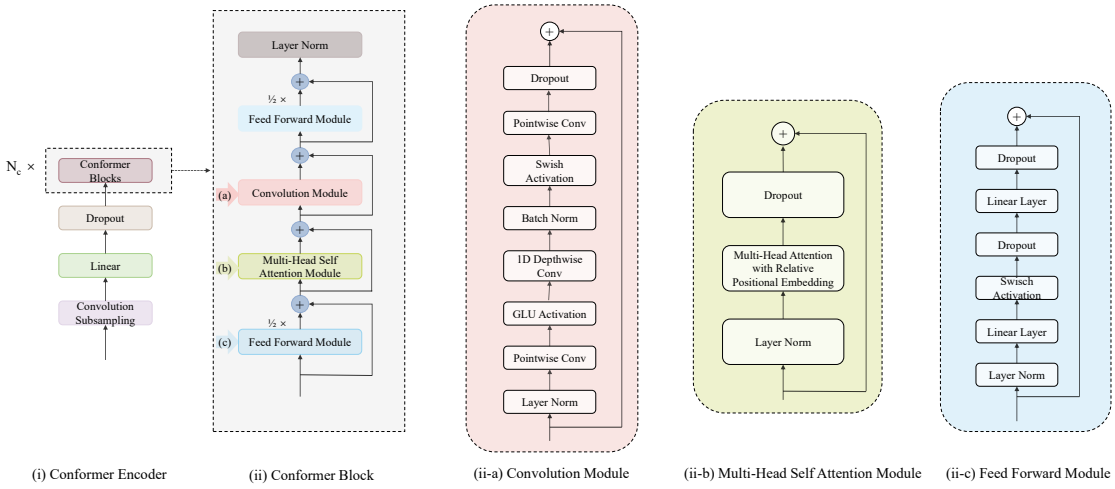


Figure 5.4: Illustration of the Conformer encoder model architecture. (i) Conformer encoder architecture. (ii) Conformer block architecture. (ii-a) Convolution module of the Conformer block. (ii-b) Multi-headed self-attention module of the Conformer block. (ii-c) Feed forward module of the Conformer block.

in Fig. 5.4 (i). The Conformer block (Fig. 5.4 (ii)) consists of a multi-head self-attention module (MHSA), a convolution module, sandwiched by a pair of Macaron-feedforward module [144]. The layer normalization is applied before each module and the dropout is followed by a residual connection afterwards (pre-norm) [322, 291]. Mathematically, let $\boldsymbol{x}_i$ be the input to the ith Conformer block, the output $\boldsymbol{y}_i$ of this block is:

$$\boldsymbol{x}'_i = \boldsymbol{x}_i + \frac{1}{2}\mathrm{FFN}(\boldsymbol{x}_i), \tag{5.11}$$

$$\boldsymbol{x}''_i = \boldsymbol{x}'_i + \mathrm{MHSA}(\boldsymbol{x}'_i), \tag{5.12}$$

$$\boldsymbol{x}'''_i = \boldsymbol{x}''_i + \mathrm{Conv}(\boldsymbol{x}''_i), \tag{5.13}$$

$$\boldsymbol{y}_i = \mathrm{Layer\,Norm}(\boldsymbol{x}'''_i + \frac{1}{2}\mathrm{FFN}(\boldsymbol{x}'''_i)). \tag{5.14}$$

$\mathrm{FFN}(\cdot)$, $\mathrm{MHSA}(\cdot)$, $\mathrm{Conv}(\cdot)$, and $\mathrm{Layer\,Norm}(\cdot)$ denote the Macaron-feedforward module, the multi-head self-attention module, the convolution module, and the layer normalization module, respectively. The multi-head self-attention module is the same as in Section 5.3.4.1 and is demonstrated in Fig. 5.4 (ii-b). Sections 5.3.5.1 and 5.3.5.2 will introduce the convolution module and the Macaron-feedforward module, respectively.

5.3.5.1 Convolution Module

Fig. 5.4 (ii-a) demonstrates the details of the convolution module. The convolution module starts with a 1-dim pointwise convolution layer and a gated linear units (GLU) activation [50]. The 1-dim pointwise convolution layer doubles the input channels,

and the GLU activation splits the input along the channel dimension and executes an element-wise product. What follows are a 1-dim depthwise convolution layer, a batch normalization layer, a Swish activation, and another 1-dim pointwise convolution layer. As mentioned before, the layer normalization is applied before each module and the dropout is followed by a residual connection afterwards (pre-norm).

5.3.5.2 Macaron-Feedforward Module

Unlike the FFN module in Transformer encoder [54], which comprises two linear transformations with a ReLU activation in between (Eq. 5.9), the Conformer encoder [82] introduces another FFN module and substitutes the ReLU activation with the Swish activation. Furthermore, inspired by Macaron-Net [144], this pair of FFN modules are following a half-step scheme and sandwiching the MHSA and the convolution modules. The detail of the FFN is illustrated in Fig. 5.4 (ii-c).

5.4 Adversarial Joint Training

GANs map samples $\tilde{\boldsymbol{x}}$ from the distribution $\tilde{\mathcal{X}}$ to samples $\boldsymbol{x}^*$ from another distribution $\mathcal{X}^*$. The generator is tasked to learn an effective mapping that can imitate the real data distribution to generate novel samples from the manifold defined in $\mathcal{X}^*$, by means of an adversarial training exerted by the discriminator. During back-propagation, the discriminator classifies real samples from the fake samples more accurately; in return, the generator updates its parameters towards the real data manifold, till the mixed Nash equilibria are reached [74]. For more details of the adversarial training, please refer to Section 4.1.3.1 and references therein.

In the proposed robust end-to-end speech recognition scheme, the discriminant network first acts as the local guide for the enhancement module, where the discriminator shifts the training of the generator towards the distribution of clean data; thereafter, it is deployed as the global guide for the whole scheme, where the discriminator instructs the generator to output pertinent enhanced data for the subsequent ASR task.

This work first trains the enhancement module, which contains both the generator and the discriminator. To solve the problem of vanishing gradients caused by sigmoid cross-entropy loss for training, the least-squares GAN (LSGAN) with binary coding (1 for real, 0 for fake) is utilized instead of the cross-entropy loss. Consequently, the loss function of the discriminator component changes to

$$\begin{aligned}\min_{\mathrm{D}} \mathcal{L}(\mathrm{D}) = &\frac{1}{2}\mathbb{E}_{\boldsymbol{x}^*,\tilde{\boldsymbol{x}}\sim p_{data}(\boldsymbol{x}^*,\tilde{\boldsymbol{x}})}[\mathrm{D}(\boldsymbol{x}^*,\tilde{\boldsymbol{x}})-1]^2+ \\ &\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z}),\tilde{\boldsymbol{x}}\sim p_{data}(\tilde{\boldsymbol{x}})}[\mathrm{D}(\mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}),\tilde{\boldsymbol{x}})]^2,\end{aligned} \tag{5.15}$$

where $\boldsymbol{z}$ is a latent variable. To minimize the distance between its generations and the clean examples, it is beneficial to add a secondary component to the loss of the generator. Inspired by the effectiveness of L_1 norm in the image manipulation domain [111, 197],

this study deploys it in the generator to gain more fine-grained and realistic results. The magnitude of the L_1 norm is controlled by a new hyper-parameter λ. Hence, the loss function of the generator component becomes

$$\begin{aligned}\min_{\mathrm{G}} \mathcal{L}(\mathrm{G}) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z}),\tilde{\boldsymbol{x}}\sim p_{data}(\tilde{\boldsymbol{x}})}[\mathrm{D}(\mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}),\tilde{\boldsymbol{x}})-1]^2 \\ &+ \lambda \parallel \mathrm{G}(\boldsymbol{z},\tilde{\boldsymbol{x}}) - \boldsymbol{x}^* \parallel_1 .\end{aligned} \tag{5.16}$$

In the joint training, the enhancement module is initialized from the trained generator component, while the global discriminant module is initialized from the trained discriminator component. The training of the ASR component is based on the cross-entropy criterion, namely

$$\mathcal{L}_{asr} = -\ln P(\boldsymbol{y}^*|\boldsymbol{u}) = -\sum_n \ln P(\boldsymbol{y}_n^*|\boldsymbol{u}, \boldsymbol{y}_{1:n-1}^*), \tag{5.17}$$

where $\boldsymbol{y}^*$ is the ground truth of a whole sequence of output labels and $\boldsymbol{y}_{1:n-1}^*$ is the ground truth from output step 1 to $n-1$. $\boldsymbol{u}$ is the extracted normalized log FBank features by Eq. 5.2. In the proposed framework, the parameters of all procedures, e.g., enhancement, feature extraction, ASR, and the discriminant network, are updated by stochastic gradient descent calculated by the loss function of the whole scheme. It is composed of three losses: $\mathcal{L}_{asr}$, $\mathcal{L}_{enh}$, and $\mathcal{L}_{gan}$, which correspond to Eqs. 5.17, 5.16, and 5.15, i.e.,

$$\mathcal{L} = \mathcal{L}_{asr} + \zeta\mathcal{L}_{enh} + \rho\mathcal{L}_{gan}, \tag{5.18}$$

where ζ and ρ are two hyper-parameters weighting the magnitude of the enhancement loss and adversarial loss. Notably, the scheme targets the recognition performance, and the loss function of the discriminant network adapts the enhancement module implicitly. As a result, the discriminant network guides the enhancement module to serve the subsequent ASR task more properly. Accordingly, the unnecessary speech distortion caused by the enhancement process is alleviated.

5.5 Experimental Setups

This study systematically evaluates the robustness of the adversarial joint training framework, and ablation tests are conducted to validate the effects of (i) the enhancement front-end on the ASR task, (ii) the joint training on the whole scheme, and (iii) the GAN on the joint training.

5.5.1 Corpus

All experiments are executed on the open source Mandarin speech corpus, AISHELL-1 [35]. This corpus is 178 h long, and its utterances contain 11 domains, e.g., smart home, autonomous driving, industrial production, etc. 400 speakers from different accent areas in China participate in the recording. The corpus is divided into training, development,

and test sets. The training dataset contains 120,098 utterances from 340 speakers, the development dataset contains 14,326 utterances from 40 speakers, and the test dataset contains 7,176 utterances from 20 speakers.

For the noisy data, this study contaminates clean utterances in AISHELL-1 with 9 sorts of intrusions from the NOISEX-92 dataset [276] artificially as noisy utterances. The noisy training, development, and test sets are created in the same manner. Note that besides the "matched" noisy test set, which is contaminated by the same intrusions as the training dataset, this study also corrupts the test set with the rest 5 sorts of intrusions in the NOISEX-92 dataset as "unmatched" test materials. Table 5.1 exhibits the sorts the intrusions mixed in "match" and "unmatch" cases. All utterances are mixed with the intrusions at SNRs randomly sampled between [0dB, 20dB]. To sum up, there are two sorts of datasets for training:

- Clean: Clean utterances from the training dataset of AISHELL-1.
- Match: Contaminated clean utterances (training dataset) with "matched" noises of Table 5.1.

For test datasets, there are:

- Clean: Clean utterances from the test dataset of AISHELL-1.
- Match: Contaminated clean utterances of the test set with the same intrusions ("matched" noises of Table 5.1) as "matched" training set.
- Unmatch: Contaminated clean utterances of the test set with different intrusions ("unmatched" noises of Table 5.1) from "matched" training set.

5.5.2 Baseline

For the comparison purpose, the competitive work from [26] is taken as the baseline model.

The mask-based enhancement network is deployed as the front-end. It estimates a masking function to multiply the frequency-domain feature of the noisy speech to form an estimate of the clean speech. For the ASR task, the ESPnet model [296] is employed. It consists of an encoder network that maps the input feature sequence into a higher-level representation. Then a location-based attention layer integrates the representation into a context vector with the attention weight vector. In the end, the decoder predicts the next output conditioned on the full sequence of previous predictions. Besides, there is an extra discriminant network, whose loss is weighted in the loss function of the whole scheme to optimize the joint training.

Importantly, the baseline model does not contain any self-attention layer. Furthermore, the discriminant work in the baseline model is an extra auxiliary module, which does not participate in the enhancement training directly. By contrast, this work benefits from self-attention mechanism and the discriminant module exits innately, which is a component of the enhancement front-end. It acts as the local guide for the enhancement

training, leading the enhancement network to output towards the distribution of the clean samples. Simultaneously, it also plays the role of the global guide, instructing the enhancement module and the ASR module to be better matched.

Table 5.1: The demonstration of categories of intrusions utilized in "match" and "unmatch" cases.

match	
Intrusion	Description
White Noise	Analog noise generator
Factory Floor Noise 1	Plate-cutting and electrical welding
Cockpit Noise 1	Buccaneer jet traveling at 190 knots
Cockpit Noise 3	F-16
Engine Room Noise	Destroyer
Military Vehicle Noise	Leopard 1 vehicle
Machine Gun Noise	Gun
Vehicle Interior Noise	Volvo 340
HF Channel Noise	HF radio channel
unmatch	
Intrusion	Description
Pink Noise	Analog noise generator
Factory Floor Noise 2	Car production hall
Cockpit Noise 2	Buccaneer jet traveling at 450 knots
Operations Room Background Noise	Destroyer
Military Vehicle Noise	M109

5.5.3 Configurations

5.5.3.1 Baseline

For the enhancement front-end, the input is the 257-dim logarithmic STFT features, and all input vectors are normalized to have the zero mean and the unit variance. The network is composed of 3-layer LSTM with 128 nodes, followed by a linear layer with the sigmoid activation function. The network outputs the masking estimate, whose size is equal to the input size, multiplying by the STFT feature of the noisy speech to estimate the clean speech.

For the ASR network, the input is the 80-dim normalized log FBank features transformed from the enhanced STFT features. The encoder is composed of 4-layer BLSTM

with 320 cells, while the decoder is composed of 1-layer unidirectional LSTM with 320 cells. After each BLSTM layer, a linear projection layer with 320 nodes is used to combine the forward and backward LSTM outputs. The location-based attention mechanism comprises 10 centered convolution filters of width 100. Besides, this study also adopts a joint CTC-attention multitask loss function [120] with the CTC loss weight as 0.1.

The discriminant network consists of a 4-layer convolution network, each of which is followed by the ReLU activation function [172].

For decoding, this work [26] uses a beam search algorithm with the beam size 12. CTC rescores the hypotheses with 0.1 weight [120]. Besides, an external RNN language model is also adopted with 0.2 weight during decoding.

5.5.3.2 The Proposed Joint Training Scheme

SASEGAN SASEGAN [202] is trained for 86 epochs with RMSprop [270] and a learning rate of 0.0002. The batch size is 50. During training, this study extracts 1-second chunks of raw waveforms ($L = 16,384$ samples) with a 50 % overlap. During the test, the window is slid without overlapping through the whole duration of test utterances and the outputs are concatenated at the end of the stream. During both training and test, a high-frequency preemphasis filter is employed with a coefficient of 0.95 to all inputs. For the self-attention layer in SASEGAN, this work sets $b = 8$ and $p = 4$ for memory reduction. Phan et al. [202] suggest that the placement of the self-attention layer does not show a clear difference on the performance, which indicates that applying the self-attention to the higher-level (de)convolutional layer is expected to be as good as to a lower layer. Compromising between the computation time and memory requirement and the performance, the self-attention layer is placed in the 10th layer ($\bar{l}$=10).

FBank extraction network The FBank feature extraction network is a linear layer to transform the raw outputs from the upstream SASEGAN to the downstream ASR procedure. 80-dim filterbanks are extracted with the window size of 25 ms and the window shift of 10 ms, extended with the temporal first- and second-order differences. Thereafter, the logarithmic calculation and the global mean and variance normalization are performed according to Eq. 5.5.

Transformer For training the Transformer [54], the Adam optimizer is adopted [121] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, and vary the learning rate over the course of training according to the formula:

$$lr = \varpi \cdot d_{model}^{-0.5} \cdot \min(\tau^{-0.5}, \tau \times warmup_{\tau}^{-1.5}), \tag{5.19}$$

where τ denotes the step number. ϖ is a tunable scalar, which is set to be 10 initially and is declined to 1 when the model converges. The learning rate increases linearly during the first $warmup_{\tau} = 25000$ steps, and afterwards, it decreases proportionally to the inverse square root of the step number. The residual dropout is applied to each sub-block before adding the residual information, while the attention dropout is performed on

the softmax activation in each attention. Both of these aforementioned dropouts are set to be 0.1. Additionally, this work guides the system to be more attentive on closer positions by punishing the attention weights of more distant position-pairs. Similar to the baseline model, a joint CTC-attention multi-task loss function is also adopted [120], with the CTC loss weight as 0.3. In the decoding, the beam size is set to be 12 and length penalty η to be 1.0 [304]. Besides, an external RNN language model with 0.3 weight is also integrated. The training procedure is stopped after 30 epochs.

Conformer The model hyper-parameters of the Conformer [83] are: N_{e}=12, N_{d}=6, h=4, d_k=256 and d_{ff}=2048. The convolution subsampling layer possesses a 2-layer CNN with 256 channels, stride with 2, and kernel size with 3. The kernel size of the convolution module is 31. The dropout is applied in each residual unit of the Conformer with the weight 0.1. The same as Transformer, the Conformer is trained with the Adam optimizer [121] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$ and a Transformer learning rate schedule [278] with 10000 warm-up steps. The learning rate is peaked at $0.05/\sqrt{d_c}$, where d_c is the model dimension in the Conformer encoder. Please note that neither the speed perturbation [125] nor SpecAugment [191] for the data augmentation is included to exclude extra tricks that could cause performance improvements. The training procedure is stopped after 30 epochs.

5.6 Results

CER is utilized to quantify the system performance in all experiments. This study reports CER of the AISHELL-1 test set on three conditions: "clean" refers to the original clean test dataset of the corpus, "match" denotes the noisy test dataset contaminated by "matched" sorts of intrusions in Table 5.1, and "unmatch" means the noisy test set corrupted by the rest of "unmatched" sorts of intrusions in Table 5.1. To validate the efficacy of the enhancement front-end, this study also introduces multi-condition training (MCT), a popular training strategy for robust speech recognition, for comparison. Different from the training data generation of the speech enhancement front-end, the training data of MCT contains 10 % clean utterances that are chosen randomly from the training set of AISHELL-1. Except for the 10 % partition, the remaining 90 % of the training data is generated in the same manner as that of the enhancement front-end, namely being corrupted by the "matched" intrusions from Table 5.1 at an SNR in the range [0dB, 20dB].

Firstly, this study trains the ASR network with the original clean utterance and MCT strategy. The results are shown in Table 5.2.

Ranking these three models from the aspect of ASR performance, the first is Conformer, then Transformer, and the last is the baseline model, consistent with observations in [54, 82]. However, their performance deteriorates rapidly in the noisy test set, demonstrating the necessity of the robustness investigation. The MCT training considerably improves the system's robustness. Its performance on the "matched" test set outperforms the clean training by 63.2 % and 62.9 % relative, in cases of Transformer and Conformer

Table 5.2: CER[%] results of ASR system trained by clean data and multi-condition training (MCT) without the enhancement.

ASR	Training	CER[%]		
		clean	match	unmatch
baseline	clean	14.0	60.0	61.6
	MCT	14.6	20.8	27.3
Transformer	clean	8.0	35.6	37.1
	MCT	7.8	13.1	16.2
Conformer	clean	6.5	32.6	31.7
	MCT	6.9	12.1	13.7

model, respectively; while on the "unmatched" dataset, it outperforms the clean training by 56.3 % and 56.8 % relative, in cases of Transformer and Conformer model, respectively.

Table 5.3: The impacts of the enhancement front-end on ASR systems trained by clean data and MCT strategy. Results are in CER[%].

ASR	Training	CER[%]		
		clean	match	unmatch
baseline	clean	13.9	25.8	53.8
	MCT	14.9	23.5	34.3
Transformer	clean	8.1	19.1	33.7
	MCT	7.9	14.9	20.7
Conformer	clean	6.5	17.6	28.8
	MCT	7.0	14.2	17.9

Secondly, this study trains SASEGAN with the training data contaminated by "matched" intrusions in Table 5.1 to enhance the noisy speech. Then the enhanced features are used for the downstream ASR task. Importantly, the ASR models are taken over from the same well-trained model as in Table 5.2, which means that the enhancement front-end and the ASR back-end are trained separately by different objectives. As exhibited in Table 5.3, the enhancement module tremendously improves the performance of the ASR component that is trained by the clean data merely. Compared to Table 5.2, it outperforms all of the three ASR modules (baseline, Transformer, Conformer) without the enhancement front-end. The improvement achieved in the "matched" dataset is more remarkable than that achieved in the "unmatched" test set. For instance, it outperforms the Conformer without the enhancement module by 46.0 % in the "matched"

test set while 9.1 % in the "unmatched" test set. This difference is due to the fact that SASEGAN is trained with the "matched" intrusions and can enhance the data contaminated by the same intrusions better during the test. All these improvements confirm the efficacy of the enhancement module for improving the robustness of the ASR system. Nevertheless, improving the robustness of the framework in unseen noisy environments still remains to be a challenge. Additionally, the speech enhancement module deteriorates the performance of ASR_MCT networks, which stays in accordance with the observations in [53] and [174]. Donahue et al. [53] and Narayanan and Wang [174] hypothesize that the enhancement front-end might be introducing hitherto-unseen distortions that compromise performance. Furthermore, the author believes that this latent distortion is derived from the independent training of the enhancement module, demonstrating the necessity of the joint training strategy.

Table 5.4: CER[%] results of the SE-ASR system retraining with and without noisy features.

ASR	Retraining	CER[%]		
		clean	match	unmatch
Transformer_MCT	no	7.8	13.1	16.2
	enhanced	7.8	12.9	15.8
	enhanced+noisy	7.8	12.9	15.6
Conformer_MCT	no	6.9	12.1	13.7
	enhanced	6.7	12.0	13.5
	enhanced+noisy	6.7	11.8	13.3

To remedy the deterioration of the performance of the ASR_MCT, this study retrains the network with the enhanced features. Assuming that the network may also benefit from the knowledge of the noisy features, this study also experiments with ingesting both enhanced and noisy features. Results are displayed in Table 5.4. Either the Transformer_MCT model or the Conformer_MCT model is initialized from their existing well-trained MCT checkpoints respectively, setting the additional parameters to zero to ensure the fair training start. As presented in Table 5.4, the retraining with the enhanced features improves the performance in both "matched" and "unmatched" cases, and the retraining with both enhanced and noisy features improves the performance slightly further.

Lastly, this study jointly trains the whole scheme with and without adversarial training according to Eq. 5.18. In the framework, the enhancement front-end is initialized from the generator (G component) of SASEGAN, the ASR back-end is initialized from the corresponding ASR_MCT checkpoint (without retraining), and the adversarial module is initialized from the discriminator (D component) of SASEGAN. When the adversarial module does not participate in the training, the magnitude of the loss function are set as ζ=6.0 and ρ=0; by contrast, when it participates in the training, this

Table 5.5: The impacts of the joint training with and without GAN on SE-ASR pipeline. Results are in CER[%].

SE	ASR	joint training with GANs	CER[%]		
			clean	match	unmatch
SASEGAN	baseline	no	12.8	18.7	25.3
		yes	**12.8**	**18.7**	**24.8**
	Transformer	no	**7.0**	12.4	15.6
		yes	7.2	**12.4**	**15.5**
	Conformer	no	**6.8**	11.9	13.3
		yes	6.9	**11.8**	**13.0**

work sets ζ=6.0 and ρ=3.0. Results are presented in Table 5.5. Compared to Table 5.2, the joint training mitigates the distortion problem existing in the MCT strategy. Additionally, the adversarial joint training improves the performance further; and exceeds the performance of retraining with both enhanced and noisy features in either Transformer or Conformer case. Taking Conformer for example, compared to Conformer trained with clean data merely, the adversarial joint training yields 63.8 % relative and 59.0 % relative improvements on "matched" and "unmatched" datasets, respectively; meanwhile, the adversarial joint training outperforms the MCT strategy by 2.5 % relative and 5.2 % relative on "matched" and "unmatched" datasets, separately. These results indicate the efficacy of the adversarial joint training in improving the robustness of the end-to-end ASR scheme.

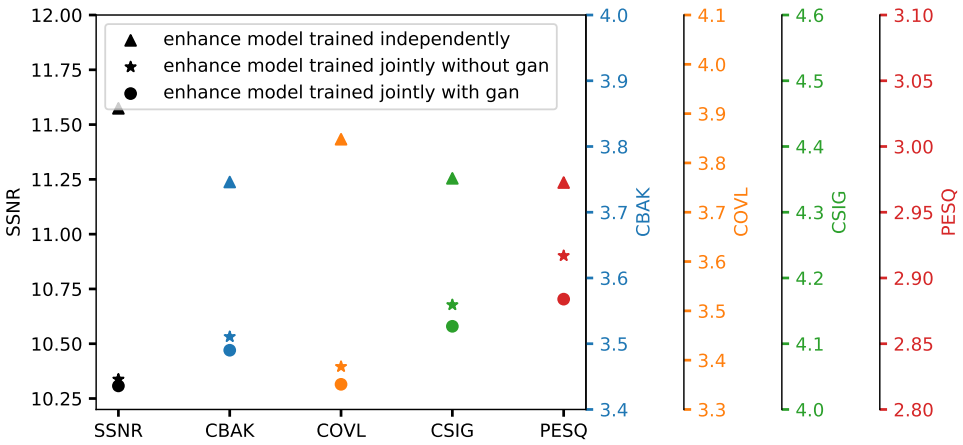


Figure 5.5: The performance comparison of the enhancement model trained independently and the enhancement models trained jointly with the baseline ASR model without and with GAN.

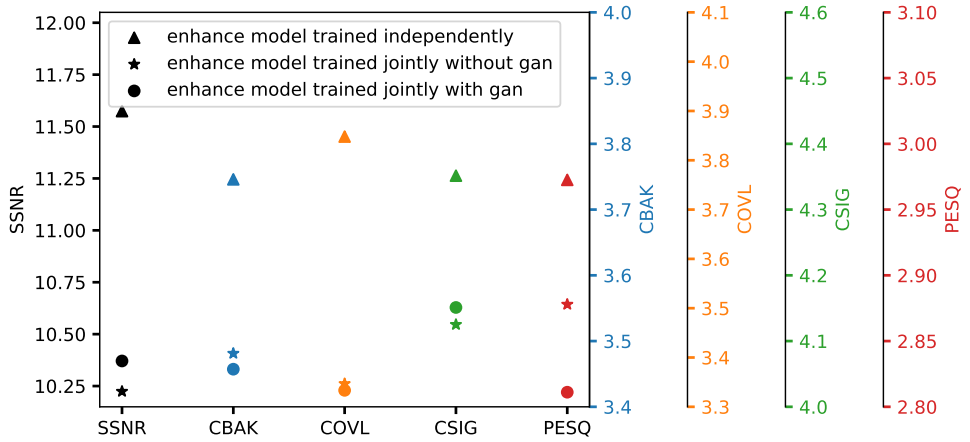


Figure 5.6: The performance comparison of the enhancement model trained independently and the enhancement models trained jointly with Transformer ASR model without and with GAN.

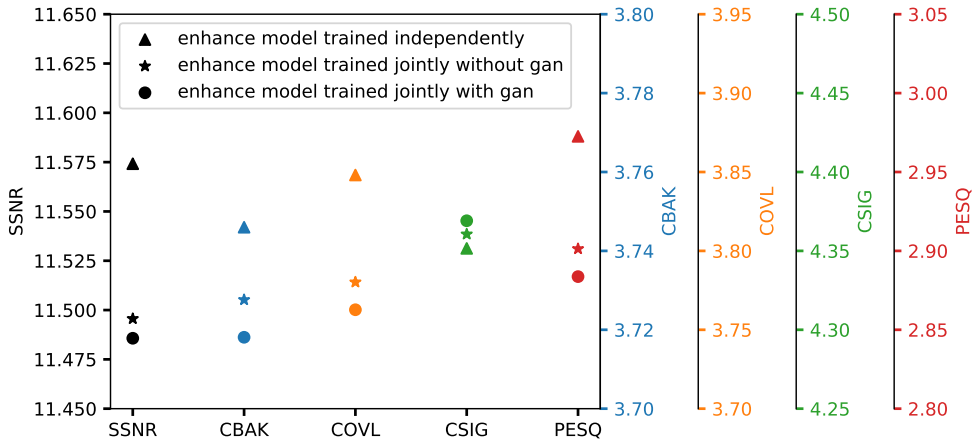


Figure 5.7: The performance comparison of the enhancement model trained independently and the enhancement models trained jointly with Conformer ASR model without and with GAN.

5.7 Discussion

To analyse the difference between these enhancement modules that are trained independently, jointly without GANs, and jointly with GANs, this study quantifies their performance on the five classic objective criteria of speech enhancement: SSNR [211], CBAK [106], CSIG [106], COVL [106], and PESQ [225] (For more details, please refer to Section 4.1.4.2).

All criteria are computed based on the implementation in [142], available at the publisher website[1]. This study quantifies the performance of the enhancement front-end

[1]https://www.crcpress.com/downloads/K14513/K14513_CD_Files.zip

that is trained independently, trained jointly with and without GANs in cases of baseline, Transformer, and Conformer schemes. As exhibited in Figs. 5.5, 5.6, and 5.7, the joint training slightly degrades the enhancement module's performance on SSNR, CBAK, COVL, CSIG, and PESQ generally. These results suggest that these objective criteria cannot indicate the suitability of the enhanced data for ASR task, which vindicates that it is hard for independent training to lead the enhancement module to the global optimum. Another phenomenon which is worth noting is that the discrepancies on CBAK and SSNR suggest that there are conflicts between erasing the noise contamination and averting the speech distortion. Therefore, an equilibrium between these two goals should be sought. The experimental results in Section 5.6 validate the efficacy of the adversarial joint training with a global discriminant guide for reaching the equilibrium point.

Table 5.6: Evolution of performance in unmatched test dataset of Conformer along the value of ζ and ρ.

ζ ($\rho = 0$)	4	5	6	7
CER[%]	15.7	13.9	**13.3**	13.8
$\rho(\zeta = 6)$	1	2	3	4
CER[%]	14.5	14.9	**13.0**	13.9

Besides, this study also conducts ablation tests for the values of ζ and ρ and the results are shown in Table 5.6. The results explain why this work sets the magnitude of the loss function to ζ=6.0 and ρ=0 for the joint training without GAN; and sets ζ=6.0 and ρ=3.0 for the adversarial joint training experiments.

5.8 Summary

This chapter concentrates on the joint training technique, and presents an adversarial joint training framework with the self-attention mechanism to boost the noise robustness of the end-to-end ASR system. The jointly compositional scheme is a fusion of the back- and front-end techniques, and in this chapter, it consists of an enhancement front-end, an ASR back-end, and the discriminant network. A highlight of this proposed framework is that the discriminant component first acts as the guide of the enhancement front-end training; afterwards, it participates in the adversarial joint training as the global instructor, which leads the enhancement front-end to output appropriate enhanced features for the downstream ASR task. Experimental results validate the efficacy of the proposed adversarial joint training strategy.

"Live your life as an Exclamation rather than an Explanation."

— Isaac Newton

6

Conclusion and Outlook

Following years of research, ASR has made significant breakthrough and achieved even human-comparable performance. However, existing ASR systems still suffer from performance degradation in adverse environments, limiting their performance in everyday situations. Therefore, improving robustness of ASR systems is a crucial step towards flexible human-machine interfaces. Techniques of robust ASR can be categorized into three clusters: back-end, front-end, and joint front- and back-end techniques, according to the processing stages of an ASR pipeline. This thesis has contributed to each stage by: (i) proposing the deep neural fenonic baseform growing, a novel approach to optimize HMM-based back-end for robust speech recognition; (ii) proposing two novel lightweight GAN-based architectures to optimize the speech enhancement front-end; (iii) proposing adversarial joint training framework for robust end-to-end speech recognition. Each contribution has been fully introduced and discussed in a separate chapter.

In this chapter, Section 6.1 summarizes the aforementioned contributions of this thesis, and potential directions for future work will be discussed in Section 6.2.

6.1 Summary and Discussions

After an introduction of the general research background, challenges, and related works in Chapter 1, the general framework of automatic speech recognition is introduced in Chapter 2. Chapters 3, 4, and 5 introduce back-end, front-end, and joint-training techniques, respectively. The main contributions of this thesis can be summarized as follows:

Proposing the deep Neural Fenonic Baseform Growing (NFBG) approach.

NFBG is an approach to optimize the HMM structure, leveraging deep learning. This study introduces the concept *fenone* for representing sub-phones as the basic acoustic unit. Fenones are the building block of phones and each fenone is modeled as one state of HMMs, which can be obtained automatically through the deep neural network vector quantizer (DNNVQ) (cf. Section 3.3). NFBG starts with aligning all data against different phones (cf. Section 3.4.2) and then applies DNNVQ on each phone's data.

Next, a DNNVQ generates the fenone labels for every phone, and accordingly, the fenone sequence of every HMM model is obtained. After compacting the fenone sequence to the fenonic baseform by eliminating all successive duplicated fenones, the customized HMM topology of the phone is gained (cf. Section 3.4.3). The comparison among elementary Markov models is also given (cf. Section 3.4.4). Experiments are exerted on Tedlium and TIMIT to evaluate the proposed approach. Firstly, fenonic baseform results of each monophone in Tedlium and TIMIT are given and analyzed in Section 3.6. Next, ablation tests are conducted on the effects of the elementary HMM topology of the fenone (cf. Section 3.7.1) and the number of DNNVQ prototypes (cf. Section 3.7.2). Thereafter, experimental results show that NFBG-based HMMs improve ASR performance in monophone systems (cf. Section 3.7.3), systems with context-dependent inputs (cf. Section 3.7.4), and triphone systems (cf. Section 3.7.5). Last but not least, the efficacy of NFBG for improving robustness of ASR systems is validated (cf. Section 3.7.6).

Proposing the lightweight self-attention augmented GANs for speech enhancement.

Inspired by previous works on speech enhancement GANs (SEGANs), and recent studies on the relation between convolutional layers and self-attention mechanism, this thesis presents a series of SEGANs equipped with the self-attention mechanism in three manners: first, this thesis deploys the stand-alone self-attention layer in SEGAN (cf. Section 4.1.3.3). Next, the study employs locality modeling on the stand-alone self-attention layer (cf. Section 4.1.3.4). Finally, the thesis investigates the functionality of the self-attention augmented convolutional SEGAN (cf. Section 4.1.3.5). The goal here is to probe the performance of the SEGAN equipped (i) with stand-alone standard self-attention layers, (ii) with stand-alone hybrid (global and local) self-attention layers, and (iii) with self-attention augmented convolutional layers. In addition, the parameter scales of these proposed models are also calculated. Systematic experimental results reveal that equipped with the stand-alone self-attention layer, the proposed system outperforms baseline systems in terms of various objective evaluation criteria with up to 95 % fewer parameters. In addition, the locality modeling on the stand-alone self-attention layer delivers further performance improvements without any parameter incrementation. Moreover, the self-attention augmented SEGAN outperforms all baseline systems and achieves the best results on SSNR and STOI of this work, with acceptably increased parameters (cf. Section 4.1.5).

Proposing the lightweight end-to-end GANs using Sinc-convolution for speech enhancement.

Sinc convolution [221] is proposed to learn more meaningful filters in the input layer. However, the implementation of Sinc convolution on speech enhancement tasks is still under-explored. This study aims to transfer the success achieved by the Sinc convolution in the field of speech and speaker recognition to the field of end-to-end speech enhancement.

To achieve this goal, this thesis first optimizes the SEGAN architecture from the seminal work [193], and then enhances the original Sinc convolution layer to fit the advanced SEGAN (cf. Section 4.2.4). Ablation tests are exerted on the configuration of Sinc convolution, including the number of Sinc filters, the kernel size, the input length, and the placement of Sinc convolution layers (cf. Section 4.2.6.1). Experimental results show that the proposed Sinc-SEGAN overtakes a set of competitive baseline models, especially on higher-level perceptual quality and speech intelligibility. Additionally, the system parameters reduce drastically up to merely 17.7 % of the baseline system (cf. Section 4.2.6.2). Moreover, data augmentation methods further boost the system performance (cf. Section 4.2.6.3). Analysis of the learnt Sinc filters reveals that the learnt filter-bank is tuned precisely to select narrow-band speech clues, and hence suitable for the speech enhancement task in the time domain (cf. Section 4.2.6.4).

Proposing self-attention based adversarial joint training framework for robust end-to-end speech recognition.

To obtain the global optimum and circumvent unnecessary distortion introduced by independent training of the front- and back-ends, this study proposes an adversarial joint training framework with self-attention mechanism to boost the robustness of the end-to-end ASR systems, which consists of a self-attention speech enhancement GAN (SASEGAN) (cf. Section 5.3.2) and a self-attention end-to-end ASR model (SA_ASR) (cf. Section 5.3.4 and Section 5.3.5). There are two advantages which are worth noting in this proposed framework. For one thing, it benefits from the advancement of both self-attention mechanism and GANs for the first time. For another, the discriminant component does not concentrate on the enhancement front-end exclusively, but also plays the role of the global discriminant network in the stage of the adversarial joint training, which guides the enhancement front-end to capture more compatible structures for the subsequent ASR module and thereby offsets the limitation of the separate training and handcrafted loss functions (cf. Section 5.4). Experimental results show that on the artificial noisy test set, the proposed framework achieves the relative improvements of up to 66 % compared to the ASR model trained by clean data solely, up to 35.1 % compared to the speech enhancement & ASR scheme without joint training, and up to 5.3 % compared to multi-condition training (cf. 5.5). Moreover, the impacts of the joint training on the speech enhancement training are also analyzed (cf. Section 5.7). Based on all the results, it is safe to draw the conclusion that with the adversarial joint optimization, the proposed framework learns more robust representations suitable for the ASR task.

6.2 Directions for Future Works

Robust speech recognition has always been a challenge for flexible human-machine communication. This thesis aims to address this issue by optimizing the ASR system itself, the auxiliary speech-enhancement front-end, and the training strategy of the whole

scheme. However, the accomplished work reaches only the tip of the iceberg of the robust ASR study, and here are several promising future directions where the further research can proceed.

To begin with, microphone arrays and multi-channel processing techniques also play a significant role in the development of robust ASR. The primary approach of microphone-array techniques is acoustic beamforming. Namely, spatio-temporal filters operate on the outputs of microphone arrays and convert them to single-channel signals while amplifying the waveform from the desired direction and attenuating the noise from other directions. The beamformer outputs are often further enhanced by a microphone array post-filter, and thereafter, the back-end techniques for single-channel speech can be applied to this enhanced data for speech recognition. For future studies, it would be interesting to consider microphone-array approaches and extend the proposed techniques to integrate them, e.g., the Direction-of-Arrival (DOA) estimator and the Delay-and-Sum (DS) beamforming approach [274].

Another special research direction, *adversarial examples*, has been drawing more and more attention. Adversarial samples were first proposed by Szegedy et al. [264] in image classification. The authors find that a formerly correctly classified example could be misclassified easily by a neural network if the image pixels are only slightly skewed from the original ones, even unnoticeable to humans. This kind of perturbation is called *adversarial perturbation*. Experimental results show that many state-of-the-art deep learning models are vulnerable to such adversarial examples [164, 127, 163, 4, 37]. The existence of adversarial examples indicates that there are *blind spots* in the input space. In other words, the models are unsmooth because a tiny perturbation in input space could trigger a drastic change in the output space. Plenty of attempts have been executed on improving the robustness of ASR system against adversarial test samples [75, 112, 209, 176]. Therefore, one direction in this line can be generating and implementing adversarial samples in the training of proposed systems to make them invariant to adversarial perturbation. As the result, the overall robustness of ASR systems will be improved accordingly.

Last but not least, unsupervised learning is a natural way to mitigate the performance degradation of ASR systems due to substantial mismatches existing between training and test conditions. Unsupervised learning attempts to extract knowledge from unlabeled data, and can potentially discover representations that capture the underlying structure of such data. Several approaches have been proposed for unsupervised learning in the last decade. Notable examples are deep autoencoders [24], restricted Boltzmann machines [100], variational autoencoders [122], and generative adversarial networks [74]. A related sub-field of unsupervised learning is self-supervised learning, where targets are computed from the signal itself [52, 72]. Pascual et al. [195] propose to jointly tackle multiple self-supervised tasks using an ensemble of neural networks that cooperate to discover good speech representations, named the problem-agnostic speech encoder (PASE). Since the proposed approach imposes several constraints into the learnt representations, it is more likely to learn general, robust, and transferable features, and less likely to focus on superficial features of the signal which may be sufficient for the given training data but are insufficient when considering broader types of data. Another related sub-field

is domain adaption. Ghorbani et al. [71] study several domain expansion techniques which exploit only the data of the new domain to build a stronger model for all domains. These techniques are aimed at learning the new domain with a minimal forgetting effect (i.e., they maintain original model performance). One direction of future work can be leveraging problem-agnostic speech representations and domain expansion to make ASR systems more robust across all acoustic conditions.

This thesis has investigated techniques of the back-end, front-end, and the joint training of the whole ASR framework for robust speech recognition. Hopefully, the community can benefit from this work. Although combining human and machine intelligence to improve the ASR system robustness in every acoustic condition is still a long way off, the proposed algorithms may inspire future works.

"One day, in retrospect, the years of struggle will strike you as the most beautiful."

– Sigmund Freud

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016.

[2] K. T. Abou-Moustafa, M. Cheriet, and C. Y. Suen. On the structure of hidden Markov models. *Pattern Recognition Letters*. Elsevier, 2004.

[3] K. Ait-Mohand, T. Paquet, and N. Ragot. Combining structure and parameter adaptation of HMMs for printed text recognition. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2014.

[4] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

[5] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International conference on machine learning*. PMLR, 2016.

[6] K. T. Andersen and M. Moonen. Robust speech-distortion weighted interframe Wiener filters for single-channel noise reduction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2017.

[7] A. Arora, D. Raj, A. S. Subramanian, K. Li, B. Ben-Yair, M. Maciejewski, P. Żelasko, P. García, S. Watanabe, and S. Khudanpur. The JHU multi-microphone multi-speaker ASR system for the CHiME-6 challenge. *arXiv preprint arXiv:2006.07898*, 2020.

[8] A. R. Avila, M. J. Alam, D. D. O'Shaughnessy, and T. H. Falk. Investigating speech enhancement and perceptual quality for speech emotion recognition. In *INTERSPEECH*, 2018.

[9] D. Baby. isegan: Improved speech enhancement generative adversarial networks. *arXiv preprint arXiv:2002.08796*, 2020.

[10] D. Baby and S. Verhulst. Sergan: Speech enhancement using relativistic generative adversarial networks with gradient penalty. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[11] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[12] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016.

[13] L. Bahl, P. Brown, P. De Souza, and M. Picheny. Acoustic Markov models used in the Tangora speech recognition system. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*. IEEE Computer Society, 1988.

[14] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer, and M. A. Picheny. A method for the construction of acoustic Markov models for words. *IEEE Transactions on Speech and Audio processing*. IEEE, 1993.

[15] R. Bakis. Continuous speech recognition via centisecond acoustic states. *The Journal of the Acoustical Society of America*. Acoustical Society of America, 1976.

[16] J. Barker, R. Marxer, E. Vincent, and S. Watanabe. The third "CHiME" speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015.

[17] J. Barker, E. Vincent, N. Ma, H. Christensen, and P. Green. The PASCAL CHiME speech separation and recognition challenge. *Computer Speech & Language*. Elsevier, 2013.

[18] J. Barker, S. Watanabe, E. Vincent, and J. Trmal. The fifth "CHiME" speech separation and recognition challenge: dataset, task and baselines. *arXiv preprint arXiv:1803.10609*, 2018.

[19] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Statist.* The Institute of Mathematical Statistics, 12 1966.

[20] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics*. JSTOR, 1970.

[21] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[22] Y. Bengio. A connectionist approach to speech recognition. In *Advances in Pattern Recognition Systems Using Neural Network Technologies*, pages 3–23. World Scientific, 1993.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The journal of machine learning research*. JMLR. org, 2003.

[24] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, 2007.

[25] A. Biem. A model selection criterion for classification: Application to HMM topology optimization. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* IEEE, 2003.

[26] L. Bin, N. Shuai, L. Shan, L. Wenju, Y. Meng, C. Lianwu, P. Shouye, L. Changliang, et al. Jointly adversarial enhancement training for robust end-to-end speech recognition. ISCA, 2019.

[27] E. Bocchieri and D. Dimitriadis. Investigating deep neural network based transforms of robust audio features for LVCSR. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.

[28] S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1979.

[29] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*. Nimes, 1991.

[30] L. Bottou, F. F. Soulié, P. Blanchet, and J.-S. Lienard. Experiments with time delay networks and dynamic time warping for speaker independent isolated digits recognition. In *First European Conference on Speech Communication and Technology*, 1989.

[31] H. Bourlard and C. J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Transactions on pattern analysis and machine intelligence*. IEEE, 1990.

[32] H. A. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

[33] J. S. Bridle. Alpha-nets: a recurrent "neural" network architecture with a hidden Markov model interpretation. *Speech Communication*. Elsevier, 1990.

[34] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[35] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng. Aishell-1: An open-source Mandarin speech corpus and a speech recognition baseline. In *20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, 2017.

[36] M. Cai and J. Liu. Maxout neurons for deep convolutional and LSTM neural networks in speech recognition. *Speech Communication*. Elsevier, 2016.

[37] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018.

[38] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[39] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

[40] W. Chan and I. Lane. Deep recurrent neural networks for acoustic modelling. *arXiv preprint arXiv:1504.01482*, 2015.

[41] J. Cheng, R. Liang, and L. Zhao. DNN-based speech enhancement with self-attention on feature dimension. *Multimedia Tools and Applications*. Springer, 2020.

[42] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[43] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

[44] N. Cirera, A. Fornés, and J. Lladós. Hidden Markov model topology optimization for handwriting recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015.

[45] J.-B. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

[46] Z. Cui and C. Bao. Power exponent based weighting criterion for DNN-based mask approximation in speech enhancement. *IEEE Signal Processing Letters*. IEEE, 2021.

[47] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*. Springer, 1989.

[48] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013.

[49] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*. IEEE, 2011.

[50] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*. PMLR, 2017.

[51] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*. IEEE, 1980.

[52] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[53] C. Donahue, B. Li, and R. Prabhavalkar. Exploring speech enhancement with generative adversarial networks for robust speech recognition. In *Proc. ICASSP*. IEEE, 2018.

[54] L. Dong, S. Xu, and B. Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *Proc. ICASSP*. IEEE, 2018.

[55] L. Dong, S. Zhou, W. Chen, and B. Xu. Extending recurrent neural aligner for streaming end-to-end speech recognition in Mandarin. *arXiv preprint arXiv:1806.06342*, 2018.

[56] J. Droppo and A. Acero. Joint discriminative front end and back end training for improved speech recognition accuracy. In *Proc. INTERSPEECH*. IEEE, 2006.

[57] J. L. Elman. Finding structure in time. *Cognitive science*. Wiley Online Library, 1990.

[58] Y. Ephraim. A minimum mean square error approach for speech enhancement. In *Proc. ICASSP*. IEEE, 1990.

[59] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proc. ICASSP*. IEEE, 2015.

[60] F. Eyben, M. Wöllmer, B. Schuller, and A. Graves. From speech to letters-using a novel neural network architecture for grapheme based ASR. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2009.

[61] J. Fainberg, O. Klejch, E. Loweimi, P. Bell, and S. Renals. Acoustic model adaptation from raw waveforms with SincNet. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019.

[62] C. Fan, B. Liu, J. Tao, J. Yi, Z. Wen, and Y. Bai. Noise prior knowledge learning for speech enhancement via gated convolutional generative adversarial network. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019.

[63] C. Fan, J. Yi, J. Tao, Z. Tian, B. Liu, and Z. Wen. Gated recurrent fusion with joint training framework for robust end-to-end speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2020.

[64] S.-W. Fu, T.-y. Hu, Y. Tsao, and X. Lu. Complex spectrogram enhancement by convolutional neural network with multi-metrics learning. In *2017 IEEE 27th international workshop on machine learning for signal processing (MLSP)*. IEEE, 2017.

[65] M. Fujimoto and H. Kawai. One-pass single-channel noisy speech recognition using a combination of noisy and enhanced features. In *INTERSPEECH*, 2019.

[66] S. Furui. Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1986.

[67] T. Gao, J. Du, L. Dai, and C. Lee. Joint training of front-end and back-end deep neural networks for robust speech recognition. In *Proc. ICASSP*. IEEE, 2015.

[68] J. S. Garofolo. TIMIT acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

[69] J. Geiger, J. Schenk, F. Wallhoff, and G. Rigoll. Optimizing the number of states for HMM-based on-line handwritten whiteboard recognition. In *2010 12th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2010.

[70] J. T. Geiger, Z. Zhang, F. Weninger, B. Schuller, and G. Rigoll. Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[71] S. Ghorbani, S. Khorram, and J. H. Hansen. Domain expansion in DNN-based acoustic models for robust speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[72] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[73] T. Goehring, F. Bolner, J. J. Monaghan, B. Van Dijk, A. Zarowski, and S. Bleeck. Speech enhancement based on neural networks improves speech intelligibility in noise for cochlear implant users. *Hearing research*. Elsevier, 2017.

[74] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 2014.

[75] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[76] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[77] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[78] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

[79] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*. PMLR, 2014.

[80] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013.

[81] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*. Elsevier, 2005.

[82] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented Transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[83] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi, et al. Recent developments on ESPnet toolkit boosted by Conformer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[84] Q. Guo, X. Qiu, X. Xue, and Z. Zhang. Low-rank and locality constrained self-attention for sequence modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2019.

[85] P. Haffner. Connectionist word-level classification in speech recognition. In *1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1992.

[86] T. Hain, P. C. Woodland, G. Evermann, M. J. Gales, X. Liu, G. L. Moore, D. Povey, and L. Wang. Automatic transcription of conversational telephone speech. *IEEE Transactions on Speech and Audio Processing*. IEEE, 2005.

[87] J. Han and C. Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*. Springer, 1995.

[88] K. J. Han, J. Huang, Y. Tang, X. He, and B. Zhou. Multi-stride self-attention for speech recognition. In *Proc. INTERSPEECH*, 2019.

[89] K. J. Han, R. Prieto, and T. Ma. State-of-the-art speech recognition using multi-stream self-attention with dilated 1D convolutions. In *Proc. ASRU*. IEEE, 2019.

[90] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. *arXiv preprint arXiv:1408.2873*, 2014.

[91] S. Hazmoune, F. Bougamouza, S. Mazouzi, and M. Benmohammed. A new hybrid framework based on hidden Markov models and K-nearest neighbors for speech recognition. *International Journal of Speech Technology*. Springer, 2018.

[92] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 2015.

[93] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[94] T. He and J. Droppo. Exploiting LSTM structure in deep neural networks for speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[95] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *the Journal of the Acoustical Society of America*. Acoustical Society of America, 1990.

[96] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[97] J. Heymann, L. Drude, and R. Haeb-Umbach. A generic neural acoustic beamforming architecture for robust multi-channel speech processing. *Computer Speech & Language*. Elsevier, 2017.

[98] T. Higuchi, K. Kinoshita, M. Delcroix, and T. Nakatani. Adversarial training for data-driven speech enhancement without parallel corpus. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017.

[99] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*. IEEE, 2012.

[100] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2006.

[101] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*. MIT Press, 1997.

[102] T. Hori, W. Wang, Y. Koji, C. Hori, B. Harsham, and J. R. Hershey. Adversarial training and decoding strategies for end-to-end neural conversation models. *Computer Speech & Language*. Elsevier, 2019.

[103] T. Hori, S. Watanabe, Y. Zhang, and W. Chan. Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. *arXiv preprint arXiv:1706.02737*, 2017.

[104] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*. Elsevier, 1989.

[105] H. Hu, T. Tan, and Y. Qian. Generative adversarial networks based data augmentation for noise robust speech recognition. In *Proc. ICASSP*. IEEE, 2018.

[106] Y. Hu and P. C. Loizou. Evaluation of objective quality measures for speech enhancement. *IEEE Transactions on audio, speech, and language processing*. IEEE, 2007.

[107] X. Huang, A. Acero, H.-W. Hon, and R. Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[108] K. Hwang and W. Sung. Character-level incremental speech recognition with recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[109] M.-Y. Hwang and X. Huang. Shared-distribution hidden Markov models for speech recognition. *IEEE Transactions on Speech and Audio Processing*. IEEE, 1993.

[110] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[111] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[112] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[113] Z. Jiang, X. Ding, L. Peng, and C. Liu. Analyzing the information entropy of states to optimize the number of states in an HMM-based off-line handwritten Arabic word recognizer. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012.

[114] B.-H. Juang and L. R. Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on acoustics, speech, and signal Processing*. IEEE, 1990.

[115] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix. Semi-supervised end-to-end speech recognition. In *Interspeech*, 2018.

[116] J. Katona. Examination and comparison of the EEG based attention test with CPT and TOVA. In *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2014.

[117] J. Katona and A. Kovari. The evaluation of BCI and PEBL-based attention tests. *Acta Polytechnica Hungarica*, 2018.

[118] J. Katona, T. Ujbanyi, G. Sziladi, and A. Kovari. Examine the effect of different web-based media on human brain waves. In *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE, 2017.

[119] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*. IEEE, 1987.

[120] S. Kim, T. Hori, and S. Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017.

[121] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[122] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[123] K. Kinoshita, M. Delcroix, S. Gannot, E. A. Habets, R. Haeb-Umbach, W. Kellermann, V. Leutnant, R. Maas, T. Nakatani, B. Raj, et al. 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*. Springer, 2016.

[124] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *1995 international conference on acoustics, speech, and signal processing*. IEEE, 1995.

[125] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[126] Y. Koizumi, K. Yatabe, M. Delcroix, Y. Masuyama, and D. Takeuchi. Speech enhancement using self-adaptation and multi-head self-attention. In *Proc. ICASSP*. IEEE, 2020.

[127] A. Kurakin, I. Goodfellow, S. Bengio, et al. Adversarial examples in the physical world, 2016.

[128] Y.-H. Lai and W.-Z. Zheng. Multi-objective learning based speech enhancement method to increase speech quality and intelligibility for hearing aid device users. *Biomedical Signal Processing and Control*. Elsevier, 2019.

[129] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural networks*. Elsevier, 1990.

[130] D. Le, X. Zhang, W. Zheng, C. Fügen, G. Zweig, and M. L. Seltzer. From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019.

[131] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in neural information processing systems*, 1990.

[132] Y. LeCun et al. Generalization and network design strategies. *Connectionism in perspective*. Elsevier Zurich, Switzerland, 1989.

[133] H. Lee, P. Pham, Y. Largman, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*. Citeseer, 2009.

[134] K.-F. Lee. On large-vocabulary speaker-independent continuous speech recognition. *Speech communication*. Elsevier, 1988.

[135] B. Li, T. N. Sainath, K. C. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao. Multi-dialect speech recognition with a single sequence-to-sequence model. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018.

[136] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero. HMM adaptation using a phase-sensitive acoustic distortion model for environment-robust speech recognition. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008.

[137] J. Li, H. Zhang, X. Cai, and B. Xu. Towards end-to-end speech recognition for Chinese Mandarin using long short-term memory recurrent neural networks. In *Sixteenth annual conference of the international speech communication association*, 2015.

[138] P. Li, Z. Jiang, S. Yin, D. Song, P. Ouyang, L. Liu, and S. Wei. PAGAN: A phase-adapted generative adversarial networks for speech enhancement. In *ICASSP 2020*

- *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[139] Z.-X. Li, L.-R. Dai, Y. Song, and I. McLoughlin. A conditional generative model for speech enhancement. *Circuits, Systems, and Signal Processing*. Springer, 2018.

[140] J. Lim and A. Oppenheim. All-pole modeling of degraded speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. IEEE, 1978.

[141] B. Liu, S. Nie, Y. Zhang, D. Ke, S. Liang, and W. Liu. Boosting noise robustness of acoustic model via deep adversarial training. In *Proc. ICASSP*. IEEE, 2018.

[142] P. C. Loizou. *Speech enhancement: theory and practice*. CRC press, Boca Raton, FL, USA, 2013.

[143] X. Lu, Y. Tsao, S. Matsuda, and C. Hori. Ensemble modeling of denoising autoencoder for speech spectrum restoration. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[144] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, and T.-Y. Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

[145] H. Luo, S. Zhang, M. Lei, and L. Xie. Simplified self-attention for transformer-based end-to-end speech recognition. *arXiv preprint arXiv:2005.10463*, 2020.

[146] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney. RWTH ASR systems for Librispeech: Hybrid vs attention-w/o data augmentation. *arXiv preprint arXiv:1905.03072*, 2019.

[147] A. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng. Lexicon-free conversational speech recognition with neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[148] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, 2013.

[149] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 2008.

[150] J. Malek and J. Zdansky. On practical aspects of multi-condition training based on augmentation for reverberation-/noise-robust speech recognition. In *International Conference on Text, Speech, and Dialogue*. Springer, 2019.

[151] C. Manning and H. Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[152] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[153] W. J. Masek and M. S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*. Elsevier, 1980.

[154] I. A. McCowan and H. Bourlard. Microphone array post-filter based on noise field coherence. *IEEE Transactions on Speech and Audio Processing*. IEEE, 2003.

[155] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. Springer, 1943.

[156] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, I. Podluzhny, et al. The

STC system for the CHiME-6 challenge. In *CHiME 2020 Workshop on Speech Processing in Everyday Environments*, 2020.

[157] H. Miao, G. Cheng, P. Zhang, T. Li, and Y. Yan. Online hybrid CTC/attention architecture for end-to-end speech recognition. In *INTERSPEECH*, 2019.

[158] Y. Miao, M. Gowayyed, and F. Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015.

[159] D. Michelsanti and Z.-H. Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. *arXiv preprint arXiv:1709.01703*, 2017.

[160] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[161] S. Mirsamadi and J. H. Hansen. Multi-domain adversarial training of neural network acoustic models for distant speech recognition. *Speech Communication*. Elsevier, 2019.

[162] S. Mittermaier, L. Kürzinger, B. Waschneck, and G. Rigoll. Small-footprint keyword spotting on raw audio data with sinc-convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[163] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2018.

[164] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[165] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*, 2014.

[166] A.-r. Mohamed, G. Dahl, and G. Hinton. Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications*. Vancouver, Canada, 2009.

[167] A.-r. Mohamed, G. E. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*. IEEE, 2011.

[168] A.-r. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny. Deep belief networks using discriminative features for phone recognition. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011.

[169] R. K. Moore and L. Skidmore. On the use/misuse of the term 'Phoneme'. *arXiv preprint arXiv:1907.11640*, 2019.

[170] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. *Advances in neural information processing systems*, 1989.

[171] N. Morgan and H. Bourlard. Continuous speech recognition. *IEEE signal processing magazine*. IEEE, 1995.

[172] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
[173] A. Narayanan and D. Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.
[174] A. Narayanan and D. Wang. Joint noise adaptive training for robust automatic speech recognition. In *Proc. ICASSP*. IEEE, 2014.
[175] S. Nayak, S. Bhati, and K. S. R. Murty. An investigation into instantaneous frequency estimation methods for improved speech recognition features. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017.
[176] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar. Universal adversarial perturbations for speech recognition systems. *arXiv preprint arXiv:1905.03828*, 2019.
[177] H. Ney, U. Essen, and R. Kneser. On the estimation of 'small' probabilities by leaving-one-out. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 1995.
[178] H. Ney and S. Ortmanns. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*. IEEE, 1999.
[179] S. Nie, S. Liang, W. Liu, X. Zhang, and J. Tao. Deep learning based speech separation via NMF-style reconstructions. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2018.
[180] S. Nie, S. Liang, W. Xue, X. Zhang, W. Liu, et al. Two-stage multi-target joint learning for monaural speech separation. In *Proc. INTERSPEECH*, 2015.
[181] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey. Multichannel end-to-end speech recognition. *arXiv preprint arXiv:1703.04783*, 2017.
[182] J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young. A one pass decoder design for large vocabulary recognition. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.
[183] D. Oglic, Z. Cvetkovic, P. Bell, and S. Renals. A deep 2D convolutional network for waveform-based speech recognition. *Proc. Interspeech 2020*, 2020.
[184] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
[185] D. O'Saughnessy. Speech communication-human and machine. *Reading, PA: Addison-Wesley*, 1987.
[186] D. Palaz. Towards end-to-end speech recognition. Technical report, EPFL, 2016.
[187] K. Paliwal, K. Wójcicki, and B. Shannon. The importance of phase in speech enhancement. *speech communication*. Elsevier, 2011.
[188] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.
[189] A. Pandey and D. Wang. Dense CNN with self-attention for time-domain speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2021.

[190] T. Parcollet, M. Morchid, and G. Linares. E2E-SINCNET: Toward fully end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[191] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[192] S. R. Park and J. Lee. A fully convolutional neural network for speech enhancement. *arXiv preprint arXiv:1609.07132*, 2016.

[193] S. Pascual, A. Bonafonte, and J. Serra. SEGAN: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[194] S. Pascual, A. Bonafonte, J. Serrà, and J. A. Gonzalez. Whispered-to-voiced alaryngeal speech conversion with generative adversarial networks. *arXiv preprint arXiv:1808.10687*, 2018.

[195] S. Pascual, M. Ravanelli, J. Serra, A. Bonafonte, and Y. Bengio. Learning problem-agnostic speech representations from multiple self-supervised tasks. *arXiv preprint arXiv:1904.03416*, 2019.

[196] S. Pascual, J. Serrà, and A. Bonafonte. Towards generalized speech enhancement with generative adversarial networks. *arXiv preprint arXiv:1904.03418*, 2019.

[197] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[198] V. Peddinti, D. Povey, and S. Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[199] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[200] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*, 2019.

[201] H. Phan, I. V. McLoughlin, L. Pham, O. Y. Chén, P. Koch, M. De Vos, and A. Mertins. Improving GANs for speech enhancement. *IEEE Signal Processing Letters*. IEEE, 2020.

[202] H. Phan, H. L. Nguyen, O. Y. Chén, P. Koch, N. Q. Duong, I. McLoughlin, and A. Mertins. Self-Attention Generative Adversarial Network for Speech Enhancement. *arXiv preprint arXiv:2010.09132*, 2020.

[203] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, 2018.

[204] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The Kaldi speech recognition toolkit.

In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[205] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur. A time-restricted self-attention layer for ASR. In *Proc. ICASSP*. IEEE, 2018.

[206] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*, 2017.

[207] Y. Qian, T. Tan, and D. Yu. Neural network based multi-factor aware joint training for robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016.

[208] S. Qin and T. Jiang. Improved Wasserstein conditional generative adversarial network speech enhancement. *EURASIP Journal on Wireless Communications and Networking*. Springer, 2018.

[209] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*. PMLR, 2019.

[210] Z. qiu Wang and D. Wang. Joint training of speech separation, filterbank and acoustic model for robust automatic speech recognition. In *Proc. INTERSPEECH*. IEEE, 2015.

[211] S. R. Quackenbush. *Objective measures of speech quality*. PhD thesis, Georgia Institute of Technology, 1995.

[212] L. Rabiner. Fundamentals of speech recognition. *Fundamentals of speech recognition*. PTR Prentice Hall, 1993.

[213] L. Rabiner and S. Levinson. A speaker-independent, syntax-directed, connected word recognition system based on hidden Markov models and level building. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. IEEE, 1985.

[214] L. Rabiner and R. Schafer. *Theory and applications of digital speech processing*. Prentice Hall Press, 2010.

[215] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*. Ieee, 1989.

[216] L. R. Rabiner, R. W. Schafer, et al. *Digital processing of speech signals*. Prentice-hall, 1978.

[217] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[218] A. Raganato, J. Tiedemann, et al. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. The Association for Computational Linguistics, 2018.

[219] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[220] M. Ravanelli and Y. Bengio. Interpretable convolutional filters with SincNet. *arXiv preprint arXiv:1811.09725*, 2018.

[221] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018.

[222] M. Ravanelli and Y. Bengio. Speech and speaker recognition from raw waveform with SincNet. *arXiv preprint arXiv:1812.05920*, 2018.

[223] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. Batch-normalized joint training for DNN-based distant speech recognition. In *Proc. SLT*. IEEE, 2016.

[224] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio. Multi-task self-supervised learning for robust speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[225] I. Rec. P. 862.2: Wideband extension to recommendation P. 862 for the assessment of wideband telephone networks and speech codecs. *International Telecommunication Union, CH–Geneva*, 2005.

[226] C. K. A. Reddy, N. Shankar, G. S. Bhat, R. Charan, and I. Panahi. An individualized super-Gaussian single microphone speech enhancement for hearing aid users with smartphone as an assistive device. *IEEE signal processing letters*. IEEE, 2017.

[227] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in HMM speech recognition. *IEEE transactions on speech and audio processing*. IEEE, 1994.

[228] D. Rethage, J. Pons, and X. Serra. A wavenet for speech denoising. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[229] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural computation*. MIT Press, 1991.

[230] G. Rigoll, C. Neukirchen, and J. Rottland. A new hybrid system based on MMI-neural networks for the RM speech recognition task. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. IEEE, 1996.

[231] A. J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*. IEEE, 1994.

[232] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*. American Psychological Association, 1958.

[233] A. Rousseau, P. Deléglise, Y. Esteve, et al. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *LREC*, 2014.

[234] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[235] S. Sahu, R. Gupta, and C. Espy-Wilson. On enhancing speech emotion recognition using generative adversarial networks. *arXiv preprint arXiv:1806.06626*, 2018.

[236] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran. Learning filter banks within a deep neural network framework. In *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013.

[237] T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012.

[238] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for LVCSR. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013.

[239] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR arXiv preprint arXiv:1402.1128*, 2014.

[240] H. Sak, M. Shannon, K. Rao, and F. Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *Interspeech*, 2017.

[241] J. Salazar, K. Kirchhoff, and Z. Huang. Self-attention networks for connectionist temporal classification in speech recognition. In *Proc. ICASSP*. IEEE, 2019.

[242] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, 2016.

[243] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny. The IBM 2015 English conversational telephone speech recognition system. *arXiv preprint arXiv:1505.05899*, 2015.

[244] G. Saon, T. Sercu, S. Rennie, and H.-K. J. Kuo. The IBM 2016 English conversational telephone speech recognition system. In *Interspeech 2016*, 2016.

[245] P. Scalart and J. Filho. Speech enhancement based on a priori signal to noise estimation. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, 1996.

[246] M.-P. Schambach. Model length adaptation of an HMM based cursive word recognition system. In *ICDAR*, 2003.

[247] M. Schroeder and B. Atal. Code-excited linear prediction (CELP): High-quality speech at very low bit rates. In *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1985.

[248] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*. Ieee, 1997.

[249] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1985.

[250] F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

[251] M. L. Seltzer. Bridging the gap: Towards a unified framework for hands-free speech recognition using microphone arrays. In *Hands-Free Speech Communication and Microphone Arrays*. IEEE, 2008.

[252] M. L. Seltzer, A. Acero, and K. Kalgaonkar. Acoustic model adaptation via linear spline interpolation for robust speech recognition. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010.

[253] M. L. Seltzer, D. Yu, and Y. Wang. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[254] P. Shen, X. Lu, S. Li, and H. Kawai. Conditional generative adversarial nets classifier for spoken language identification. In *Proc. INTERSPEECH*. IEEE, 2017.
[255] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*, 2018.
[256] M. H. Soni, N. Shah, and H. A. Patil. Time-frequency masking-based speech enhancement using generative adversarial network. In *Proc. ICASSP*. IEEE, 2018.
[257] F. K. Soong and E.-F. Huang. A Tree. Trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.
[258] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel. Self-attentional acoustic models. *arXiv preprint arXiv:1803.09519*, 2018.
[259] A. Sriram, H. Jun, Y. Gaur, and S. Satheesh. Robust speech recognition using generative adversarial networks. In *Proc. ICASSP*. IEEE, 2018.
[260] N. Srivastava. Improving neural networks with dropout. *University of Toronto*, 2013.
[261] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*. JMLR. org, 2014.
[262] L. Sun, J. Du, L.-R. Dai, and C.-H. Lee. Multiple-target deep learning for LSTM-RNN based speech enhancement. In *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*. IEEE, 2017.
[263] S. Sun, P. Guo, L. Xie, and M.-Y. Hwang. Adversarial regularization for attention based end-to-end robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2019.
[264] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
[265] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*. IEEE, 2011.
[266] H. Taherian, Z.-Q. Wang, J. Chang, and D. Wang. Robust speaker recognition based on single-channel and multi-channel speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2020.
[267] T. Tan, Y. Qian, H. Hu, Y. Zhou, W. Ding, and K. Yu. Adaptive very deep convolutional residual network for noise robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2018.
[268] J. Thiemann, N. Ito, and E. Vincent. The Diverse Environments Multi-channel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings. In *Proceedings of Meetings on Acoustics ICA2013*. Acoustical Society of America, 2013.
[269] Z. Tian, J. Yi, J. Tao, Y. Bai, and Z. Wen. Self-attention transducers for end-to-end speech recognition. *arXiv preprint arXiv:1909.13037*, 2019.

[270] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

[271] Y.-H. Tu, J. Du, L. Sun, F. Ma, J. Pan, and C.-H. Lee. A space-and-speaker-aware iterative mask estimation approach to multi-channel speech recognition in the CHiME-6 challenge. *Proc. Interspeech 2020*, 2020.

[272] C. Valentini-Botinhao et al. Noisy speech database for training speech enhancement algorithms and TTS models. University of Edinburgh. School of Informatics. Centre for Speech Technology, 2017.

[273] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi. Investigating RNN-based speech enhancement methods for noise-robust text-to-speech. In *SSW*, 2016.

[274] B. D. Van Veen and K. M. Buckley. Beamforming: A versatile approach to spatial filtering. *IEEE assp magazine*. IEEE, 1988.

[275] J. Vaněk, J. Zelinka, D. Soutner, and J. Psutka. A regularization post layer: An additional way how to make deep neural networks robust. In *International Conference on Statistical Language and Speech Processing*. Springer, 2017.

[276] A. Varga, H. Steeneken, and D. Jones. The NOISEX-92 study on the effect of additive noise on automatic speech recognition system. *Reports of NATO Research Study Group (RSG. 10)*, 1992.

[277] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens. Scaling local self-attention for parameter efficient visual backbones. In *CVPR*, 2021.

[278] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[279] C. Veaux, J. Yamagishi, and S. King. The voice bank corpus: Design, collection and data analysis of a large regional accent speech database. In *2013 international conference oriental COCOSDA held jointly with 2013 conference on Asian spoken language research and evaluation (O-COCOSDA/CASLRE)*. IEEE, 2013.

[280] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. In *Interspeech*, 2013.

[281] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni. The second "CHiME" speech separation and recognition challenge: Datasets, tasks and baselines. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.

[282] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer. An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language*. Elsevier, 2017.

[283] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 2008.

[284] T. Vintsyuk. Element-wise recognition of continuous speech composed of words from a specified dictionary. *Cybernetics*. Springer, 1971.

[285] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*. IEEE, 1967.

[286] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*. IEEE, 1989.

[287] D. Wang and J. H. Hansen. Speech enhancement for cochlear implant recipients. *The Journal of the Acoustical Society of America*. Acoustical Society of America, 2018.

[288] D. Wang and J. Lim. The unimportance of phase in speech enhancement. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1982.

[289] D. Wang, X. Wang, and S. Lv. An overview of end-to-end automatic speech recognition. *Symmetry*. Multidisciplinary Digital Publishing Institute, 2019.

[290] K. Wang, J. Zhang, S. Sun, Y. Wang, F. Xiang, and L. Xie. Investigating generative adversarial networks based speech dereverberation for robust speech recognition. In *Proc. INTERSPEECH*. IEEE, 2018.

[291] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

[292] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, and M. L. Seltzer. Transformer-based acoustic modeling for hybrid speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[293] Y. Wang, A. Narayanan, and D. Wang. On training targets for supervised speech separation. *IEEE/ACM transactions on audio, speech, and language processing*. IEEE, 2014.

[294] Y. Wang, L. Zhang, B. Zhang, and Z. Li. End-to-end Mandarin recognition based on convolution input. In *MATEC Web of Conferences*. EDP Sciences, 2018.

[295] Z.-Q. Wang and D. Wang. A joint training framework for robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2016.

[296] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai. ESPnet: End-to-end speech processing toolkit. In *Proc. INTERSPEECH*. IEEE, 2018.

[297] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj, et al. CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. *arXiv preprint arXiv:2004.09249*, 2020.

[298] T. Watzel, L. Li, L. Kürzinger, and G. Rigoll. Deep neural network quantizers outperforming continuous speech recognition systems. In *International Conference on Speech and Computer*. Springer, 2019.

[299] C. Weng, D. Yu, S. Watanabe, and B.-H. F. Juang. Recurrent deep neural networks for robust speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014.

[300] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller. Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR. In *International conference on latent variable analysis and signal separation*. Springer, 2015.

[301] F. Weninger, F. Eyben, and B. Schuller. Single-channel speech separation with memory-enhanced recurrent neural networks. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014.

[302] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller. Discriminatively trained recurrent neural networks for single-channel speech separation. In *Proc. GlobalSIP*. IEEE, 2014.

[303] P. Woodland, M. Gales, D. Pye, and S. Young. The development of the 1996 HTK broadcast news transcription system. In *DARPA speech recognition workshop*, 1997.

[304] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[305] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.

[306] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. PMLR, 2015.

[307] M. Xu, D. F. Wong, B. Yang, Y. Zhang, and L. S. Chao. Leveraging local and global patterns for self-attention networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[308] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2014.

[309] B. Yang, Z. Tu, D. F. Wong, F. Meng, L. S. Chao, and T. Zhang. Modeling localness for self-attention networks. *arXiv preprint arXiv:1810.10182*, 2018.

[310] B. Yang, L. Wang, D. F. Wong, L. S. Chao, and Z. Tu. Convolutional Self-Attention Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[311] Y.-H. Yang and H. H. Chen. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*. ACM New York, NY, USA, 2012.

[312] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. L. Seltzer. Transformer-transducer: End-to-end speech recognition with self-attention. *arXiv preprint arXiv:1910.12977*, 2019.

[313] S.-K. A. Yeung and M.-H. Siu. Improved performance of Aurora 4 using HTK and unsupervised MLLR adaptation. In *Eighth International Conference on Spoken Language Processing*, 2004.

[314] D. Yin, C. Luo, Z. Xiong, and W. Zeng. Phasen: A phase-and-harmonics-aware speech enhancement network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[315] T. Yoshioka, A. Sehr, M. Delcroix, K. Kinoshita, R. Maas, T. Nakatani, and W. Kellermann. Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition. *IEEE Signal Processing Magazine*. IEEE, 2012.

[316] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, et al. The HTK Book (for HTK Version. 3.3), Cambridge University Engineering Department, 2005. 2006.

[317] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy modelling. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[318] A. W. Yu, D. Dohan, Q. Le, T. Luong, R. Zhao, and K. Chen. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*, 2018.

[319] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. QANet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[320] D. Yu and M. L. Seltzer. Improved bottleneck features using pretrained deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

[321] K. Yu. *Adaptive training for large vocabulary continuous speech recognition*. PhD thesis, University of Cambridge, 2006.

[322] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney. A comparison of transformer and LSTM encoder decoder models for ASR. In *Proc. ASRU*. IEEE, 2019.

[323] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International conference on machine learning*. PMLR, 2019.

[324] H. Zhang, C. Liu, N. Inoue, and K. Shinoda. Multi-task autoencoder for noise-robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[325] S. Zhang, C.-T. Do, R. Doddipatla, and S. Renals. Learning noise invariant features through transfer learning for robust end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[326] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014.

[327] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

[328] Z. Zhang, C. Deng, Y. Shen, D. S. Williamson, Y. Sha, Y. Zhang, H. Song, and X. Li. On loss functions and recurrency training for GAN-based speech enhancement systems. In *Proc. Interspeech 2020*, 2020.

[329] Z. Zhao, H. Liu, and T. Fingscheidt. Convolutional neural networks to enhance coded speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 2018.

[330] W. Zhou, W. Michel, K. Irie, M. Kitza, R. Schlüter, and H. Ney. The RWTH ASR system for TED-LIUM release 2: Improving hybrid HMM with specaugment. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[331] M. Zimmermann and H. Bunke. Hidden Markov model length optimization for handwriting recognition systems. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*. IEEE, 2002.

Publications

[1†] L. Kürzinger, E. R. Chavez Rosas, L. Li, T. Watzel, and G. Rigoll. Audio adversarial examples for robust hybrid CTC/attention speech recognition. In A. Karpov and R. Potapova, editors, *Speech and Computer*, pages 255–266, Cham, 2020. Springer International Publishing.

[2†] L. Kürzinger, T. Watzel, L. Li, R. Baumgartner, and G. Rigoll. Exploring hybrid CTC/attention end-to-end speech recognition with Gaussian processes. In A. A. Salah, A. Karpov, and R. Potapova, editors, *Speech and Computer*, pages 258–269, Cham, 2019. Springer International Publishing.

[3†] L. Kürzinger, D. Winkelbauer, L. Li, T. Watzel, and G. Rigoll. CTC-segmentation of large corpora for German end-to-end speech recognition. In *International Conference on Speech and Computer*, pages 267–278. Springer, 2020.

[4†] L. Li, Y. Kang, Y. Shi, L. Kürzinger, T. Watzel, and G. Rigoll. Adversarial joint training with self-attention mechanism for robust end-to-end speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 07 2021.

[5†] L. Li, L. Kürzinger, T. Watzel, and G. Rigoll. A global discriminant joint training framework for robust speech recognition. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 85–92, 2021.

[6†] L. Li, Z. Lu, T. Watzel, L. Kürzinger, and G. Rigoll. Light-weight self-attention augmented generative adversarial networks for speech enhancement. *Electronics*, 2021.

[7†] L. Li, T. Watzel, L. Kürzinger, and G. Rigoll. Towards constructing HMM structure for speech recognition with deep neural fenonic baseform growing. *IEEE Access*, 2021.

[8†] L. Li, Wudamu, L. Kürzinger, T. Watzel, and G. Rigoll. Lightweight end-to-end speech enhancement generative adversarial network using Sinc convolutions. *Applied Sciences*, 2021.

[9†] L. Li, X. Zhou, Z. Song, T. Watzel, L. Kürzinger, and G. Rigoll. Deep neural fenonic baseform growing: A novel approach to construct HMM topologies for speech recognition. In *2020 International Conference on High Performance Computing Simulation (HPCS)*, pages 27–32, 2021.

[10†] T. Watzel, L. Kürzinger, L. Li, and G. Rigoll. Synchronized forward-backward Transformer for end-to-end speech recognition. In A. Karpov and R. Potapova,

editors, *Speech and Computer*, pages 646–656, Cham, 2020. Springer International Publishing.

[11†] T. Watzel, L. Kürzinger, L. Li, and G. Rigoll. Induced local attention for Transformer models in speech recognition. In A. Karpov and R. Potapova, editors, *Speech and Computer*, pages 795–806, Cham, 2021. Springer International Publishing.

[12†] T. Watzel, L. Kürzinger, L. Li, and G. Rigoll. Regularized forward-backward decoder for attention models. In A. Karpov and R. Potapova, editors, *Speech and Computer*, pages 786–794, Cham, 2021. Springer International Publishing.

[13†] T. Watzel, L. Li, L. Kürzinger, and G. Rigoll. Deep neural network quantizers outperforming continuous speech recognition systems. In A. A. Salah, A. Karpov, and R. Potapova, editors, *Speech and Computer*, pages 530–539, Cham, 2019. Springer International Publishing.