

VOICE CONVERSION AND TEXT-TO-SPEECH FOR PRIVACY PROTECTION APPLICATIONS

by
FRANCESCO NESPOLI

A Thesis submitted in fulfilment of requirements for the degree of
Doctor of Philosophy
of
Imperial College London

Speech and Audio Processing Group
Department of Electrical and Electronic Engineering
Imperial College London
2025

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Statement of Originality

As far as the author is aware, the following aspects of the thesis are believed to be original contributions:

1. Two-Stage Voice anonymization. A fully neural pipeline was developed, combining two state-of-the-art voice conversion systems. One of the stages in this process consists of a zero-shot voice conversion system, marking the first application of such an architecture in the privacy and anonymization domain.
2. Accent Conversion in the context of data augmentation and privacy applications. The functionality of the zero-shot voice conversion model was extended to simulate various accents, offering significant benefits for data augmentation in ASR. By integrating a more diverse range of accented data, this approach aims to improve the fairness of ASR systems and privacy preservation measures across different accents, ensuring more equitable performance regardless of the user's country of origin. Additionally, the use of accent embeddings was explored to enhance speech privacy preservation.
3. Low-Complexity Features for Privacy Preservation. A suite of low-complexity features was explored as a method for privacy preservation on devices with limited computational resources. The method, proposed in the context of conversation analysis, ensures that privacy can be maintained without the need for heavy processing, making it suitable for use on constrained devices. Departing from the conventional approach of using the Word Error Rate (WER) from an ASR model to evaluate speech utility, this research evaluated the utility of feature extraction methods through a voice activity detection and speaker diarization system. It was demonstrated that combining the McAdams coefficient with spectral smoothing not only preserves utility

but also enhances privacy.

4. Inter-Device Distance Estimation. The enhanced proportionate normalized least mean square adaptive filter was utilized to estimate the relative room impulse responses from audio recordings across two smart devices. This methodology allowed for precise inter-device distance estimation through a novel set of features, developed by extending the definition of certain acoustic attributes of the room impulse response to their relative counterparts. Moreover, this application underscores the potential of the distance estimation technique to contribute to more secure and privacy-conscious smart home ecosystems: distance estimation between devices can help mitigate accidental activation of smart speakers thereby enhancing user privacy by reducing unintended interactions.

The author affirms that this work is the result of his own independent research. Any ideas, data, or content derived from other sources have been acknowledged and appropriately referenced.

“The question isn’t ‘What do we want to know about people?’ It’s ‘What do people want to tell about themselves?’.”

Mark Zuckerberg

Abstract

In recent years, the need for privacy preservation when manipulating or storing personal data has become a major issue. Particularly, speech data might contain significant Personal Identifiable Information (PII) such as full name, social security number or geographical positioning. This happens not only in the semantic but also in the acoustic domain: voice characteristics such as prosody, speaking rate, accent and intonation inherently contain a variety of PII such as personality, physical characteristics, emotional state, age and gender that can be identified and potentially used for malicious privacy attacks. This thesis explores the possibility of applying voice conversion, text-to-speech and signal processing-based approaches to remove the biometric identity from speech signals therefore protecting speaker's privacy against identification attacks. While protecting the speaker's identity against fraudulent access, the methods presented in this thesis aim to preserve the usefulness of speech on a variety of downstream tasks. The tools developed in this thesis explore new voice anonymization techniques with the aim to help, on one side, companies to comply with the most stringent privacy regulations, and, on the other, provide individuals the possibility to enforce their right to privacy.

Acknowledgment

I would like to begin by expressing my sincere gratitude to my supervisors, Professor Patrick A. Naylor, Daniel Dalmendro Barreda, and Professor Joerg Bitzer. I am deeply thankful for their time, patience, and belief in my work, which helped me grow as a researcher and navigate the many challenges of this project.

A special thank you goes to Jule Polhausen from the Jade University of Applied Sciences, Oldenburg, Germany, for her invaluable collaboration on Chapter 4. Working with her has been a rewarding experience, and her contributions greatly enriched this part of the thesis.

I am also deeply thankful to my family, friends, and colleagues for their unwavering support and encouragement, not only during the process of writing this thesis but throughout my entire academic journey. Their belief in me kept me motivated and grounded.

Finally, my deepest thanks go to Ieva, my life companion. Her constant support, encouragement, and patience throughout this journey have been invaluable. I am profoundly grateful for her understanding and belief in me during this challenging process.

Thank you all for being part of this journey with me.

Context of Research

The research presented in this thesis was conducted within the framework of the Marie Skłodowska-Curie Actions European Training Network (MSCA-ETN) Service-Oriented, Ubiquitous, Network-Driven Sound (SOUNDS) project (grant agreement No. 956369). This program provided me with three years of dedicated funding and the opportunity to work across multiple institutions, fostering both academic and industrial collaboration.

My primary research was hosted by Nuance Communications UK, which was later acquired by Microsoft UK. The majority of the research work that underpins this thesis, specifically Chapters 2 and 3, was carried out during this period. These chapters focus on the foundational aspects of the study, which were shaped by the advanced technological environment and the resources provided by Nuance/Microsoft.

In addition, the PhD program also comprehended two secondments, each lasting six months. The first secondment was hosted by the Fraunhofer Institute in Oldenburg, Germany. My time there contributed to the development of Chapter 4, which was particularly influenced by the applied research and industry-oriented projects typical of Fraunhofer. The second was at Imperial College London, UK, where I developed the research presented in Chapter 5.

Contents

Copyright Declaration	2
Statement of Originality	3
Abstract	6
Acknowledgment	7
Context of Research	8
Contents	9
List of Figures	13
List of Tables	20
List of Notation	24
Abbreviations	25
Introduction	28
Motivation	28
Objectives	29
Summary of contributions and thesis structure	29
Chapter 1. Speech and Neural Networks	31
1.1 Introduction to Speech Processing	31
1.2 Human Speech Production	31
1.3 Speech Perception	34
1.4 Time-Frequency Processing	38
1.5 Artificial Neural Networks	41
1.5.1 Biological Neurons	41
1.5.2 Perceptron	42
1.5.3 Training Procedure: Gradient Descent	44

1.6 Fundamental Deep-Learning Architectures	47
1.6.1 Convolutional Neural Networks	47
1.6.2 Recurrent Neural Networks	50
1.6.3 Transformers	55
1.7 Automatic Speech Recognition	61
1.8 Automatic Speaker Verification	68
1.8.1 Speaker Embedding Extractors	70
1.8.2 Scoring Functions: Cosine Similarity and Probabilistic Linear Dis-	
criminant Analysis	73
1.9 Basic Concepts and Architectures of Text-To-Speech Technology	77
1.9.1 Introduction to Text-To-Speech	77
1.9.2 Neural Speech Synthesis	80
1.10 Basic Concepts and Architectures of Voice Conversion Technology	89
1.10.1 Introduction to Voice Conversion	89
1.10.2 Voice Conversion Architectures	91
1.10.3 Disentanglement in Voice Conversion	93
1.11 An Introduction to Privacy in Speech Technology	94
Chapter 2. Two-Stage Voice Anonymization	97
2.1 The anonymization task	98
2.2 The Voice Privacy Challenge	99
2.3 Scoring Pipeline	102
2.3.1 The attacker model: privacy evaluation	102
2.3.2 The downstream task: utility evaluation	104
2.3.3 Privacy-To-Utility Trade Off Metric	105
2.4 Advancements in Voice Privacy and Anonymization Techniques	107
2.5 Two-Stage Voice Anonymization	109
2.5.1 X-vector Speaker Anonymization	110
2.5.2 Zero-Shot Voice Conversion	111
2.6 Experiments	112
2.6.1 Primary Anonymization Metric	113
2.6.2 Primary Utility Metric	113
2.6.3 Secondary Utility Metric	114
2.6.4 Privacy Utility Trade-Off	116
2.7 Discussion and Conclusions	117
Chapter 3. Accented Speech Simulation: From Data Augmentation To	
Privacy	120

3.1	Accented Speech Recognition	121
3.2	TTS Accented Data Augmentation for ASR Applications	123
3.3	Accent Simulation with Zero Shot TTS	124
3.4	Experiments	125
3.4.1	Single Accent	126
3.4.2	Multiple accents	128
3.4.3	Combine TTS with other audio augmentations	131
3.5	Accent Simulation and Privacy	132
3.6	Privacy Preservation Through Accent Modification	132
3.7	Experiments	133
3.7.1	Language Interpolation	134
3.7.2	Accent Embeddings	138
3.8	Discussion and Conclusions	143
Chapter 4.	Conversation Analysis: Exploring Utility and Privacy	145
4.1	Conversational Data Anonymization	146
4.2	Privacy and Utility Evaluation	147
4.2.1	Voice Activity Detection	148
4.2.2	Speaker Diarization	148
4.3	Experiments	149
4.4	Conclusions	153
Chapter 5.	Relative Acoustic Features for Indoor Applications	155
5.1	Inter-Device Distance Estimation	155
5.2	Relative Acoustic Features Extraction	160
5.3	Improved Proportionate Normalized Least Mean Square Algorithm	161
5.4	Distance Estimation and Explainability	163
5.5	Experiments	168
5.5.1	Acoustic Channel Estimation	169
5.5.2	Distance Estimation for White Noise Inputs	170
5.5.3	Distance Estimation with Speech Signals	173
5.6	Conclusions	174
Chapter 6.	Discussion and Conclusions	175
6.1	Future Work	176
Appendix A.	Feature Correlation and Optimization Convergence	200
A.1	Benefits of Decorrelated Features in Optimization	200
A.2	Instability in ill-conditioned problems	201

Appendix B. Probabilistic Linear Discriminant Analysis	203
B.1 Mathematical Formulation of Complete Data Likelihood	203
B.1.1 Learn model parameters	204
B.1.2 Example	204

List of Figures

1.1	The Source-Filter Model of speech production, showing the anatomy of the human vocal apparatus on the left, and its corresponding schematic representation on the right. The diagram includes the nasal cavity, oral cavity, and pharynx, which act as resonating filters, and the larynx, which serves as the sound source. This model encapsulates how the airflow from the lungs is modulated into sound by the glottis and further shaped by the vocal tract into speech with distinct resonances or formants. Adapted from [9].	32
1.2	Spectra of a flute and a clarinet playing the same note. Taken from [12].	35
1.3	Illustration of the human ear anatomy, showing the three main sections: outer ear, middle ear, and inner ear. The pinna directs sound waves into the ear canal, impacting the eardrum, visible here as the tympanic membrane. These vibrations are then transmitted through the ossicles (malleus, incus, and stapes) to the cochlea in the inner ear, which is linked to the auditory nerve. Taken from [13].	36

1.4	Cochlear structure, showcasing the cochlea, vestibular organ, and organ of Corti within the inner ear. The image highlights the scala vestibuli, scala media, and scala tympani, which are essential for the cochlea's role as a spectral analyzer. The basilar membrane, depicted at the bottom, varies in stiffness and is crucial for frequency analysis of incoming sound waves. Inner hair cells (IHC) on the basilar membrane transduce mechanical movements into neural signals by bending their stereocilia, initiating a chemical cascade that results in auditory perception. This complex process allows for the encoding of sound waves into electrical signals that travel via the auditory nerve. Taken from [14].	37
1.5	Representation of speech signal processing from waveform to Mel Frequency Cepstral Coefficients. At the top, the time-domain representation. The spectrogram details the spectrum of frequencies present in the signal over time. Further down, the Mel Spectrogram transforms this information into a format that aligns more closely with human auditory perception. Bottom, the Mel-frequency Cepstral Coefficients abstract the audio into a set of features that capture the essential spectral properties.	39
1.6	Transformation from the STFT power spectrum to the Mel-scale power spectrum using triangular Mel-scale filter banks. This transformation adjusts the frequency bins of the original power spectrum (top) to align with human auditory perception, which is less sensitive to higher frequencies. The triangular filters cover wider ranges as frequency increases to mimic this perceptual characteristic. Each filter's output contributes to a corresponding bin in the Mel-scale power spectrum (bottom), effectively summarizing the spectral content in a way that reflects human hearing. This processed spectrum forms the basis for extracting MFCCs, which are crucial for various audio processing tasks.	40

1.7	Threshold Logic Unit. The process begins with inputs (x_1 to x_4), each multiplied by corresponding weights (ω_1 to ω_4). These weighted inputs are summed together with a bias (b_m) to compute the input to the activation function (z), which applies a threshold to produce the binary output ($\hat{y}$).	42
1.8	Example of a Multi-Layer Perceptron with a single hidden layer with M neurons h_i^1 and output layer with one unit o_1^1 .	43
1.9	Gradient Descent: 1D example of the MSE loss function. The red arrow indicates the $\nabla_{\mathbf{w}}\text{MSE}$ computed in the initial random point.	45
1.10	Gradient Descent: 1D example of a non-convex loss function. The red arrow indicates the $\nabla_{\mathbf{w}}\text{MSE}$. A and B indicate two different random initializations of the parameter vector $\mathbf{w}$.	46
1.11	Measured receptive fields of mice visual cortex neurons and CNN features. Panel A, from left to right: low, middle and high level features extracted from a CNN architecture. Adapted from [27]. Panel B: Measured receptive fields of visual cortex neurons in mice. Adapted from [28], Copyright 2009 Society for Neuroscience.	48
1.12	CNN processing of an input image: f_h, f_w are the filter height and width and s_h, s_w are the vertical and horizontal strides.	49
1.13	RNN architecture. Right: single RNN cell and its enrolling through time with $y_t \in R$. Left: RNN layer and its enrolling through time. Note that in this case the output at each time step t is a vector.	50
1.14	Schematic of an LSTM cell. From left to right: $\mathbf{c}_t$ is the long-term cell state, $\mathbf{h}_t$ is the short-term cell state, σ is the sigmoid activation function, $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{f}_t$ and $\mathbf{g}_t$, are the outputs of the <i>gate controllers</i> and the <i>analyser</i> , $\mathbf{x}_t$ is the current input and $\mathbf{y}_t$ its associated output.	52

1.15	Encoder-Decoder architecture based on RNN networks. The original English word sequence is first encoded into fixed-size vectors $\mathbf{x}_t$ by the embedding layer. Then, the RNN-encoder consumes the encoded sequence and generates a compressed representation $\mathbf{h}_2$ that is passed to the RNN-decoder. This network processes the hidden states $\mathbf{h}'_t$ in combination with a forward-shifted version of the target sequence and predicts, one word at the time, the translated Italian sentence.	53
1.16	Encoder-Decoder architecture with concatenative attention [35] mechanism. Before feeding the hidden encoder state $\mathbf{h}_t$ to the decoder an alignment network (green box) computes alignment scores $a_{i,t}$ between each encoder output $\mathbf{y}_t$ and the decoder previous hidden state $\mathbf{h}'_t$. This new vector $\mathbf{ws}_t$ contains context information useful for the translation task.	54
1.17	Original transformer architecture with its principal components: encoder, decoder, positional encoding and multi-head attention. Adapted from [36]	56
1.18	Scaled dot-product self-attention example: calculation of the context vector $\mathbf{z}_t$ at time step $t = 1$.	58
1.19	Multi-head attention example with $h = 3$ attention heads. Example for computing the context vector $\mathbf{z}_t$ at time step $t = 1$.	58
1.20	Basic ASR Pipeline. The input audio x_t is first preprocessed to enhance signal quality. The resulting features are then passed into the acoustic model. In the decoding step, the acoustic model, lexicon, and an optional Language Model (LM) are combined to predict the output label y_t .	62
1.21	ASR model combining CTC, attention and language modeling blocks. The orange arrows indicate that the previous label y_{n-1} is recursively fed into both the attention-based decoder and the LM.	66

1.22 Basic speaker verification pipeline with a <i>closed-set</i> example. During the enrollment phase, each utterance is processed through a feature extraction module. Subsequently, a speaker extractor generates a unique speaker embedding, which serves as a voice fingerprint for each speaker. In the verification phase, when a new utterance is introduced, its embedding is extracted, and a similarity score is calculated. If the score exceeds a predefined threshold, the system attributes the new utterance to the corresponding enrolled speaker.	69
1.23 Basic components of SPSS and neural TTS systems. The text analysis module maps text into linguistic features (such as phonemes) containing pronunciation and prosody information. The acoustic model transforms the linguistic features into acoustic features such as Mel spectrograms. Finally, the vocoder synthesizes a waveform given the acoustic features.	78
1.24 Schematic overview of the Tacotron model [126].	81
1.25 Basic components of a VC system. The content encoder extracts semantic features from the source utterance, while the speaker encoder generates a speaker representation from the target utterance. The decoder then synthesizes an audio signal by combining the outputs of the content and speaker encoders, producing a waveform that replicates the target speaker's voice as if they were speaking the words from the source audio.	90
2.1 Semi-Informed attacker model. Panel 1: the attacker has access to the anonymization system but without the specific parameters used by the user for the anonymization of each enrollment utterance. Panel 2: the attacker uses the ASV model fine-tuned on an anonymized version of LibriSpeech-train-clean-360 ASV_{eval}^{anon} to compute log-likelihood ratios between enrollment and trial anonymized utterances.	103
2.2 Semi-Informed x-vector attacker model structure.	104
2.3 Overview of voice anonymization components: Zero Shot Voice Conversion (ZS-VC) and baseline system (B1a). The red dotted lines (1) and (2), are mutually exclusive and refer, in Section 2.6, to system VB and BV respectively.	109

2.4	Privacy-to-utility trade-off for development (Dev) and test (Test) sets evaluated for $\lambda \in [0.1, 0.3, 0.5, 0.7, 0.9]$.	117
3.1	Language embeddings interpolation for accent simulation: 2D example. The degree of accentedness in speech can be adjusted by varying the relative weights of the English and French embeddings.	134
3.2	Privacy-to-Utility trade-off for language interpolation and accent embedding at varying WER/EER trade-offs λ . First row, language embedding interpolation with oracle transcriptions. Second row, language embedding interpolation with ASR-TTS transcripts. Third row, accent embedding with ASR-TTS transcripts. Lower is better.	142
5.1	Example of array of randomly distributed smart devices. Adapted from [299]	156
5.2	Gradient Boosted Regression Trees. Left column shows predictions of three trees; the right column shows the ensemble's combined predictions. Adapted from [330].	165
5.3	Example of marginal contribution computation for <i>player 1</i> .	165
5.4	Computation example of the Shapley value associated with <i>player 1</i> . First, all the possible coalitions with <i>player 1</i> need no be computed. Once all the 8 coalitions have been formed, the next step consists of computing all the marginal contributions δ_i , $i = 1, 2, \dots, 8$. Finally, the mean marginal contribution ϕ (or Shapley value) for <i>player 1</i> , is computed by averaging all the 8 marginal contributions.	166
5.5	Example of KS computation of Shapley values. Note that when computing the prediction for coalition $(1, 0, 1, 0)$, f_1 and f_3 are taken from the instance $\mathbf{x}$ and, since ML models can not omit features at inference time, the original values of f_2 and f_4 of $\mathbf{x}$ are replaced with values drawn from the background set. Once selected a finite number N of coalitions z_i , the marginal contributions δ_i can be computed from the ML model output $\hat{y}_i$ by employing the training loss function $l(z_i, \hat{y}_i) = \delta_i$. Finally, the weights w_{z_i} can be computed following equation [5.11] and can be used in the linear model [5.12] to compute the Shapley values ϕ .	168

5.6	Acoustic channel estimation. A : Estimation error, B : Normalized Projection	
	Misalignment (NPM), C : Ground truth relative RIR, D : Estimated relative	
	RIR.	170
5.7	Relative error comparison for GCC-PHAT (gcc), XGBoost (xgb), and a	
	DeNN (neural) for broad-band white noise inputs.	172
5.8	SHAP values for XGBoost and DeNN calculated for the small room and	
	white noise inputs.	172
5.9	Relative error comparison for XGBoost (xgb) and a DeNN (neural) for	
	speech input signals.	173
5.10	SHAP values for XGBoost and the DeNN calculated for the small room and	
	speech inputs.	174

List of Tables

1.1	ASR decoding results for the models introduced in Section 1.7	68
2.1	Voice Privacy Challenge datasets. Number of speakers and utterances for each dataset.	101
2.2	Voice Privacy development and test datasets. Number of speakers and utterances for each dataset.	102
2.3	Primary anonymization metric (EER, higher is better) results for development (light gray) and test (dark gray) sets. Orig: Non-anonymized original data. B1a: baseline from [201]. V_R : ZS-VC with random target speaker selection. V_{GP} : ZS-VC with random target speaker selection and gender preservation. BV : B1a followed by ZS-VC. VB : ZS-VC followed by B1a. VCTK subsets comprise a set of utterances with same (comm.) and different (diff.) linguistic content across the speakers.	114
2.4	Primary utility metric (WER, lower is better).	115
2.5	Pitch correlation (ρ^{F_0} , higher is better). Average results computed with the same weights as in Table 2.3.	116

3.1 ASR decoding results for the *test-IND* set. The ASR model was fine-tuned on various partitions of *train-IND* that were augmented with TTS-generated data. R indicates the model trained exclusively on real data (some % of *train-IND*). R – TTS_c represents the model trained on real data combined with synthetic data generated by TTS_c. R – TTS_l represents the model trained on real data combined with synthetic data generated by TTS_l. The final column shows the relative Word Error Rate Reduction (WERR) achieved by each augmentation approach in comparison to the model without augmentation. For reference, the line below the table reports the WER for a model trained exclusively on the full partition of real data (*train-IND*), representing the lower bound for ASR models trained on a mix of real (< 100%) and synthetic data. 127

3.2 ASR decoding results for the *test-IND* set using an ASR model fine-tuned exclusively on TTS-generated data. TTS_c: model fine-tuned with synthetic data generated by TTS_c. TTS_l: model fine-tuned with synthetic data generated by TTS_l. The final column presents the WERR achieved by each augmentation method in comparison to the model trained solely on the [68] dataset. For reference, the line below the table shows the WER for a model trained exclusively on LibriSpeech 960, representing the upper bound beyond which fine-tuning with synthetic data alone begins to degrade ASR performance on genuine accented speech. 129

3.3 ASR decoding results for the Interspeech 2020 test sets. The upper section (light grey) presents ASR results using only real data. *C*: model trained on the 960-hour LibriSpeech dataset [68]. *FT*: model *C* fine-tuned separately on each accented training set. *FM*: model *C* fine-tuned on a combination of all accented training sets. The lower section (dark grey) shows ASR results with TTS data augmentation (75% TTS / 25% real data). *FT_{aug}* and *FM_{aug}* correspond to the previously mentioned models with TTS augmentation. *FM_{aug+db}*: *FM_{aug}* with batches composed of 75% TTS and 25% real data. 130

3.4	WER results for the <i>test-IND</i> set. $R - TTS_{INI}$: ASR model fine-tuned on a combination of real and synthetic speech, with the synthetic data generated by a TTS model fine-tuned on a small subset (3 speakers) of the <i>train-IND</i> dataset.	130
3.5	WER results for the <i>test-IND</i> set. R_{aug} : ASR model fine-tuned on real data augmented using the methods described in Section 3.4.3. $R_{aug} - TTS_{INI}$: ASR model fine-tuned on a combination of real and synthetic augmented speech, with the synthetic speech generated by a TTS model adapted to a subset of 3 speakers from the <i>train-IND</i> dataset.	131
3.6	WER results for language embedding interpolation and oracle transcription (O-TTS) for en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding.	135
3.7	EER results for language embedding interpolation and oracle transcription. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. . .	136
3.8	WER results for language embedding interpolation and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with way2vec2.0 ASR.	137
3.9	EER results for language embedding interpolation and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with way2vec2.0 ASR.	138
3.10	WER results for the accent embedding case and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with way2vec2.0	139

3.11	Primary anonymization metric (EER, higher is better) results for development (light gray) and test (dark gray) sets. Accent Embedding ASR-TTS.	
	en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding.	
	Transcriptions extracted with wav2vec2.0	140
4.1	ASR decoding on test subsets of [68] for different Mel input feature dimensions, olMEGA and anonymized speech with models trained on clean (dark gray rows) and fine-tuned (FT) on anonymized (white rows) data.	150
4.2	ASV performance on the Voice Privacy Challenge [201] dataset for different Mel feature dimensions, olMEGA and anonymized speech. O: original data. AN: anonymized data and model trained on unprocessed speech. FT: anonymized data and model fine-tuned on anonymized speech.	151
4.3	VAD performance measured with MCC on the LibriParty test set. The models were trained for different Mel input feature dimensions and olMEGA using the original signals (O) and after McAdams' anonymization (M). . . .	151
4.4	DER on the AMI corpus development and evaluation sets and the VPC simulated meetings sets for different Mel input feature dimensions, olMEGA and anonymized speech. The number of speakers per meeting is estimated and a forgiveness collar of 250 ms is applied. O: original data. AN: anonymized data and model trained on unprocessed speech. FT: anonymized data and model fine-tuned on anonymized speech.	153
5.1	Cross correlation matrix for white noise input signal	170

List of Notation

- **Boldface Letters:**
 - **Bold capital letters** (e.g., $\mathbf{A}$) represent **matrices**.
 - **Bold lowercase letters** (e.g., $\mathbf{a}$) represent **vectors**.
- Non-bold letters (e.g., a, x) represent **scalar values** or **single numbers**.
- $x[n]$ — Represents the n -th entry of a vector $\mathbf{x}$.
- $X[n, k]$ — Represents the entry in the n -th row and k -th column of matrix $\mathbf{X}$.
- $P(a|x)$ — Represents the **conditional probability** of a given x .
- $\hat{a}$ — The **estimate** of scalar a .
- $\|\cdot\|$ — Represents the **norm-2** (Euclidean norm) of a vector.
- $\mathcal{N}(y|m, \Phi_b)$ — The **normal distribution** of variable y with mean m and standard deviation Φ_b .

Abbreviations

PII	Personal Identifiable Information
ASR	Automatic Speech Recognition
EU-GDPR	European Union - General Data Protection Regulation
MSCA-ETN	Marie Skłodowska-Curie Actions European Training Network
SOUNDS	Service-Oriented, Ubiquitous, Network-Driven Sound
F0	Fundamental Frequency
DeNN	Dense Neural Network
FC	Fully Connected
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transform
LM	Language Model
SE	Squeeze-and-Excitation
LE	Label Encoder
AE	Audio Encoder
SVM	Support Vector Machine
UBM	Universal Background Model
EM	Expectation-Maximization
MAP	Maximum A Posteriori
ECAPA	Emphasized Channel Attention, Propagation and Aggregation
FAR	False Acceptance Rate
FRR	False Rejection Rate
SPSS	Statistical Parametric Speech Synthesis
MCC	Matthews Correlation Coefficient
GRU	Gated Recurrent Unit
FeFoT	Feed-Forward Transformer
GAN	Generative Adversarial Networks
DDPM	Denoising Diffusion Probabilistic Models
LLR	Log-Likelihood Ratio

OHNN	Orthogonal Householder Neural Network
ZS-VC	Zero Shot Voice Conversion
VC	Voice Conversion
BN	Bottle-Neck
SSAM	Speech Synthesis Acoustic Model
RIR	Room Impulse Response
ToF	Time of Flight
XGBoost	eXtreme Gradient Boosting
SHAP	SHapley Additive exPlanations
POS	Part Of Speech
NSF	Neural Source Filter
GCC	Generalized Cross Correlation
PHAT	Phase Transform
MTL	Multi-Task Learning
WERR	Word Error Rate Reduction
ZS-TTS	Zero-Shot Text-To-Speech
TTS	Text To Speech
NLU	Natural Language Understanding
ASV	Automatic Speaker Verification
STFT	Short Time Fourier Transform
DFT	Discrete Fourier Transform
MFCC	Mel Frequency Cepstral Coefficients
ANN	Artificial Neural Networks
TLU	Threshold Logic Unit
ISTFT	Inverse Short Time Fourier Transform
VAE	Variational Autoencoder
EER	Equal Error Rate
WER	Word Error Rate
MLP	Multi-Layer Perceptron
DNN	Deep Neural Network
MSE	Mean Squared Error
SGD	Stochastic Gradient Descent
MBGD	Mini-Batch Gradient Descent
TDNN	Time Delay Neural Network
PLDA	Probabilistic Linear Discriminant Analysis
SVD	Singular Value Decomposition
CNN	Convolutional Neural Network

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

Introduction

Motivation

Humans connect for a variety of reasons such as survival, procreation and for creating social bonds. Central to all these human interactions is speech, i.e. communication through an articulatory–acoustic–auditory channel between a sender and a receiver [1]. Furthermore, since the origins of the computer era, scientists dreamt of a conversational machine that could engage in a spoken natural language conversation. For example, Alan Turing’s well-known Imitation Game to test computational intelligence, contemplated a machine that could conduct such a human-like conversation, that people could not distinguish it from a human [2].

In recent years, advancements in speech technologies have significantly enhanced human-machine interaction, enabling the control of systems and devices through voice commands [3]. While voice assistants and smart speakers have simplified daily tasks and improved accessibility [4], they have also raised substantial privacy concerns among the public and policymakers [5]. These concerns stem from the fact that speech inherently carries extensive PII, both semantically and acoustically. Specific identifiers such as full names, social security numbers, or geographical locations can facilitate speaker identification. Additionally, voice characteristics including prosody, speaking rate, accent, and intonation encapsulate various types of PII, such as personality traits, physical attributes, emotional states, age, and gender, which can be extracted and potentially exploited for privacy violations [6]. In light of these privacy challenges, stringent data protection regulations like the European Union - General Data Protection Regulation (EU-GDPR) [7] mandate robust privacy-preserving measures for handling speech data. Suppressing PII in speech

signals not only enhances privacy but also ensures that personal data handling aligns with rigorous privacy standards. This approach is crucial for maintaining trust in digital communication technologies while complying with regulatory requirements.

Objectives

Due to the speech privacy requirements outlined in the [Motivation](#) Section above, this thesis is dedicated to devising more private speech recording solutions through voice anonymization. Anonymization in this context refers to the process of removing or altering PII from speech data to protect individual privacy. While the EU-GDPR does not mandate the systematic anonymization of personal data, it recognizes anonymization as a viable method that enables the processing of personal data while safeguarding the privacy rights of individuals. Throughout this thesis, the proposed anonymization techniques are designed to obscure speaker identity in speech signals, ensuring that the anonymization aligns with additional utility measures specific to the tasks for which anonymization is employed.

Summary of contributions and thesis structure

- Chapter 1 starts with the basics of speech production and speech perception in humans. Furthermore, fundamental speech processing algorithms and artificial neural networks are introduced. Finally, basic approaches behind speech recognition, speaker recognition and speech synthesis are briefly discussed.
- Chapter 2 describes the Voice Privacy Challenge initiative that motivated the research presented in this thesis and frames the specific anonymization requirements and assumptions in its framework. Furthermore, a new approach to the Voice Privacy Challenge is introduced and its anonymization results are presented.
- Chapter 3 introduces a zero-shot voice conversion and a text-to-speech model adapted for simulating different accents. Results obtained using the aforementioned model will be presented in the context of data augmentation and privacy enforcement.
- Chapter 4 and 5 present two side projects that have been developed during the two Secondments included in the MSCA-ETN SOUNDS project. The first describes

a privacy-preserving feature extraction method based on input feature dimension reduction, spectral smoothing and a low-cost speaker anonymization technique. The second introduces the task of inter-device distance estimation by exploiting a new set of features obtained by extending the definition of some acoustic attributes of the room impulse response to its relative version.

- Finally, Chapter 6 includes an overall discussion and conclusions.

Chapter 1

Speech and Neural Networks

1.1 Introduction to Speech Processing

In this Section, basic human speech production, speech perception and speech processing tools are introduced. First, the physiology of speech production and perception are described in combination with the basic mathematical sound wave processing. Second, the chapter offers a concise overview of the fundamental principles underpinning the theory of artificial neural networks. Third, we describe some widespread artificial neural network architectures employed in the methods presented in this thesis. Finally, the last Sections describe the technology behind the three fundamental speech applications enabling privacy-preservation evaluation namely, automatic speech recognition, automatic speaker recognition and speech synthesis.

1.2 Human Speech Production

Talking is a unique human ability. Speech, is a complex motor activity [8] that any healthy human acquires: as a species, humans not only do have evolved brains that process and produce language and syntax, but they can also produce an exceptionally vast range of sounds and tones that are used to form hundreds of thousands of words. To articulate all those sounds, humans use the same basic apparatus that chimps have: lungs, throat, voice box, tongue and lips which are shown in Fig. 1.1. However, over thousands of

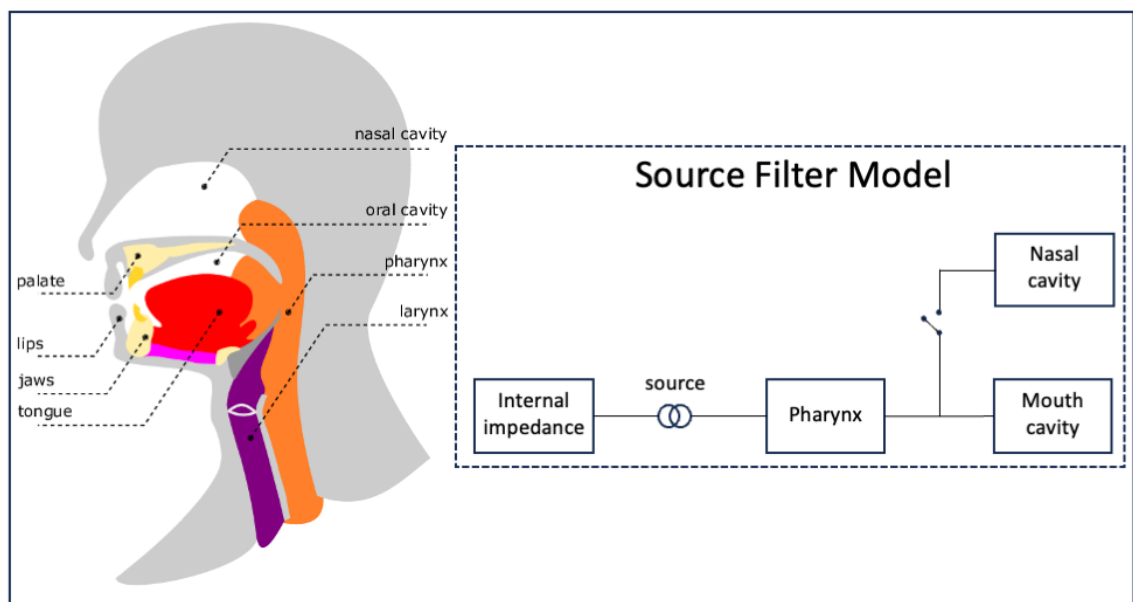


Figure 1.1: The Source-Filter Model of speech production, showing the anatomy of the human vocal apparatus on the left, and its corresponding schematic representation on the right. The diagram includes the nasal cavity, oral cavity, and pharynx, which act as resonating filters, and the larynx, which serves as the sound source. This model encapsulates how the airflow from the lungs is modulated into sound by the glottis and further shaped by the vocal tract into speech with distinct resonances or formants. Adapted from [9].

years, humans have evolved a longer throat and smaller mouth better suited for shaping sound. When humans talk or sing, they release controlled puffs of air from the lungs through the larynx, or voice box. The larynx is about the size of a walnut and, in males, it is immediately visible: it's the Adam's apple which is mostly made up of cartilage and muscle. Stretched across the top of the larynx, there are the vocal chords. These structures are essentially two folds of mucous membrane which, when we expel air from the lungs and push it through the larynx, vibrate, producing the sound. The fundamental characteristic of the vocal cord is its periodicity expressed by the duration T_0 of a complete voice period or by the inverse value called the voice fundamental frequency $F_0 = \frac{1}{T_0}$ Hz. The frequency of this buzzing is what gives sound the pitch. Humans can modulate the pitch by tightening the vocal cords to make the voice higher and loosening them to make a lower sound. However, fundamental pitch and fundamental frequency are not synonyms: these terms are usually used interchangeably due to the close one-to-one relationship but, in more strict terminology, pitch is a tonal sensation and frequency a property of the sound stimulus. Mathematically, the speech wave can be interpreted as the response of the vocal tract filter to one or more sound sources. This simple idea of modeling speech in the terminology of acoustic and electrical engineering, implies that the speech wave may be uniquely expressed in terms of source and filter characteristics. This model, also known as the *Source-Filter theory* of speech production, is exemplified in the block diagram of Fig. [1.1](#): it contains interconnected filter blocks, each one representing a part of the vocal tract. In this representation, the source is the air flow from the lungs, modulated at the glottis to produce a quasi-periodic sound, or above the glottis to produce a noisy turbulence. The filter is the supra-laryngeal vocal tract which, with its varying configurations, gives rise to varying resonances (formants). The complete filter function, also known as the transfer function, measures how sound pressure varies with frequency at a specific point in front of the speaker, relative to the pressure right at the sound source. This function illustrates how different sound frequencies are either diminished or maintained as they travel from the source to the designated point [\[10\]](#). By indicating a sound source with the

letter S and the transfer function T , the equation,

$$P(f, t) = S(f, t) \cdot T(f, t)$$

describes the speech sound P at any given frequency f and time t . Here, $P(f, t)$ is the resulting speech sound, $S(f, t)$ is the sound pressure level at the source, and $T(f, t)$ is the transfer function that modulates the sound as it travels from the source to the observer. Finally, it is worth mentioning that each voice is characterized by the spectrum envelope $S(f, t)$ which describes the amplitude of the source harmonics as a function of their frequency. This quantity incorporates personal characteristics of the speaker but can vary in combination with voice register, volume and fundamental pitch [10] which can further reflect speaker's emotional state [11].

1.3 Speech Perception

Sound waves are fluctuations in the air pressure which are generated by a moving or vibrating object modulating the air density. This modulation carries information about the event that disturbed the medium. The simplest describable sound wave is the sine tone. This wave is described by its frequency, amplitude and phase. The frequency reveals the rate at which the object is vibrating and changes in frequency of the sine wave are perceived as changes in pitch. The amplitude describes the magnitude of the variations in air pressure and its perceptual correlate is the loudness. Finally, the phase is the point in the sine cycle at which the wave starts. For single sounds, changes in the starting phase do not result in noticeable perceptual changes. This does not mean that phase is not encoded by the auditory system. In fact, the phase delay between sounds reaching each ear result in differences in perceived location of the sound source [12]. Single sine wave sounds alone are not ecologically valid as we do not experience them in the real world. However, Fourier's theorem guarantees each sound $s(t)$ can be decomposed in an infinite superposition of sine

and cosine waves

$$s(t) = A_0 + \sum_{n=1}^{\infty} A_n \sin 2\pi n F_0 t + \sum_{n=1}^{\infty} B_n \cos 2\pi n F_0 t. \quad (1.1)$$

Figure 1.2 shows the spectra of two instruments playing the same note. Both of the

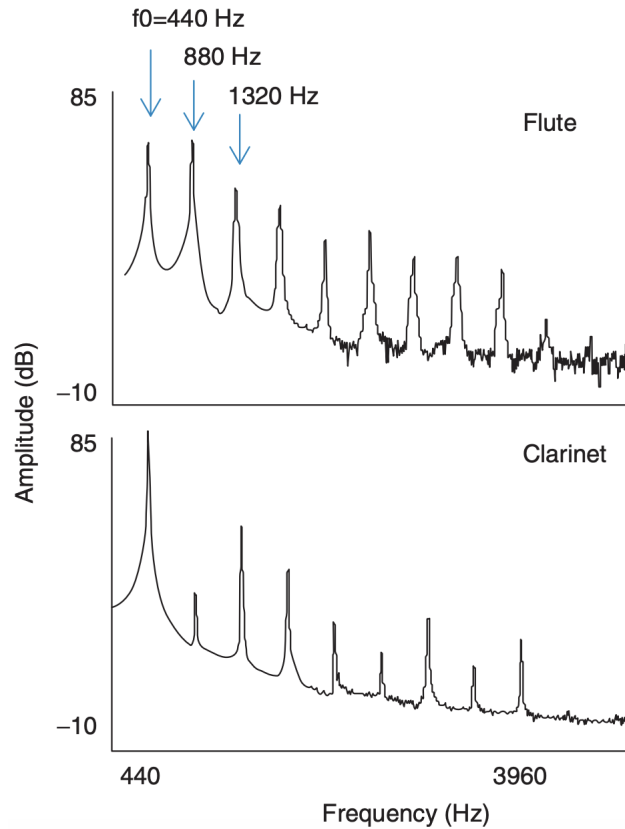


Figure 1.2: Spectra of a flute and a clarinet playing the same note. Taken from [12].

sounds are composed of a Fundamental Frequency (F_0) and harmonics at multiples of F_0 . When all harmonics superimpose, listeners do not hear a number of simultaneous pitches corresponding to each sine wave. Instead, the listener perceives one pitch that matches that of the F_0 . Hence, the two sounds presented in the figure will have the same pitch and, thus, be perceived as the same musical note. The feature humans use to distinguish between the sound produced by the two instruments is called timbre which is a measure of the amplitudes of the harmonics. However, contrarily to music, for speech sounds the variation in timbre is more important than the variation in pitch. In fact, vowels differs

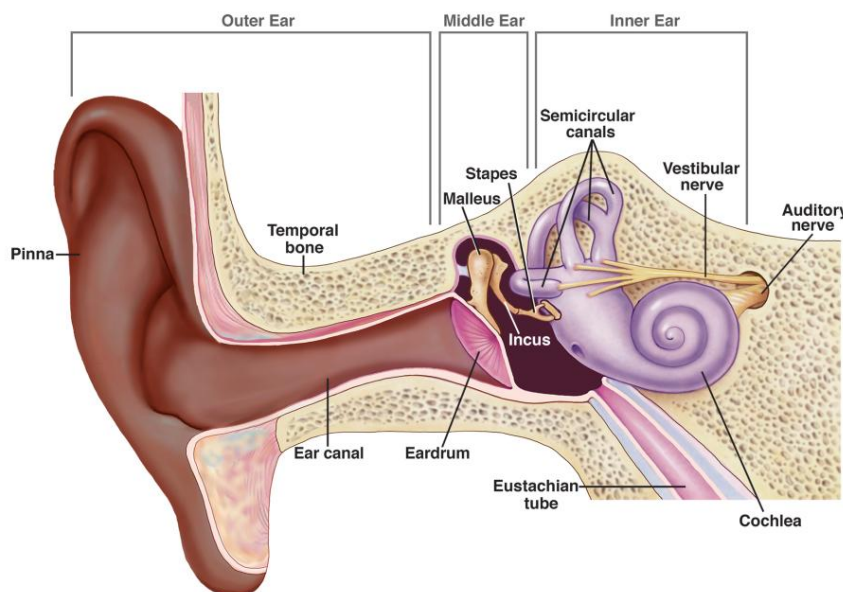


Figure 1.3: Illustration of the human ear anatomy, showing the three main sections: outer ear, middle ear, and inner ear. The pinna directs sound waves into the ear canal, impacting the eardrum, visible here as the tympanic membrane. These vibrations are then transmitted through the ossicles (malleus, incus, and stapes) to the cochlea in the inner ear, which is linked to the auditory nerve. Taken from [13].

mainly in timbre: by appropriately shaping the vocal tract, humans can generate different resonances which result in relative amplitude patterns across the harmonics. However, voice pitch does play a role in speech communication, especially for tone languages such as Mandarin Chinese, but changes in timbre carry most of the linguistic information [12]. At the neural level, in humans, perception begins with sound pressure waves entering the outer ear: the pinna in Fig. [1.3]. This wave sets into vibration the tympanic membrane (eardrum, Fig. [1.3]), which transfers its movement to a chain of three bones (the ossicles). This bone structure has the role of amplifying, with a lever mechanism, and transferring the mechanical movement generated by the sound pressure, to the fluid-filled inner ear: the cochlea. The role of the cochlea is to encode the physical stimulus generated by sound pressure into an electrical signal. When the ossicles transduce the sound wave to the cochlea, illustrated in Fig. [1.4], this vibration causes the displacement of the basilar membrane. This structure has a flexible design with varying mechanical impedance. As a result, higher frequencies stimulate the stiffer base more than lower frequencies. This

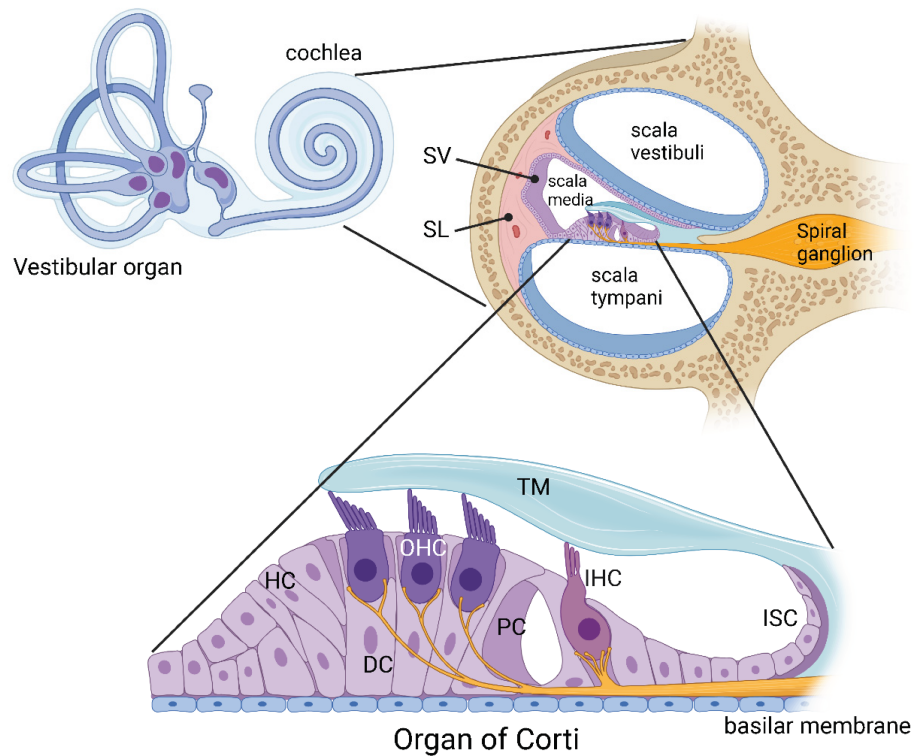


Figure 1.4: Cochlear structure, showcasing the cochlea, vestibular organ, and organ of Corti within the inner ear. The image highlights the scala vestibuli, scala media, and scala tympani, which are essential for the cochlea's role as a spectral analyzer. The basilar membrane, depicted at the bottom, varies in stiffness and is crucial for frequency analysis of incoming sound waves. Inner hair cells (IHC) on the basilar membrane transduce mechanical movements into neural signals by bending their stereocilia, initiating a chemical cascade that results in auditory perception. This complex process allows for the encoding of sound waves into electrical signals that travel via the auditory nerve. Taken from [14].

creates a "place code" along the basilar membrane, allowing the cochlea to act like a Fourier analyzer. Finally, auditory receptors called inner hair cells are located on the basilar membrane. They have hair-like stereocilia which, when the basilar membrane is displaced by the wave, are bent setting up a cascade of chemical signals that translate the mechanical energy into a neural code along the auditory nerve.

1.4 Time-Frequency Processing

In a similar way as human cochlea does, a microphone converts sound into a small electrical current. In some designs, sound waves hit a diaphragm that vibrates, moving a magnet near a coil: the varying magnetic field causes a current in the coil which encodes sound information. In condenser microphones instead, the incoming sound causes the vibration of one plate of a capacitor and the varying capacitance is converted into the corresponding electrical signal. When dealing with speech, a complex and highly redundant signal, it is essential to derive a more compact representation. Traditionally, scientist hand-crafted speech features ad-hoc for specific algorithmic applications. If, on one side, those features are still largely in use, on the other, the advent of machine and deep learning brought the possibility of learning task-specific features directly from the data [15].

Speech features are typically extracted by transforming the signal from the time domain to the time-frequency domain using the Short Time Fourier Transform (STFT). The STFT segments the original signal into usually overlapping frames of a fixed size, applying the Fourier transform to each segment individually. For a discrete-time signal $\mathbf{x}[p]$, sampled from the continuous-time speech signal $\mathbf{x}(t)$, the STFT is defined as:

$$\mathbf{X}[n, k] = \sum_{m=-\infty}^{\infty} \mathbf{x}[m] \mathbf{w}[n - m] e^{-j2\pi km/N} \quad (1.2)$$

where $\omega_k = \frac{2\pi k}{N}$ represents the angular frequency, $k \in [0, N - 1)$ where N is the number of frequency points used in the Discrete Fourier Transform (DFT) for each segment of the signal, $w[n - m]$ is a window function designed to reduce frequency leakage in the DFT and n is the time frame index. Figure 1.5 illustrates both the time-domain representation of a

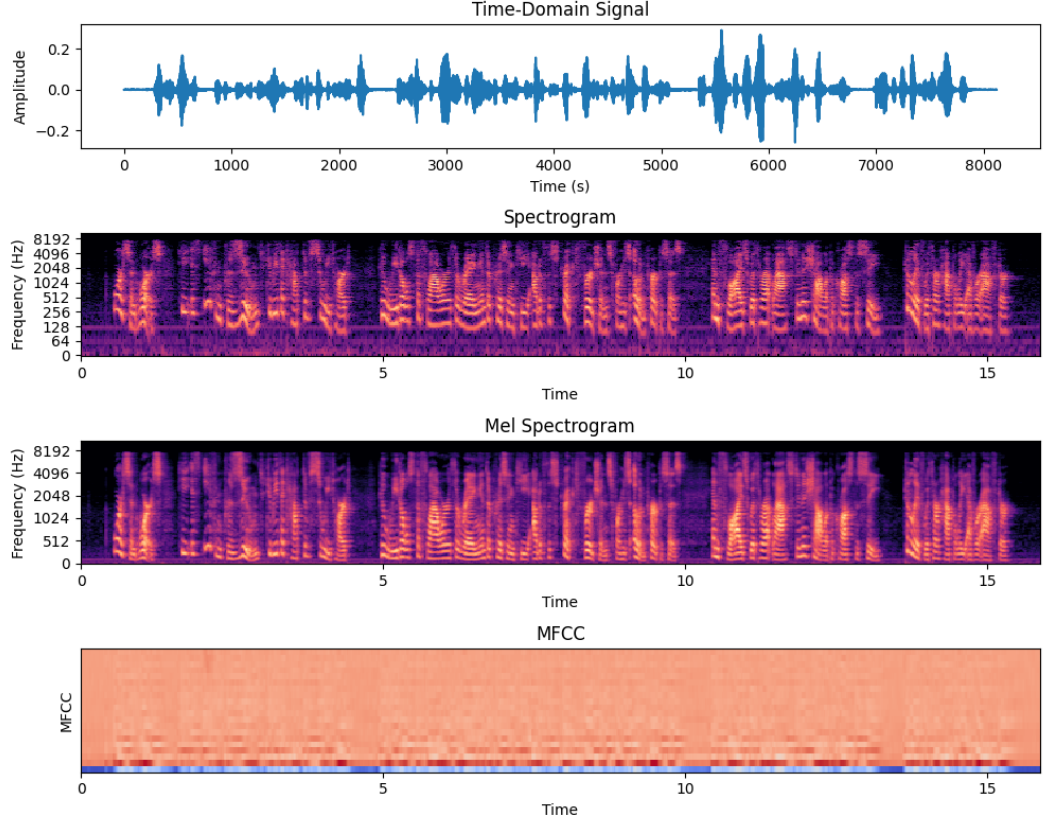


Figure 1.5: Representation of speech signal processing from waveform to Mel Frequency Cepstral Coefficients. At the top, the time-domain representation. The spectrogram details the spectrum of frequencies present in the signal over time. Further down, the Mel Spectrogram transforms this information into a format that aligns more closely with human auditory perception. Bottom, the Mel-frequency Cepstral Coefficients abstract the audio into a set of features that capture the essential spectral properties.

speech signal and its power spectrum, $|\mathbf{X}[n, k]|^2$. Human perception of loudness varies with frequency, and our sensitivity to frequency changes decreases at higher frequencies. This auditory phenomenon is captured in the Mel-scale power spectrum, where the frequency response is adjusted using triangular Mel-scale filter banks. This transforms the power spectrum into a Mel-scale representation, as shown in Fig. 1.6:

$$\mathbf{Y}[n, j] = \sum_{k=1}^N \mathbf{W}[j, k] |\mathbf{X}[n, k]|^2 \quad (1.3)$$

where k indicates the DFT bin number, j ranges from 1 to M , each corresponding to a different Mel-filter bank and n is the time frame index. The Mel-scale power spectrum can

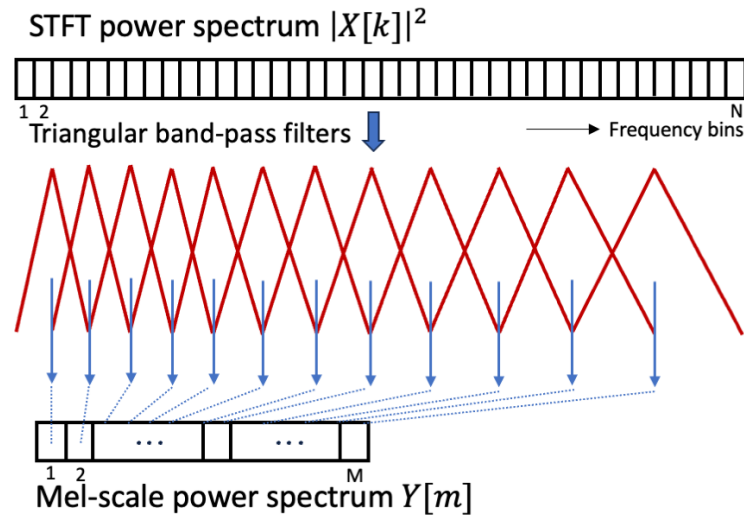


Figure 1.6: Transformation from the STFT power spectrum to the Mel-scale power spectrum using triangular Mel-scale filter banks. This transformation adjusts the frequency bins of the original power spectrum (top) to align with human auditory perception, which is less sensitive to higher frequencies. The triangular filters cover wider ranges as frequency increases to mimic this perceptual characteristic. Each filter’s output contributes to a corresponding bin in the Mel-scale power spectrum (bottom), effectively summarizing the spectral content in a way that reflects human hearing. This processed spectrum forms the basis for extracting MFCCs, which are crucial for various audio processing tasks.

be further processed to extract the Mel Frequency Cepstral Coefficients (MFCC). Once the Mel-scale power spectrum is computed, the next step involves taking the logarithm of each Mel power spectrum value. This logarithmic step is crucial as it closely mimics the human ear’s response to different loudness levels, which is logarithmic in nature. After applying the logarithm to the Mel-scale power spectrum, a Discrete Cosine Transform (DCT) is applied to the log-Mel spectrum. The DCT helps in de-correlating the log-Mel spectrum and results in a compact representation of the filter banks, emphasizing the amplitudes of the spectral envelope of the original signal. Further advantages of decorrelated features can be found in App. A. The coefficients obtained from the DCT are the MFCCs. Typically, only the first few coefficients (e.g., the first 12-13) are retained, as they contain the most significant information about the spectral shape, while higher-order coefficients are discarded as they often represent fine details and noise. The process of extracting MFCCs from the Mel-scale power spectrum effectively captures the important characteristics of the speech signal, making MFCCs invaluable for various tasks such as speech recognition, speaker

identification, and audio classification.

1.5 Artificial Neural Networks

In this Section, we introduce the main concepts behind modern Artificial Neural Networks (ANN)s, from conceptual similarities with biological neurons to the mainstream training and optimization procedures.

ANNs were first conceptualized in 1943, drawing inspiration from biological neural networks. The first public demonstration of an ANN occurred in July 1958 when the U.S. Office of Naval Research showcased a computer that learned to differentiate between punch cards marked on the left and those marked on the right, using Frank Rosenblatt’s “perceptron” model. Despite this early success, ANNs experienced periods of decline, known as AI winters during the 1960s and 1990s, due to limited computational power, inefficient training methods, and insufficient data. However, the advent of backpropagation and parallel computing revitalized ANNs, leading to their widespread adoption across various scientific fields, including speech processing.

1.5.1 Biological Neurons

Biological neurons are the basic unit of the nervous system for most animals. They are composed by a cell body, also called *soma*, the *dendrites* and an *axon*. The *soma* contains the cell’s nucleus and all the other organuli sustaining cell’s metabolism. The *dendrites* are short-range branches extending from the cell body towards the outside space. The *axon* is a long-range, single extension of the cell body which terminates by splitting in many branches. At the end of each branch, there are the synaptic terminals, also known as *synapses*. In the case of excitatory connections, when the pre-synaptic neuron produces an electrical signal (the *action potential*), this travels through the axon until the *synapse* where it triggers the release of a class of chemicals: the *neurotransmitters*. These molecules travel from the synapse of the pre-synaptic neuron to the post-synaptic cell. If present in a high enough concentration, an *action potential* is generated in the post-synaptic neuron.

If, on one side, individual neurons seem to behave in a rather simple way, on the other, their complex organization in networks allows complex computations to be performed [16].

1.5.2 Perceptron

Drawing inspiration from the operational principles of biological neurons, the perceptron, or the Threshold Logic Unit (TLU) [17], is one of the simplest ANN architecture. The

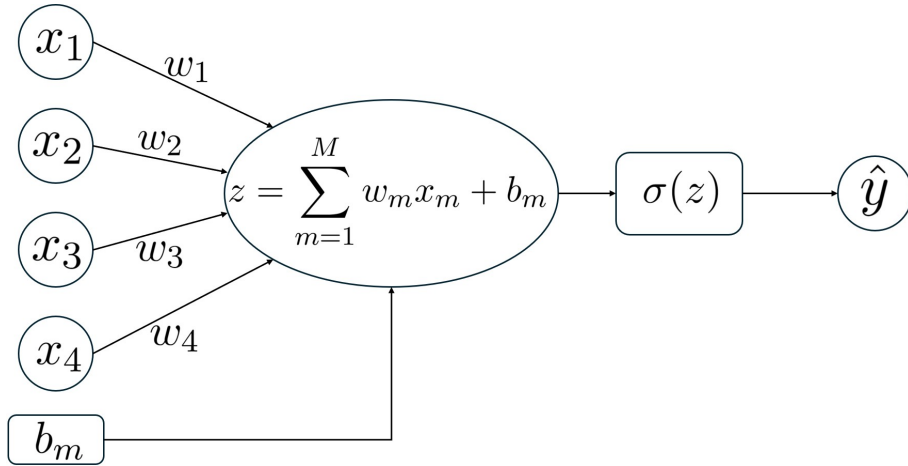


Figure 1.7: Threshold Logic Unit. The process begins with inputs (x_1 to x_4), each multiplied by corresponding weights (w_1 to w_4). These weighted inputs are summed together with a bias (b_m) to compute the input to the activation function (z), which applies a threshold to produce the binary output ($\hat{y}$).

TLU (Figure [1.7]) computes a weighted sum of the inputs

$$z = \sum_{m=1}^M w_m x_m + b_m, \quad (1.4)$$

where x_m are the inputs, w_m the weights associated with each input and b_m are bias terms. Then, it applies to z the *activation function* σ leading to the output

$$\hat{y} = \sigma(z). \quad (1.5)$$

Originally, the activation function was the Heaviside step function [18]. This was motivated by the fact that in biological neurons, when the dendrites integrate electrical signals coming from neighbouring cells, if the cumulative voltage exceeds a threshold, an action potential

is generated. A single TLU cell can therefore be used for simple binary classification: by computing a linear combination of the inputs, if the result is greater than a threshold, it outputs the positive class otherwise, it outputs the negative class. Regarding the training procedure, this was also inspired by biological neurons and more specifically by Hebbian Learning [19]. By summarizing this principle in a single sentence “*cells that fire together, wire together*” [20]. Mathematically, the perceptron learning rule

$$w_{i,j}^{k+1} = w_{i,j}^k + \eta(y_j - \hat{y}_j)x_i \quad (1.6)$$

states that the updated weight $w_{i,j}$ describing the connection between neurons i and j , depends on the weight at the previous iteration k , an error term weighted by η (the learning rate) and the input x_i . One limitation of this model is that the decision boundary is linear

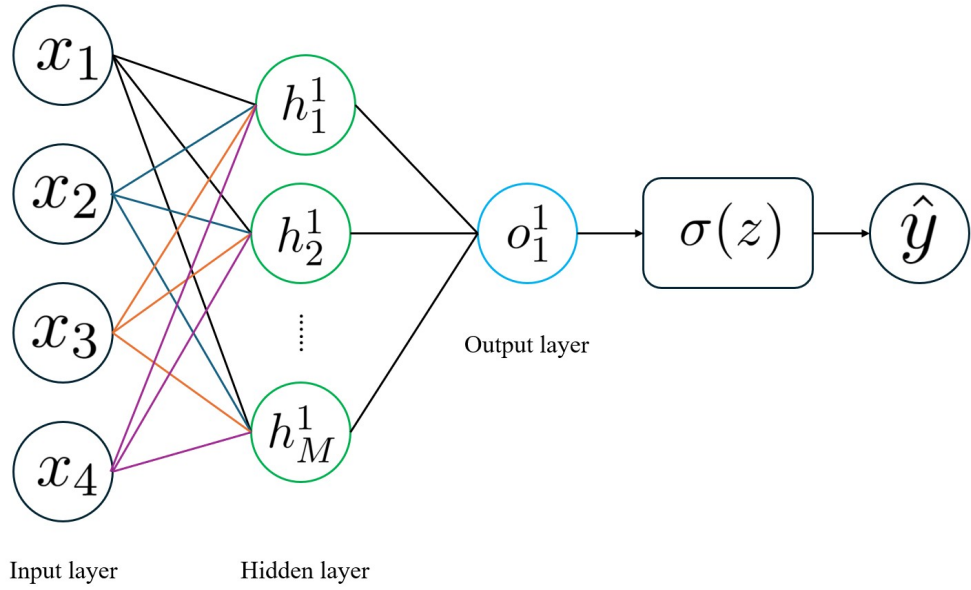


Figure 1.8: Example of a Multi-Layer Perceptron with a single hidden layer with M neurons h_i^1 and output layer with one unit o_1^1 .

therefore, this architecture is incapable of learning complex non-linear patterns. This limitation can be overcome by adding a non-linear activation function such as a ReLU, sigmoid, or tanh [21]. Moreover, stacking multiple perceptrons in an ANN hierarchical structure leads to the Multi-Layer Perceptron (MLP), Figure 1.8. This architecture is composed of an input layer, one or more hidden layers and one final output layer. Each layer

but the output layer, is fully-connected to the next layer and has a bias term per-neuron. Given the stack structure of the network, models of this type fall under the field called Deep Neural Network (DNN). Originally, MLP training consisted of applying equation 1.6 to each layer while freezing all the other weights of the network. This was highly inefficient and contributed to the first AI winter during the '60s. The modern training procedure of DNN is Gradient Descent combined with the *generalized delta rule* [22] for efficient gradient computations.

1.5.3 Training Procedure: Gradient Descent

Modern DNN training procedures are based on the idea of minimizing a cost function from which the network weights can be computed. Consider a general, supervised regression task with input data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K\}$, $\mathbf{x}_i \in R^N$ and targets $\mathbf{y} = \{y_1, y_2, \dots, y_K\}$, $y_i \in R^1$. Given a generic regression model ρ with parameter vector $\mathbf{w}$, $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_K\}$, $\hat{y}_i \in R^1$ can be computed as $\hat{y}_i = \rho(\mathbf{x}_i; \mathbf{w})$. In order to evaluate the error the model commits by estimating $\mathbf{y}$ with $\hat{\mathbf{y}}$ a commonly used cost (loss) function is the Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^K (\hat{y}_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^n (\rho(\mathbf{x}_i; \mathbf{w}) - y_i)^2. \quad (1.7)$$

The objective is then to compute $\mathbf{w}^* = \arg \min_{\mathbf{w}} \text{MSE}(\mathbf{X}, \mathbf{y})$. The general idea behind the gradient descent is to estimate the parameters vector iteratively employing the update rule

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \nabla_{\mathbf{w}} \text{MSE}(\mathbf{X}, \mathbf{y}) \quad (1.8)$$

to tweak the model parameters in order to minimize the cost function. This minimization relies on the computation of the gradient of the cost function with respect to the model parameters $\mathbf{w}$

$$\nabla_{\mathbf{w}} \text{MSE}(\mathbf{X}, \mathbf{y}) = \begin{pmatrix} \frac{\partial \text{MSE}(\mathbf{X}, \mathbf{y})}{\partial w_1} \\ \frac{\partial \text{MSE}(\mathbf{X}, \mathbf{y})}{\partial w_2} \\ \dots \\ \frac{\partial \text{MSE}(\mathbf{X}, \mathbf{y})}{\partial w_M} \end{pmatrix}, \quad (1.9)$$

which indicates the direction of steepest variation of the cost function. Gradient descent (Equation 1.8) computes the optimal weight's update by extracting the gradient from the loss function, invert its sign to move towards a local minimum and multiply the gradient vector by η to determine the size of the downhill step. Figure 1.9 depicts the gradient descent iterative procedure. First, to compute the initial value of the loss function, the

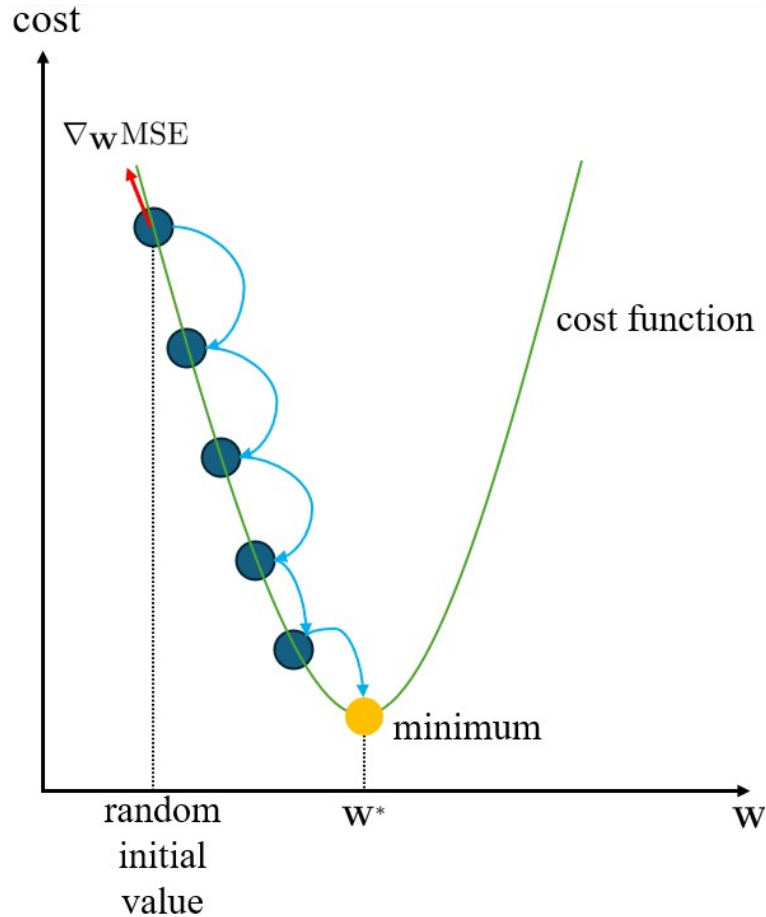


Figure 1.9: Gradient Descent: 1D example of the MSE loss function. The red arrow indicates the $\nabla_{\mathbf{w}} \text{MSE}$ computed in the initial random point.

weight vector $\mathbf{w}$ is randomly initialized. Then, step-by-step, the algorithm attempts to gradually improve the value of the loss function until it converges to a minimum. What controls the speed for the algorithm to reach the minimum is η : if the learning rate is too small, the algorithm has to go through many steps before reaching the minimum. On the other side, if η is too large, the algorithm might start jumping back and forth across the two sides of the loss function never reaching the minimum and, potentially, starting

to diverge. Finally, not all the cost functions are convex as the MSE. If, in the case of convex losses, gradient descent is guaranteed to approach arbitrarily close to the global minimum, for non-convex functions (Figure 1.10) there is no such guarantee. Moreover, random initialization (Figure 1.10, A or B) might profoundly affect the convergence of the algorithm to the global minimum. Operatively, in gradient descent $\nabla_{\mathbf{w}}\text{MSE}$ is computed on

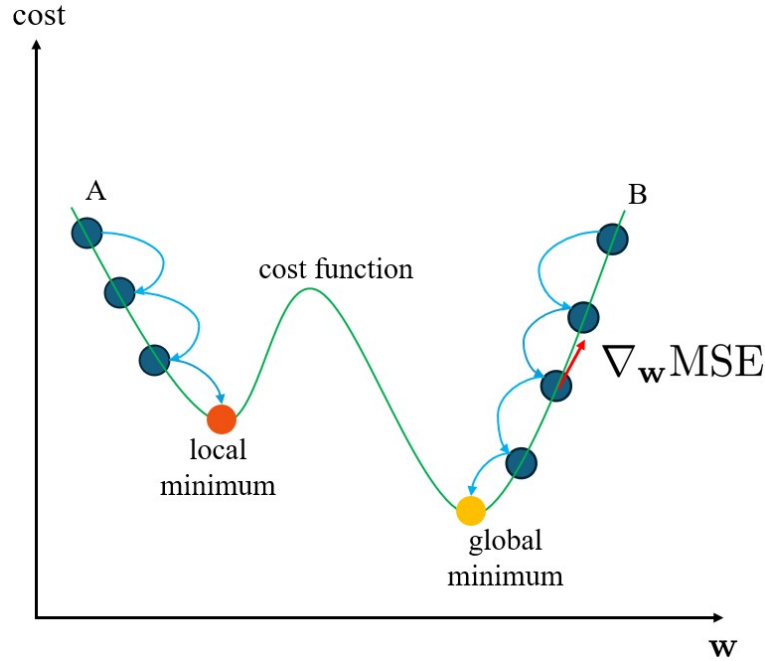


Figure 1.10: Gradient Descent: 1D example of a non-convex loss function. The red arrow indicates the $\nabla_{\mathbf{w}}\text{MSE}$. A and B indicate two different random initializations of the parameter vector $\mathbf{w}$.

the whole training set $\mathbf{X}$ at each iteration step. This is a serious problem when the training set is large. To solve this problem, a solution is to use Stochastic Gradient Descent (SGD) [23]. In this case, for each optimization step, a single training example is used to compute the gradient. If, SGD makes training on large datasets very fast and its randomness helps to escape from local optima ($\nabla_{\mathbf{w}}\text{MSE} = 0$), on the other, convergence is much more erratic than gradient descent. Specifically, when the algorithm ends up close to the minimum, it will continue bouncing around it leading to a non-optimal parameter vector $\mathbf{w}$. For practical applications, Mini-Batch Gradient Descent (MBGD) is used: instead of using the full dataset $\mathbf{X}$ or a single instance $\mathbf{x}_i$, MBGD computes the gradient on a small random set of $\mathbf{X}$ called a *mini-batch*. This has the advantage of speeding up the training process with

respect to gradient descent while mitigating the problems of SGD specifically if combined with learning rate scheduling techniques such as [24]. To efficiently compute gradients, *backpropagation* [22] implements a forward and a backward pass through the network to compute, one layer at a time, $\nabla_{\mathbf{w}}\text{MSE}$ with respect to all model parameters $\mathbf{w}$. The forward pass consists of calculating $\hat{\mathbf{y}}$ given the input matrix $\mathbf{X}$ and the model weights $\mathbf{w}$. In the backward pass, the algorithm exploits the chain rule of derivatives to efficiently calculate $\nabla_{\mathbf{w}}\text{MSE}$ avoiding redundant calculations of intermediate terms. Following these ideas, by considering a single example $y \in \mathbf{Y}$, each term in equation [1.8] can be computed as

$$\frac{\partial \text{MSE}}{\partial w_i} = \frac{\partial (\hat{y} - y)^2}{\partial w_i} = 2 \cdot \frac{\partial \hat{y}}{\partial w_i} = 2 \cdot \frac{\partial \sigma}{\partial z} \frac{\partial z}{\partial w_i} \quad (1.10)$$

where σ is the activation function and z is the term in equation [1.4]. Finally, equation [1.10] can easily be extended to whatsoever cost function and multi-layer architecture.

1.6 Fundamental Deep-Learning Architectures

In this Section, a brief overview is provided of the fundamental deep learning architectures commonly used in speech technology. Furthermore, three fundamental speech processing fields will be introduced: ASR, Automatic Speaker Verification (ASV) and Speech Synthesis.

1.6.1 Convolutional Neural Networks

Convolutional Neural Network (CNN), as the original MLP, have been inspired from biological neurons specifically, the visual cortex neurons. Those cells have a local receptive field making them sensitive to visual stimuli located in a specific region of the visual field and with a specific orientation [25]. Moreover, the visual cortex has a hierarchical structure: cells react only to specific line orientations others have large receptive fields and can detect more complex patterns that are combinations of the low-level features [25], [26].

The main feature differentiating MLPs and CNNs is that for the latter the neurons in the first layer are not connected to all pixels of the input image but only to pixels in their receptive fields. As visible in Fig [1.12], neurons in the second convolutional layer

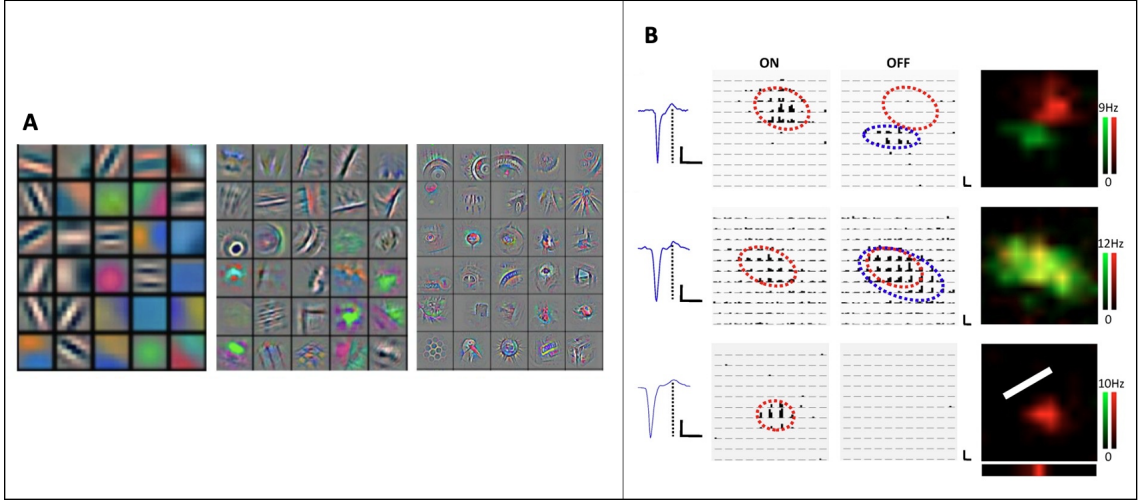


Figure 1.11: Measured receptive fields of mice visual cortex neurons and CNN features. Panel A, from left to right: low, middle and high level features extracted from a CNN architecture. Adapted from [27]. Panel B: Measured receptive fields of visual cortex neurons in mice. Adapted from [28], Copyright 2009 Society for Neuroscience.

are connected only to units in a small rectangular window of the first layer: this specific connection scheme allows the network to select low-level features in the first hidden layer and then assemble those into higher-level features in the upper layers. The weights of each unit in a CNN can be represented as small images the size of the receptive fields and are called *filters* or *convolution kernels* (as depicted in Fig. 1.11). When a filter is applied to the input image, it originates a *feature map*. Importantly, all neurons within the same feature map share the same weights: this means that the CNN is able to detect the same feature at multiple locations in the input image. To mathematically derive the output of a general unit in the architecture, it can be noticed that neurons located in row i, j of a given layer, are connected to neurons in rows $i * s_h$ to $i * s_h + f_h - 1$ and columns $j * s_w$ to $j * s_h + f_w - 1$ where f_h, f_w are the filter height and width, and s_h, s_w are the vertical and horizontal strides (i.e. the shifts between two consecutive receptive fields, Fig. 1.12). Now the output of a neuron in the CNN can be computed as:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_n-1} x_{i',j',k'} * w_{u,v,k',k} \quad (1.11)$$

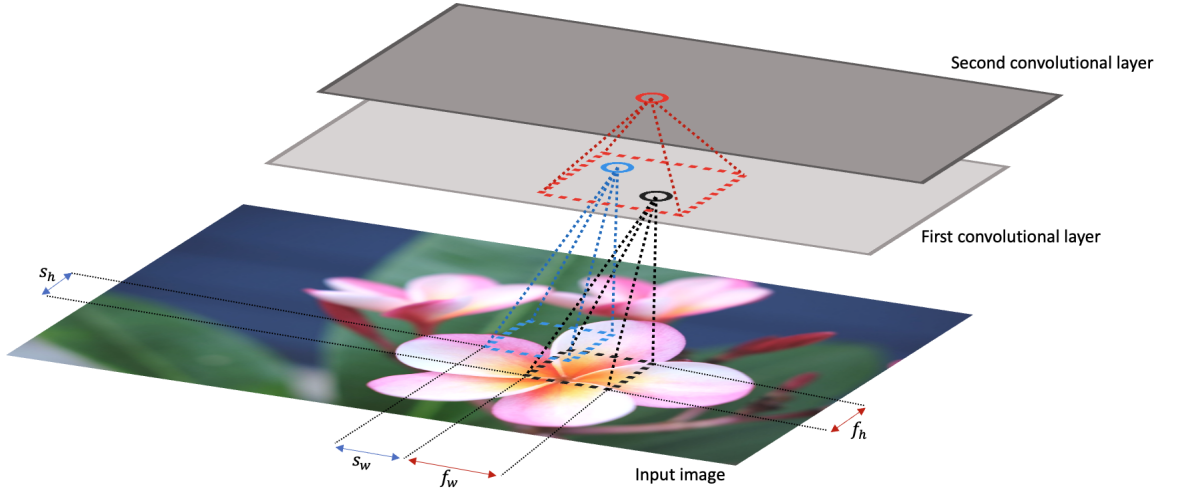


Figure 1.12: CNN processing of an input image: f_h, f_w are the filter height and width and s_h, s_w are the vertical and horizontal strides.

where $z_{i,j,k}$ is the output on the unit located at row i , column j and feature map k , f'_n is the number of feature maps in the previous layer, $x_{i',j',k'}$ is the output of the neuron in the previous layer (located at row i' , column j' and in feature map k'), b_k is the bias term for the feature map k and $w_{u,v,k',k}$ is the connection weight between any neuron in the feature map k and its input located in row u , column v and feature map k' .

If CNNs have traditionally excelled in processing spatial data through 2D convolutions, particularly for image analysis, the adaptation of convolutional architectures to 1D convolutions has significantly advanced the processing of sequential data, such as audio signals. WaveNet [29], a model that operates directly on raw audio waveforms, utilizes a deep network of 1D convolutions combined with dilated causal convolutions. Dilated 1D convolutions enable the model to handle long-range temporal dependencies efficiently by exploiting an exponentially growing receptive field without increasing computational demand. Moreover, the causal aspect of these convolutions ensures that the prediction for the current audio sample does not use future data allowing real-time audio processing. Additionally, the dilation allows the model to synthesize audio that maintains coherent structures over extended time periods, far surpassing the capabilities of previous audio processing models.

1.6.2 Recurrent Neural Networks

The Recurrent Neural Network (RNN) is an architecture developed mainly to process time series. They differ from MLP and CNN because information doesn't flow in a single direction. As shown in Fig 1.13, panel A, the network can be *unrolled through time*. In this representation, for each time step t , the network receives the input $\mathbf{x}(t)$ and the output generated at the previous time step $\mathbf{y}(t-1)$. Since the output of time step t is a function of all previous time steps, this type of network holds some *memory*. In RNNs, each unit has two sets of weights, one for the input $\mathbf{x}(t)$ and one for the output of the previous time step $\mathbf{y}(t-1)$. These single-unit weights can be combined leading to the matrices $\mathbf{W}_x$ and $\mathbf{W}_y$. With these two quantities, the output can be computed as

$$\mathbf{y}(t) = \sigma \left(\mathbf{W}_x^T \mathbf{x}(t) + \mathbf{W}_y^T \mathbf{y}(t-1) + \mathbf{b} \right) \quad (1.12)$$

where $\mathbf{b}$ is the bias term and σ the activation function. The training procedure for RNNs

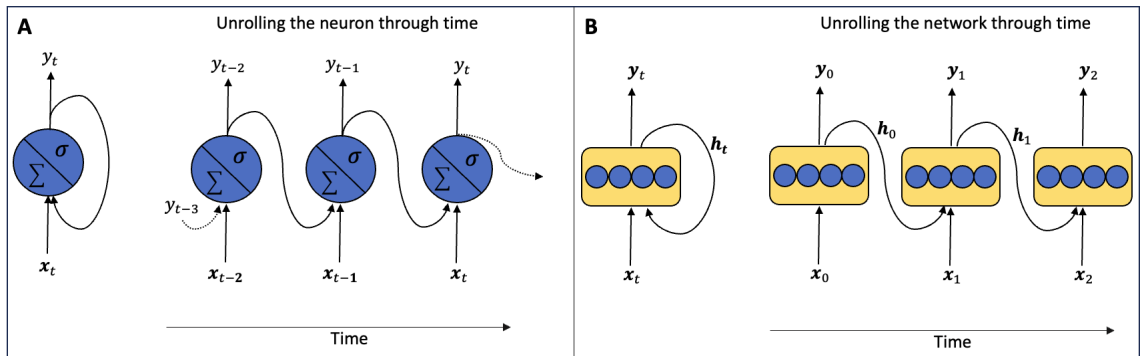


Figure 1.13: RNN architecture. Right: single RNN cell and its enrolling through time with $y_t \in \mathbb{R}$. Left: RNN layer and its enrolling through time. Note that in this case the output at each time step t is a vector.

is the same as the other deep architectures introduced before: back propagation. However, here the network is first unrolled through time and a forward pass is performed. Then, a loss is computed, differentiated and the gradients are back propagated *through time*, which is the fundamental difference when compared to the previous architectures. So far, the simplest case of RNN has been considered in which the feedback information comes from the output of the previous time step. However, more sophisticated architectures use *cell*

states ($\mathbf{h}(t)$, Fig. 1.13, panel B). This vanilla RNN suffers of two main limitations:

- handling long sequences: in this case the network unrolled through time becomes very deep and, as any deep network, this might result in unstable gradients [30–32];
- short term memory: due to the sequential processing of the data while traversing the RNN, some information gets lost at each time step. This means that after some steps the RNN state contains little to no information about the original input [33].

One way to mitigate the memory problem is the Long Short-Term Memory (LSTM) cell [34]. In Fig. 1.14, the cell state has been split in two, $\mathbf{h}_t$ and $\mathbf{c}_t$, which can be interpreted as the short and long term states of the cell. The idea behind the LSTM cell is that it can learn what information to store, what information to disregard and what information to read from the long-term state. Traversing Fig. 1.14 from left to right, $\mathbf{c}_t$ passes through a *forget gate* ($\mathbf{F}$, Fig. 1.14) which has the role of removing some information from $\mathbf{c}_t$. After that, some new information flows into $\mathbf{c}_t$ coming from the *input gate* ($\mathbf{I}$, Fig. 1.14) through the summation node. Therefore, for each time step, some memories are removed from the cell state $\mathbf{c}_t$ and some new are added. The short-term memory state $\mathbf{h}_t$ (which equals the current time step output of the cell $\mathbf{y}_t$) is calculated from $\mathbf{c}_t$, filtered through an hyperbolic tangent activation function, element-wise multiplied with the $\mathbf{o}_t$: the output of the *output gate* ($\mathbf{O}$, Fig. 1.14). The LSTM cell relies on four fully connected ($\mathbf{FC}$, Fig. 1.14) layers three of which are *gate controllers* ($\mathbf{FC}$ blue boxes, Fig. 1.14) and one is the *analyser* ($\mathbf{FC-tanh}$ block, Fig. 1.14) which have different roles

- *analyser*: this network processes $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ and extracts its most important parts ($\mathbf{g}_t$, Fig. 1.14) to be stored in $\mathbf{c}_t$ through the *input gate*;
- *gate controllers*: they rely on the sigmoid activation which outputs values in the range $[0, 1]$. Because the outputs ($\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$) of these gate are directly fed to an element-wise multiplication, if their output is 0s, they close the specific gate and if the output is 1s they open the gate: the *forget gate* controls which part of $\mathbf{c}_t$ should be erased, the *input gate* decides which parts of $\mathbf{g}_t$ should be integrated into $\mathbf{c}_t$ and the *output gate* controls which information in $\mathbf{c}_t$ should be conveyed to $\mathbf{h}_t$ and $\mathbf{y}_t$.

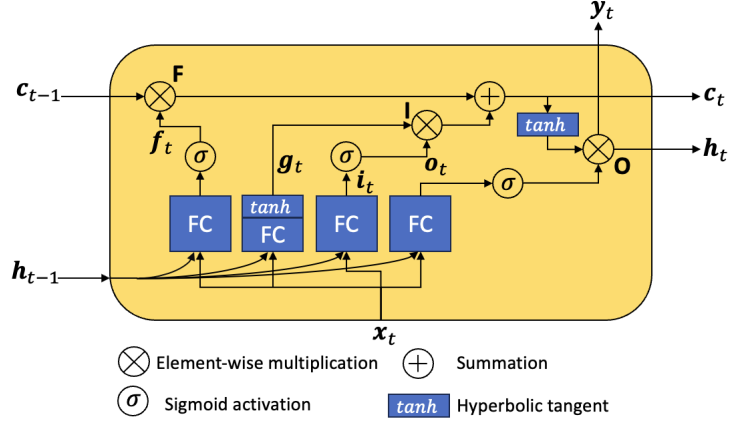


Figure 1.14: Schematic of an LSTM cell. From left to right: c_t is the long-term cell state, h_t is the short-term cell state, σ is the sigmoid activation function, i_t , o_t , f_t and g_t , are the outputs of the *gate controllers* and the *analyser*, x_t is the current input and y_t its associated output.

This particular structure allows the LSTM cell to learn important input features, store them in the long-term state, preserve it for as long as it is needed extract and remove it whenever is needed. Equation 1.13 summarizes how each of the terms in Fig. 1.14 can be calculated

$$\begin{aligned}
 i(t) &= \sigma \left(\mathbf{W}_{x_i}^T \mathbf{x}(t) + \mathbf{W}_{h_i}^T \mathbf{h}(t-1) + \mathbf{b}_i \right) \\
 f(t) &= \sigma \left(\mathbf{W}_{x_f}^T \mathbf{x}(t) + \mathbf{W}_{h_f}^T \mathbf{h}(t-1) + \mathbf{b}_f \right) \\
 o(t) &= \sigma \left(\mathbf{W}_{x_o}^T \mathbf{x}(t) + \mathbf{W}_{h_o}^T \mathbf{h}(t-1) + \mathbf{b}_o \right) \\
 g(t) &= \tanh \left(\mathbf{W}_{x_g}^T \mathbf{x}(t) + \mathbf{W}_{h_g}^T \mathbf{h}(t-1) + \mathbf{b}_g \right) \\
 c(t) &= f(t) \otimes c(t-1) + i(t) \otimes g(t) \\
 y(t) &= h(t) = o(t) \otimes \tanh(c(t))
 \end{aligned} \tag{1.13}$$

where

- $\mathbf{W}_{x_i}$, $\mathbf{W}_{x_f}$, $\mathbf{W}_{x_o}$ and $\mathbf{W}_{x_g}$ are the weight matrix of the four FC layers (Fig. 1.14, FC blocks) applied to the input $\mathbf{x}(t)$;
- $\mathbf{W}_{h_i}$, $\mathbf{W}_{h_f}$, $\mathbf{W}_{h_o}$ and $\mathbf{W}_{h_g}$ are the weight matrix of the four FC layers (Fig. 1.14, FC blocks) applied to the short-term memory $\mathbf{h}(t)$;

- $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o$ and $\mathbf{b}_g$ are the bias terms for each FC layer.

Encoder-Decoder Architecture and Attention Mechanism

Although LSTM cells greatly improve the memory capability when compared to vanilla RNN, their memory is still limited to few (~ 100) time steps. To overpass this limitation, the attention mechanism [35] was introduced for the first time in the encoder-decoder architecture (Fig. 1.15) for machine translation applications. Each word is associated with an integer index which is then passed through an embedding layer outputting the vector which will be consumed by the LSTM cells in the encoder and decoder. Fig. 1.15 shows an English sentence fed to the encoder and the decoder outputs the Italian transcription. The English sentence is input in reversed order to the encoder so that the first input to the decoder is the beginning of the sentence which is the part that needs to be translated first. At each time step, the decoder outputs a score for each word in the Italian vocabulary and a softmax layer (red box, Fig. 1.15) turn each score into a probability. At training

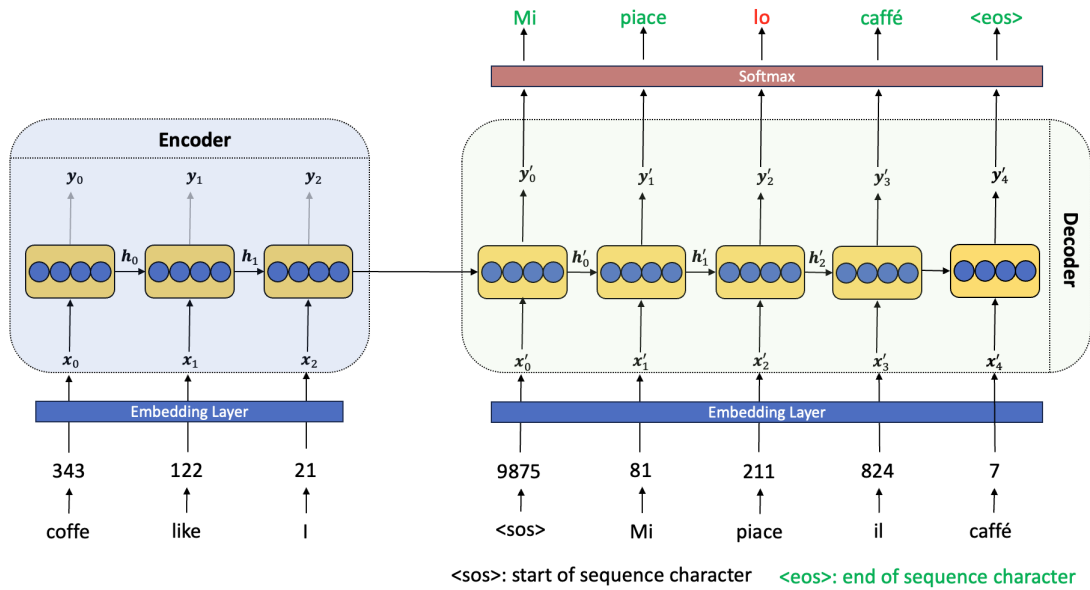


Figure 1.15: Encoder-Decoder architecture based on RNN networks. The original English word sequence is first encoded into fixed-size vectors x_t by the embedding layer. Then, the RNN-encoder consumes the encoded sequence and generates a compressed representation h_2 that is passed to the RNN-decoder. This network processes the hidden states h'_t in combination with a forward-shifted version of the target sequence and predicts, one word at the time, the translated Italian sentence.

time, the target transcription is used as input to the decoder. This is shifted one time step forward which is the word the decoder should have produced at the previous time step. This is useful because at test time we do not have the target Italian sentence therefore, the output of the decoder is fed back as input to generate the next word prediction. The main limitation of this architecture is that the information required to translate each word (the original word in English and its context) needs to be carried over many time steps before it is actually used (in Fig. 1.15 the word *caff * needs to be remembered through the whole architecture before being translated by the decoder). In order to solve this issue, the first attention mechanism was introduced [35] which allowed the decoder to focus on the appropriate words at each time step. Figure 1.16 shows the modified encoder-decoder architecture employing the attention mechanism. Compared to Fig. 1.15, here the decoder has as input all the encoder outputs and the last hidden state which are all consumed by the block A. This component computes a weighted sum of all encoder outputs and

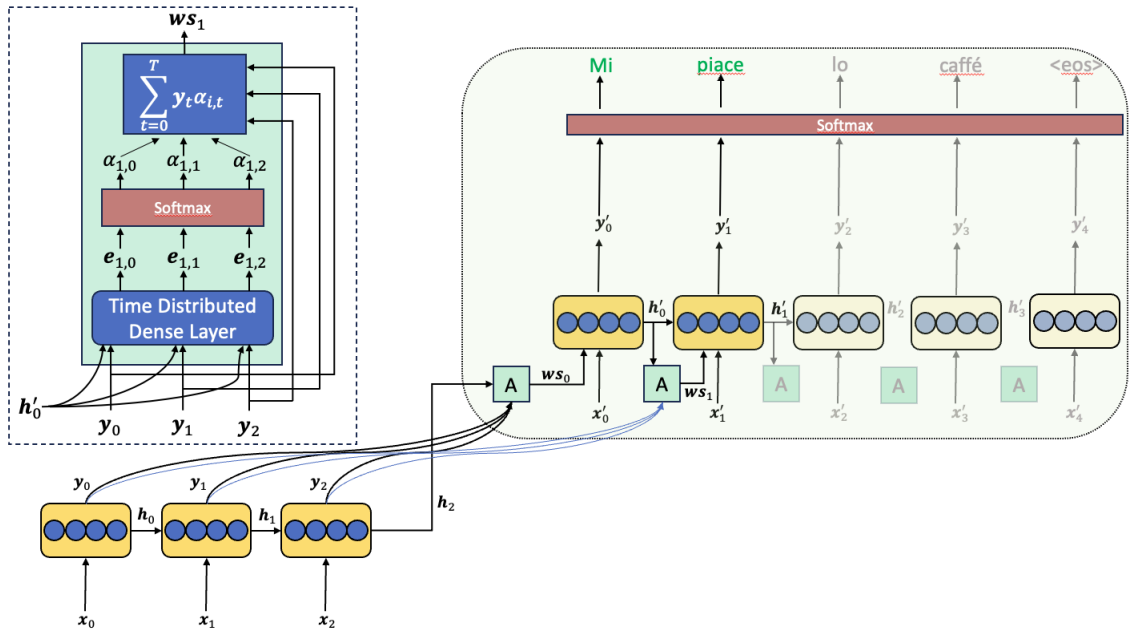


Figure 1.16: Encoder-Decoder architecture with concatenative attention [35] mechanism. Before feeding the hidden encoder state h_t to the decoder an alignment network (green box) computes alignment scores $a_{i,t}$ between each encoder output y_t and the decoder previous hidden state h'_t . This new vector ws_t contains context information useful for the translation task.

determines which word the decoder needs to focus on at each time step. The weights $\alpha_{i,t}$ for the i^{th} encoder output and t_{th} decoder time step, are learnt jointly while training the whole

encoder-decoder architecture by the *time-distributed* dense layer (also called *alignment model*) in Fig. 1.16. For each time step, the *alignment model* measures how well each encoder output is aligned to the decoder previous hidden state (scores $e_{i,j}$ terms, Fig. 1.16). Afterwards, the scores go through a softmax layer which computes the final weights for each encoder output $\alpha_{i,t}$. Going back to the example of the word *coffe*, we would expect the weight $\alpha_{0,3}$ (corresponding to the word *caff  *) to be much larger than $\alpha_{1,3}$ and $\alpha_{2,3}$ resulting in the decoder to focus more on the word *coffe* than to the other two words at time step $t = 3$. Since the encoder output is concatenated with the decoder's previous time step, this type of attention is called *concatenative attention* [35] (or *Bahdanau attention* from the name of its inventor).

1.6.3 Transformers

Recently, the advent of large language models has underscored the already recognized capabilities of the transformer architecture [36]. As visible in Fig. 1.17, this architecture doesn't rely on RNN or CNN layers but just the attention mechanism and other simple structures such as embedding, dense and normalization layers: because of this reason, the architecture is fast to train (if compared to RNN) and is easy to parallelize. The transformer architecture, similarly to the RNN, has two main components: an encoder and a decoder. The role of the encoder is to produce a richer and more context-aware input representation (word embeddings in the translation example in Section 1.6.2). The decoder takes as inputs the encoded inputs and the target sequence (shifted one time step forward, Fig. 1.15) and outputs a probability distribution for each word in the vocabulary. Before diving into the details and mechanisms of the transformer we need to introduce few more key concepts:

- Multi-Head Attention: self-attention and cross attention,
- Positional Encoding and
- Layer Normalization.

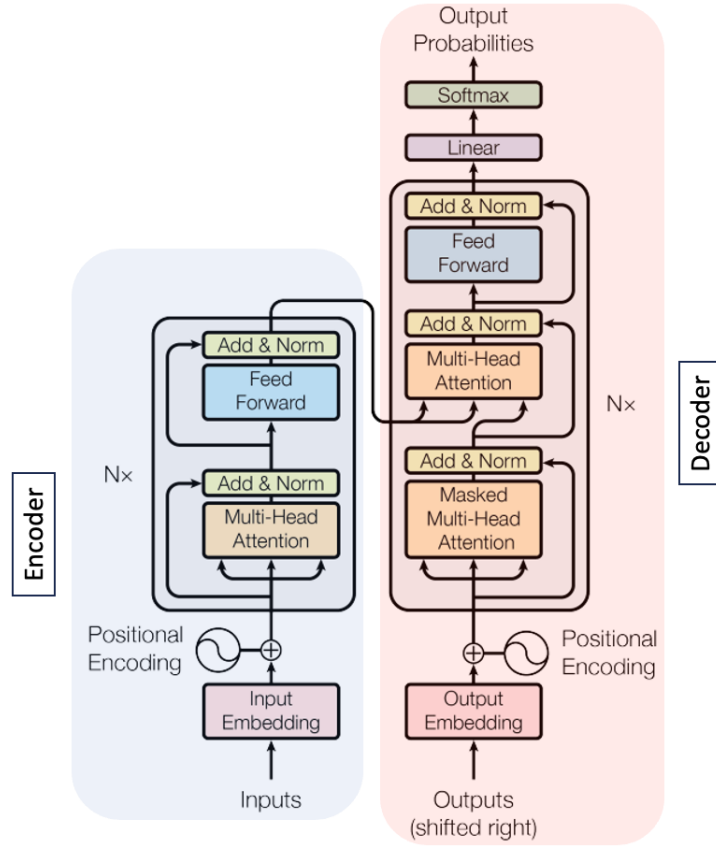


Figure 1.17: Original transformer architecture with its principal components: encoder, decoder, positional encoding and multi-head attention. Adapted from [36]

Multi-Head Attention

The multi-head attention layers in Fig. 1.17 are based on the *scaled dot-product self-attention*^[1]. Given the input sequence $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$, $\mathbf{x}_t$ is the d -dimensional word embeddings and t is the time step index. We define three trainable^[2] matrices $\mathbf{U}_q$, $\mathbf{U}_k$ (of dimension $d_k \times d$) and $\mathbf{U}_v$ (of shape $d_v \times d$ ^[3]) which are used to project each input $\mathbf{x}_t$ into *query*, *key*

¹This type of attention can be also applied to the RNN encoder-decoder architecture. In that case, instead of calculating the term $\alpha_{t,i}$ (Fig. 1.16) with the network A, the weights are obtained as the dot product between the encoder outputs and the last hidden state of the decoder.

²These matrices are usually implemented with a simple linear layer.

³Usually, d_k, d_v and d assume the same value. In [36], $d = 512$.

and *value* sequences as

$$\begin{aligned}\mathbf{q}_t &= \mathbf{U}_q \mathbf{x}_t \\ \mathbf{k}_t &= \mathbf{U}_k \mathbf{x}_t \\ \mathbf{v}_t &= \mathbf{U}_v \mathbf{x}_t\end{aligned}\tag{1.14}$$

where $\mathbf{q}_t, \mathbf{k}_t$ have shape $d_k \times 1$ and $\mathbf{v}_t$ has shape $d_v \times 1$. After projecting each input vector $\mathbf{x}_t$ to *query*, *key* and *value* vectors the un-normalized weights $\omega_{t,j}$ can be computed as the dot product between $\mathbf{q}_t$ and $\mathbf{k}_j$. Fig 1.18 exemplifies how to calculate the new context-aware embedding vector $\mathbf{z}_t$ at time step $t = 1$. In this example, the term $\omega_{1,0}$ measures the importance of $\mathbf{v}_0$ in modeling the input $\mathbf{x}_1$ at time step $t = 1$. To ensure the Euclidean length of weight vectors to be approximately the same, the terms $\omega_{t,j}$ are scaled by d_k and then normalized using the softmax function

$$a_{t,j} = \frac{\exp(\omega_{t,j}/d_k)}{\sum_{i=0}^T \exp(\omega_{t,i}/d_k)}.\tag{1.15}$$

Finally, the context vector

$$\mathbf{z}_t = \sum_{i=0}^T a_{t,i} \mathbf{v}_i\tag{1.16}$$

is the average of the *value* vectors weighted by the terms $a_{t,j}$. When compared to the input embeddings, $\mathbf{z}_t$ is context-aware meaning it incorporates knowledge coming from the whole input sentence⁴. In the case of multi-head attention, we can think about $\mathbf{U}_q, \mathbf{U}_k$ and $\mathbf{U}_v$ as one *attention head*. Intuitively, we now have a set of *query*, *key* and *value* matrices similar to the case of CNN in which we can have multiple kernels. Considering six attention heads⁵, we need to compute a set of $h = 6$ *query*, *key* and *value* sequences (one for each attention head)

$$\begin{aligned}\mathbf{q}_t^m &= \mathbf{U}_{q_m} \mathbf{x}_t \\ \mathbf{k}_t^m &= \mathbf{U}_{k_m} \mathbf{x}_t \\ \mathbf{v}_t^m &= \mathbf{U}_{v_m} \mathbf{x}_t\end{aligned}\tag{1.17}$$

⁴This is also the reason why this mechanism is called *self-attention*: the input sequence attends itself.

⁵The number of heads $h = 6$ was the choice the authors made in the original "Attention is all you need" paper [36].

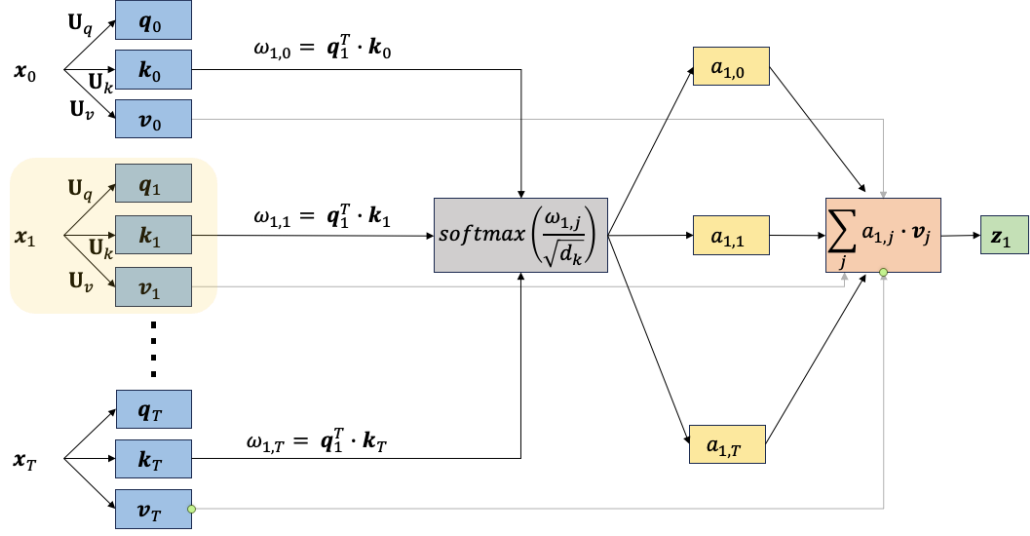


Figure 1.18: Scaled dot-product self-attention example: calculation of the context vector z_t at time step $t = 1$.

where $m \in [1, \dots, h]$. Figure 1.19 shows an example with three attention heads: by

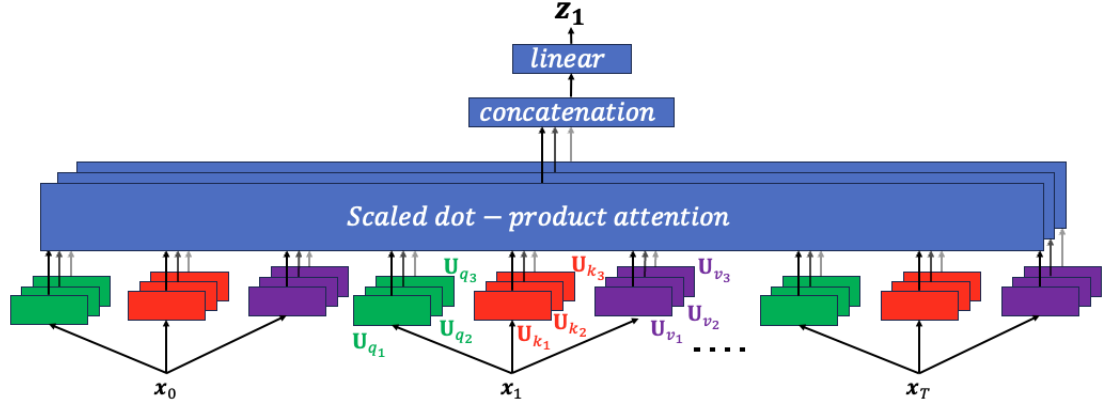


Figure 1.19: Multi-head attention example with $h = 3$ attention heads. Example for computing the context vector z_t at time step $t = 1$.

considering $d_q = d_k = d_v = d$, each matrix $\mathbf{U}$ has shape $(3, d, d)$ and, after the scaled dot-product the context vector has shape $(d, t, 3)$. To restore the original input shape (d, t) , the intermediate context vector are first concatenated and then a linear layer projects the output of the concatenation layer to the original input dimension. The multi-head attention mechanism described above, is used in the encoder module and the upper part of the decoder (Fig. 1.19). In the lower decoder network, the multi-head attention needs to be *masked*. This is necessary due to the fact that, at inference time, the decoder doesn't have

access to future time steps but only the current and past inputs. This is done by summing an upper triangular matrix⁶ to the scaled dot product tensor (the tensor composed by $\omega_{t,j}$ terms). In the upper decoder network, the multi-head attention receives as input the output of the masked multi-head attention and the output of the encoder network (Fig. 1.17). In this case, the attention mechanism takes the name of *cross-attention*. The fundamental difference when compared to the *self-attention* is that here $\mathbf{v}_t^m$ and $\mathbf{k}_t^m$ are generated from the context-vectors generated by the encoder and $\mathbf{q}_t^m$ is computed from the output of the masked multi-head attention from the decoder. Intuitively, the cross-attention weights incorporate the knowledge of which entries of the input sentence are most useful in decoding the target sequence.

Positional Encoding

Due to the fact that both multi-head attention and dense layers are permutation-invariant, a tool for incorporating the order of the words appearing in the input sequence is needed. For doing this, the *positional encoding* was introduced [36]. These are vectors with small values that are added to the word embeddings at the beginning of the encoder and decoder blocks and allow the transformer architecture to produce slightly different encodings for the same word in different positions in the sequence. In the the original architecture [36], *sinusoidal* positional encodings were employed:

$$\begin{aligned} PE_{i,2k} &= \sin(i/10^{8k/d_{model}}) \\ PE_{i,2k+1} &= \cos(i/10^{8k/d_{model}}) \end{aligned} \tag{1.18}$$

where i is the position of the word in the sentence, k denotes the length of the encoding vector⁷ and d_{model} is the dimension of the word embedding. The positional embeddings can also be learnt by the model [36] and can be implemented as *absolute* (as above) or *relative* encodings which model the relative position of words in the sentence.

⁶This is a matrix with *-inf* values in all the elements above the main diagonal and zeros elsewhere. The choice of this particular matrix comes from the fact that, by applying the softmax operation to the scaled dot-product tensor, we will force all *-inf* values to zero effectively suppressing all the future time steps for each input x_t .

⁷ k has the same dimension as the input word embedding so that the positional encoding and the word embedding can be added together.

Skip Connections and Layer Normalization

Covariate shift refers to the situation where the distribution of the input features (covariates) to a model changes between the training and testing phases⁸. This shift in distribution can be problematic because machine learning models often assume that the training and testing data come from the same distribution. When this assumption is violated, the model's performance may degrade, as it has not seen the variations present in the testing data during training. Batch normalization [37] is one technique that aims to address the issue of covariate shift. By normalizing the inputs to a layer across batches during training, batch normalization helps maintain a more consistent distribution of inputs, reducing internal covariate shift. This, in turn, can lead to improved training stability and faster convergence. However, being the normalization statistics calculated along the batch dimension, batch normalization is not ideal in the context of transformer models. In fact, transformers are usually trained on large corpora with parallel computing and this can be challenging to achieve with batch normalization which has a dependency across training examples. Layer normalization [38] is another important component of the transformer architecture. Layer normalization normalizes the inputs of a layer by computing the mean and standard deviation across all features for a single data point: this means that the main difference between batch and layer normalization is that the former normalizes along the batch dimension, while in the latter the normalization is performed along the feature dimension. Given an input vector $(x_1, x_2, \dots, x_n)$ for a single example, layer normalization independently normalizes each feature x_i based on its mean μ_i and standard deviation σ_i :

$$\hat{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

where ϵ is a small constant added for numerical stability. The normalized output $\hat{x}$ is then scaled and shifted using learnable parameters γ (scale) and β (shift):

$$y_i = \gamma \hat{x}_i + \beta$$

⁸In other words, the statistical properties of the input data may differ between the training and inference (testing) sets.

where γ and β are learnable parameters for each feature. Layer normalization is particularly useful in scenarios where the statistics of different features may vary, and it helps mitigate the vanishing/exploding gradient problems. Another important component of the transformer architecture are skip connections. These also help mitigate the vanishing/exploding gradient problems. Moreover, by maintaining a more direct and efficient flow of information from the input to the output, they also facilitate the training of deep networks. Finally, due to the fact that the output of a transformer block is computed as the sum of the input and the transformed (non-linear) output of the block, this formulation encourages the model to focus on learning the residual between the input and the desired output.

1.7 Automatic Speech Recognition

ASR is the task of converting human speech into text and this field holds significant importance for enhancing human-to-machine interaction. Initial approaches focused on methods such as Gaussian Mixture Model (GMM) combined with Hidden Markov Model (HMM) [39] and Dynamic Time Warping (DTW) [40]. In more recent times, there has been a shift towards the application of neural networks, including RNN, CNN, and Transformers, demonstrating impressive performance improvements. The general sequence of ASR can be seen in Fig. 1.20. The primary objective of an ASR system is to convert an audio input signal $\mathbf{x} = (x_1, x_2, \dots, x_T)$ of a specific length T into a sequence of words or characters (labels) $\mathbf{y} = (y_1, y_2, \dots, y_N)$, where $y_n \in V$ where V is the lexicon: the complete set of words and their pronunciations that the ASR system can recognize. It often includes details like phonetic representations and may also contain information about word forms, such as inflections or variations. Labels can be represented at various granularities, including word-level, character-level (individual letters), or sub-word units [41]. A word-level representation constrains the system to generate only the words observed during training, reducing ASR effectiveness in handling rare or out-of-vocabulary words. Character-level modeling addresses the limitations of word-level representations; however, it tends to generate long sequences, complicating the model's ability to learn and capture long-term dependencies effectively. Sub-word units provide an intermediate solution between

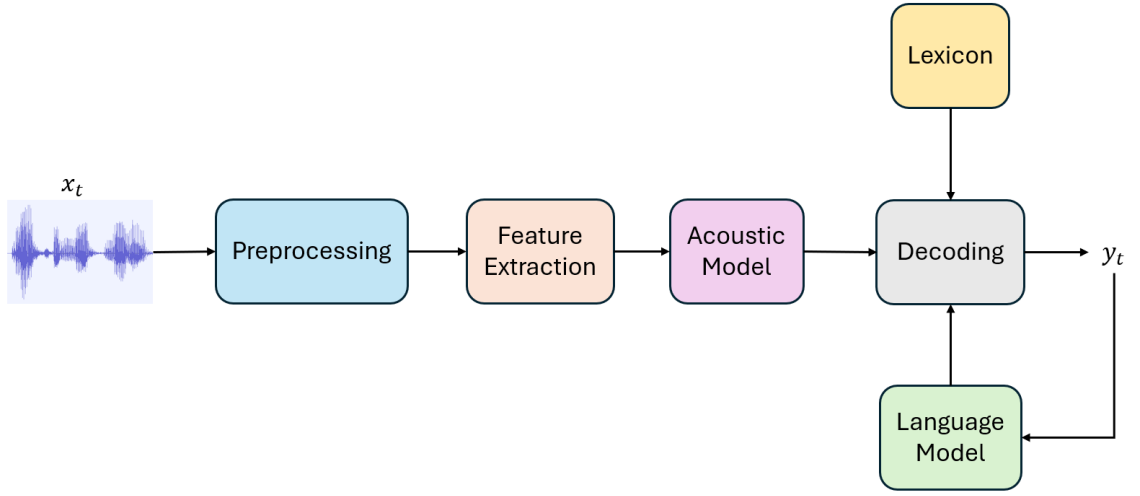


Figure 1.20: Basic ASR Pipeline. The input audio x_t is first preprocessed to enhance signal quality. The resulting features are then passed into the acoustic model. In the decoding step, the acoustic model, lexicon, and an optional LM are combined to predict the output label y_t .

word-level and character-level modeling, offering a balanced approach. Furthermore, sub-word level representations can be effectively learned from large text corpora [41], [42]. Given the audio input $\mathbf{x}$, the most probable output sequence is determined by:

$$\hat{\mathbf{y}} = \arg \max_{y_i \in V} p(\mathbf{y} | \mathbf{x}).$$

The performance of the ASR model is evaluated using the WER metric which is calculated from the ASR output transcription as

$$\text{WER} = \frac{N_{\text{sub}} + N_{\text{ins}} + N_{\text{del}}}{N_{\text{tok}}} \%$$

where N_{sub} , N_{ins} , N_{del} are the number of substitutions, insertions and deletions in the ASR output and N_{tok} is the number of tokens in the reference transcript. A typical ASR system (Fig. 1.20) undergoes the following processing steps: pre-processing, feature extraction, decoding (or classification) and, optionally, language modeling. The pre-processing step aims to enhance the audio signal. This step might contain

- voice activity detection to remove non-speech frames and to reduce the computational cost of ASR [43],

- noise removal to enhance the robustness of the speech recognition system [44],
- pre-emphasis to flatten the magnitude spectrum and equalize the balance between high and low-frequency components, since the frequency components of speech signals are predominantly in the higher frequency range and
- normalization, to reduce the impact of noise, minimize speech signal distortion, and mitigate channel distortions [45].

Next, after the input audio has been preprocessed, the signal is fed into the feature extraction stage. Here, the audio is first divided into frames of 10-30ms and then, for each frame, a certain number of features are extracted. Popular features employed in ASR are MFCCs (Section 1.4) and Discrete Wavelet Transform (DWT) [46]. Moreover, recent work showed it is possible to obtain effective speech representations using self-supervised learning [47]. The primary goal of the classification block is to identify the spoken text present in the input signal. It utilizes the features extracted during the feature extraction stage to produce the output text. Finally, the classification stage can be complemented with a pretrained LM

$$\hat{\mathbf{y}} = \arg \max_{y_i \in V} p(\mathbf{y} \mid \mathbf{x}) \cdot p(\mathbf{y})^\alpha$$

where the term $p(\mathbf{y})^\alpha$ captures both the grammatical and semantic information inherent in the language. LMs play an important role in refining the output tokens from the decoding stage by making necessary corrections to the generated text. This is particularly important in low-resource scenarios where only a limited amount of training audio is available [48].

The recent advancements in speech recognition can be primarily attributed to the progress in neural network architectures. Various approaches, such as CNNs and RNNs, have been utilized [49], [50]; however, transformer networks have demonstrated exceptional performance in this domain. As mentioned in Section 1.6.2, one drawback of RNNs is that, for each time step the next output token, is generated based only on the left context of the input sequence. However, in speech recognition, usually the information of the right context (future tokens in the sequence) is equally significant as the past context [51]. This issue can be partially overcome by bidirectional RNNs (BiRNNs) where the

input vector is processed in both directions, forward and backward, retaining the hidden state vectors for each direction. Another shortcoming of RNNs, is that they typically require an audio-word sequence alignment [52]. One of the first solutions to tackle this problem were the HMMs [53] which are able to compute alignment between the input audio and its transcribed output [54]. However, more recent approaches employ Connectionist Temporal Classification (CTC) [55]. A key advantage of CTC is its independence from prior data segmentation or alignment. CTC operates as an objective function evaluating the alignment between the input speech signal and the corresponding output word sequence with the aim of maximising the total probability of correct alignments. CTC introduces a blank label, representing silent time-steps (when the speaker is not speaking) or transitions between tokens. Given the input sequence $\mathbf{x}$ and the output probability sequence of words $\mathbf{y}$, the probability of an alignment path α is computed as:

$$P(\alpha \mid \mathbf{x}) = \prod_{t=1}^T P(\alpha_t \mid \mathbf{x})$$

where, α_t represents a single alignment at time-step t . For a given transcription sequence, multiple possible alignments exist, as labels can be separated from blanks in different ways. For instance, the alignments $(a, -, b, c, -, -)$ and $(-, -, -, -, a, -, b, c)$ both correspond to the character sequence (a, b, c) , where $-$ denotes the blank symbol. Ultimately, the total probability of all paths is determined by summing the individual path probabilities

$$P(\mathbf{y} \mid \mathbf{x}) = \sum_{\alpha} P(\alpha \mid \mathbf{x}).$$

In order to predict the most appropriate word sequence, different decoding algorithms can be used, with the best-path decoding algorithm being the most prevalent. In this algorithm, maximum probabilities are utilized at each time-step. Beam search [56] is another method employed for CTC decoding. The search for the most likely translation occurs from left-to-right time-steps, maintaining a limited number B of partial hypotheses. These hypotheses represent incomplete versions of the output sentence, or prefixes. As the model progresses through each time step, it expands each hypothesis by considering every possible word from the vocabulary as the next word in the sequence. Since exploring every

possible option is computationally expensive, the beam search algorithm maintains only the top B hypotheses at each step which are the likely under the model's probabilities. The process repeats until the model reaches the end of the sentence, selecting the most likely full translation from the set of completed hypotheses.

Furthermore, recent works focused on combining CTC and sequence modeling neural architectures such as RNNs: an architecture usually referred to as RNN-Transducer [52]. In this paradigm, the encoder network transforms the acoustic feature x_t at time-step t into a representation $h_e = f_{enc}(x_t)$. Additionally, a prediction network takes the previous label y_{k-1} and produces a new representation $h_p = f_p(y_{k-1})$. Encoder and prediction networks are RNN-based modules while the joint network is a fully-connected layer that combines these two representations to generate the posterior probability $P(y_k|t, k) = f_{joint}(h_e : h_p)$. This enables the RNN-Transducer to generate subsequent words by effectively fusing acoustic and language modeling information from both the encoder and the prediction network, leading to superior performance when compared to sequence modeling with CTC [51]. Another approach combines the attention mechanism with the encoder-decoder architecture of the RNN (Section 1.6.2), eliminating the need for pre-segmented data alignment. By using attention, the model can implicitly learn the soft alignment between the input and output sequences determining which temporal locations in the input sequence should be used to update the RNN's hidden state and predict the next output value [57]. The key distinction between attention-based and CTC-based ASR models lies in their token generation approach: attention-based models generate tokens in an autoregressive manner, meaning each token depends on the previously generated tokens, while CTC models generate tokens independently, without following a sequential dependency. Finally, it is also possible to combine CTC, attention and a LM in a single decoder [50]: in Fig. 1.21, a CNN/BLSTM encoder network processes the input audio sequence $\mathbf{x}$ and produces the shared hidden sequence $\mathbf{h}$ that is utilized by the joint decoder modules. The decoder network iteratively generates the label sequence $\mathbf{y}$ based on the hidden sequence $\mathbf{h}$ from the BLSTM. The joint decoder incorporates elements of CTC, attention, and language modeling to enhance the alignment between the input and output and improve the output sequence. The network is trained to maximize the joint posterior probabilities p_{ctc} and

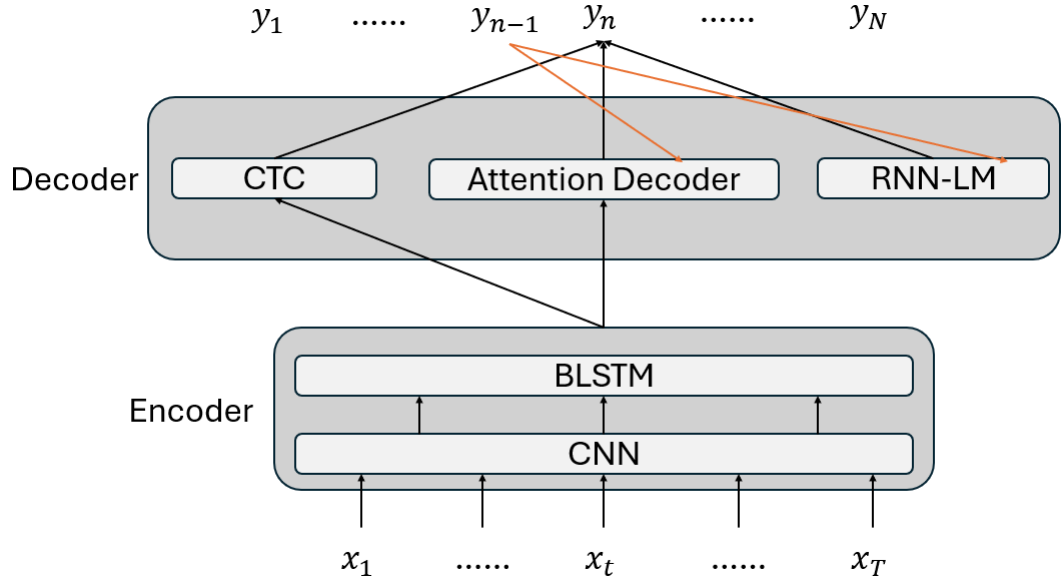


Figure 1.21: ASR model combining CTC, attention and language modeling blocks. The orange arrows indicate that the previous label y_{n-1} is recursively fed into both the attention-based decoder and the LM.

p_{att} :

$$L = \lambda \cdot \log p_{\text{ctc}}(\mathbf{y}|\mathbf{x}) + (1 - \lambda) \cdot \log p_{\text{att}}(\mathbf{y}|\mathbf{x})$$

with the tunable parameter $\lambda \in [0, 1]$. During inference, to determine the most probable word sequence

$$\hat{\mathbf{y}} = \arg \max \left(\lambda \cdot \log p_{\text{ctc}}(\mathbf{y}|\mathbf{x}) + (1 - \lambda) \cdot \log p_{\text{att}}(\mathbf{y}|\mathbf{x}) + \gamma \cdot \log p_{\text{lm}}(\mathbf{y}) \right)$$

the LM probability $p_{\text{lm}}(\mathbf{y})$ is also incorporated into the calculation.

It is worth noting that CNNs have also been employed as building blocks in ASR encoders. However, while CNNs used for image processing share the same set of weights across all parts of the input, speech processing employs a different approach. In speech processing, only the convolutional filters connected to the same pooling region share weights [58]. The reason for this difference is that, in images, the same visual patterns or features can appear in various regions, and weight sharing allows the network to learn these shared features across the entire image. In contrast, speech signals vary significantly across different frequency ranges therefore, limited weight sharing is applied. Other

CNN-based encoder architectures employ residual connections [59], separable convolutions [60], 1D-convolutions [61] and fully convolutional ASR networks [62]. ContextNet [63], is the CNN-RNN-transducer architecture bridging the gap in performance between CNN and RNN based models. ContextNet features a fully convolutional encoder that incorporates global context into its layers using Squeeze-and-Excitation (SE) [64] modules with a relatively small number of parameters (approximately 10M). This network comprises K layers, with the feature generation process described by

$$\mathbf{h} = C_K(C_{K-1}(\dots(C_1(\mathbf{x}))))$$

where, C_k denotes a convolutional block that is sequentially followed by batch normalization [37] and activation function. Additionally, after every C_k , an SE block calculates a global channel-wise weight, θ , through a global average pooling layer. This weight is then applied to the input $\mathbf{x}$ resulting in a modified output that encapsulates global sequence information. This approach addresses the limitation of CNNs, where the receptive field is constrained by the kernel size, by incorporating broader contextual information. Furthermore, the advent of transformer networks (Section 1.6.3) has brought notable enhancements in the fields of speech recognition. Transformer models for speech recognition typically follow the encoder-decoder architecture, as in more traditional sequence-to-sequence models (Section 1.6.2). More specifically, these models rely on the self-attention mechanism (Section 1.6.3) as opposed to the recurrence employed by RNNs. The Speech-Transformer [65] employs the encoder-decoder architecture and, similarly to the architectures described above, the encoder transforms a sequence of acoustic features $\mathbf{x}$ into a hidden representation $\mathbf{h}$ fed to the decoder sequentially generating an output sequence $\mathbf{y}$ autoregressively by incorporating previously emitted characters as additional inputs at each step.

However, as a non-recurrent sequence to sequence model, the Speech-Transformer distinguishes itself from recurrent models primarily in two key aspects: firstly, both the encoder and decoder consist of multi-head attention and position-wise feed-forward networks, secondly, the encoder's outputs $\mathbf{h}$ is attended to by each decoder block, replacing the single-step intermediary attention mechanism found in recurrent models. Similar

ASR model	Dataset	PER[%]	WER	Reference
RNN+CTC	[66]	18.4	X	[51]
RNN-Transducer	[66]	17.7	X	[51]
RNN+Attention	[66]	17.6	X	[57]
LAS	Google Voices	X	10.3	[67]
RNN+CTC+Att.	[68]	X	5.3	[50]
Fully-CNN ASR	[68]	X	3.26	[62]
Contexnet	[68]	X	2.1	[63]
Speech-Transformer	[69]	X	10.92	[65]
Transformer-Transducer	[68]	X	2.0	[70]
Conformer	[68]	X	1.9	[71]

Table 1.1: ASR decoding results for the models introduced in Section 1.7

to the RNN-Transducer, a Transformer-Transducer model [70] has also been applied for speech recognition. This model combines the outputs of the Audio Encoder (AE) at time-step t_i and the previously predicted label sequence y_{i-1} , which is generated by a feedforward network and a softmax layer represented after the transformer-based Label Encoder (LE). The Conformer [71] is a variation of the original transformer with attention ASR, designed to incorporate both CNNs and transformers, effectively modeling local and global speech dependencies with greater efficiency and fewer parameters. Table 1.1 contains ASR decoding results for the models presented in this Section.

1.8 Automatic Speaker Verification

Speaker verification is a biometric task that involves addressing whether a claimed identity matches the speaker based on their voice characteristics. This technology operates on the premise that each individual’s voice is unique, due to distinctive physiological and behavioral traits that manifest in their speech patterns. It plays a crucial role in various applications, ranging from security systems to access control and from forensic analysis to customer authentication in banking and telecommunication services. This Section provides an introduction to the speaker verification task and explores the models used for speaker verification in the rest of the manuscript. Speaker recognition can be divided into *closed-set* and *open-set* verification. In *closed-set* speaker verification, the system is tasked with verifying whether a voice sample belongs to one of the known speakers in a predefined set.

Here, the universe of possible speakers is limited to those already known and enrolled in the system. This scenario implies that every time an unknown voice sample is presented, the system assumes it must match one of the enrolled profiles, Fig 1.22. *Closed-set* verification is typically used in more controlled environments where the number of speakers is limited and well-defined, such as in access control systems within a company [72]. *Open-set* speaker verification, on the other hand, addresses a more challenging scenario. It not only determines whether the voice sample matches an enrolled speaker but also has to contend with the possibility that the sample may not belong to any of the known speakers at all. In other words, the system must distinguish between known and unknown speakers, adding an extra layer of complexity. We can further divide speaker verification in *text-dependent* and *text-independent*. In *text-dependent* speaker verification, the speaker is required to utter a specific phrase or set of words during both the enrollment and verification phases. This approach relies on comparing the speaker's voice characteristics disentangling acoustic

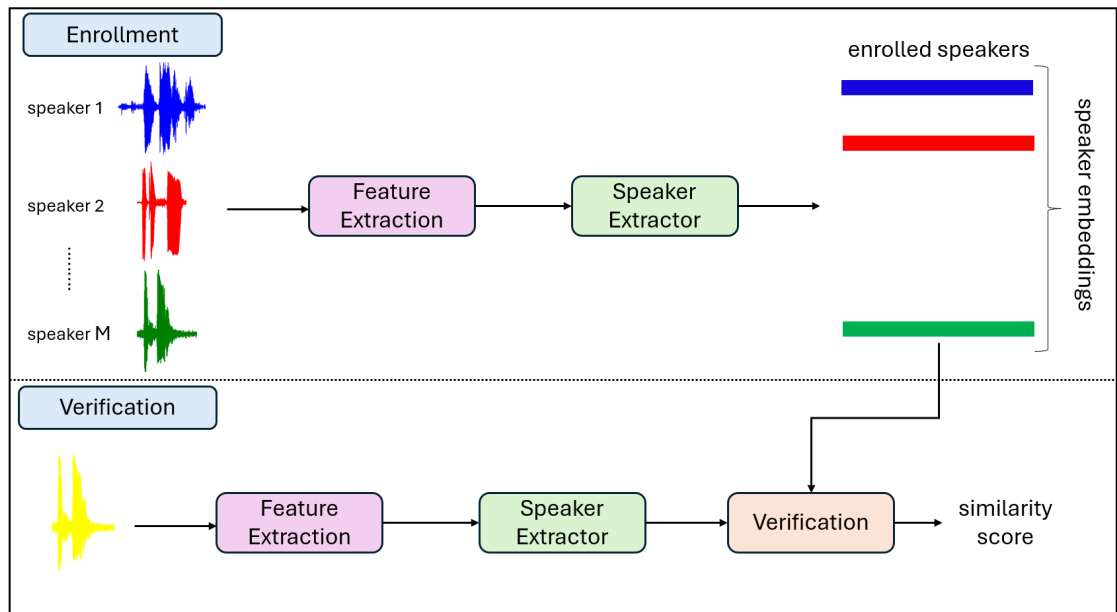


Figure 1.22: Basic speaker verification pipeline with a *closed-set* example. During the enrollment phase, each utterance is processed through a feature extraction module. Subsequently, a speaker extractor generates a unique speaker embedding, which serves as a voice fingerprint for each speaker. In the verification phase, when a new utterance is introduced, its embedding is extracted, and a similarity score is calculated. If the score exceeds a predefined threshold, the system attributes the new utterance to the corresponding enrolled speaker.

features and linguistic content. The main advantage of *text-dependent* verification is its

potential for higher accuracy [73], as the system can analyze both the speaker’s unique vocal features and the way they pronounce the specific phrase. It is commonly used in situations where high security is necessary, and the inconvenience of repeating a specific passphrase is not a significant impediment, such as secure access controls. *Text-independent* speaker verification does not require the speaker to use the same words during enrollment and verification. The system analyzes the speaker’s voice on more general characteristics, such as tone, pitch, and speech patterns, allowing for a more flexible verification process that can accommodate any spoken content. This approach is particularly advantageous in scenarios where it’s impractical to ask for specific phrases to be repeated, such as in forensic applications or when verifying a speaker’s identity in natural conversation over customer service calls. As of the above definitions, the rest of the Section focuses on a *text-independent* speaker verification model in an *open-set* scenario. Speaker verification systems consist of three primary components, illustrated in Fig. 1.22. Initially, the speech signal is converted to a sequence of acoustic features. Subsequently, a speaker embedding extractor transforms the variable length sequence of multiple acoustic feature frames into a fixed vector referred to as a speaker embedding. These embeddings constitute high-level speaker representations, encoding speaker discriminative information while discarding irrelevant incidental details. Finally, the speaker embeddings undergo comparison to yield a score indicating the similarity between two speakers. Based on the comparison score, a binary decision is made to affirm or negate the correspondence. The subsequent Sections expound on speaker embedding extractor, scoring functions, and metrics employed to evaluate ASV system performance.

1.8.1 Speaker Embedding Extractors

Speaker verification models have evolved significantly in the last few years, reflecting advancements in signal processing, machine learning, and computational capabilities. Initial techniques focused on simple acoustic feature matching, using metrics like pitch [74] and formant frequencies [75]. Afterwards, Support Vector Machine (SVM) [76] and especially GMM Universal Background Model (GMM-UBM) became prevalent for modeling voice

characteristics. GMMs provided a way to represent the distribution of spectral features of a speaker’s voice, leading to improved verification performance [77]. However, the primary challenge associated with GMM-based systems is inter-session variability [78]. This phenomenon refers to the variation in speech characteristics exhibited by the same speaker across different recording sessions. These variations can arise due to factors such as changes in speaking style, emotional state, recording environment, or recording equipment. One major breakthrough mitigating this phenomenon were i-vectors [79]. The i-vectors are used to capture variations in acoustic features attributed to differences in speakers and channels. The objective is to identify discrepancies between the features of a specific utterance and a general set of utterances to isolate speaker or channel-specific characteristics. The underlying approach involves initially employing a GMM-UBM to derive fixed-length embeddings from variable-length sequences of acoustic features, ensuring that these embeddings are independent of both speaker and session. Subsequently, the UBM is adapted to the characteristics of a specific speaker, followed by the application of intersession compensation techniques to mitigate session variability. For the first step, a large corpus of unlabeled utterances is employed to compute features, and these feature vectors are assumed to be independently and identically distributed under the GMM with a predetermined number of components. The Expectation-Maximization (EM) algorithm [80] is then used to derive the parameters of this GMM, forming what is referred to as the UBM. Next, a speaker and session-independent supervector m is constructed by concatenating the means of all UBM components. Mathematically, a speaker utterance is modelled as

$$M = m + Tw$$

where T represents a low-rank total variability matrix, and w is the i-vector. The matrix T can be estimated in the same way as eigenvoices are computed [81] whereas w is a hidden variable that can be computed by a Maximum A Posteriori (MAP) estimate [79]. Finally, channel and session compensation can be achieved by applying Linear Discriminant Analysis (LDA) [82] in the total variability space. While i-vectors are mathematically attractive and useful in the sense that they do not need a labeled training set, they are computationally

expensive. Training the UBM model and the parameters of the GMM is time-consuming, and since they model the total variability (i.e., speaker and channel), it is hard to tease apart the speaker-specific information. Deep neural network based speaker embeddings have become popular as a discriminatively-trained alternative to i-vectors. Given a sequence of feature vectors, a neural network is used to obtain an utterance-level representation, which is then fed into a softmax layer that estimates a distribution over a large set of speakers. The parameters of the network are trained using gradients of a classification-style loss. Two widespread neural network-based speaker verification systems are x-vectors [83] and Emphasized Channel Attention, Propagation and Aggregation (ECAPA) combined with time delayed neural networks (TDNNs) [84]. Conventionally, the x-vector neural network consists of TDNN (1-D strided CNN) which is specifically structured to handle variations in input lengths and to capture long-range dependencies. TDNN operates over varying temporal contexts through layers that apply convolutions across time and is followed by a statistics pooling layer. This layer aggregates over the whole input frame and computes its mean and standard deviation. These frame-level stats are concatenated together and passed to further linear hidden layers that operate at the sequence level and produce the final representation that is fed to the softmax for classification (Enrollment, Fig. 1.22). The ECAPA-TDNN architecture [84] is designed to focus more effectively on salient speaker-specific features. Similar to x-vectors, it processes frame-level features but enhances the model's focus on relevant features through attention mechanisms and dense connections. The attention mechanism allows the network to weigh different temporal segments differently, giving more importance to more informative parts of the speech signal. Furthermore, ECAPA-TDNN extend the original TDNN approach by addressing some of the limitations of x-vectors. First, a channel-dependent attention mechanism [35] is introduced, extending the statistics pooling layer to jointly operate across both the channel and temporal dimensions. Second, since in the original x-vector implementation the temporal context was limited to 15 frames, frame-level features are rescaled given global sequence properties computed by SE blocks [64]. Finally, x-vectors rely solely on the feature maps generated by the last frame layer to compute the pooled statistics. However, shallow features have been shown to enhance the robustness of speaker embeddings [85].

Consequently, in ECAPA-TDNN, the statistical pooling layer leverages the concatenation of feature maps from all SE blocks, rather than just the final layer, to improve performance. During training, both the x-vector and ECAPA architectures are trained in a supervised manner by predicting speaker labels and minimizing the cross-entropy loss function. At inference time, the classification layer is removed from the model, allowing for the extraction of speaker embeddings. This enables pairs of embeddings to be compared in order to assess whether they correspond to the same speaker (as shown in Fig. 1.22 Verification). To perform this comparison and make a decision, various scoring functions can be applied to the speaker embeddings.

1.8.2 Scoring Functions: Cosine Similarity and Probabilistic Linear Discriminant Analysis

After the speaker embedding extraction stage, a verification stage (Fig. 1.22) is needed in order to assess whether the two embeddings can be attributed to the same speaker. The two main verification strategies used in this manuscript are:

- cosine similarity
- probabilistic linear discriminant analysis (PLDA)

Cosine similarity is a metric used to measure how similar two vectors are, irrespective of their magnitude. In the context of speaker verification, it is particularly useful for comparing dense vector representations. This is done using the dot product of the vectors divided by the product of their norms. Given two N-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$, the cosine similarity can be computed as

$$\text{Cosine Similarity} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}.$$

Cosine similarity produces a score that ranges from -1 to 1. A score close to 1 signifies a high degree of similarity between the embeddings, suggesting that the samples likely belong to the same speaker. Conversely, a score near -1 indicates significant dissimilarity. To make

a decision, a threshold—typically determined through experimental calibration—is applied to classify whether the embeddings correspond to the same speaker.

PLDA [86] is based on LDA [82]: a statistical method extensively used for dimensionality reduction and classification. LDA focuses on finding linear combinations of features that best separate multiple classes. Specifically, LDA aims to identify features that maximize the separation between class means while minimizing the variance within each class. In a typical setting, consider a dataset with N examples, each represented as a column vector $\mathbf{x}_i$ in a d -dimensional space and labeled according to one of K distinct classes. For each class k , defined by a subset C_k of examples, the mean $\mathbf{m}_k$ is computed along with the overall mean $\mathbf{m}$ of the dataset. The effectiveness of LDA hinges on how well it computes the within-class scatter matrix $\mathbf{S}_w$ and the between-class scatter matrix $\mathbf{S}_b$

$$\mathbf{S}_w = \frac{\sum_k \sum_{i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T}{N}, \quad \mathbf{S}_b = \frac{\sum_k n_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T}{N} \quad (1.19)$$

using these means. The goal is to find a linear transformation $\mathbf{x} \mapsto \mathbf{W}^T \mathbf{x}$ that maximizes the ratio of between-class variance to within-class variance. Here, $\mathbf{W}$ is a $D \times d$ matrix, where d represents the desired number of dimensions. It can be shown that the optimal columns of $\mathbf{W}$ are the generalized eigenvectors that satisfy the equation:

$$\mathbf{S}_b \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}$$

corresponding to the d largest eigenvalues. As a consequence, $\mathbf{W}$ simultaneously diagonalizes the scatter matrices $\mathbf{W}^T \mathbf{S}_b \mathbf{W}$ and $\mathbf{W}^T \mathbf{S}_w \mathbf{W}$, meaning that LDA decorrelates the data both between and within classes. LDA projections can also be derived by maximum likelihood estimation assuming Gaussian distribution for each class [87]. The resulting model can be employed to classify examples from the classes represented in the training data, but it is not capable of handling novel, unseen classes. In that case, a different probabilistic model is necessary. PLDA is a generalization of LDA, extending it into a probabilistic framework that handles both the dimensionality reduction and classification problems with a generative modeling approach. PLDA reformulates the LDA problem by introducing a GMM where the means of the Gaussians are the class centroids $\mathbf{y}$ [87], and all classes share

a common covariance matrix Φ_w . The latent variable in PLDA is therefore $\mathbf{y}$. The key innovation in PLDA is the treatment of $\mathbf{y}$ as a random variable with its own distribution, specifically a Gaussian distribution with mean $\mathbf{m}$ and covariance Φ_b . This contrasts with traditional LDA, where class centroids are fixed points estimated from the training data. In PLDA, the class-conditional and the prior distributions are defined as

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{y}, \Phi_w)$$

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \Phi_b).$$

The complete data likelihood in PLDA is given by integrating over the latent variable $\mathbf{y}$, leveraging the Gaussian properties to simplify to closed-form expressions. The process of fitting a PLDA model involves maximizing this likelihood, which can be efficiently done using EM algorithms. An important aspect of PLDA is its ability to handle new classes not seen during training. This is achieved by the generative nature of the model, where the continuous nature of the class variable $\mathbf{y}$ allows the model to generalize beyond the specific classes on which it was trained. In practical terms, this means PLDA can be used for tasks like "one-shot learning," where a model of a new class (a new speaker model) can be built from a very small number of examples (utterances), or even a single example. Moreover, PLDA weights features according to their discriminative power. PLDA operates on a foundation that each feature of the input data can contribute differently to distinguishing between classes. During the training phase, PLDA learns to identify which features are most effective at separating classes by analyzing the variance between and within classes, as captured in the covariance matrices Φ_b and Φ_w . On the one hand, Φ_b (between-class covariance) highlights how much each feature contributes to distinguishing between different classes: features that show significant variations across different classes (i.e., have high between-class variance) are deemed more discriminative. PLDA prioritizes these features by assigning them higher weights. This means that during the transformation of the data in the PLDA model, these features are amplified relative to others. On the other, Φ_w (within-class covariance) provides insight into the variability of features within the same class: features with high variance within the same class do not help as much in distinguishing between

classes, as their variability could overlap across classes. In PLDA, these features are given lower weights during the transformation process, effectively reducing their influence on the outcome. The effectiveness of the verification stage in Fig 1.22 is measured by computing the Equal Error Rate (EER). The EER metric represents the point at which the rate of false acceptances (FAR, where an imposter is incorrectly verified as the genuine speaker) equals the rate of false rejections (FRR, where the genuine speaker is incorrectly denied). Mathematically, EER can be described by setting the FAR equal to the FRR:

$$\text{FAR}(\theta) = \text{FRR}(\theta)$$

where θ is the decision threshold. This threshold is the value at which the similarity score (calculated using PLDA or cosine similarity) between two speaker samples is compared to decide whether they match. For PLDA, the similarity score is the log-likelihood ratio of the hypotheses that the samples are from the same speaker versus from different speakers [86]. For cosine similarity, it is the cosine of the angle between two speaker vectors. These scores determine whether the scores indicate the same identity or different identities, based on θ :

- FAR is calculated by:

$$\text{FAR}(\theta) = \frac{\text{Number of imposter trials scored above } \theta}{\text{Total number of imposter trials}}$$

- FRR is calculated by:

$$\text{FRR}(\theta) = \frac{\text{Number of genuine trials scored below } \theta}{\text{Total number of genuine trials}}$$

where the goal is to find the threshold θ that balances these outcomes, minimizing both the risk of admitting an imposter and rejecting a valid user. In practice, calculating EER involves adjusting θ and computing FAR and FRR until they are equal. Finally, it is worth mentioning that log-likelihood and cosine similarity scores can benefit from a calibration stage [88] which role is to mitigate the mismatch between training and evaluation populations.

1.9 Basic Concepts and Architectures of Text-To-Speech Technology

Text To Speech (TTS) is a technology aimed at generating intelligible and natural speech from written text. Building a TTS system entails a deep understanding of languages, human speech production, and various other disciplines such as linguistics, acoustics, digital signal processing, and machine learning. The purpose of this Section is to provide an overview and basic understanding of TTS technologies, and it does not aim to be an exhaustive review of the field.

1.9.1 Introduction to Text-To-Speech

Early computer-based speech synthesis methods encompassed articulatory synthesis [89], formant synthesis [90], and concatenative synthesis [91]. Subsequently, with the advancement of statistical machine learning, Statistical Parametric Speech Synthesis (SPSS) emerged [92], [93]. More recently, in the 2010s, neural network-based speech synthesis gradually gained prominence as the dominant method, yielding significantly improved voice quality and establishing as the default method for TTS. Initially, articulatory synthesis [89] aimed to generate speech by emulating the actions of human articulators, such as the lips, tongue, and vocal tract. However, practical implementation poses significant challenges. One of the main difficulties lies in generating control parameters from specifications and finding a balance between creating a highly precise model that closely mirrors human physiology and a more practical, simplified representation that is easier to design and manage [94]. Formant synthesis [90] operates on linguist-developed rules that govern a simplified source-filter model (Fig. 1.1) to replicate the spectral properties of speech, including formant structure. The synthesis process involves an additive synthesis module and an acoustic model with adjustable parameters such as F0 and noise levels. Despite its ability to produce intelligible speech with modest computational requirements [95], formant synthesis suffers from sounding less natural and may exhibit artifacts [96]. Furthermore, crafting precise rules for synthesis presents a notable challenge [96]. Concatenative synthesis [91] constructs

speech by piecing together segments stored in a database, ranging from individual syllables to complete sentences [97]. During inference, the system selects and concatenates speech units to match the input text, generating the speech waveform. While concatenative TTS systems excel in producing highly intelligible audio with authentic timbre akin to the original voice, they necessitate extensive recording databases to encompass all possible speech unit combinations [98]. Moreover, the resulting voice may lack naturalness and emotional depth due to potential discontinuities in stress, prosody, and emotion resulting from concatenation [96]. SPSS [92], [93] emerges as a solution to the limitations encountered in concatenative TTS systems. Instead of directly synthesizing speech waveforms through concatenation, SPSS operates by first generating essential acoustic parameters [99] needed for speech production and subsequently reconstructing speech from these generated parameters using specific algorithms [100], [101]. Typically, SPSS comprises three key

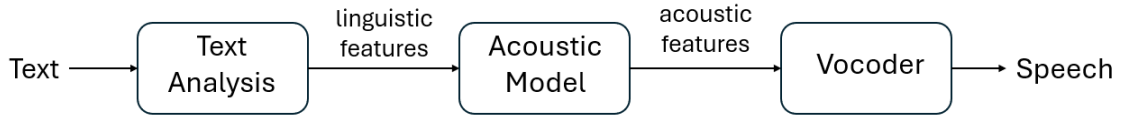


Figure 1.23: Basic components of SPSS and neural TTS systems. The text analysis module maps text into linguistic features (such as phonemes) containing pronunciation and prosody information. The acoustic model transforms the linguistic features into acoustic features such as Mel spectrograms. Finally, the vocoder synthesizes a waveform given the acoustic features.

components as shown in Fig. 1.23:

- The **text analysis module** is responsible for transforming text into linguistic features which contain information about pronunciation and prosody easing the speech synthesis task. Text analysis includes tasks like *text-normalization* [102], *grapheme-to-phoneme conversion* [103], *word segmentation* [104], *prosody* and *duration prediction*, and *part-of-speech* (POS) *tags*. *Text normalization* converts non-standard words (such as numbers "2024") into their spoken form ("Twenty twenty four") which facilitate TTS pronunciation. *Grapheme-to-phoneme* is the task of converting characters (graphemes) into pronunciations (phonemes). This is particularly important in alphabetic languages like English, where it is impossible for a lexicon (a mapping of written words to their pronunciations) to cover every word due to irregularities

in spelling and pronunciation. *Word segmentation* helps the system identify the boundaries between individual words in the input text. This is essential for accurately synthesizing speech, as each word needs to be processed separately to determine its pronunciation, intonation, and other linguistic characteristics. This becomes particularly critical for character-based languages like Chinese, where the outcome of word segmentation directly influences prosody prediction and grapheme-to-phoneme conversion. *Prosody and duration prediction* of elements like rhythm, stress, and intonation in speech are reflected in variations of syllable duration, loudness, and pitch. These variations are essential for human speech communication, influencing perception (Section 1.3). Finally, POS involves assigning a grammatical category (such as noun, verb, adjective, adverb, etc.) to each word in a given text or sentence based on its syntactic role and its relationship with other words in the sentence and affects the quality of grapheme-to-phoneme and prosody prediction [105].

- The **acoustic model** is a component that learns to map linguistic features, typically represented by phonemes, to acoustic features of speech signals such as F0, spectrum, or cepstrum, extracted through vocoder analysis [100] from speech.
- The **vocoder** module synthesizes speech from the predicted acoustic features.

SPSS offers several advantages over previous TTS systems, including enhanced naturalness, flexibility in parameter modification, and lower data requirements compared to concatenative synthesis [96]. However, SPSS also presents drawbacks such as reduced speech intelligibility due to artifacts and a perceptible robotic quality in the generated voice [96]. With the rapid advancements in neural networks and deep learning, some studies integrated deep neural networks into SPSS, including models based on DNNs [106] and RNNs [107]. However, these models still retain the SPSS paradigm of predicting acoustic features from linguistic features and just replace HMMs with neural networks. The advent of fully-neural speech synthesis, was initiated by WaveNet [29]: the first model generating a waveform directly from linguistic features. WaveNet achieves this by using an autoregressive model, where each sample is generated based on previous ones. It employs fully convolutional neural networks with dilated convolutions [108]. However, a drawback of WaveNet is its

slow inference speed due to the autoregressive nature. Meanwhile, other models continue to adhere to the three components found in statistical parametric synthesis, but enhance them by integrating neural network-based models [109]. However, the most advanced TTS technologies now include end-to-end (E2E) speech synthesis models, which generate acoustic features directly from phoneme sequences instead of relying on intermediate linguistic features [110]–[112]. Additionally, diffusion-based TTS systems [113], [114] and TTS technologies that leverage foundation models [115], [116] have also emerged as cutting-edge approaches.

1.9.2 Neural Speech Synthesis

This Section introduces the evolution of neural TTS models, highlighting the modifications applied to the three main components discussed earlier: *text analysis*, the *acoustic model*, and the *vocoder*. Additionally, it presents some mainstream E2E TTS architectures and concludes with a broad overview of the latest TTS technologies.

Text Analysis

The *text analysis* module is responsible for transforming text into linguistic features. In neural TTS, the text analysis module has been integrated into the model’s front-end. Some approaches adopt multi-task and unified front-end models [117], [118], which include all text analysis tasks within a single framework. Other methods incorporate prosodic features such as pitch [111], duration [119], phrase breaks [120], and pauses directly into the text encoder, enhancing the naturalness of synthesized speech. Lastly, additional approaches for integrating prosodic features involve utilizing reference encoders to acquire prosody representations from reference speech [121], text pre-training [122], and integrating syntax information using specialized modeling techniques such as graph networks [123].

Acoustic Model

The *acoustic model* is a component that learns to associate linguistic characteristics, often represented by phonemes, with the acoustic features which are subsequently further

converted into waveform using vocoders. The selection of acoustic features significantly influences the design of TTS pipelines. Various types of acoustic features have been experimented with, including mel-cepstral coefficients, mel-generalized coefficients, band aperiodicity, F0, voiced/unvoiced, bark-frequency cepstral coefficients [124], [125]. Nevertheless, in neural-based end-to-end TTS systems, acoustic models predominantly utilize mel-spectrograms as acoustic features. This transition to a more comprehensive representation as the target for the acoustic model became feasible due to the escalating modeling capabilities of neural networks, enabling the transformation of acoustic features from low-dimensional and compressed cepstrums to highly dimensional spectrograms. Moreover, unlike conventional acoustic models that necessitate alignments between linguistic and acoustic features, sequence-to-sequence-based neural models implicitly learn the alignments through attention [126] or predict the duration jointly [127]. One of the first neural TTS model was Tacotron [126], an RNN-based encoder-attention-decoder framework, taking characters as input and producing linear-spectrograms as output, while employing the Griffin-Lim algorithm [128] for waveform generation. Figure 1.24 illustrates the architecture of the Tacotron model. This is an autoregressive model generating one frame $\mathbf{y}_i$ at each

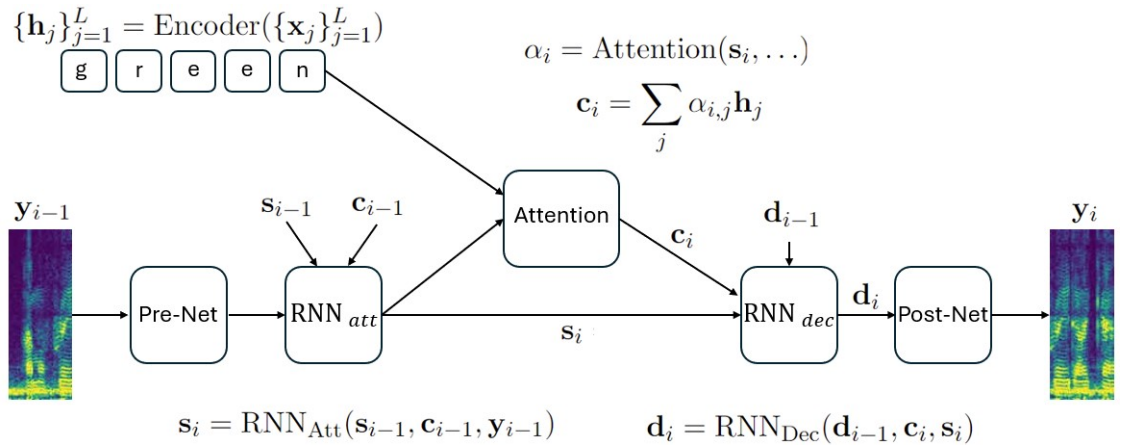


Figure 1.24: Schematic overview of the Tacotron model [126].

time step i relying on the information in the previous frame $\mathbf{y}_{i-1}$ and

$$\{\mathbf{h}_j\}_{j=1}^L = \text{Encoder}(\{\mathbf{x}_j\}_{j=1}^L)$$

which denotes the current character ($\mathbf{x}_j$) representation. The architecture comprises an encoder, attention-based decoder, and a post-processing network, which together generate a spectrogram from text input. The encoder transforms character sequences $\mathbf{x}_i$ into robust representations $\mathbf{h}_i$ relying on the CBHG module. This module serves as a crucial component in both the encoder and decoder parts of the model and combines convolutional filters, highway networks [129], and bidirectional Gated Recurrent Unit (GRU) layers for sequence representation. This architecture is specifically designed to aid in the model's training convergence and enhance its generalization capabilities [126]. The decoder uses the attention mechanisms [35] and GRUs to generate a linear spectrogram. First, the previous frame $\mathbf{y}_{i-1}$ is projected into a latent representation

$$\mathbf{s}_i = \text{RNN}_{\text{Att}}(\mathbf{s}_{i-1}, \mathbf{c}_{i-1}, \mathbf{y}_{i-1})$$

where

$$\mathbf{c}_i = \sum_j \alpha_{i,j} \mathbf{h}_j$$

denotes the context vector and

$$\alpha_{i,j} = \text{Attention}(\mathbf{s}_i, \mathbf{h}_j)$$

the attention weights computed by concatenating the embedded input sequence $\mathbf{h}_j$ and the encoded spectrogram frame $\mathbf{s}_i$ (Bahdanau attention, Section 1.6.2). Second, the RNN decoder generates the current frame representation

$$\mathbf{d}_i = \text{RNN}_{\text{Dec}}(\mathbf{d}_{i-1}, \mathbf{c}_i, \mathbf{s}_i)$$

from the current context vector $\mathbf{c}_i$, the latent representation $\mathbf{s}_i$ and the previous time step prediction $\mathbf{d}_{i-1}$. Finally, the post-net converts the decoder's output $\mathbf{d}_i$ into a linear-frequency spectrogram $\mathbf{y}_i$ which is then converted into waveforms utilizing the Griffin-Lim algorithm [128]. However, the model has some limitations. Autoregressive models generate output sequences one element at a time, conditioning each step on the outputs of all previous steps. In the context of speech synthesis, this means Tacotron generates speech sequentially,

which can be computationally intensive and slow, especially for longer sentences. Moreover, the Griffin-Lim vocoder outputs audible artifacts [126]. Tacotron 2 [130], on the other hand, introduces a novel approach by generating mel instead of linear spectrograms and converting them into waveform using an additional WaveNet [29] model. This advancement significantly enhances voice quality compared to previous methods, including concatenative TTS, parametric TTS, and earlier neural TTS models like Tacotron. Subsequent research have further refined Tacotron in various aspects such as improving expressiveness of speech synthesis [131], [132], employing transformer-based encoder-decoder architectures [133] and non-autoregressive generation [134].

Removing the text-speech attention mechanism has also been suggested as a way of improving synthesised speech quality and robustness [119], [135]. The attention mechanism, despite improvements like Location-Sensitive Attention [136] and Dynamic Convolution Attention [137], is not immune to errors producing inconsistencies in the generated speech. The issue arises from the imprecise alignment of attention between text and mel-spectrograms in autoregressive generation based on encoder-attention-decoder architecture. This problem can be overcome by substituting the attention mechanism with a duration predictor [138]. A significant leap forward in TTS technology was achieved with FastSpeech [119], [135], which effectively addressed many of the issues encountered with attention-based and autoregressive models namely slow inference due to autoregressive generation and robustness [139], often attributed to error propagation [140] and incorrect attention alignments between text and speech during autoregressive generation. FastSpeech utilizes a feed-forward structure based on the self-attention mechanism and 1D convolutional networks. This design, referred to as Feed-Forward Transformer (FeFoT), replaces the traditional encoder-attention-decoder architecture. Each FeFoT block comprises a multi-head self-attention layer for capturing cross-position relationships and a 1D convolutional network for local feature extraction. Unlike the dense layers in the original Transformer, FastSpeech uses 1D convolutions with ReLU activation, aiming to better capture the closely related adjacent states in phoneme and mel-spectrogram sequences. The model stacks multiple FeFoT blocks on both the phoneme input and mel-spectrogram output sides. Between these blocks, a length regulator is employed to address sequence length

disparities. This component solves the length mismatch problem existing between the shorter phoneme sequences and the longer mel-spectrogram sequences. This is achieved by upsampling the phoneme sequence based on the predicted phoneme durations and aligning it with the length of the mel-spectrogram sequence. This mechanism supports dynamic control over speech speed and aspects of prosody by adjusting phoneme durations predicted by the duration predictor. This module predicts the duration of each phoneme in the input sequence and is crucial for the length regulator’s operation. It consists of a 2-layer 1D convolutional network with ReLU activation, followed by layer normalization, dropout and by a linear layer to output the predicted duration. The duration predictor is trained using ground-truth phoneme durations extracted from an autoregressive teacher TTS model.

The workflow from input text to waveform generation involves several stages. Initially, the input text is converted into a sequence of phoneme representations, capturing linguistic information. This phoneme sequence then undergoes processing through feed-forward transformer blocks, allowing for the modeling of the acoustic features. Subsequently, the length regulator adjusts the sequence based on predicted phoneme durations, ensuring alignment with the target mel-spectrogram’s length. Further processing through additional FeFoT blocks enables the generation of the mel-spectrogram, capturing detailed frequency information. Finally, the synthesized mel-spectrogram is transformed into a waveform using a vocoder [141], resulting in the production of speech. FastSpeech 2 [135] introduces several key improvements over FastSpeech to address its limitations. Firstly, FastSpeech 2 eliminates the teacher-student distillation pipeline, directly training the model with ground-truth data instead of simplified output from a teacher model. This change simplifies the training process and avoids information loss, potentially enhancing voice quality. Secondly, it incorporates more detailed information, including pitch, energy, and more accurate duration, as conditional inputs. This additional information helps in better solving the one-to-many mapping problem inherent in text-to-speech synthesis. Lastly, FastSpeech 2 also introduces a variant: FastSpeech 2s [135]. Unlike its predecessor and other similar models that typically generate mel-spectrograms which are then converted into speech waveforms using a vocoder, FastSpeech 2s streamlines this process. This model integrates the entire pipeline, from text to waveform generation, within a single framework, thus

eliminating the need for separate vocoder models.

Vocoder

The role of the vocoder is to synthesize speech from the features predicted by the acoustic model. One of the first vocoders used in neural TTS was the Griffin-Lim algorithm [128], which was designed for reconstructing a waveform from its magnitude spectrogram by iteratively estimating the phase information. In the Griffin-Lim algorithm, the phase estimation is indeed done frame by frame for each iteration n . For each frame S_i of the acoustic features, the algorithm starts by assuming an arbitrary phase $\theta_i^{(0)}$ and combining it with the magnitude spectrogram $|S_i|$ to construct an initial complex spectrogram $S_i^{(0)} = |S_i|e^{j\theta_i^{(0)}}$. For each iteration n , the algorithm computes the inverse STFT (ISTFT) of $S_i^{(n)}$ to obtain a time-domain signal $x_i^{(n)}$. The ISTFT reconstructs the signal from the complex spectrogram as:

$$x_i^{(n)}(t) = \sum_{k=0}^{N-1} S_i^{(n)}(k, t) e^{j2\pi kt/N},$$

where N is the number of points in the Fourier transform, and t is the time index for the window. The algorithm then applies the STFT to $x_i^{(n)}$ to obtain a new complex spectrogram, $S_i^{(n+1)}$. From this spectrogram, it extracts the updated phase $\theta_i^{(n+1)}$. The original magnitude spectrogram $|S_i|$ is combined with this new phase to form $S_i^{(n+1)} = |S_i|e^{j\theta_i^{(n+1)}}$ for the next iteration. This iterative process is repeated until the phase $\theta_i^{(n)}$ converges, leading to a reconstructed signal that closely approximates the original audio from its magnitude spectrogram. The Griffin-Lim algorithm has been fundamental in early TTS systems but has largely been replaced by more advanced neural vocoders, which can generate high-quality audio directly from spectrograms without iterative phase estimation. Modern vocoders are based on neural network architectures and can be categorized into four main types:

- **autoregressive**: WaveNet [29], as the first neural-based vocoder, introduced a novel approach to generating waveform points autoregressively using dilated convolution

[108]. One of the main drawbacks of the original WaveNet architecture was its slow inference speed which was improved using hierarchical RNN [142] for unconditional waveform generation [143]. Further advancements in vocoder technology have focused on enhancing robustness and universality. The introduction of conventional digital signal processing techniques into neural networks, as seen in LPC-Net [144], leveraged linear prediction coefficients for next waveform point calculation and a lightweight RNN for residual computation.

- **flow-based:** normalizing flows [145] operate by transforming a simple, known probability distribution (such as a Gaussian distribution) into a more complex distribution through a sequence of invertible transformations. The mathematical foundation of normalizing flows is rooted in the change-of-variables formula, which allows for the computation of the probability density of the transformed variable. Given an initial random variable $z \in \mathbb{R}^D$ with a simple probability density function $f_Z(z)$, we apply a series of transformations $g_k(\cdot; \theta_k)$, where each g_k is a bijective and continuously differentiable function parameterized by θ_k . After K transformations, the transformed variable z_K has a density function $f_{Z_K}(z_K)$, which can be computed using the change of variables rule as

$$f_{Z_K}(z_K) = f_Z(z) \prod_{k=1}^K \left| \det \frac{\partial g_k^{-1}(z_k; \theta_k)}{\partial z_k} \right|,$$

or equivalently:

$$f_{Z_K}(z_K) = f_Z(z) \prod_{k=1}^K \left| \det \frac{\partial g_k(z_{k-1}; \theta_k)}{\partial z_{k-1}} \right|_{z_{k-1}=g_k^{-1}(z_k; \theta_k)}^{-1}.$$

Here, $\det \frac{\partial g_k^{-1}(z_k; \theta_k)}{\partial z_k}$ is the determinant of the Jacobian matrix of the inverse transformation g_k^{-1} evaluated at z_k , which accounts for the change in volume (density) caused by each transformation g_k . This rule allows to track the evolution of the probability density function through successive transformations, with the Jacobian determinant of each transformation adjusting the density appropriately for changes in

volume in the transformed space. In practice, designing f_k to ensure the invertibility and efficiently computing the Jacobian determinant are two critical steps for the applicability of normalizing flows. Normalizing flows can theoretically model any complex distribution given enough transformations. Because of this, they have been used in various machine learning tasks, including generative modeling and variational inference [146], where the goal is to approximate complex posterior distributions. Flow-based models are broadly categorized into autoregressive transforms and bipartite transforms. Autoregressive transforms, such as the inverse autoregressive flow [147], allow for efficient parallel sampling but require sequential inference for likelihood estimation. This contrasts with bipartite transforms [141], [148], which are characterized by their use of affine coupling layers to ensure the invertibility of the transforms, enabling high-quality voice synthesis and fast inference speeds. Despite their advantages, each approach has its trade-offs. Autoregressive transforms, while expressive, necessitate complex teacher-student training strategies, whereas bipartite transforms often require a larger number of parameters to achieve comparable model capacities.

- **GAN-based:** Generative Adversarial Networks (GAN) are a class of neural networks consisting of two competing models: a generator and a discriminator engaged in a minimax game [149]. The generator learns to produce synthetic data samples while the discriminator learns to distinguish between real and generated samples. This architecture is largely used in vocoder applications [150]–[152]. Leading-edge architectures introduce multi-scale generators [153], [154], progressively generating waveform sequences at different resolutions, thereby offering flexibility in balancing synthesis efficiency and sample quality. On the discriminator front various discriminator variants have been explored, including random window discriminators [150], multi-scale discriminators [155], multi-period discriminators [152] and hierarchical discriminators [153]. These discriminator architectures improve audio analysis, thereby providing better guidance for the generator models. Random window discriminators assess audio inputs through multiple complementary windows, simplifying binary scoring

compared to analyzing the entire audio stream. Multi-scale discriminators work on distinct frequency ranges, whereas multi-period discriminators analyze diverse segments of an audio input at varying intervals, capturing implicit structural features. Additionally, hierarchical discriminators evaluate generated waveforms across multiple resolutions, aiding the generator in understanding the relationship between acoustic characteristics and waveforms.

- **diffusion-based:** diffusion-based vocoders utilize Denoising Diffusion Probabilistic Models (DDPM) [156] to generate waveforms from spectrograms [114]. The diffusion process gradually transforms a mel-spectrogram $\mathbf{X}_0$ into a Gaussian noise $\mathbf{X}_T$ through a series of Markov transitions, each with a predefined variance schedule β_t . The transformation at each step is described by the Markov transition probability

$$q(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t; \sqrt{1 - \beta_t}\mathbf{X}_{t-1}, \beta_t\mathbf{I}).$$

This process is modeled as:

$$q(\mathbf{X}_{1:T}|\mathbf{X}_0) = \prod_{t=1}^T q(\mathbf{X}_t|\mathbf{X}_{t-1}).$$

The reverse process, involves learning to denoise, i.e., to recover the original mel-spectrogram from the noise. This is described by the conditional distributions $p_\theta(\mathbf{X}_{0:T-1}|\mathbf{X}_T) = \prod_{t=1}^T p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t)$, where $p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t)$ models the step from noisy $\mathbf{X}_t$ back to a less noisy $\mathbf{X}_{t-1}$, with the goal of eventually reconstructing $\mathbf{X}_0$. While diffusion-based vocoders exhibit exceptional voice quality in speech generation, they encounter a drawback in their slow inference speed due to the prolonged iterative process [96]. Consequently, various studies on diffusion models are actively exploring methods to mitigate inference time without compromising generation quality [157], [158].

Recent advances in TTS technology involve the integration of neural audio codecs [159], [160] and language modeling techniques [161]. Neural audio codecs compress audio into discrete tokens that preserve fine acoustic details, including speaker identity and environmental

sounds. Building on this, models such as [162] frame TTS as a language modeling task using these discrete audio tokens. Rather than predicting waveforms or spectrograms directly, these systems generate sequences of tokens derived from neural audio codecs. By treating TTS as a language modeling problem, these models leverage autoregressive transformers commonly used in natural language processing [163] to predict audio tokens conditioned on text inputs. This approach enables more efficient and flexible TTS systems, facilitating applications such as speech editing, personalization, and content creation [115].

1.10 Basic Concepts and Architectures of Voice Conversion Technology

The purpose of this Section is to introduce the evolution of Voice Conversion (VC) models and to provide an overview and a basic understanding of VC technologies, without aiming to be an exhaustive review of the field.

1.10.1 Introduction to Voice Conversion

VC refers to the process of transforming a speaker's voice (source) into the voice of another speaker (target) while preserving the linguistic content [164]. The primary challenge in VC lies in modifying speaker-specific characteristics, such as timbre, pitch, and prosody, while maintaining the semantic content.

In mathematical terms, VC is the transformation of an acoustic feature sequence $\mathbf{X}_s = \{\mathbf{x}_{s1}, \mathbf{x}_{s2}, \dots, \mathbf{x}_{sn}\}$ from the source speaker into an acoustic feature sequence $\mathbf{X}_t = \{\mathbf{x}_{t1}, \mathbf{x}_{t2}, \dots, \mathbf{x}_{tm}\}$ for the target speaker. The mapping function f can be formalized as:

$$\mathbf{X}_t = f(\mathbf{X}_s)$$

where f is the model responsible for learning the transformation. A simplified schematic of a VC system is reported in Fig. 1.25 where the function f is implemented by the pipeline *context encoder*, *speaker encoder* and *decoder*. VC is also divided in parallel and non-parallel VC. Parallel data-based VC [165] requires aligned pairs of speech utterances from the

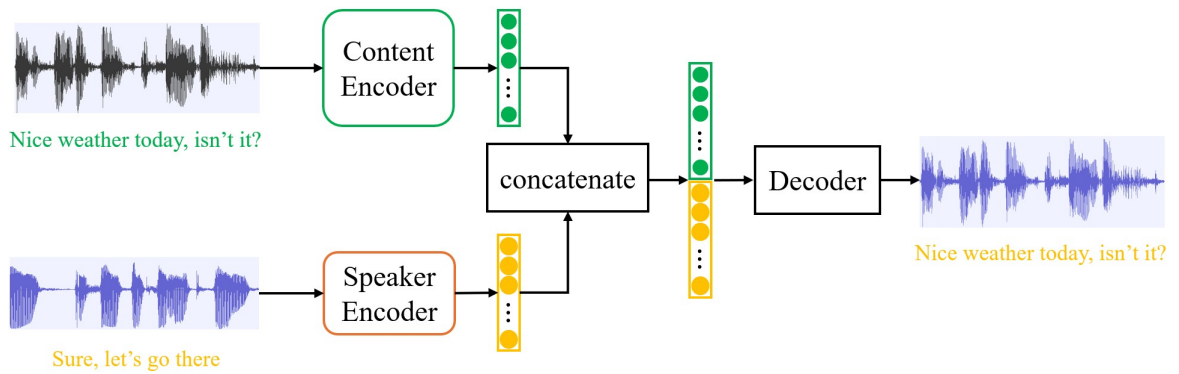


Figure 1.25: Basic components of a VC system. The content encoder extracts semantic features from the source utterance, while the speaker encoder generates a speaker representation from the target utterance. The decoder then synthesizes an audio signal by combining the outputs of the content and speaker encoders, producing a waveform that replicates the target speaker’s voice as if they were speaking the words from the source audio.

source and target speakers, enabling direct feature mapping between the two. The goal is to estimate the function $f(\mathbf{X}_s)$ by minimizing the distance between the converted feature sequence and the target feature sequence $\mathbf{X}_t$. Non-parallel VC [166] methods do not require aligned speech pairs, making them more flexible. These methods utilize unsupervised learning techniques such as CycleGAN [167], Variational Autoencoders (VAEs) [168], or adversarial training [150] to learn a mapping between the source and target without explicit pairing of data.

Moreover, in VC, there are various types of conversion approaches that differ based on the nature of speaker transformation. The most common forms include one-to-one VC, many-to-many VC, and any-to-any VC:

- One-to-one VC refers to converting the voice of one specific speaker (source) into that of another predefined speaker (target). This approach is common in systems that work with known speaker pairs and are typically trained with parallel data for the specific speakers.
- Many-to-many VC involves converting voices among a larger set of speakers. For example, any speaker from a predefined group can be converted into any other speaker within the group.

- Any-to-any VC extends the flexibility further by enabling voice conversion between completely unseen speakers (i.e., speakers not present in the training data). This type is often referred to as zero-shot or one-shot VC and requires more sophisticated models that generalize well across unseen conditions, relying on speaker embeddings such as i-vectors [79], x-vectors [83] or ECAPA embeddings [84] (briefly covered in Section 1.8) to represent speaker identity without needing large amounts of data from new speakers.

Any-to-any transformation is the VC type used in Chapter 2 and 4.

1.10.2 Voice Conversion Architectures

In this Section some VC architectures are explored, covering their key components and the associated mathematical formulations.

In traditional VC, GMMs are used to model the joint probability distribution of source and target features [169]. A GMM is used to learn the relationship between corresponding feature vectors of the source and target speakers. For a frame feature vector $\mathbf{x}_s$, the GMM is used to predict the target feature vector $\mathbf{x}_t$ by computing the conditional distribution:

$$P(\mathbf{x}_t|\mathbf{x}_s) = \sum_{i=1}^K P(i|\mathbf{x}_s) \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_t^i + \mathbf{A}_i(\mathbf{x}_s - \boldsymbol{\mu}_s^i), \boldsymbol{\Sigma}_t^i)$$

where:

- $\boldsymbol{\mu}_s^i, \boldsymbol{\mu}_t^i$ are the means of the i -th (out of K) Gaussian component for the source and target,
- $\boldsymbol{\Sigma}_t^i$ is the covariance matrix for the target,
- $\mathbf{A}_i$ is a linear transformation matrix representing the relationship between source and target features in the i -th component.

Conversion is performed by transforming the source feature vector using the parameters of the GMM:

$$\hat{\mathbf{X}}_t = \boldsymbol{\mu}_t^i + \mathbf{A}_i(\mathbf{X}_s - \boldsymbol{\mu}_s^i).$$

Variational Autoencoder (VAE)s are a type of autoencoder used in VC for learning latent representations of speech [170], [171]. The encoder $q(\mathbf{z}|\mathbf{X}_s)$ learns a latent variable $\mathbf{z}$ that captures speaker-invariant content, while the decoder $p(\mathbf{X}_t|z)$ reconstructs the target speech. The VAE loss function consists of two terms. The reconstruction loss

$$L_{rec} = \mathbb{E}_{q(z|X_s)}[\log p(\mathbf{X}_t|\mathbf{z})]$$

measures how well the decoder reconstructs the target speech while the Kullback–Leibler divergence

$$L_{KL} = D_{KL}(q(\mathbf{z}|\mathbf{X}_s)||p(\mathbf{z})).$$

ensures that the latent variable follows a prior distribution (usually a Gaussian distribution). The total VAE loss is then:

$$L_{VAE} = L_{rec} + \beta L_{KL}$$

where β is a weight balancing the two terms.

GANs consist of two neural networks: the generator G and the discriminator D . In VC, the generator tries to produce realistic target speech from the source speech, while the discriminator attempts to distinguish between real and generated speech. The GAN objective is:

$$L_{GAN} = \mathbb{E}_{\mathbf{X}_t \sim P_{data}}[\log D(\mathbf{X}_t)] + \mathbb{E}_{\mathbf{X}_s \sim P_{gen}}[\log(1 - D(G(\mathbf{X}_s)))]$$

where P_{data} represents the real data distribution and P_{gen} represents the generated data distribution. In VC, the generator is tasked with generating realistic target speech, and the discriminator helps improve the quality of the generated speech by providing feedback. CycleGANs [167], [172] are widely used for non-parallel VC, as they do not require paired data. The key idea of CycleGANs is to use cycle consistency [173] to ensure that the converted speech can be mapped back to the original speech. CycleGANs consist of two generators: $G_{\mathbf{X} \rightarrow \mathbf{Y}}$ (source-to-target conversion) and $G_{\mathbf{Y} \rightarrow \mathbf{X}}$ (target-to-source conversion). Cycle consistency ensures that converting $\mathbf{X}_s$ to $\mathbf{X}_t$ and back should yield the original

input:

$$L_{cycle} = \|G_{Y \rightarrow X}(G_{X \rightarrow Y}(\mathbf{X}_s)) - \mathbf{X}_s\|^2 + \|G_{X \rightarrow Y}(G_{Y \rightarrow X}(\mathbf{X}_t)) - \mathbf{X}_t\|^2.$$

The total loss for CycleGAN includes adversarial loss, cycle consistency loss, and identity loss:

$$L_{total} = L_{GAN} + \lambda_{cyc}L_{cycle} + \lambda_{id}L_{identity}$$

where $L_{identity}$ helps preserve the identity of the converted speech:

$$L_{identity} = \|G_{X \rightarrow Y}(\mathbf{X}_t) - \mathbf{X}_t\|^2 + \|G_{Y \rightarrow X}(\mathbf{X}_s) - \mathbf{X}_s\|^2.$$

StarGAN-VC [174], [175] is an extension of CycleGAN that can handle many-to-many voice conversion tasks. The generator in StarGAN takes both the source speech and a target speaker label $\mathbf{y}_t$ as inputs:

$$G(\mathbf{X}_s, \mathbf{y}_t) = \mathbf{X}_t.$$

The model is trained using an adversarial loss to ensure the generated speech is indistinguishable from real target speech, the cycle consistency loss [173] to ensure that the converted speech can be mapped back to the and a classification loss to ensure that the generated speech is correctly classified as belonging to the target speaker. Moreover, diffusion-based models [176]–[178], self-supervised learning [179]–[181] and end-to-end [127] with language modeling [182], [183] architectures have recently gained attention in the VC domain.

1.10.3 Disentanglement in Voice Conversion

Although VC and the vocoder stage in Section 1.9 present a parallel evolution given shared technical challenges and goals, disentanglement is a fundamental stage typical of only VC. Disentanglement techniques such as [184] aim to separate content (speaker-independent) from style (speaker-specific) attributes in speech. Style characteristics that disentanglement

typically targets include emotion [185], accent [186], prosody [187] pitch and rhythm [188]. This separation allows for flexible manipulation of speaker identity without altering the linguistic content. Disentanglement can be achieved through adversarial training [189], [190] or by introducing an information bottleneck [171], [191], [192].

1.11 An Introduction to Privacy in Speech Technology

Privacy protection cannot be considered separately from technological advancement [193]. Due to the development of science, technology and the advent of the social media era, the possibility to invade people privacy has increased massively. This led, in recent years, to mounting calls towards privacy protection resulting in novel privacy legislation, e.g., the EU-GDPR [7]. However, to enforce and secure privacy protection, it is necessary to first define the subject of protection: privacy. Historically, privacy is as old matter although it was not always a legally protected right. Moreover, it is important to highlight that what is considered to be private and what is legally protected as private can differ. Privacy existed long before our modern era. We can track the origins of privacy back to ancient societies. The Hammurabi Code contained a section against the intrusion into individual's home and even the Roman laws contemplated the same question [194]. Of course, the distinction between private and public life changes with each era and society, leading to evolving perceptions of privacy throughout history [195]. The modern privacy notion, was originally defined as “the right to be let alone” [196]. From here on, the right to privacy has become universally acknowledged and became a fundamental human right in western culture and law. In spite of the fact that legal systems ensure privacy protection, there is no consensus on what exactly privacy is [197]. Even though several attempts have been made to establish a broad definition of privacy, no universal description could be established. This implies that privacy needs to be interpreted in the light of the current era and be examined in each specific context. For example, sharing the same information in different situations might be considered private differently [198]. In the scientific and technical fields, after the appearance of computers in the 1970s, it was questioned whether the right to privacy could ensure private life protection. Because of this technological change, a

separate individual right appeared: the right to data protection [193]. On this manner, the EU-GDPR states that privacy protection applies to all “personal data” which include any information relating to a living, identified or identifiable person. Examples include name, social security number, other identification numbers, location data, IP addresses, online cookies, images, email addresses, and content generated by the data subject. Moreover, privacy regulations include more stringent protections for special categories of personal data such as racial or ethnic origin, physical or mental health data, political opinions, trade union membership, sex life and sexual orientation, religious or philosophical beliefs, genetic and biometric data.

In this context, speech becomes a highly sensitive type of personal data that must be protected. In fact, speech contains significant PII both in the semantic (linguistic) and acoustic domain. Specifically, semantic personal identifiers such as full name, social security number or geographical positioning can alone allow speaker identification. Moreover, voice characteristics such as prosody, speaking rate, accent and intonation inherently contain a variety of acoustic PII such as personality, physical characteristics, emotional state, age and gender that can be identified [6] and therefore used for malicious privacy attacks. In recent years, the topic of speech privacy raised privacy concerns for the public and policy makers because of the advent of voice assistants and smart speakers [5]. Those devices have built-in microphones continuously listening for an activation word which, when detected, activates the recording of a voice request. This audio is then broadcasted to a remote server which transcribes the audio with an ASR engine, analyses the linguistic content with a Natural Language Understanding (NLU) model and process user’s initial request which can range from controlling a smart device (such as turning on a light) or giving a voice reply (such as the current weather forecast) by synthesizing artificial TTS signals. Typically, the ASR, NLU, and TTS modules are powered by sophisticated deep learning techniques that demand extensive data sets to achieve optimal performance. This necessity pushes companies to process and store users speech data in centralized remote servers to continuously improve the aforementioned algorithms. However, speech contains such a variety of PII that even if privacy policies are in place, huge scandals came up due to poor

data handling within large tech companies⁹. Moreover, according to the market research firm Voicebot, more than 50% of the U.S. population has access to one or many voice assistant devices. These devices operate by continuously listening for their designated "wake-up word," a feature that enables them to respond promptly to user commands. However, this continuous listening poses inherent privacy risks, as the devices can inadvertently activate and record conversations not intended as commands. Recent research highlights that such misactivations are not merely hypothetical but occur with measurable frequency, approximately 1.43 times every 10,000 words [199]. Furthermore, it has been observed that about 10% of these unintended activations last longer than 10 seconds, during which time conversations can be recorded and transmitted to remote servers without user awareness or consent. This issue of accidental recording is particularly concerning because these devices, once activated, whether intentionally or not, can capture and process highly sensitive information. The data collected may inadvertently include PII, which can range from names and locations to more subtle details like personal preferences and behaviors. Similarly, connected toys like CloudPets experienced data breaches, exposing over 2 million voice recordings of children and parents due to unsecured databases, highlighting vulnerabilities in devices designed for children¹⁰. The transmission of such sensitive information to remote servers not only compromises user privacy but also increases the risk of data breaches and unauthorized access. Given the potential for such privacy violations, suppressing PII in speech signals would improve privacy.

⁹The Guardian article: <https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings>

¹⁰The Telegraph article: <https://www.telegraph.co.uk/news/2017/02/17/germany-bans-internet-connected-dolls-fears-hackers-could-target/?ref=troyhunt.com>

Chapter 2

Two-Stage Voice Anonymization

The rapid evolution of speech technologies and artificial intelligence has transformed daily life through the convenience of voice-activated devices and automated systems. However, these advancements have raised complex privacy challenges, prompting legislative responses such as the EU-GDPR, which imposes strict requirements on the handling of personal data. Addressing these issues, this chapter presents a dedicated investigation into speech anonymization, which is central to safeguarding sensitive information embedded in audio data. Building on prior frameworks, this work introduces an advanced anonymization system built during the author time at Nuance Communication UK and an earlier version of this work was presented in [200]. The system is tailored to obscure personal identifiers within speech, drawing on recent methodologies and best practices in data protection.

Furthermore, this chapter contributes to the field by rigorously analyzing privacy vulnerabilities specific to speech data, offering a detailed technical overview of anonymization processes, and presenting empirical results that underscore the system's effectiveness. By situating this research within the current landscape of AI-driven privacy challenges, the chapter provides a comprehensive, state-of-the-art perspective aimed at advancing the field of speech privacy technology. Key contributions include the application of innovative anonymization techniques, the assessment of their alignment with regulatory standards, and an evaluation framework that benchmarks these methods against existing solutions in terms of privacy protection and data utility.

This anonymization system can be applied in various domains, including healthcare, where protecting patient confidentiality in speech-based diagnostic applications is crucial.

2.1 The anonymization task

Formally, anonymisation is defined as the process of rendering personal data anonymous. According to the European Union’s data protection laws, in particular the EU-GDPR, anonymous data is *“information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable”*. This definition is particularly important because, once data is truly anonymous and individuals are no longer identifiable, the data will not fall within the scope of the EU-GDPR regulation. Furthermore, the EU-GDPR defines pseudonymisation as *“the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person”*. This means that the use of ‘additional information’ can lead to the identification of the individuals, which is why pseudonymous personal data are still personal data. It is however important that the expression “anonymous data” is not intended as if anonymized datasets could be binary labelled as anonymous or not. In fact, each example in any dataset have a probability of being re-identified. Because of this reason, any robust anonymisation system should assess the re-identification risk, which should be managed and controlled over the time: this is particularly important due to the ever improving computing capability and algorithmic performance available to an attacker that could try to re-identify individuals within an anonymous dataset. To further develop the concept of anonymization described in the EU-GDPR, encryption can not be considered as an anonymization technique. Because the original data needs to be accessible, the transformations applied by the encryption algorithms are designed to be reversible, in what is called decryption stage. The secret keys used for encryption/decryption are the ‘additional information’ introduced above in the

context of pseudonymisation, which can make the personal data readable and, consequently, the identification possible.

2.2 The Voice Privacy Challenge

The Voice Privacy Challenge [201] initiative has been established to promote research in speech privacy and its first edition took place in 2020. The Voice Privacy Challenge focuses on voice anonymization, defined as the task of altering speaker’s voice to hide their identity, while preserving other speech attributes such as speech content and pitch. The challenge framed the problem by establishing formally the definition of the anonymization task, providing a state-of-the-art privacy attack model, and by constituting a common dataset, protocols and metrics to measure new anonymization algorithms capability.

Considering a situation in which semantic personal identifiers are not present or has been obfuscated, initial acoustic privacy protection approaches explored several research directions such as extracting privacy-preserving features [202], working with encrypted speech signals [203], learning adversarial features [204], or performing federated learning [205]. However, feature or model-level privacy protection have limitations. Privacy preserving features can, in principle, be used for any downstream application but there is no guarantee they retain any useful information to efficiently address the new task: this means that either the downstream application is known beforehand, or a new set of features needs to be developed (or learned) for different applications. Encryption, although allowing data manipulation in the encrypted domain, introduces significant computational overheads and, due to the definition of Section 2.1, can not be considered as an anonymization technique. Adversarial features have been reported to increase privacy in closed-set classification scenarios but lack in generalization [204] when extended to other datasets. Federated models, despite avoiding direct access to the data, can leak original data information through the higher level representation used for learning (e.g., local gradients) [206]. Therefore, state-of-the-art anonymization systems are based on the idea of disentangling the speaker identity information from the linguistic and prosodic content thus producing synthetic utterances in which the speaker identity has been altered [201],

[207], [208] while other speech characteristics such as pitch and linguistic content [209], possibly important for downstream tasks, are preserved.

The privacy attack model is structured as a game (Figure 2.1). Two users exchange utterances (*trial utterances*) through a channel which gets hacked by an attacker. The users, before transmitting their data, process them with the anonymization system which should remove all the personal information while preserving collateral information useful for a downstream task. The attacker can access the audio recordings of the users and tries to infer users personal information using any additional prior knowledge. Furthermore, the Voice Privacy Challenge framework establishes the following rules:

- the anonymization system needs to output a speech waveform: processed utterances sound as if they were uttered by a different speaker, the *pseudo-speaker*;
- the anonymized utterance needs to hide the original speaker identity;
- the anonymization system needs to preserve speaker characteristics collateral to the identity which might be useful for downstream tasks;
- trial utterances from each speaker need to be anonymized by the same, unique pseudo-speaker.

Another important contribution of the Voice Privacy Challenge is the establishment of a carefully designed dataset to assess privacy protection capability of each anonymization system. Table 2.1 describes the composition of the training, development and evaluation sets. All the data used in the design of the anonymization system and in the privacy evaluation stages come from publicly available datasets. These comprise subsets from:

- LibriSpeech [68] is an English speech dataset obtained by recording actors reading book passages. It encompasses 960 hours of speech, which are categorized into 'clean' and 'other' subsets to distinguish between recordings with clear audio quality and those with background noise;
- LibriTTS [210] is derived from LibriSpeech and its original application was text-to-speech model development;

	Subset	Female	Male	Total	N° of Utt.
Training	VoxCeleb-1,2	2912	4451	7363	1281762
	LibriSpeech train-clean-100	125	126	251	28539
	LibriSpeech train-other-500	564	602	1166	148688
	LibriTTS train-clean-100	123	124	247	33236
	LibriTTS train-other-500	560	600	1160	205044
	LibriSpeech train-clean-360*	439	482	921	104014
Dev.	LibriSpeech dev-clean	35	34	69	2321
	VCTK dev	15	15	30	11972
Test	LibriSpeech test-clean	36	33	69	1934
	VCTK test	15	15	30	12048

Table 2.1: Voice Privacy Challenge datasets. Number of speakers and utterances for each dataset.

- VCTK [211] contains 109 native English speakers with various accents with each speaker reading out about 400 sentences from a newspaper. The dataset was recorded for the purpose of building text-to-speech synthesis systems;
- VoxCeleb [212], [213] is an audio-visual dataset consisting of short clips of human speech, extracted from interview videos uploaded on YouTube designed for speaker verification research. It contains approximately 2,770 hours of speech sampled at 16 kHz collected from 7,363 speakers, covering a wide range of accents and languages.

Furthermore, the development and test datasets have been divided in more subsets as shown in Table 2.2. In the case of LibriSpeech dev/test-clean, all the speakers in the enrollment set are a subset of those in the trial set. Regarding VCTK-dev/test, the same speakers are present in the enrollment and trial. Moreover, two different trial sets have been considered: common and different. On one hand, the common subset consists of utterances in the VCTK corpus that are common to all speakers. On the other, the different subset contains distinct utterances for each speakers (the same is valid for the enrollment set).

	Subset	Split	Female	Male	Total	N° of Utt.
Development	LibriSpeech dev-clean	enrollment	15	14	29	343
		trial	20	20	40	1978
	VCTK dev	enrollment	15	15	30	600
		trial different	15	15	30	10677
		trial common	15	15	30	695
Test	LibriSpeech test-clean	enrollment	16	13	29	438
		trial	20	20	40	1496
	VCTK test	enrollment	15	15	30	600
		trial different	15	15	30	10748
		trial common	15	15	30	700

Table 2.2: Voice Privacy development and test datasets. Number of speakers and utterances for each dataset.

2.3 Scoring Pipeline

The two primary scores to evaluate an anonymization system are the EER of an ASV system and the WER of an ASR model. The first assesses the privacy protection capability of the anonymization pipeline whereas the second measures the utility of the anonymized speech to perform downstream tasks. The next sections introduce the attacker model and the ASV system in combination with the ASR model used to assess the utility of the anonymized speech.

2.3.1 The attacker model: privacy evaluation

In the Voice Privacy Challenge and for the experiments presented in Section 2.6 the attacker model is an ASV model. For each user of interest, the attacker model is supposed to have access to a set of original utterances, the *enrollment utterances*, and a certain number of *trial utterances* the attacker intercepts, Figure 2.1. The ASV algorithm used in the Voice Privacy Challenge is based on the x-vector speaker embeddings [83] coupled with PLDA [86] and was implemented in Kaldi [214]. For the experiments presented in Section 2.6, a similar architecture was implemented in Speechbrain [215]. Moreover, the considered attacker model is *Semi-Informed* [216]. This model is the strongest known to date, which is the reason why it is the most reliable for privacy assessment. In this condition, each

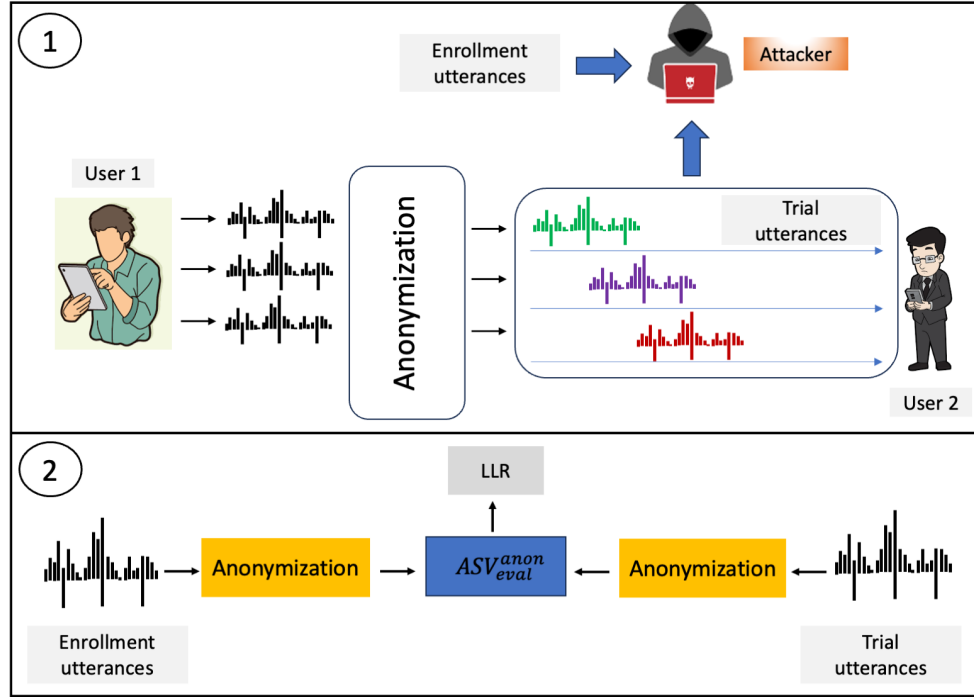


Figure 2.1: Semi-Informed attacker model. Panel 1: the attacker has access to the anonymization system but without the specific parameters used by the user for the anonymization of each enrollment utterance. Panel 2: the attacker uses the ASV model fine-tuned on an anonymized version of LibriSpeech-train-clean-360 ASV_{eval}^{anon} to compute log-likelihood ratios between enrollment and trial anonymized utterances.

user anonymizes their *trial utterances* before transmitting them (Figure 2.1, panel 1). The attacker intercepts those data and attempts to re-identify the users. Assuming the attacker has access to original, unprocessed, enrollment data and to the anonymization system, the attacker performs speaker-level anonymization of the enrollment data (Figure 2.1, panel 2) before feeding them to the ASV system: this is done in order to minimize the mismatch between the original and anonymized audios. In this scenario, the trial and enrollment data might be anonymized using different pseudo-speakers, since the attacker can not know the pseudo-speaker used in the anonymization stage by the user. Initially, the attacker trains the ASV model on Voxceleb1,2. However, being the anonymization system publicly available, the attacker uses that system for utterance-level anonymization of the LibriSpeech-train-clean-360 dataset (LS_{anon}^{360}) and uses it to fine-tune the ASV system on anonymized data. For every pair of enrollment and trial x-vectors extracted from the respective utterances, the ASV outputs a Log-Likelihood Ratio (LLR) score. This is used,

with a thresholding mechanism, to discriminate between same and different speaker label for that utterance pair. Privacy protection is measured with the EER metric, Section 1.8.

The ASV model used in the Results section is reported in Figure 2.2. This is composed

Layer ID	Output Channels	Kernel Size	Dilation Rate
1	512	5	1
2	512	3	2
3	512	3	3
4	512	1	1
5	1500	1	1

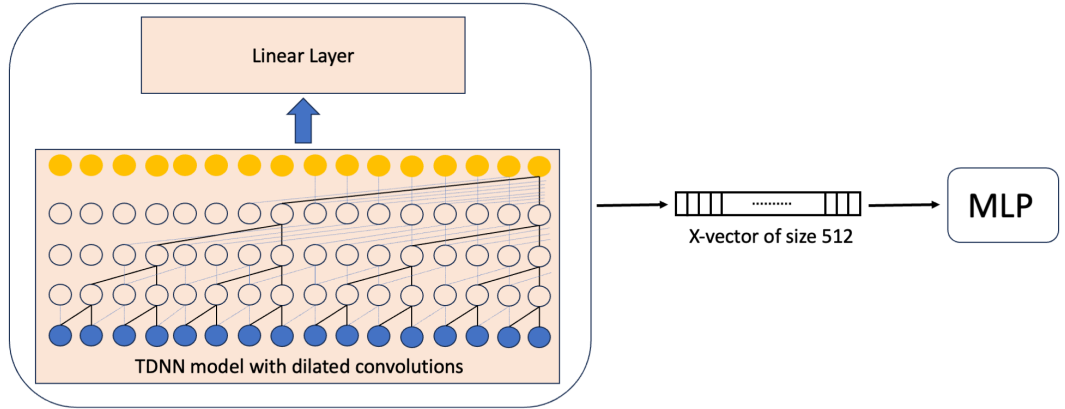


Figure 2.2: Semi-Informed x-vector attacker model structure.

by 5 TDNN layers with varying dilation rates (layer architecture in Table 2.3.1) coupled with a lineal layer with 512 outputs mapping variable length audio input sequences to fixed length output embedding: the x-vectors. At training time, the embedding model is coupled with a simple MLP classifier with number of output neurons equal to the number of speakers in the training dataset. This classifier is not employed at testing time but is substituted with the PLDA probabilistic model.

2.3.2 The downstream task: utility evaluation

In the Voice Pricavy Challenge, privacy evaluation is combined with the evaluation of a utility metric. This measures the capability of the anonymization system to preserve information in the input audio signals useful to perform a specific downstream task. For

doing this, an ASR system is used and its WER is considered to be the evaluation score. Contrarily to the Voice Privacy Challenge in which an outdated TDNN-ASR model was used, here an ASR system based on a transformer acoustic model encoder and a joint transformer decoder was employed. This model exploits CTC [55], with decoding stage integrating also CTC probabilities, Section 1.7. Furthermore, the ρ^{F_0} was calculated, also known as the pitch correlation metric. This score assesses intonation preservation of the anonymization process. Preserving a wide range of pitches is particularly important when using anonymized data to train ASR models which are then applied to real speech, as it enhances model generalization across diverse speakers and contexts, ensures robustness against anomalies, and upholds inclusivity by accurately reflecting the natural variability found in human speech [217], [218]. Computing ρ^{F_0} requires to first extract F_0^k from each utterance using [219]. Then

$$\rho_{i,j}^{F_0} = \frac{\sum_{i=1}^n (F_0^i[k] - \bar{F}_0^i)(F_0^j[k] - \bar{F}_0^j)}{\sqrt{\sum_{i=1}^n (F_0^i[k] - \bar{F}_0^i)^2} \sqrt{\sum_{i=1}^n (F_0^j[k] - \bar{F}_0^j)^2}}$$

is computed as the Pearson correlation between the F_0 of original (F_0^i) and anonymized (F_0^j) speech.

2.3.3 Privacy-To-Utility Trade Off Metric

The two primary scores used to evaluate an anonymization system are the EER and the WER as defined in Sections 2.3.1 and 2.3.2. The first assesses the privacy protection capability of the anonymization pipeline whereas the second measures the utility of the anonymized speech to perform a downstream task namely, ASR. Moreover, it has been shown that when manipulating speech data to enhance privacy (higher EER), the utility of the anonymized signals drops (higher WER) [201]. Therefore, the choice of the most appropriate anonymization system heavily depends on the specific application: when privacy is paramount, large WER degradation might be tolerated in favour of privacy. However, for other applications when even small WER variations heavily impact the performance, an anonymization system with a limited WER deterioration might be the preferred choice

despite its lower privacy protection capability. Here, the Privacy-to-Utility Trade Off (PU_{tr}) is introduced. This is a compressed metric combining the primary anonymization metric (EER) and the primary utility score (WER), designed to evaluate the anonymization system's performance across different operational points. The WER measures the impact on speech utility by quantifying how much the anonymized utterances deviate from the original utterances in terms of intelligibility and recognition accuracy. The EER assesses the anonymization strength, reflecting how much harder it becomes for a speaker verification system to correctly identify the speaker after anonymization. Given WER_i and EER_i for $i \in \{0, 1\}$, where $i = 0$ represents the metrics calculated on the original (non-anonymized) utterances and $i = 1$ represents the metrics on the anonymized utterances, we define the metric

$$PU_{tr} = \lambda \frac{\log \left(1 + \frac{WER_1}{WER_0} \right)}{\log \left(1 + \frac{1}{WER_0} \right)} - (1 - \lambda) \frac{\log \left(1 + \frac{EER_1}{EER_0} \right)}{\log \left(1 + \frac{1}{EER_0} \right)}$$

where $\lambda \in [0, 1]$ is a parameter that controls the trade-off between the two metrics. A value of λ close to 1 emphasizes the importance of utility (WER), favoring systems that preserve speech intelligibility and content, while a λ closer to 0 prioritizes anonymization strength (EER), making the system focus more on hiding the speaker's identity. The use of logarithmic ratios in both WER and EER calculations smooths out large disparities in absolute values, making the metric more balanced and scalable. The log-ratios express the relative changes in WER and EER due to anonymization. A lower PU_{tr} value (closer to -1) indicates a more favorable trade-off, where both anonymization is strong and utility is preserved. Conversely, a higher PU_{tr} value suggests either significant loss in utility or poor anonymization. This metric is structured to assess the performance of anonymization systems at different operational points, allowing system designers to adjust the weight λ based on whether privacy (anonymization strength) or utility (intelligibility) is the priority in their application. The flexibility of using the factor λ makes PU_{tr} suitable for evaluating various use cases where the balance between privacy and utility may differ.

2.4 Advancements in Voice Privacy and Anonymization Techniques

This Section discusses the recent advancements in voice anonymization, focusing on various techniques that strive to balance the privacy-utility trade-off, including disentangled representations, adversarial perturbation, and deep learning approaches such as generative adversarial networks and orthogonal transformations.

One of the fundamental approaches to voice anonymization is the use of disentangled representations introduced in Section 1.10.3, where the components of speech are divided into distinct features such as content, prosody (intonation, rhythm), and speaker identity. This technique assumes that by isolating these features, it is possible to alter the speaker identity without affecting the linguistic content. Disentangled representation learning underpins many modern speaker anonymization systems. However, despite the appeal of disentangled representations, research has shown that these features are not always perfectly separated [208], [220]. For instance, phonetic bottleneck features can still carry residual speaker information, limiting the effectiveness of anonymization. Methods such as vector quantization [221] have been proposed to further reduce speaker information within the content features by constraining the representation space, which forces the model to suppress speaker identity more effectively. This approach has been found to improve privacy while maintaining the utility of the speech for ASR tasks [222]. Nevertheless, there is still an ongoing debate on the extent to which content and speaker features can be fully disentangled, as perfect separation remains elusive in practical applications.

Beyond disentangled representations, deep learning models have driven significant advancements in speaker anonymization. For example, GANs have been employed to create artificial speaker embeddings that do not correspond to any real speakers, thus anonymizing the speech. In [223], the authors demonstrated the effectiveness of GAN-based approaches in generating new speaker embeddings that lead to nearly perfect anonymization without compromising the intelligibility or quality of the speech but heavily damaging pitch correlation. Similarly, the Orthogonal Householder Neural Network (OHNN) has

been proposed for speaker anonymization [224]. The OHNN acts as a transformation that rotates the original speaker vectors in a way that ensures they remain within the original speaker vector space while being anonymized. This approach has been shown to effectively anonymize speaker identities [224] while preserving the naturalness of the speech. It also addresses some of the limitations seen in earlier selection-based anonymizers, which often suffer from reduced speaker diversity and privacy leakage when the external speaker pool is not well-suited for the target data. The OHNN-based system demonstrated robust performance across various languages, including unseen ones, making it a strong candidate for real-world multilingual applications. Another important dimension of speaker anonymization is the manipulation of prosody, particularly the F_0 , which is closely tied to speaker identity. For instance, systems using pitch manipulation based on phase vocoders or time-scaling algorithms adjust the F_0 trajectories to anonymize the speaker’s voice without altering the linguistic content. An example of this is [225] where a deep neural network is used to predict F_0 trajectories that align better with anonymized x-vectors and linguistic features, thereby improving the naturalness of anonymized speech. However, challenges remain in maintaining the natural variability in speech patterns after pitch manipulation, particularly in cross-gender conversions, where mismatched F_0 trajectories can lead to unnatural-sounding voices.

Despite these advancements, one of the key limitations across anonymization methods is the trade-off between privacy and utility. Methods that offer strong privacy protections often degrade the quality or intelligibility of the anonymized speech. For example, systems that focus on maximizing speaker anonymization by heavily modifying the speaker embeddings or pitch trajectories may inadvertently reduce the speech’s naturalness or increase the WER when used in ASR systems [225], [226]. Furthermore, systems like the OHNN that introduce strong orthogonal transformations to anonymize speaker embeddings may face challenges in ensuring that the transformed vectors retain enough diversity to avoid re-identification [224]. Moreover, the issue of language and accent adaptability presents a significant challenge. Many anonymization systems are trained on American English datasets and face difficulties when applied to non-English languages or non-American accents. Techniques that rely on ASR models trained on specific languages struggle when

exposed to unseen accents and languages due to the lack of appropriate training data. In conclusion, while voice anonymization techniques have advanced significantly, there remains no one-size-fits-all solution. Disentangled representations, deep learning models like GANs and OHNNs, and pitch manipulation techniques all offer unique strengths but are also accompanied by limitations in terms of privacy-utility trade-offs, language and accent adaptability, and speaker diversity.

2.5 Two-Stage Voice Anonymization

In [227], it is demonstrated that combining multiple speech modification techniques enhances privacy. Building on these findings, in this Section, a cascade of two deep learning-based voice conversion systems is proposed, designed to address specific scenarios with varying anonymization requirements. Figure 2.3 provides a schematic overview of the components used in the proposed two-stage system. The first component is a Zero Shot Voice Conversion

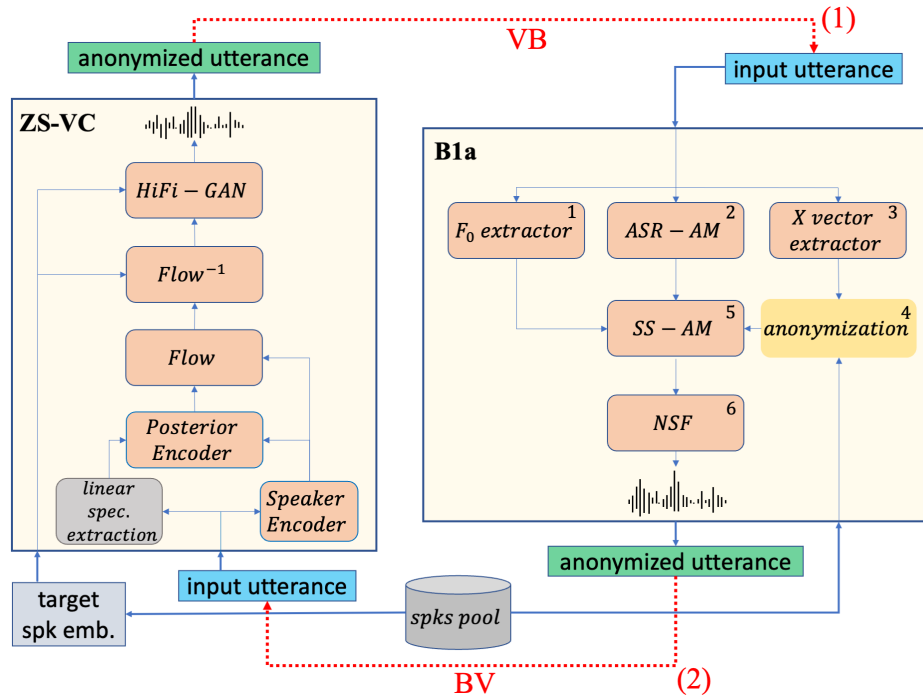


Figure 2.3: Overview of voice anonymization components: ZS-VC and baseline system (B1a). The red dotted lines (1) and (2), are mutually exclusive and refer, in Section 2.6, to system VB and BV respectively.

(ZS-VC) [112] whereas the second is the baseline B1a from [201]. Speech anonymization is achieved by altering the original speaker identity. To do this, both components rely on a set of external speakers (*spks pool*, Figure 2.3) utterances. Speakers in *spks pool* are not included in the set to be anonymized: they are used as targets to alter the original speaker identity. Target speaker selection is handled differently by the two components and this is described in detail in the next Sections. The red dotted lines in Figure 2.3 are mutually exclusive: when (1) is active, ZS-VC is applied first; when (2) is operating, B1a is the first stage of the two-stage anonymization process.

2.5.1 X-vector Speaker Anonymization

This process employs techniques to extract speaker identity, linguistic, and prosodic information from the original speech signal, followed by re-synthesizing a new utterance with modified speaker content. Blocks 1-6 in Figure 2.3 are derived from [201]. These blocks include two main stages: the first focuses on extracting speaker and acoustic information (blocks 1-3), and the second involves re-synthesizing the speech (blocks 4-6). The goal is to merge the original speaker’s acoustic features with a new speaker embedding to achieve effective voice anonymization. The first stage comprises several components: a F_0 extractor (block 1), based on [219], a TDNN-ASR model used to extract Bottle-Neck (BN) features that capture the speech acoustics (block 2), and a speaker encoder (block 3), which computes x-vectors [83] using a second factorized TDNN architecture. The synthesis of anonymized speech is carried out in two steps. First, a Speech Synthesis Acoustic Model (SSAM) (block 5) calculates Mel-filterbank features using F_0 , BN features, and the anonymized x-vector. Second, a Neural Source Filter (NSF) (block 6) generates the waveform from the F_0 , anonymized x-vector, and Mel-filterbank. In this case, the anonymized x-vector is obtained through a distance-based rule: a set of N closest speaker embeddings is selected for each original speaker representation, followed by a random sampling of a subset M from N , which is then averaged to produce the target speaker embedding ($N=200$, $M=20$).

2.5.2 Zero-Shot Voice Conversion

This stage utilizes YourTTS [112], a multi-speaker TTS architecture that supports multilingual functionalities by integrating variational inference and adversarial learning techniques [127]. The architecture comprises five core components: a text encoder, posterior encoder, alignment mechanism, invertible flow-based decoder, and a vocoder [152]. During the training process, the posterior encoder takes the input linear spectrogram and outputs a latent variable, Z . This representation is then used by both the vocoder and the flow-based decoder, whose output Z_p is aligned with the text encoder’s representation via monotonic alignment search [228]. At the inference stage, the text encoder generates the alignment Z_p , which is subsequently utilized by the inverted flow decoder to recreate Z . This latent representation is passed into the vocoder to produce the final audio. Since all five main components are conditioned on speaker embeddings, the same architecture can be repurposed for voice conversion. In the case of voice conversion, where an utterance rather than text is provided at inference, Z_p is derived from the posterior encoder and the direct flow output, both conditioned on the original speaker embedding. By conditioning the inverted flow decoder and vocoder on the target speaker embedding, effective voice conversion is achieved. For this system, the speaker encoder was built using the H/ASP architecture [229], based on ResNet-34. Two different methods for selecting the target speaker were investigated, addressing distinct trade-offs between privacy and utility. We hypothesize that selecting the target speaker randomly while maintaining gender consistency could result in higher-quality speech generation, as noted in other voice conversion studies [168]. This approach could lead to a lower WER by reducing errors in ASR. On the other hand, relaxing the gender constraint might increase the EER due to the larger mismatch between the target and original speaker embeddings, which could negatively affect the performance of the ASV embedding model.

2.6 Experiments

In this Section, the training and testing subsets used are identical to those listed in Table 2.1 adhering to the same constraints. Initially, ZS-VC was trained on the *LibriTTS-100-clean* dataset [210] for 700,000 steps with a batch size of 52, followed by fine-tuning on *LibriTTS-500-others* for an additional 300,000 epochs. Both subsets were resampled at 16 kHz and RMS normalized to a target level of -27dB, as outlined in [112]. Additionally, the silences were removed using the method described in [230]. The speaker pool shown in Figure 2.4 corresponds to the *LibriTTS-train-other-500* subset, which was used to extract target speaker representations. The process involved computing a speaker embedding for each utterance and then averaging these embeddings to obtain the target speaker representation for each individual. This process was applied for both the H/ASP in ZS-VC and the TDNN x-vector extractor in the B1a system. Unlike the approach taken in [201], this study incorporated pre-training for both the ASV and ASR models. Instead of training solely on the anonymized *LibriSpeech-360-clean* subset [68], the ASV speaker embedding model was first pre-trained on the VoxCeleb1 and VoxCeleb2 datasets [212], [213] based on the guidelines of [215], followed by fine-tuning on LS_{anon}^{360} for 20 additional epochs. Similarly, the ASR model was pre-trained on *LibriTTS-100-clean* and *LibriTTS-500-others* for 60 epochs, with further fine-tuning on LS_{anon}^{360} for an additional 30 epochs. The Adam optimizer was utilized with an initial learning rate of 0.001, alongside the learning rate scheduler from [36], and a batch size of 128. Pre-training and fine-tuning on LS_{anon}^{360} were conducted independently for each anonymization condition for both ASR and ASV models. The final anonymization results were evaluated using the *clean* development and test sets of *LibriSpeech*, as well as a subset of the *VCTK* dataset [211]. Furthermore, two distinct mapping strategies between the original and target speakers were explored in the ZS-VC system: one involved random speaker selection, and the other incorporated random selection with gender preservation. Additionally, the study tested two scenarios: first anonymizing with ZS-VC followed by applying B1a (*VB*, Table 1), and the reverse scenario where B1a was applied first (*BV*).

2.6.1 Primary Anonymization Metric

The privacy protection performance of each model was evaluated using the EER metric, and the corresponding results are presented in Table 2.3. As shown, the ZS-VC model, when used independently (columns V_R and V_{GP} in Table 2.3), delivers significantly better privacy protection compared to the baseline system (column B1a in Table 2.3), exhibiting a twofold increase in the EER, which indicates a substantial enhancement in its ability to anonymize data. This improvement highlights ZS-VC’s effectiveness as a stand-alone privacy solution. In addition, when the two anonymization systems are combined (as represented by columns VB and BV in Table 2.3), the privacy protection level exceeds that of the individual systems. The concatenated system, particularly in the VB configuration, achieves near-perfect anonymization, with an EER reaching 50%, which is the theoretical limit for effective anonymization, across various test and development datasets. This demonstrates that the integration of multiple systems can synergize their strengths and significantly enhance data privacy outcomes. Moreover, when gender characteristics are maintained (as reflected in the V_{GP} column of Table 2.3), the anonymization results are partially stronger compared to when targets are selected randomly (seen in the V_R column of Table 2.3). This suggests that preserving certain attributes, such as gender, during the anonymization process might enhance the system’s ability to obscure identity.

2.6.2 Primary Utility Metric

Linguistic information preservation was evaluated by calculating the WER using the ASR model detailed in Section 2.3.2 and fine-tuned for each anonymization condition individually. The WER results for each anonymization stage are presented in Table 2.4. As anticipated, all anonymization processes result in a degradation of ASR performance. However, when applying ZS-VC while preserving the gender of the original speakers after conversion, a 14.7% reduction in WER (averaged across development and test sets) is observed in comparison to random gender mapping. Furthermore, although two-stage processing (columns BV and VB in Table 2.4) tends to degrade WER in relation to single-stage processes, it can yield improved utility scores when ZS-VC is used as the second stage

Table 2.3: Primary anonymization metric (EER, higher is better) results for development (light gray) and test (dark gray) sets. Orig: Non-anonymized original data. Bla: baseline from [201]. V_R : ZS-VC with random target speaker selection. V_{GP} : ZS-VC with random target speaker selection and gender preservation. BV : Bla followed by ZS-VC. VB : ZS-VC followed by Bla. VCTK subsets comprise a set of utterances with same (comm.) and different (diff.) linguistic content across the speakers.

Set	Gender	Weight	EER[%]					
			Orig	Bla	V_R	V_{GP}	BV	VB
Libri-dev	female	0.25	4.12	14.6	21.6	22.3	32.95	52.6
	male	0.25	0.93	10.2	16.2	18.3	34.6	43.9
Vctk-dev	female	0.20	0.84	9.1	26.9	30.4	35.1	53.3
	diff.	0.20	0.64	8.1	27.9	30.0	19.52	41.4
Vctk-dev	female	0.05	0.87	10.2	26.8	28.7	25.6	42.4
	comm.	0.05	0.58	9.7	30.5	29.1	32.48	51.3
<i>Avg^W_{dev}</i>			1.63	10.64	23.3	25.1	30.7	47.8
Libri-test	female	0.25	2.55	12.7	18.8	14.8	31.8	46.0
	male	0.25	0.43	10.5	11.4	11.8	30.0	48.5
Vctk-test	female	0.20	1.59	14.7	30.4	32.3	29.8	35.2
	diff.	0.20	0.97	12.2	27.1	28.3	32.9	39.0
Vctk-test	female	0.05	0.34	13.8	29.5	30.9	27.8	41.6
	comm.	0.05	0.28	7.1	27.4	29.4	31.3	48.9
<i>Avg^W_{test}</i>			1.29	12.2	21.9	21.8	31.0	43.0

of the anonymization pipeline (column BV , Table 2.4)). This demonstrates that while multi-stage anonymization can negatively impact speech recognition accuracy, the strategic placement of certain models within the pipeline can enhance overall utility and maintain better linguistic preservation under specific configurations.

2.6.3 Secondary Utility Metric

Intonation preservation was evaluated by calculating the correlation coefficient ρ^{F_0} between the original and anonymized utterances. Although it has been established that F_0 contains speaker-specific information [231], all ZS-VC models exhibit high ρ^{F_0} values while simultaneously enhancing the EER. This indicates that effective anonymization can be achieved without significantly compromising prosodic features (see Table 2.5). Notably, two-stage anonymization methods achieve EERs comparable to state-of-the-art ASR+TTS anonymization systems [232], while significantly improving ρ^{F_0} . This improvement in

Table 2.4: Primary utility metric (WER, lower is better).

Set	WER[%]					
	Orig.	B_{1a}	V_R	V_{GP}	BV	VB
Libri-dev	2.33	2.77	4.22	4.04	4.72	6.51
Vctk-dev	8.21	9.59	14.58	13.95	16.24	19.39
<i>Avg dev</i>	5.27	6.18	9.4	9.0	10.5	12.95
Libri-test	2.47	2.85	3.84	3.78	4.17	7.19
Vctk-test	7.63	9.39	13.65	9.86	9.63	18.12
<i>Avg test</i>	5.1	6.12	8.75	6.82	6.92	12.7

prosody preservation is crucial, particularly when ASR models trained on anonymized data need to be applied to real speech [217], [218]. The ability to maintain prosodic information while anonymizing speaker identity enhances the practical applicability of these systems in real-world scenarios where both privacy and data utility are critical.

Table 2.5: Pitch correlation (ρ^{F_0} , higher is better). Average results computed with the same weights as in Table 2.3.

Set	Gnd	ρ^{F_0}				
		B_{1a}	V_R	V_{GP}	BV	VB
Libri-dev	F	0.77	0.8	0.81	0.8	0.75
	M	0.73	0.78	0.75	0.77	0.75
Vctk-dev	F	0.84	0.81	0.85	0.82	0.79
comm.	M	0.78	0.78	0.77	0.78	0.74
Vctk-dev	F	0.79	0.77	0.81	0.79	0.76
diff.	M	0.72	0.75	0.74	0.75	0.71
Avg dev		0.78	0.79	0.79	0.79	0.76
Libri-test	F	0.77	0.8	0.85	0.81	0.76
	M	0.69	0.71	0.72	0.73	0.67
Vctk-test	F	0.84	0.83	0.86	0.82	0.81
comm.	M	0.79	0.78	0.78	0.78	0.76
Vctk-test	F	0.79	0.8	0.83	0.79	0.77
diff.	M	0.70	0.74	0.72	0.73	0.71
Avg test		0.77	0.78	0.80	0.78	0.75

2.6.4 Privacy Utility Trade-Off

One of the key challenges in privacy evaluation lies in the simultaneous assessment of two interdependent factors: as privacy improves (indicated by a higher EER), it typically results in a decline in ASR performance, reflected by a higher WER (Tables 2.3 and 2.4). To address this trade-off, a new compressed metric introduced in Section 2.3.3, PU_{tr} , is proposed to balance both privacy and utility. Figure 2.4 illustrates the PU_{tr} results for various anonymization systems across different values of λ , which adjusts the weighting between privacy and utility. In cases where a low WER is critical ($\lambda = 0.9$), the baseline

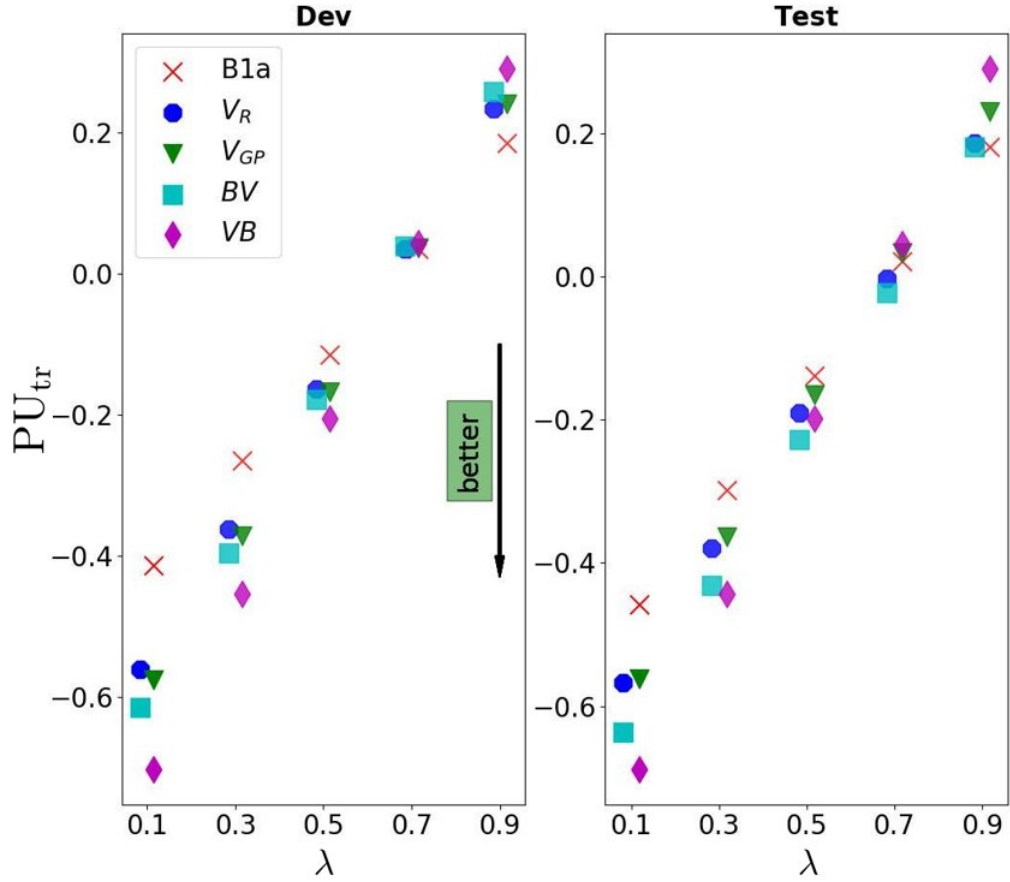


Figure 2.4: Privacy-to-utility trade-off for development (Dev) and test (Test) sets evaluated for $\lambda \in [0.1, 0.3, 0.5, 0.7, 0.9]$.

system B1a emerges as the optimal choice. However, in scenarios where stronger privacy is prioritized and a higher WER is acceptable (corresponding to lower values of λ), the ZS-VC model and two-stage processes offer a better balance in terms of the PU_{tr} metric. This highlights the flexibility of the proposed metric in accommodating varying levels of privacy requirements depending on the specific application.

2.7 Discussion and Conclusions

The Voice Privacy Challenge has significantly contributed to advancing research in speech privacy by providing a standardized framework for voice anonymization. Through well-defined protocols, datasets, and metrics, the challenge has allowed for consistent evaluation of anonymization systems effectiveness in concealing speaker identities while maintaining

speech utility. This framework has inspired the development of novel techniques designed to balance the competing objectives of privacy protection and speech utility preservation. Among the approaches explored, feature extraction, encrypted speech signals, and adversarial learning have each shown promise but are accompanied by notable limitations, particularly in ensuring privacy while retaining utility for downstream tasks. To address these challenges, state-of-the-art systems have focused on disentangling speaker identity from linguistic and prosodic content. This method, foundational to modern anonymization systems, has proven effective in altering speaker identity while preserving key speech attributes necessary for downstream tasks.

In this Chapter, a novel anonymization pipeline was presented, consisting of two fully neural anonymization systems. This pipeline combines zero-shot voice conversion with a state-of-the-art anonymization model, providing a robust solution to speaker anonymization. The two-stage process demonstrated that it is possible to preserve prosodic information—such as intonation and rhythm—while concealing speaker identity. This method achieves EER scores comparable to advanced ASR+TTS anonymization systems, underscoring its potential as a highly effective anonymization technique. The evaluation of privacy protection, using the EER metric, revealed that the ZS-VC model, when used independently, significantly enhances privacy protection compared to baseline systems. Moreover, the two-stage anonymization processes provided an even stronger level of protection, with the potential to reach near-perfect anonymization in some configurations. Gender-preserving configurations, in particular, demonstrated enhanced performance, emphasizing the importance of preserving certain speaker characteristics during anonymization to achieve a better balance between privacy and utility. In terms of utility, WER assessments showed that anonymization degrades ASR performance. However, preserving gender during anonymization with ZS-VC resulted in a WER reduction compared to random speaker selection. Additionally, the two-stage anonymization processes demonstrated that, when strategically structured, it can improve utility scores while maintaining privacy protection. To evaluate prosodic preservation, the pitch correlation metric ρ^{F_0} was computed. Results showed that the two-stage systems were able to achieve high levels of anonymization without compromising prosodic features. This is particularly important for applications

where anonymized data are used to train ASR models that must generalize to real speech, as preserving prosody enhances robustness and model adaptability. A key contribution of this research is the introduction of the Privacy-to-Utility Trade-Off (PU_{tr}) metric, which compresses the privacy and utility evaluation into a single score. This metric combines EER and WER, allowing system designers to adjust the balance between privacy protection and speech utility by tuning the λ parameter. Depending on the specific anonymization requirements—whether stronger privacy protection (higher EER) or better speech utility (lower WER) is prioritized—the optimal anonymization technique can be selected. As demonstrated in Figure 2.4, baseline systems excel in scenarios where maintaining low WER is critical, while ZS-VC and two-stage processes perform better when stronger privacy protection is required.

In conclusion, this work has demonstrated that the combination of zero-shot voice conversion and a state-of-the-art anonymization model in a two-stage pipeline offers a powerful solution for balancing privacy and utility. The introduction of the PU_{tr} metric provides a flexible tool for evaluating and selecting anonymization techniques based on specific application needs.

Chapter 3

Accented Speech Simulation: From Data Augmentation To Privacy

In recent years, ASR models have seen significant advancements, improving transcription accuracy in both clean, low-noise environments and more challenging reverberant conditions. However, these systems heavily rely on the availability of large, labeled datasets tailored to specific acoustic environments. When such data is limited, ASR performance deteriorates, particularly for underrepresented acoustic settings or speaker populations in the training data. This issue is especially pronounced when dealing with accented speech. In this chapter, we examine the impact of accented speech on the performance of a commercial ASR system and introduce a zero-shot text-to-speech approach to generate synthetic training data for accented speech. By producing synthetic data that mimics various accents, this method addresses the challenge of data scarcity in accented speech and enhances the system's ability to handle diverse speech patterns. In addition to improving ASR robustness, the generation of synthetic accented data can promote fairness across different subpopulations and ethnic groups. Moreover, this technique holds potential for privacy protection. Using voice conversion models to modify speech accents, users can obscure their identity while maintaining intelligibility. By converting a speaker's natural accent to a different one, the system adds an extra layer of security, making it more difficult for malicious actors to identify or track the speaker based on vocal characteristics. This

dual benefit of fairness and privacy underscores the critical role of accent modification and synthetic data generation in modern ASR applications. An earlier version of this work was presented in [233].

3.1 Accented Speech Recognition

English is one of the most widely spoken languages in the world [234] and, like many other languages, it is diverse and includes multiple dialects. Modern ASR systems require large volumes of carefully annotated speech data to produce accurate transcription results [235], [236]. Datasets like LibriSpeech [68] are frequently employed as essential resources during training. Despite these advancements, a marked reduction in performance is often observed when ASR systems are deployed in real-world scenarios. This performance drop is attributed to several factors, including varied acoustic environments [237] and the under-representation of certain language groups within the training data [238]. Although bias in training data does not always lead to biased models [239], subtle speech variations such as regional accents [240] and demographic characteristics like racial or ethnic backgrounds [241] may still introduce unintended biases. Furthermore, empirical studies demonstrate that ASR models primarily trained on native English speech data tend to underperform when confronted with non-native accents, highlighting the inadequacy of such models to generalize effectively across underrepresented speech patterns [238] raising ethical concerns as this bias can disproportionately affect marginalized communities and immigrants [242], [243]. Moreover, ASR systems exhibiting bias against non-native English accents is particularly alarming given the growing number of L2 English speakers [244], [245]. Similar trends are observed in other high-resource languages [246]. These findings underscore the necessity for more inclusive model development approaches.

Several techniques have been proposed to mitigate the degradation in ASR performance due to accented speech. A possible solution is to include more varied English speakers to adapt current ASR models to better handle different accents [247]. This, however, requires labeled speech data for various English accents, which is still scarce [248]. Another method, as detailed in [249], involves introducing accent-specific embeddings, which are computed

from pre-trained models and provided as additional input to the ASR system. This method has been shown to improve robustness by allowing the model to better accommodate accent variations.

Another approach involves the use of domain adversarial neural networks (DANN) [250]. In this framework, a domain discriminator is used alongside the ASR model to encourage the extraction of domain-invariant features. The discriminator is trained to distinguish between source and target domains based on the feature representations, while the ASR model learns to produce task-relevant predictions. During optimization, the loss from the domain discriminator is incorporated into the ASR loss, encouraging the ASR model to generate domain-agnostic representations, thereby reducing the mismatch between accented and standard speech features [239].

Multi-Task Learning (MTL) [251] is another approach offering a promising framework to enhance ASR performance on accented speech by jointly optimizing multiple tasks that share related information [252], [253]. In this context, an ASR model can be trained alongside an accent classification task, enabling the model to learn accent-specific features while simultaneously focusing on the main transcription task. The model is encouraged to extract shared representations that are useful for both tasks, leading to improved robustness when encountering different accents. MTL also helps the ASR system generalize better to unseen accents [253], thus mitigating performance degradation on accented speech.

Moreover, self-supervised learning [254] approaches have been applied for improving ASR performance on accented data. These methods focus on leveraging vast amounts of unlabeled speech data to learn generalizable representations that can then be fine-tuned for specific tasks like accent identification and accented ASR. By pre-training models on large datasets, these approaches reduce reliance on labeled data, which is often scarce for accented speech. Techniques like wav2vec 2.0 [255] learn rich acoustic features from raw audio without supervision, which can then be fine-tuned on smaller accented datasets to boost ASR performance, even for unseen accents [256], [257]. The use of accent embeddings and frame-level adjustments enables the model to dynamically adapt to accent-specific variations, improving transcription accuracy across diverse accents while maintaining the

flexibility to handle untranscribed data [257].

3.2 TTS Accented Data Augmentation for ASR Applications

To improve ASR performance for various accents, particularly for underrepresented ones, researchers have also explored data augmentation techniques [258]. One promising approach is using synthetic speech generated by TTS systems to simulate accented speech for ASR model training [259]–[262]. This method has gained attention because collecting real-world, transcribed speech data for a wide range of accents is often expensive and time-consuming. By using TTS systems, speech data for different accents can be produced more efficiently and at scale, offering a potential solution to the scarcity of authentic accented speech data.

In [259], the authors propose a novel method where English inputs are passed through a TTS system pre-trained in another language (the L1 of the target accent) to produce synthetic L2-accented English speech. For example, passing English text through a Spanish TTS system generates speech that mimics a Spanish accent in English. This method allows the creation of synthetic accented speech, which can then be used to fine-tune pre-trained ASR models, improving their performance on accented speech. The specific method involves using an existing pre-trained English ASR model and fine-tuning it with a combination of synthetic accented speech generated via TTS and, optionally, a small amount of authentic accented data. This approach can reduce error rates in ASR systems, with better results seen when more than one hour of authentic accented speech is available.

Other approaches explore semi-supervised [262] and unsupervised learning [261] techniques for improving accented and low resource languages speech recognition. The self-supervision approach presented in [262] leverages large volumes of unlabeled speech and text data, using TTS-based data augmentation to generate synthetic speech for training ASR models. By factorizing TTS models across multiple languages, speaker attributes, and prosodic features, the approach enhances speech variability and lexical diversity in the training data, leading to better performance in ASR across multiple interrelated and linguistically similar local languages. This iterative method combines pseudo-labeled speech

with labeled data to improve ASR models. The results show significant improvements, with up to a 14.4% relative reduction in WER across four low-resource speech scenarios. In the case of unsupervised TTS a small dataset of accented speech with no transcriptions is used to train TTS systems, generating pseudo-labeled accented speech data. These synthetic datasets, combined with non-accented speech data, are then used to fine-tune ASR models [261]. This method is particularly valuable in scenarios where obtaining transcribed speech data is costly and time-consuming. The results demonstrated a 6.1% relative reduction in WER compared to the baseline, showing that synthetic accented speech can significantly improve ASR models when combined with self-supervised learning offering a scalable solution to the data scarcity problem for accented speech recognition. These two approaches both leverage synthetic speech, either through unsupervised learning or multilingual TTS models, to address the challenges of data scarcity and accent variability in ASR systems.

The approach presented in this chapter, builds on concepts from previous work, leveraging synthetic speech through a multi-accent TTS model to address data scarcity and accent variability in ASR systems. Unlike the multilingual TTS models used in prior studies, this method trains a multi-accent TTS model by fine-tuning an existing multilingual TTS on a limited English-accented speech dataset [263], re-purposing the learnable language embeddings as accent embeddings.

3.3 Accent Simulation with Zero Shot TTS

In this section, we introduce a Zero-Shot Text-To-Speech (ZS-TTS) system as a method for data augmentation. ZS-TTS is particularly advantageous for accented speech corpora because it can capture the unique acoustic features of individual speakers using only a few seconds of audio. This capability is especially valuable given the limited availability of data in many accented speech datasets. Additionally, ZS-TTS can generate an extensive number of speech samples for each speaker, providing a robust solution for expanding the training dataset and mitigating the gap between training and real-world production data. In this study, we investigate the ZS-TTS augmentations provided by the system

[112] described in detail in Section 2.5.2 with a transformer-based ASR model from [215] presented in Section 2.3.2. These models were chosen due to their open-source availability, simplicity in training, and support for multiple languages, which made them well-suited for addressing the complexities of multi-accent speech recognition. The ZS-TTS model is also capable of learning multiple languages, achieved by incorporating trainable language embeddings. These embeddings, consisting of four dimensions, are appended to each input character’s representation, producing a language-specific character embedding. In cases where multiple accents are present, the language embedding is repurposed to serve as an accent embedding. To implement this, we begin with a publicly available multilingual checkpoint, which includes languages such as French, Portuguese, and English [112]. We then randomly initialize the existing N language embedding layers. For datasets containing M accents, additional $|M - N|$ random embeddings are introduced to account for each accent ensuring that all accents in the training set are adequately represented. Furthermore, by incorporating accent embeddings, the model learns accent-specific character representations, leading to a more contextually sensitive and linguistically adaptive system.

3.4 Experiments

Both single and multi-accent experiments were carried out to evaluate the performance of the model across various linguistic variations. For the single accent trials, data from the VCTK corpus [211] was employed, specifically focusing on recordings from three English speakers with an Indian accent. In contrast, the multi-accent experiments leveraged the dataset from the Interspeech 2020 Accented English Recognition Challenge [263], which comprises recordings from ten distinct accent groups: United States (USA), Canadian (CAN), Great Britain (GBR), Chinese (CHN), Japanese (JPN), Korean (KOR), Russian (RUS), Portuguese (PRT), and Spanish (ES). These recordings were collected using smartphones and other commercially available devices, where speakers read pre-determined texts. This method ensured consistency in content while capturing a broad spectrum of accents. The dataset is structured to provide approximately 20 hours of training data and 2 hours of testing data for eight of the accents, namely USA, GBR, CHN, JPN, KOR, PRT, RUS,

and CAN. For the remaining accents, specifically CAN and ES, training data is absent, but test sets of equivalent duration are provided. This variation in data availability creates an opportunity to test the model’s robustness when training data is limited or unavailable for certain accents. Consequently, the experimental design allows for an analysis of the model’s capacity to generalize across both high-resource and low-resource accents, offering insights into its adaptability to different linguistic conditions.

3.4.1 Single Accent

ASR Fine-Tuning on Mixed Real and TTS Data

Preliminary experiments, involved only a single accent namely, the Indian English accent. The ASR model was initially trained using the full 960 hours of the LibriSpeech dataset [68]. The model achieved a WER of 2.34% on the *test-clean* partition and 5.55% on the *test-others* partition of LibriSpeech, indicating robust performance on standard English speech data. However, when the same ASR model was applied to the decoding of the *test-IND* partition from the Interspeech 2020 dataset, which specifically contains Indian-accented English, a significant degradation in performance was observed, with the WER increasing to 22.24%. This decline in accuracy highlights a substantial mismatch between the acoustic and linguistic features present in the training data and those of the Indian-accented test data suggesting that adaptation techniques are necessary to bridge this gap and improve the model’s ability to generalize to diverse and previously unseen linguistic conditions.

Given these considerations, the ASR model was further fine-tuned using the entire *train-IND* subset for an additional 10 epochs before decoding the *test-IND* partition. This process resulted in a WER of 8.33%, which serves as the baseline or lower bound for subsequent TTS augmentation experiments involving Indian English accents. Two ZS-TTS models were considered for the augmentation experiments. The first model (TTS_c, Table 3.1) is a pre-trained model provided by [112], trained on a combination of the LJ-Speech dataset [264], VCTK [211], the TTS-Portuguese Corpus [265], and a portion of the M-AILABS dataset [266]. The second model (TTS_I, Table 3.1) is derived from this pre-trained checkpoint by continuing the training for 300 additional epochs, using approximately 1.5

Table 3.1: ASR decoding results for the *test-IND* set. The ASR model was fine-tuned on various partitions of *train-IND* that were augmented with TTS-generated data. R indicates the model trained exclusively on real data (some % of *train-IND*). R – TTS_c represents the model trained on real data combined with synthetic data generated by TTS_c. R – TTS_I represents the model trained on real data combined with synthetic data generated by TTS_I. The final column shows the relative WERR achieved by each augmentation approach in comparison to the model without augmentation. For reference, the line below the table reports the WER for a model trained exclusively on the full partition of real data (*train-IND*), representing the lower bound for ASR models trained on a mix of real (< 100%) and synthetic data.

real data	WER[%]			WERR[%]
	R	R-TTS _c	R-TTS _I	
10%	13.29	11.82	10.98	11/17
25%	11.80	10.34	10.15	12/14
50%	10.51	9.39	9.40	11/11
75%	9.74	8.81	8.64	9.5/11
90%	9.53	8.59	8.53	11/12
model trained on full <i>train-IND</i> set: 8.33% WER				

hours of speech from the three Indian speakers in the VCTK dataset, distributed equally across them. To explore different low-resource scenarios, various fractions of the original *train-IND* dataset were sampled, and the data was augmented up to 20 hours using speech generated by both TTS models. The text inputs for the TTS models were specifically sampled from the original *train-IND* set, ensuring consistency in the textual content across all experimental conditions. This careful selection process plays the role of controlling for the textual content, allowing the experiments to focus on the impact of the acoustic properties rather than variations in text. To generate the results for the 25% mixing condition shown in Table 3.1, 25% of the *train-IND* dataset was randomly selected and augmented with synthetic speech generated by both TTS models. The synthetic speaker for each TTS-generated utterance was also randomly chosen from the three Indian speakers in the VCTK dataset. Following the data augmentation, the ASR model, originally trained on the full 960-hour LibriSpeech dataset, was fine-tuned using each of the augmented datasets. Decoding was then performed on the *test-IND* partition with each of these fine-tuned models. The results are summarized in Table 3.1, which shows that the WERs from ASR models fine-tuned with augmented synthetic data are lower than those from models trained on real data alone (R). Notably, the model fine-tuned with data generated by the Indian

speakers from the VCTK dataset (TTS_I) achieves a greater WERR compared to the model trained with the publicly available checkpoint [112]. This suggests that using synthetic data tailored to the specific accent results in improved ASR performance.

ASR fine-tuning on TTS data

The ASR model, initially trained on the full 960-hour LibriSpeech dataset, was further fine-tuned using only TTS-generated data. The outcomes, as presented in Table 3.2, indicate that under these conditions, only the TTS model fine-tuned on the three Indian speakers (TTS_I) managed to achieve a lower WER compared to the original ASR model trained solely on LibriSpeech 960 (see dark grey rows, Table 3.2). However, this improvement is observed only when the amount of TTS-generated data used for fine-tuning is limited. It is suggested that this is due to the ASR model’s tendency to overfit when exposed to large volumes of TTS-generated data, which diminishes its ability to generalize and capture the characteristics of natural speech found in real-world corpora. This overfitting likely leads to a degradation in performance as the model becomes more reliant on the synthetic patterns in the TTS data, rather than the acoustic diversity present in actual speech recordings. Moreover, this experiments shows that augmenting ASR models with synthetic speech alone is not as effective as combining it with real accented data, and diversity in the prompts used for TTS synthesis is crucial for successful model adaptation.

3.4.2 Multiple accents

This Section expands the focus to include multiple accents, beyond just Indian-accented English. The dataset used for these experiments is the Interspeech 2020 Accented English Recognition Challenge dataset [263], as previously detailed in Section 3.4. For each accented partition, the cleanest recordings were selected based on the metadata annotations accompanying the audio files. However, this initial selection led to an imbalance in the subset sizes. To address this, the duration of all partitions was uniformly reduced to match that of the shortest accented subset, which was approximately 11 hours. This balancing was achieved through a random sampling process. To establish baseline results, the ASR

Table 3.2: ASR decoding results for the *test-IND* set using an ASR model fine-tuned exclusively on TTS-generated data. TTS_c: model fine-tuned with synthetic data generated by TTS_c. TTS_I: model fine-tuned with synthetic data generated by TTS_I. The final column presents the WERR achieved by each augmentation method in comparison to the model trained solely on the [68] dataset. For reference, the line below the table shows the WER for a model trained exclusively on LibriSpeech 960, representing the upper bound beyond which fine-tuning with synthetic data alone begins to degrade ASR performance on genuine accented speech.

tts data	WER[%]		WERR[%]
	TTS _c	TTS _I	
1.4 hours	23.67	21.13	-6.4/5.0
3.5 hours	25.91	21.90	-16.5/1.5
7 hours	28.44	22.5	-27.9/-1.2
10.5 hours	30.18	23.38	-35.7/-5.1
12.6 hours	31.47	24.20	-41.5/-8.8

model trained on real data only: 22.24% WER

model (referred to as the *C* model) was first trained on the entire 960-hour LibriSpeech dataset [68]. Initially, each accented test subset was decoded using the *C* model, with the results shown in Table 3.1 (first row). Next, the *C* model was fine-tuned on each accented training subset individually (*FT*, Table 3.1, second row). Finally, the model *C* was fine-tuned on a combination of all accented training subsets (*FM*, Table 3.1, third row). The decoding results for each accented test subset are provided in the upper section of Table 3.1 (light grey rows). To train the TTS model to generate speech with different accents, the same training subsets from the Interspeech 2020 Accented English Recognition Challenge dataset [263] were used. The training was initiated from the publicly available checkpoint [112] and continued for an additional 300 epochs. During this process, the TTS model was guided to learn distinct accent embeddings for each accent. To achieve this, the language embedding layers were modified to treat accents as distinct languages, and the previously learned language embedding weights were randomly re-initialized for the new accent embeddings. Based on the results obtained for the single accent scenario (Table 3.2, second row), 75% TTS-generated data was mixed with 25% real data for all multi-accent experiments (Table 3.1, lower section, dark grey rows). The same ASR experiments as those conducted for the non-augmented sets were repeated, and the corresponding results are presented in Table 3.3 (dark grey rows). In this case, it was observed that the TTS

Table 3.3: ASR decoding results for the Interspeech 2020 test sets. The upper section (light grey) presents ASR results using only real data. *C*: model trained on the 960-hour LibriSpeech dataset [68]. *FT*: model *C* fine-tuned separately on each accented training set. *FM*: model *C* fine-tuned on a combination of all accented training sets. The lower section (dark grey) shows ASR results with TTS data augmentation (75% TTS / 25% real data). FT_{aug} and FM_{aug} correspond to the previously mentioned models with TTS augmentation. FM_{aug+db} : FM_{aug} with batches composed of 75% TTS and 25% real data.

Model	WER[%]										
	GBR	USA	CAN	PRT	IND	CHN	ESP	KOR	RUS	JPN	Avg
<i>C</i>	7.47	17.74	18.26	19.28	23.40	24.03	26.35	33.78	35.75	36.97	24.3
<i>FT</i>	2.32	8.18	X	6.67	9.46	9.97	X	7.16	10.15	7.36	7.66
<i>FM</i>	2.06	4.95	4.51	5.31	8.51	9.87	7.39	5.40	8.98	6.31	6.33
TTS-AUG											
FT_{aug}	2.42	6.82	X	6.65	9.21	11.96	X	7.21	10.87	7.90	7.88
FM_{aug}	2.22	5.46	4.72	5.62	8.64	10.74	8.14	5.92	9.65	6.82	6.79
FM_{aug+db}	2.25	5.38	4.63	5.50	8.50	11.21	8.00	5.85	9.47	6.87	6.77

model trained on multiple accented datasets, combined with accent embeddings, did not lead to improved ASR performance when used for data augmentation. This could be due to the TTS model’s limitations in capturing the full range of accent-specific pronunciations via accent embeddings or the relatively lower quality of the Interspeech 2020 corpus. To investigate the latter hypothesis, the TTS_c model was fine-tuned on three Indian speakers selected from the Interspeech 2020 dataset for further analysis.

Table 3.4: WER results for the *test-IND* set. R – TTS_{INI}: ASR model fine-tuned on a combination of real and synthetic speech, with the synthetic data generated by a TTS model fine-tuned on a small subset (3 speakers) of the *train-IND* dataset.

real data	WER[%]	
	R	R-TTS _{INI}
10%	13.29	11.68
25%	11.80	10.31
50%	10.51	9.55
75%	9.74	8.95
90%	9.53	8.86

The experimental results, presented in Table [3.4], indicate that the augmentation performance of this model was either comparable to or worse than the pre-trained model (R-TTS_C) from [112] (as shown in Table [3.2]). This suggests that the lower data quality of

the Interspeech 2020 corpus, in comparison to the VCTK speakers, significantly affects the quality of the synthetic accented speech used for augmentation, which in turn impacts the overall effectiveness of the ASR model’s performance improvements.

3.4.3 Combine TTS with other audio augmentations

In the final set of experiments, the effects of common speech augmentation techniques namely, Room Impulse Response (RIR), noise, speed, and SpecAugment [267] were evaluated both individually and in combination with TTS-based augmentation. During the training process, augmentations were applied independently to all utterances in each batch, after which the original and augmented samples were concatenated to form a new batch. For RIR and noise augmentation, the OpenRIR dataset [268], which includes both real and simulated RIRs and noises, was utilized.

Table 3.5: WER results for the *test-IND* set. R_{aug} : ASR model fine-tuned on real data augmented using the methods described in Section 3.4.3. $R_{aug} - TTS_{INI}$: ASR model fine-tuned on a combination of real and synthetic augmented speech, with the synthetic speech generated by a TTS model adapted to a subset of 3 speakers from the *train-IND* dataset.

real	WER[%]		
	R	R_{aug}	$R_{aug} - TTS_{INI}$
10%	13.29	13.52	11.03
25%	11.80	11.90	10.08
50%	10.51	10.08	9.66
75%	9.74	9.33	9.10
90%	9.53	9.00	9.04

As shown in Table 3.5, results indicate that conventional speech augmentation techniques alone did not enhance ASR performance when dealing with accented speech (R_{aug} , Table 3.5). Moreover, the combination of traditional speech augmentations with synthetic TTS data appeared to negatively impact ASR performance. Specifically, for the R_{aug} model, traditional augmentation resulted in a degradation in performance for smaller data partitions (10% and 25%) but provided some improvement in WER for larger data sets. Conversely, in the case of $R_{aug} - TTS_{INI}$, the opposite trend was observed: improvements were noted for smaller partitions, while performance deteriorated on larger sets. These

findings suggest that while conventional augmentation methods may benefit larger datasets, combining them with TTS-based augmentation can introduce complications that hinder ASR performance, particularly with accented speech.

3.5 Accent Simulation and Privacy

This section presents the findings on employing the ZS-TTS system for privacy preservation in the context of accented speech. The ZS-TTS system, as described in Section 3.2 was used to anonymize speech data, following the approach outlined in [209]. To achieve speaker anonymization, the system first transcribes the original speech. Subsequently, the transcribed text is used to generate a new speech signal, where the speaker identity is replaced with a new synthetic identity, and accent information is retained or modified as needed. This process effectively removes identifying features from the original acoustic signals, while preserving linguistic content and accent characteristics. For the remainder of this chapter, this configuration will be referred to as ASR-TTS.

3.6 Privacy Preservation Through Accent Modification

ASR combined with TTS systems has traditionally been considered a relatively simplistic method for anonymization [208]. However, recent advancements in ASR technologies [236], TTS systems [116], [269], [270], and the increasing relevance of synthetic data in various applications [271]–[274] present new possibilities for improving this approach. Conventional ASR-TTS anonymization techniques [232] often result in the standardization of all utterances to a single accent, which is problematic when the anonymized speech is intended for tasks requiring accent diversity [217], [218]. This homogenization can introduce bias, skewing the anonymized data toward a particular accent and diminishing the effectiveness of ASR models trained on such data, especially when exposed to the variety of accents encountered in real-world scenarios [233]. The primary motivation for incorporating accent variability into ASR-TTS anonymization systems is to overcome this limitation. Retaining or introducing diverse accents into anonymized speech ensures that

the resulting data better represents the linguistic diversity present in real-world speech. This not only enhances fairness by reducing bias against underrepresented accents but also improves the utility of anonymized data for various downstream tasks, as demonstrated in Section 3.3.

In this section, the privacy implications of accent modification are explored, with the hypothesis that such techniques could provide an additional layer of security in anonymized data. Accent modification is investigated for its dual potential: improving model fairness and strengthening privacy preservation. Specifically, TTS models can modify speech accents, adding an additional layer of protection by making speaker identification based on vocal characteristics more challenging. This dual advantage—enhancing both fairness and privacy—underscores the importance of accent modification in modern ASR applications, ensuring that anonymized data remains both secure and representative of diverse linguistic backgrounds.

3.7 Experiments

In Section 3.7.1, the concept of leveraging learned character embeddings from multiple languages to simulate accents through interpolation was investigated. This approach utilized the pretrained model outlined in [112]. For instance, to simulate the speech of an English speaker with a French accent, a linear interpolation of English character embeddings with French embeddings was performed, as illustrated in Fig. 3.1. This methodology allows for the modulation of the French accent’s intensity in relation to the English base by appropriately selecting the weights a and b in the interpolation equation $interp = a \times Eng + b \times Fr$, where Eng and Fr represent the English and French character embeddings, respectively. In the initial experiment, oracle transcriptions were utilized exclusively, focusing on TTS synthesis and language interpolation. Following this, a wav2vec2.0 ASR model [215] was employed to transcribe the utterances, with the resulting transcriptions serving as inputs to the TTS model. The utility of the anonymized data was subsequently assessed using the pipeline presented in Section 2.3 where the ASR and the ASV model are fine-tuned on an anonymized version of LibriSpeech-360 before scoring the

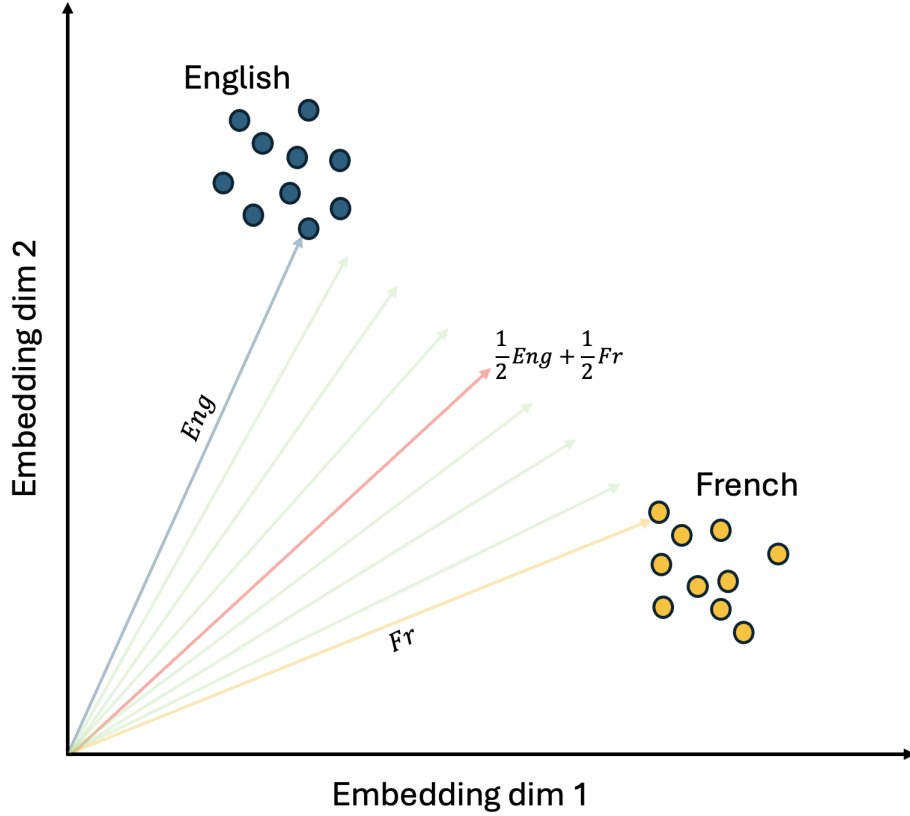


Figure 3.1: Language embeddings interpolation for accent simulation: 2D example. The degree of accentedness in speech can be adjusted by varying the relative weights of the English and French embeddings.

test sets [209]. Regarding the utility score, the ASR approach presented in Section 2.3 was used. For the privacy evaluation, the speaker verification framework [84] was implemented, specifically using the formulation from [215] that incorporates cosine similarity as the scoring function, diverging from the PLDA method outlined in Section 2.3. Subsequently, the accent embedding capabilities of the TTS system described in Section 3.3 were assessed as effective anonymizers, with evaluations conducted using both oracle and ASR transcriptions.

3.7.1 Language Interpolation

In the initial experiment, oracle transcriptions were utilized exclusively to apply TTS synthesis and language interpolation techniques (denoted as O-TTS). The results corresponding to this experimental condition are presented in Table 3.6 and Table 3.7. In the

case of the English accent (en) in Table 3.6, no language interpolation is applied; instead, the default character embeddings for English are utilized. Notably, the results from the O-TTS system demonstrate a significantly lower WER for the English accent compared to the original speech recordings, a trend particularly evident within the VCTK subsets. This phenomenon has been previously reported in [223] and can be explained by the interplay of two factors. Given that the VCTK corpus comprises various accents, the O-TTS system eliminates original accent characteristics in the synthesized outputs. Therefore, the observed reduction in WER can be attributed to the ASR model’s predominant pre-training on American-accented speech (*Libri-TTS 100-clean*, *LibriTTS 500-others*), followed by fine-tuning on synthetic standard English-accented data (LS_{anon}^{360}). This finding suggests that the O-TTS methodology successfully attenuates the presence of original accent information in synthesized recordings, which is crucial, as accents can serve as markers of speaker identity. The disparity observed for the Portuguese and the French accents can be again attributed to two main factors: first, language embedding interpolation introduces perceptible distortions in the synthesized audio, and second, the fine-tuning performed on the LS_{anon}^{360} dataset does not sufficiently address the limitations of the ASR model, which was not originally trained on accented speech.

Table 3.6: WER results for language embedding interpolation and oracle transcription (O-TTS) for en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding.

set	WER[%]				
	orig	en	pt	fr	random
libri-dev	2.33	1.89	3.46	4.57	4.81
vctk-dev	8.21	3.43	6.47	7.52	8.49
Avg – dev	5.27	2.66	4.97	6.05	6.65
libri-test	2.47	1.92	3.44	4.41	4.89
vctk-test	7.63	3.32	6.11	7.45	8.27
Avg – test	5.05	2.62	4.78	5.93	6.58

With respect to the privacy evaluation results, it was observed that the highest EER originated from the French and random interpolation schemes, as shown in Table 3.7. Although language embedding interpolation appears to enhance privacy protection, it

introduces a trade-off with speech utility, as evidenced by WER metrics. The privacy-utility trade-off is a recognized phenomenon within the domain of voice privacy [201], [209], and identifying the optimal system in terms of this trade-off is often complex. To address this, the PU_{tr} measure [200] was computed for all methodologies presented in this section; results pertaining to language embeddings with oracle transcriptions are depicted in the first row of Fig. 3.2. According to this metric, the English accent maintained the most favorable PU_{tr} across all values of $\lambda \in [0.1, 0.3, 0.5, 0.7, 0.9]$, primarily due to the low WER scores produced by the ASR-TTS method when applied to English speech.

Table 3.7: EER results for language embedding interpolation and oracle transcription. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding.

Set	Gender	Weight	EER[%]				
			orig	en	pt	fr	random
Libri-dev	female	0.25	2.70	45.74	42.88	46.31	45.91
	male	0.25	0.34	49.09	38.10	49.71	37.27
Vctk-dev	female	0.20	0.44	48.78	48.85	46.64	47.12
diff.	male	0.20	0.45	42.24	43.31	44.57	45.66
Vctk-dev	female	0.05	0.51	44.71	46.42	43.61	46.80
comm.	male	0.05	0.02	49.86	46.70	47.01	47.59
<i>Avg^W_{dev}</i>			0.97	46.64	43.34	46.78	44.43
Libri-test	female	0.25	0.37	46.00	43.40	48.36	48.52
	male	0.25	0.89	49.93	49.09	47.66	49.67
Vctk-test	female	0.20	0.93	47.68	45.49	49.12	43.49
diff.	male	0.20	0.41	42.58	43.64	44.27	46.84
Vctk-test	female	0.05	0.82	45.16	43.53	46.53	46.77
comm.	male	0.05	0.26	42.10	42.88	40.37	44.77
<i>Avg^W_{test}</i>			0.64	46.40	45.27	47.10	47.19

Subsequently, the focus shifted from oracle transcriptions to the implementation of actual

ASR-TTS methods. Initially, all data were transcribed using a wav2vec2.0-based ASR model¹. The same TTS framework employed previously was then applied. The anonymized data were subsequently evaluated using the ASR and ASV models used in the case of oracle transcription. The results from these ASR and ASV scoring models are presented in Table 3.8 and Table 3.9. When comparing the WER results, we do not observe statistical difference between the ASR results for O-TTS in Table 3.6 and ASR-TTS in Table 3.8 (T-test, p-value=0.95). A potential explanation for this is that the errors made by the wav2vec2.0-based ASR model (ASR_{wav}) may form a subset of the errors made by the scoring ASR (ASR_{score}). To test this hypothesis, all development and test sets from LibriSpeech and VCTK were transcribed using both ASR models. For each sentence, the sets of errors $Set_{ASR_{wav}}^{err}$ and $Set_{ASR_{score}}^{err}$ produced by each model were extracted. To facilitate comparison of these two sets, an inclusion score (IS), defined as the ratio of the size of the intersection of the two sets to the size of $Set_{ASR_{wav}}^{err}$, was computed. For both the development and test sets, the inclusion scores were found to be above 0.9 ($IS_{dev} = 0.964$, $IS_{test} = 0.981$). This suggests that nearly all errors identified in $Set_{ASR_{wav}}^{err}$ are also present in $Set_{ASR_{score}}^{err}$ indicating that $Set_{ASR_{wav}}^{err} \subset Set_{ASR_{score}}^{err}$.

Table 3.8: WER results for language embedding interpolation and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with wav2vec2.0 ASR.

set	WER[%]			
	en	pt	fr	random
libri-dev	1.79	3.12	4.33	4.33
vctk-dev	3.93	6.60	8.42	8.82
Avg – dev	2.86	4.86	6.38	6.58
libri-test	1.85	3.08	4.37	4.59
vctk-test	3.24	5.84	7.35	7.89
Avg – test	2.55	4.46	5.86	6.24

Regarding the EER results for ASR-TTS, they are presented in Table 3.9. In contrast with the WER statistical results documented in Table 3.8 and Table 3.6, there is insufficient evidence to assert that a statistically significant difference exists between the means of the

¹Additional information: <https://huggingface.co/speechbrain/asr-wav2vec2-librispeech>

two EER populations from O-TTS in Table 3.7 and ASR-TTS transcriptions in Table 3.9 (T-test, p-value=0.3).

Table 3.9: EER results for language embedding interpolation and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with wav2vec2.0 ASR.

Set	Gender	Weight	EER[%]			
			en	pt	fr	random
Libri-dev	female	0.25	47.58	43.76	43.05	43.91
	male	0.25	49.06	47.67	45.26	46.27
Vctk-dev	female	0.20	46.37	34.81	46.48	48.74
	diff.	0.20	47.69	38.37	45.25	42.83
Vctk-dev	female	0.05	49.13	42.73	48.20	46.83
	comm.	0.05	41.60	48.15	49.30	44.73
<i>Avg^Wdev</i>			47.51	42.04	45.08	45.44
Libri-test	female	0.25	48.33	46.72	47.74	41.42
	male	0.25	49.22	44.99	43.64	46.77
Vctk-test	female	0.20	36.21	48.82	40.11	49.95
	diff.	0.20	45.62	39.10	41.62	47.65
Vctk-test	female	0.05	47.67	44.73	44.51	48.27
	comm.	0.05	44.63	42.92	45.76	43.28
<i>Avg^Wtest</i>			45.37	44.90	44.24	46.15

3.7.2 Accent Embeddings

In this section, the TTS model presented in Section 3.3 was used in the context of ASR-TTS anonymization. Utility and privacy metrics were computed with the same systems as in Section 3.7.1. Results for this condition are shown in Table 3.10 and Table 3.11. In this case, the TTS model [112] was trained on the English-accented data provided in [263], but no French-accented English data was available. Consequently, it was not possible to

Table 3.10: WER results for the accent embedding case and ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with wav2vec2.0

set	WER[%]			
	ind	pt	usa	random
libri-dev	3.08	3.46	3.34	4.07
vctk-dev	3.53	3.39	3.23	4.14
Avg – dev	3.31	3.43	3.29	4.11
libri-test	3.51	3.81	3.68	4.50
vctk-test	2.95	2.74	2.70	3.51
Avg – test	3.23	3.28	3.19	4.01

compare the accent embedding with the language interpolation strategy for French-accented English. Additionally, within the [263] dataset, the English accents were further divided into Canadian, US, and British accents. It is suggested that the US accent is the closest match to the English partitions in the LibriSpeech and VCTK test datasets, although it is acknowledged that additional accent groups are present, particularly within the VCTK set. For this reason, the Indian accent was also included in the experiments due to its known presence in the VCTK datasets. The utility and privacy results for ASR-TTS, using the TTS model trained with learnable accent embeddings, are presented in Table [3.10] and Table [3.11]. Similar to the findings in Section [3.7.1], the lowest WER was achieved with the US accent (Table [3.10], bold values). However, in contrast to the earlier approach described in Section [3.7.1], the Portuguese and Indian accents demonstrated performance closely aligned with the US English benchmark. Unlike the results from language interpolation, as shown in Table [3.8], fine-tuning the ASR system on LibriSpeech-360—processed through the ASR-TTS system for each specific accent embedding—resulted in comparable WERs. It is posited that the use of accent embeddings leads to higher quality audio generated by the ASR-TTS system, effectively eliminating the distortions typically introduced by language interpolation. Moreover, as confirmed by the analysis in Section [3.3], training the TTS system with these accent embeddings produces accented speech that significantly improves the ASR system’s ability to capture accent-specific acoustic features. This contributes to the observed uniformity in WER outcomes, as shown in Table [3.10], where the use of

Indian and Portuguese accents leads to a reduction in performance variability, closely matching the results obtained with US English. These findings support the hypothesis that accent-specific fine-tuning enhances the performance of ASR models. Conversely, when accents are selected at random, a noticeable degradation in WER occurs as similarly observed in the case of language interpolation in Section 3.4.

Table 3.11: Primary anonymization metric (EER, higher is better) results for development (light gray) and test (dark gray) sets. Accent Embedding ASR-TTS. en: American English accent, pt: Portuguese English accent, fr: French English accent and random: random selection of the accent embedding. Transcriptions extracted with wav2vec2.0

Set	Gender	Weight	EER[%]			
			ind	pt	usa	random
Libri-dev	female	0.25	46.16	46.71	48.89	47.16
	male	0.25	47.04	41.92	45.34	48.28
Vctk-dev	female	0.20	43.41	49.64	40.26	38.18
diff.	male	0.20	46.85	45.31	40.34	37.27
Vctk-dev	female	0.05	46.54	49.37	47.38	42.73
comm.	male	0.05	45.91	43.02	46.97	43.88
<i>Avg^Wdev</i>			45.97	45.77	43.15	43.28
Libri-test	female	0.25	45.81	46.19	44.34	46.39
	male	0.25	39.91	37.64	48.98	43.65
Vctk-test	female	0.20	47.37	49.28	47.83	46.51
diff.	male	0.20	33.65	38.58	41.16	45.11
Vctk-test	female	0.05	48.23	49.42	46.82	45.37
comm.	male	0.05	43.50	49.72	47.74	48.59
<i>Avg^Wtest</i>			42.22	43.49	45.85	45.53

In terms of privacy, the EER results in Table 3.11 indicate that the highest privacy levels were achieved with the Indian accent and the US accent for the development and test sets, respectively. These results showed slightly lower scores compared to language interpolation,

with a difference of around 1.5%, likely due to the more challenging acoustic conditions introduced by the language embedding method’s artifacts. Lastly, when examining the PU_{tr} metric for the accent embeddings, shown in Figure 3.2 last row, it was observed that for lower values of λ , where EER (and thus privacy) is prioritized, the Portuguese and Indian accents emerged as the optimal choices. However, when higher values of λ are used, where WER must not be significantly compromised, the US accent becomes a better option similarly to the case of language embedding interpolation in Section 3.4.

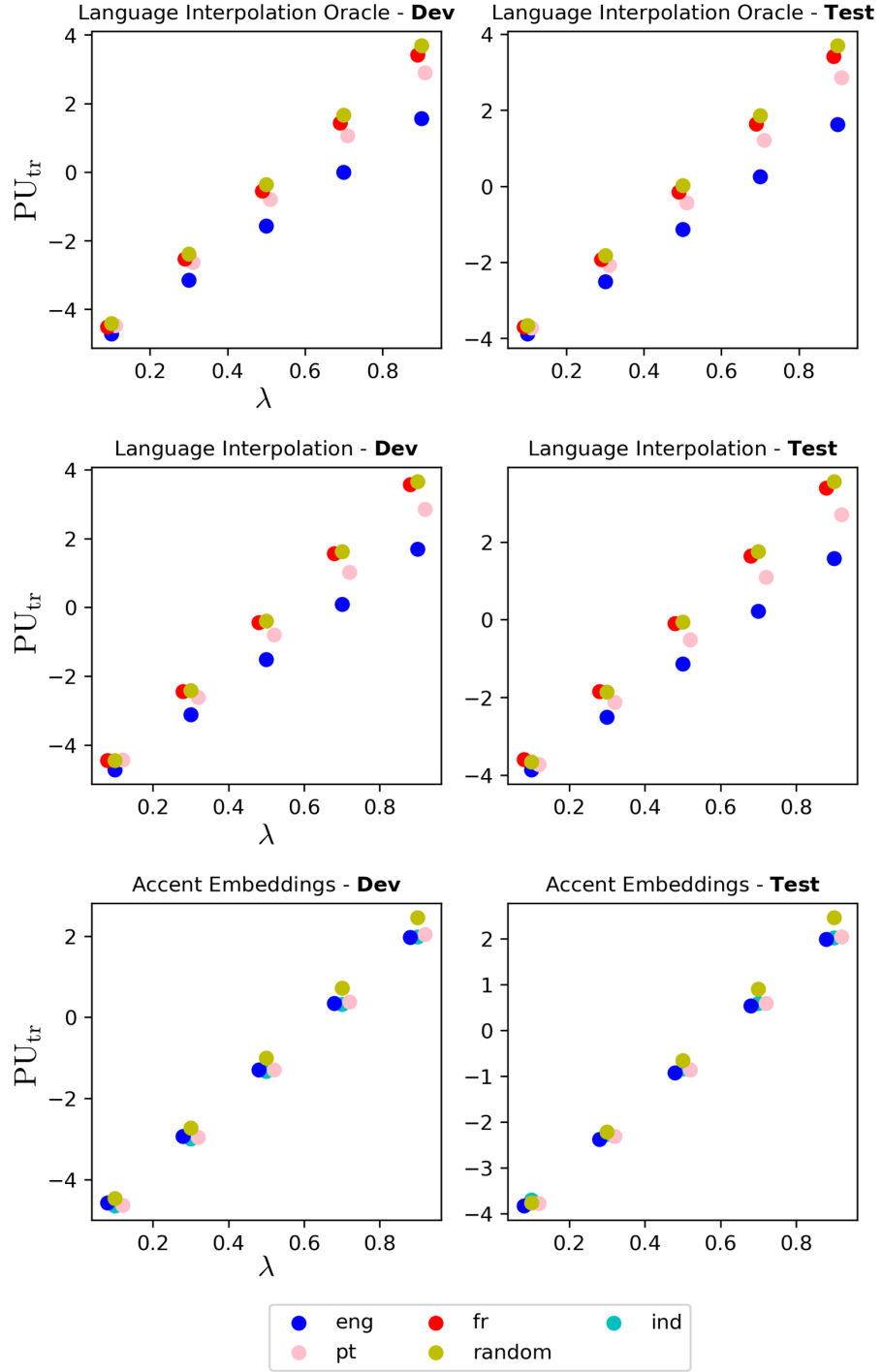


Figure 3.2: Privacy-to-Utility trade-off for language interpolation and accent embedding at varying WER/EER trade-offs λ . First row, language embedding interpolation with oracle transcriptions. Second row, language embedding interpolation with ASR-TTS transcripts. Third row, accent embedding with ASR-TTS transcripts. Lower is better.

3.8 Discussion and Conclusions

This chapter has initially explored various methods to enhance ASR systems ability to handle accented speech, focusing on data augmentation techniques and the integration of synthetic speech generated through ZS-TTS. The motivation for this research stemmed from the persistent challenges ASR models face in underrepresented and low-resource scenarios, particularly with accented speech, where data scarcity hinders model performance. By leveraging ZS-TTS, synthetic speech was generated to simulate a range of accents, addressing the issue of limited real-world training data for accent variations.

The experiments demonstrated that pre-trained, open-source models provide improvements in WER when augmented with synthetic data, particularly for single accents like Indian English. However, it was observed that fine-tuning these models on small amounts of high-quality, real accented speech before introducing synthetic data led to improvements compared to using pre-trained models alone. This suggests that while ZS-TTS is a powerful tool for data augmentation, its effectiveness is significantly enhanced when fine-tuned with real, high-quality accented data, even if in small amounts as detailed in Tables [3.1](#) and [3.2](#). Moreover, when fine-tuning the ASR models solely on synthetic speech, significant overfitting was observed, particularly when large volumes of TTS-generated data were used without real data for balancing. This overfitting led to diminished generalization, as the ASR models struggled to adapt to the acoustic properties of real-world speech. These findings emphasize the limitations of relying solely on synthetic data for model training in the absence of real, accented speech. The repurposing of language embeddings in the TTS system to encode accent-specific pronunciations did not yield improvements when applied to the full Interspeech 2020 dataset. This was attributed to the lower data quality of the Interspeech recordings compared to other datasets like VCTK. This issue was further explored by fine-tuning the TTS model on a subset of Indian-accented speech from the Interspeech dataset, which still did not improve the results. This suggests that while ZS-TTS can be an effective augmentation tool, its performance is highly dependent on the quality of the data used for training. Additionally, the combination of traditional speech augmentation techniques such as RIR, noise, speed, and SpecAugment with TTS data was

found to harm ASR performance on accented speech. Specifically, while traditional augmentations showed some benefit on larger datasets, combining them with synthetic speech tended to degrade performance, especially for smaller data partitions. This highlights the complexity of using both real and synthetic data in training ASR models, as augmentations that typically improve performance in standard speech recognition tasks may introduce challenges when applied to accented and synthetic speech data. In summary, this Chapter demonstrates that synthetic speech generation using ZS-TTS can be a valuable tool for augmenting ASR systems in accented speech scenarios, but its full potential is realized only when small amounts of high-quality real data is available and mindful of the limitations of augmentation strategies.

In addition to improving ASR performance, this chapter also investigated the use of accent modification for privacy preservation. Using the ZS-TTS system, synthetic speech was generated effectively anonymizing speaker identity while preserving or modifying accent information. This process of accent modification provides an additional layer of security, as it complicates the identification of speakers based on vocal characteristics. By adjusting accent information, accent-modified speech becomes less identifiable, thereby enhancing privacy protection. The experiments on privacy preservation, presented in Section 3.7, demonstrated that training the TTS model with learnable accent embeddings, as explored in Section 3.3, produced results that effectively balanced privacy and utility. Specifically, EER scores showed that the highest privacy levels were achieved with the Indian accent and the US accent for the development and test sets, respectively. These results showed slightly lower privacy scores compared to language interpolation, with a difference of around 1.5%, which may be attributed to the artifacts introduced by the language embedding method. The analysis of the PU_{tr} metric further confirmed that for privacy-sensitive applications, the Portuguese and Indian accents were optimal choices when privacy was prioritized. However, when maintaining lower WER scores was a priority (i.e., in scenarios where utility cannot be significantly compromised), the US accent became the preferred option. This dual benefit of enhancing privacy while improving ASR robustness to accented speech, demonstrates the potential of accent modification as a tool for both privacy protection and fairness enhancement.

Chapter 4

Conversation Analysis: Exploring Utility and Privacy

In the digital age, as extensively elucidated in Section [1.11](#), the ubiquity of recording apparatuses and the subsequent proliferation of audio data necessitate stringent privacy safeguards. This chapter concentrates on addressing privacy considerations in the analysis of conversational audio data [\[275\]](#). Specifically, the focus is on the analysis of prolonged conversational recordings from participants diagnosed with dementia, captured via portable devices across multiple days in domiciliary settings. The analytical emphasis is placed on identifying speaker turns and extracting diarization information, which are critical for detailed conversation analysis. Moreover, the pursuit of privacy protection extends beyond the conventional measures described in Section [2.3](#). In this context, in combination with acoustic privacy enforcement, it is desired to ensure enhanced privacy in the semantic domain of the conversational data. In this context, alongside acoustic privacy enforcement, it is important to enhance privacy within the semantic domain of conversational data.

Consequently, this research focuses on the development of an anonymization strategy targeting both acoustic and semantic information while preserving acoustic information necessary for speaker diarization and speaker turns computation.

Therefore this chapter employs distinct privacy utility measures diverging from those discussed in the preceding Section [2.3](#). Privacy assurance is no longer confined to maximizing

the EER of ASV model. Instead, there is an additional objective to maximize the WER of the ASR model, thereby obfuscating semantic content to safeguard sensitive information. Furthermore, the utility of the proposed methods is quantified by the performance of the diarization and VAD systems, providing critical metrics for assessing the effectiveness of these privacy-preserving techniques in real-world conversation analysis applications.

This research was detailed in [276] and was conducted during the second, 6 months, MSCA-ETN secondment at the Fraunhofer IDMT Institute in Oldenburg, Germany.

4.1 Conversational Data Anonymization

The analysis of everyday conversations has significant relevance across numerous research disciplines. Examples include the study of children’s language-learning environments [277] and the investigation of social interactions in individuals with mental health conditions or hearing impairments [278], [279]. In this chapter, long-term conversations from participants with dementia, recorded over multiple days using portable devices, are examined. The scope of the analysis extends beyond quantifying the number of conversations per day, to include other characteristics such as the duration of interactions, the diversity of communication partners, and the frequency of speaker turns [280]. However, speech data contains PII in both semantic and acoustic forms, and due to the uncontrolled and varied nature of daily life settings, these recordings present substantial privacy risks. Furthermore, the privacy protection methods at the feature or model level described in Section 2.2 are unsuitable for this scenario where on-device anonymization is required, primarily due to the computational limitations and power consumption constraints of portable recording devices. Prior research proposed a system for conversation detection and speaker diarization utilizing privacy-preserving features that prevent linguistic content reconstruction [281]. However, this approach required multiple audio streams and centralized data aggregation, which is impractical for the present application. An alternative, more feasible solution is to limit recordings to privacy-preserving acoustic features and perform offline analyses [282]. Previous human experiments with mosaic speech (comparable to pixelated images) have demonstrated reduced speech recognition accuracy due to degraded temporal and/or

spectral resolution [283], [284]. Following this approach, the contribution presented in this chapter investigates the combination of smoothed and subsampled power spectral densities (PSD) [282] for protecting linguistic PII, and the McAdams coefficient [285] for lightweight anonymization of acoustic PII. In addition, the approach presented in this Chapter considers attack and trust models that rely on the same feature extraction techniques, enabling reductions in feature resolution in the time and/or frequency domains while directly assessing the impacts on utility and privacy. As a first step towards enhancing privacy, the size of the Mel filterbank was reduced while maintaining the same frequency range. To recover the original time resolution, the smoothed and subsampled PSD was repeated according to the subsampling factor before applying the Mel filterbank, producing a feature set named olMEGA, developed as part of the open-source Ecological Momentary Assessment system [286]. Further obfuscation of linguistic PII was achieved by applying olMEGA smoothing and subsampling with repetition before the Mel filterbank extraction. Additionally, acoustic privacy was reinforced through the McAdams coefficient [285], a cost-effective signal processing technique that shifts pole positions derived from linear predictive coding (LPC) analysis of speech signals [287]. Finally, the combined use of the McAdams coefficient with olMEGA smoothing and subsampling is further explored in additional experiments, as described in Section 4.3.

4.2 Privacy and Utility Evaluation

The utility assessment utilizes the Matthews Correlation Coefficient (MCC) to evaluate the performance of the VAD model and the Diarization Error Rate (DER) computed from the Speaker Diarization (SD) system. These metrics were selected because conversation analysis requires first identifying voiced time segments and then assigning each segment to a distinct speaker, without necessarily identifying the speaker. Additionally, unlike in Section 2.3, the privacy evaluation is divided into two components: speech content, evaluated using an ASR model, and speaker identity, assessed using an ASV system. The core principle behind privacy preservation in this context is to balance utility with privacy protection, aiming to minimize DER while maximizing MCC, WER, and EER. The ASR

and ASV systems used in this chapter have already been described in Section 2.3 and section 3.7 while the following Sections provide details on the VAD and SD models.

4.2.1 Voice Activity Detection

The VAD model [215] is based on a Convolutional Recurrent Deep Neural Network (CRDNN) architecture. This model estimates frame-level posterior probabilities, which are subsequently passed through a sigmoid function for binary classification, distinguishing between speech and non-speech frames. The performance of the VAD model is evaluated using

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}} \quad (4.1)$$

where TP, TN, FP, and FN denote the number of true positive, true negative, false positive, and false negative, examples respectively.

The choice of the MCC is related to its robustness for binary classification problems, as it considers all four categories of the confusion matrix: TP, TN, FP, and FN. This approach ensures that the MCC provides a balanced assessment between positive and negative classes, even in scenarios with imbalanced datasets [288]. Unlike metrics such as accuracy and F1 score, which can be misleading when class distributions are uneven, the MCC yields a high score only when the classifier performs well across all classes [289]. This makes it particularly advantageous for applications where both positive and negative classes are of equal importance, offering a more informative and truthful evaluation of binary classifiers.

4.2.2 Speaker Diarization

The SD system employs ECAPA-TDNN embeddings [84], which are also used by the ASV model, and utilizes Spectral Clustering (SC) to assign a relative speaker label to each detected time segment [290]. The system operates using oracle VAD information. The utility of the SD model is measured using the

$$\text{DER} = \frac{\text{FA} + \text{MISS} + \text{ERROR}}{\text{TOTAL}}\%. \quad (4.2)$$

In this equation, TOTAL represents the total reference speaker time, FA refers to the speaker time falsely attributed to the reference speaker, MISS denotes the reference speaker time that was not correctly attributed to the speaker, and ERROR accounts for the reference speaker time assigned to an incorrect speaker [291]. The DER evaluation includes a forgiveness collar of 250 ms, and regions with overlapping speakers are included in the calculation. This *fair* setup [292] acknowledges that human annotation cannot realistically achieve frame-level accuracy.

4.3 Experiments

All experiments were conducted using the open-source speech-processing toolkit [215]. The models employed a consistent feature extraction process, which involved computing standard log Mel filterbank energies on time segments with a window size of 25 ms and a hop size of 10 ms. In the case of smoothed and subsampled PSD, a window size of 25 ms and a hop size of $L = 12.5$ ms were used. The segments were further smoothed using a first-order recursive filter with a smoothing time constant of $\tau = 125$ ms, and then subsampled by a factor of $\tau/L = 10$. These smoothed and subsampled PSD features were subsequently combined with the standard Mel filterbank to compute the log Mel filterbank energies. For the ASR and ASV models fine-tuning on anonymized data, the LS_{anon}^{360} dataset was utilized. Anonymization was achieved by randomly sampling a McAdams coefficient from the range (0.5, 0.9) for each utterance.

Set	WER[%]			
	20	40	80	olMEGA
test-clean	3.56	2.36	2.34	5.97
test-other	8.97	5.46	5.55	16.22
test-clean-anon	18.42	9.10	11.40	93.90
+ FT	5.70	3.03	2.96	11.66
test-other-anon	52.12	30.02	35.91	102.72
+ FT	17.31	8.48	8.04	32.95

Table 4.1: ASR decoding on test subsets of [68] for different Mel input feature dimensions, olMEGA and anonymized speech with models trained on clean (dark gray rows) and fine-tuned (FT) on anonymized (white rows) data.

Table 4.1 presents the WER results of the models trained for 90 epochs with a batch size of 256 for each input feature dimension and olMEGA. Anonymized speech was decoded using two models: one trained on the clean dataset (dark gray rows, Table 4.1) and a second model that was fine-tuned for 30 epochs on the LS_{anon}^{360} (white rows, Table 4.1), as described earlier. The reduction of the input Mel filterbank dimensions leads to a degradation in the performance of the ASR model, as shown in Table 4.1, which aligns with the findings in [284]. Additionally, the McAdams anonymization further worsens the WER (Table 4.1, dark gray rows), though partial recovery can be achieved by fine-tuning the model on anonymized speech (Table 4.1, white rows).

Table 4.2 presents the EER results for models trained on both unprocessed (O) and anonymized data (AN, FT). The ASV models were initially trained for 30 epochs with a batch size of 512 using clean data from [213]. To address potential mismatches between training and testing sets, the models were further fine-tuned for an additional 30 epochs on the anonymized LibriSpeech-360 dataset. In Table 4.2, the ASV performance remains relatively robust, even for lower-dimensional inputs such as 10 Mel filterbank features. Furthermore, a notable drop in anonymization performance is observed only under the 5 and 20 Mel conditions for the ASV and ASR models (Table 4.2 and Table 4.1 respectively).

Set	Gender	Weight	EER[%]									
			5	10	20	40	80			olMEGA		
			O	O	O	O	O	AN	FT	O	AN	FT
Libri-dev	female	0.25	7.53	3.97	2.98	3.12	2.70	9.81	13.58	5.97	14.21	15.19
	male	0.25	2.64	0.78	0.61	0.31	0.34	6.83	4.53	1.55	8.10	7.62
VCTK-dev	female	0.20	5.00	1.91	0.62	0.17	0.44	8.25	2.87	1.8	9.04	5.73
diff.	male	0.20	4.26	0.99	0.30	0.10	0.45	5.95	3.33	1.29	8.14	5.46
VCTK-dev	female	0.05	2.62	1.75	0.59	0.87	0.51	6.90	3.22	0.94	8.15	6.4
comm.	male	0.05	2.85	1.08	0.02	0.01	0.02	2.56	3.96	0.57	4.56	5.95
$Avg^{W_{dev}}$			4.67	1.88	1.1	0.96	0.97	7.47	6.12	2.57	9.65	8.56
Libri-test	female	0.25	4.57	1.32	0.54	0.69	0.37	8.76	9.35	2.19	14.57	10.95
	male	0.25	3.79	1.77	0.72	0.45	0.89	6.64	3.10	1.55	8.02	6.9
VCTK-test	female	0.20	9.98	3.35	1.49	0.98	0.93	9.41	4.68	2.74	11.53	10.09
diff.	male	0.20	4.72	1.43	0.63	0.36	0.41	11.20	3.26	1.43	8.67	4.93
VCTK-test	female	0.05	3.47	1.42	0.56	0.58	0.82	8.72	2.25	2.02	12.43	6.07
comm.	male	0.05	1.95	0.28	0.01	0.02	0.26	4.81	2.83	0.06	4.46	3.73
$Avg^{W_{test}}$			5.30	2.14	0.77	0.58	0.64	8.65	4.95	1.87	10.53	7.96

Table 4.2: ASV performance on the Voice Privacy Challenge [201] dataset for different Mel feature dimensions, olMEGA and anonymized speech. O: original data. AN: anonymized data and model trained on unprocessed speech. FT: anonymized data and model fine-tuned on anonymized speech.

Table 4.3: VAD performance measured with MCC on the LibriParty test set. The models were trained for different Mel input feature dimensions and olMEGA using the original signals (O) and after McAdams’ anonymization (M).

Model	Train	Test	MCC
5Mels	O	O	0.7001
10Mels	O	O	0.7970
20Mels	O	O	0.8259
40Mels	O	O	0.8311
	O	M	0.6761
	M	M	0.8536
olMEGA	O	O	0.8083
	M	M	0.8157
80Mels	O	O	0.8599

The VAD models were trained and tested using the train and test splits of the LibriParty dataset [215]. For the anonymized speech condition, the same anonymization strategy

applied to LibriSpeech-360 was used for the LibriParty dataset. Similarly to the observation made for the ASR and ASV models, the VAD performance is only marginally impacted by the reduced size of the Mel filterbank, with significant degradation in the MCC occurring solely in the 5 Mel condition (Table 4.3). This outcome is consistent with state-of-the-art VAD algorithms that rely on low-dimensional energy features [293]. Furthermore, the VAD model remains robust against McAdams anonymization, especially when it is trained on anonymized data (M rows, Table 4.3). The performance of the SD models was evaluated on the Augmented Multi-party Interaction (AMI) Meeting Corpus [294], using the standard *Full-corpus-ASR* partition and the *HeadsetMix* recording streams. Oracle VAD information was derived from the manual ground truth annotations. Additionally, to compare ASV and SD results on the same data, simulated conversations were generated using the Voice Privacy Challenge (VPC) [201] test and evaluation sets. In this regard, the Multi-purpose Multi-Speaker Mixture Signal Generator (MMS-MSG) [295] was used to generate meeting-like speech, featuring 3-4 speakers per meeting, without overlapping speech. In the simulation process, silences at the beginning and end of each utterance were removed. For fine-grained VAD, the pretrained VAD system described in Section 4.2.1 was used, and an energy-based threshold was applied to the detected speech segments to enhance the accuracy of onset and offset detection. Subsequently, speakers were randomly assigned to simulated meetings, following the activity-based speaker turn sampling method proposed by [295]. Utterances were randomly sampled per speaker until exhaustion, and the complete VPC test and evaluation data were utilized without repetition. This dataset is referred to as VPC simulated meetings (light and dark gray rows, Table 4.4).

The SD model utilizes the same ECAPA-TDNN embeddings as those employed for the ASV system described in Section 3.7. The pruning threshold for the affinity matrix during SC is optimized using the AMI and Libri development datasets. Table 4.4 provides a summary of the SD performance on both the AMI corpus development and evaluation sets, as well as on the VPC simulated meetings (light and dark gray rows, Table 4.4). The estimation of the number of speakers per meeting is capped at a maximum of ten. The SD performance is stable when models are trained using the olMEGA feature extraction method (Table 4.4: 80 Mels and olMEGA, O columns). Unlike EER, the DER remains almost constant despite

Set	DER[%]									
	5	10	20	40	80			olMEGA		
	O	O	O	O	O	AN	FT	O	AN	FT
AMI-dev	18.70	17.15	16.45	15.92	16.22	28.42	18.34	15.71	33.35	25.31
AMI-eval	18.38	16.98	16.71	16.59	16.32	29.13	19.91	17.12	30.55	24.86
Libri-dev	6.89	5.25	7.79	10.55	7.54	5.80	9.69	5.16	6.70	5.91
VCTK-dev	0.05	< 0.01	< 0.01	< 0.01	0.86	1.17	0.08	0.03	5.05	0.71
Libri-test	5.26	6.46	8.40	5.09	6.30	8.64	16.1	3.23	9.31	9.81
VCTK-test	0.12	0.01	1.44	1.39	0.63	11.87	0.67	0.33	15.14	1.23

Table 4.4: DER on the AMI corpus development and evaluation sets and the VPC simulated meetings sets for different Mel input feature dimensions, olMEGA and anonymized speech. The number of speakers per meeting is estimated and a forgiveness collar of 250 ms is applied. O: original data. AN: anonymized data and model trained on unprocessed speech. FT: anonymized data and model fine-tuned on anonymized speech.

the reduced Mel filterbank resolution. However, the DER on the VPC simulated meetings exhibits greater variability compared to that on the AMI corpus. In simulated meetings with oracle speaker information, the SD performance approaches perfection (results not shown), which may be attributed to the clean recording conditions in the original VPC data. Differences between the Libri and VCTK subsets could be explained by the overestimated number of speakers in the Libri subsets. Overall, the olMEGA feature extraction method contributes to privacy improvements in both the acoustic and semantic domains. This is particularly evident when olMEGA is combined with McAdams anonymization (Table 4.2: Avg^{Wdev} , Avg^{Wtest} ; Table 4.1: test-clean-anon+FT, test-other-anon+FT). Enhanced privacy protection is achieved with either a gain in DER performance (Table 4.4: Libri-dev and Libri-test, olMEGA FT column) or only minimal performance reductions (Table 4.4: VCTK-dev and VCTK-test, olMEGA FT column) on the VPC simulated dataset, when compared with the standard 80 Mel features (Table 4.1, 80 Mels, FT columns). However, these findings do not extend directly to the AMI dataset.

4.4 Conclusions

This chapter presented an exploration of the balance between privacy preservation and utility in the context of conversation analysis using feature extraction methods. The focus

was on long-term conversational recordings of dementia patients, where privacy preservation is paramount due to the presence of PII in both the semantic and acoustic domains. Various signal processing-based methods were analyzed, particularly the application of smoothed and subsampled PSD combined with the McAdams speaker anonymization technique, to assess their effectiveness in maintaining utility while enhancing privacy. The utility of the proposed methods was evaluated using performance metrics for VAD and SD. The MCC was employed to assess the VAD, while the DER measured the SD system’s performance. The experimental results demonstrated that reducing the input feature dimensions, particularly using smoothed and subsampled PSD in conjunction with McAdams anonymization, significantly enhanced privacy preservation. These methods resulted in improved privacy compared to unprocessed signals, while retaining high utility, especially in the VPC simulated meetings dataset. The VAD and SD models remained robust even with reduced Mel filterbank resolutions, and the McAdams coefficient contributed to substantial anonymization, particularly when applied in conjunction with the olMEGA feature extraction technique.

Chapter 5

Relative Acoustic Features for Indoor Applications

This chapter explores a method to estimate the distance between devices in smart home environments by analyzing the acoustic characteristics of audio recordings. Using a fixed smart speaker and wearable devices like watches or smartphones, the approach leverages the ambient noise and reverberation captured by these devices to estimate their relative distance. This estimation can improve tasks such as selecting the most effective microphone for speech capture or analyzing the acoustic scene in a room. Moreover, in the context of privacy protection, distance estimation could potentially lower the number of misactivations for voice-activated smart devices. The method is designed to be computationally efficient, making it suitable for real-time applications in smart home settings.

The techniques presented in this chapter were derived from [\[296\]](#) and developed during the first, 6 months, MSCA-ETN secondment at Imperial College London.

5.1 Inter-Device Distance Estimation

With the advent of smart portable devices such as powerful laptops, smartphones and smart wearable devices, speech and audio signal recording with traditional microphone array devices is becoming more sporadic. In fact, ad-hoc microphone arrays composed of

randomly located smart devices (Fig. 5.1) have been employed as recording platforms for many applications such as sound source tracking [297] and speech recognition in meeting scenarios [298].



Figure 5.1: Example of array of randomly distributed smart devices. Adapted from [299]

On one hand, this new technique has many advantages, such as that the array geometry is not anymore constrained as with conventional microphone arrays enabling wider spatial coverage of the acoustic environment. On the other, there are still significant research challenges for this method to be widely implemented in everyday life situations. Some of the arduous open problems to be solved are listed below:

1. the array geometry not physically constrained, it may vary abruptly [300];
2. the sampling rate of each device might be different and/or may change across time [301] leading to time-varying misalignment of the signal;
3. devices may have different and time-varying gains [302].

All the aforementioned issues prevent existing signal processing methods, such as beam-forming, to be directly applied in the field of distributed microphone networks. In the context of array geometry reconstruction, this chapter focuses on inter-microphone distance estimation, particularly for indoor smart home applications employing a smart wearable device and a loud speaker.

The problem of indoor localization using wearable devices presents a significant engineering challenge that has attracted increasing attention in recent years. Localization techniques for smart devices are typically composed of two primary components: the first estimates geographic features, such as the relative distance and angles between devices, while the second combines these measurements to compute the target device's location [303]. In this work, the focus is placed on the first component, specifically on distance estimation in indoor environments, also known as indoor-ranging.

As introduced in Chapter 2.1, smart speakers function by continuously monitoring their environment for a specific “wake-up word,” allowing them to respond promptly to user commands. However, this always-on listening capability introduces significant privacy concerns, as it can lead to unintended activations and the recording of conversations not meant as commands. Studies indicate that such misactivations are not merely theoretical but occur with measurable frequency [199]. Notably, around 10% of these unintended activations last longer than 10 seconds, during which conversations may be recorded and transmitted to remote servers without the user's knowledge or consent. This accidental recording is particularly problematic, as once activated, these devices can capture and process highly sensitive information.

A possible approach to mitigating such misactivations involves estimating the distance between the smart speaker and a smart wearable device, which many individuals now commonly wear. For instance, if the wearable device detects that the user is in a different room or has moved significantly away from the smart speaker, the system could infer that a detected wake word is likely a false activation and prevent the speaker from continuing to record. Similarly, if the wearable device indicates that the user is walking away while a request is being processed, the speaker could automatically stop recording, avoiding the capture of unintended private conversations. Distance estimation could also help differentiate between genuine commands spoken near the device and background conversations from across the room, reducing the likelihood of misinterpreting ambient speech as an activation.

Various technologies have been explored in the literature to address this issue. In [304]

a WiFi technique combines line of sight identification, range calibration and round trip time with a gaussian regression process estimating the distance. This method assumes some prior knowledge of the acoustic scenario which constrains the distance estimation with some heuristic rules and achieves a mean error of 0.862 m. A different approach for distance estimation is presented in [305] which, by employing ultra wide band radio technology, has shown good performance for short-range indoor localization applications. In [306] scientists employ light beams to localize targets. Compared with radio positioning this system based on visible light can achieve higher location accuracy and does not suffer from electromagnetic interference. Moreover, other techniques such as inertial sensors [307] have been used for the localization task. Intuitively, by exploiting the double integration of the acceleration it is possible to estimate the displacement and by applying gyroscope readings it is possible to predict the direction. However, this approach leads to significant location error with even a small measurement error [308]. Although poor localization estimates, inertial sensors have been used as auxiliary localization tools [309]. Finally, powerful computer vision algorithms based on neural network models [310] have been exploited for indoor localization. However, all of the aforementioned localization methods have significant drawbacks for smart home applications. For example, radio frequency, inertial sensors and light beams require method-specific infrastructure to be deployed within the localization environment and expensive sensors embedded in the devices to be localized. Computer vision requires huge and diverse image data sets, it is severely affected by indoor light conditions [311], might incur in privacy issues [312] and it might be only partially applied in the context of smart devices due to its high computational cost [313].

Consequently, acoustic localization has been identified as a viable alternative, offering relatively high accuracy and low latency with the use of microphones and speakers that are commonly embedded in modern smart devices. A common approach in acoustic-based indoor-ranging involves measuring the Time of Flight (ToF), which refers to the time taken by a signal to travel from the transmitter to the receiver. By knowing the transmission time, the inter-device distance can be estimated. However, this technique often produces suboptimal results due to inaccuracies in ToF estimation [303]. In smart-home environments, many devices are interconnected and can be controlled remotely. Typically,

smart speakers equipped with microphones are employed to manage other devices, and users frequently carry or wear smart devices, such as watches or glasses, that are also equipped with microphones.

In this context, a novel acoustic ranging system is introduced, based on the extraction of acoustic features from the relative RIR ($\overleftrightarrow{RIR}$) between the speaker and the wearable device, estimated in the time domain. The proposed feature set is inspired by traditional acoustic features extracted from the RIR but is adapted to represent relative measures. Specifically, the clarity index (C_{50}) and the direct-to-reverberant ratio (DRR), both reflecting the source-receiver distance and room acoustics, along with the reverberation time (T_{60}) are utilized. In addition, the signal-to-reverberation ratio (SRR) and a measure of sparseness (S) of the $\overleftrightarrow{RIR}$ are employed. The optimized distributed eXtreme Gradient Boosting (XGBoost) algorithm [314], using regression trees, is applied to estimate the distance between the smart speaker and the wearable device. The performance of this system is compared with a ToF baseline that utilizes Generalized Cross Correlation (GCC) with Phase Transform (PHAT) [315], as well as with a Dense Neural Network (DeNN) architecture. Finally, the importance of the relative features for the regression model's predictions is demonstrated through the use of SHapley Additive exPlanations (SHAP) [316]. The mathematical notation used is the following:

$$\begin{aligned}\mathbf{x}(n) &= [x(n), x(n-1), \dots, x(n-L+1)]^T, \\ \mathbf{y}(n) &= [y(n), y(n-1), \dots, y(n-L+1)]^T, \\ \boldsymbol{\nu}(n) &= [\nu(n), \nu(n-1), \dots, \nu(n-L+1)]^T, \\ \mathbf{h} &= [h_0, \dots, h_{L-1}]^T, \\ \hat{\mathbf{h}}(n) &= [\hat{h}_0(n), \dots, \hat{h}_{L-1}(n)]^T, \\ y(n) &= \mathbf{h}^T \mathbf{x}(n) + \nu(n),\end{aligned}$$

where $x(n)$, $y(n)$ and $\nu(n)$ are audio samples from a smart speaker (near-field), a wearable device (far-field) microphones and a noise component respectively, $\mathbf{h}$ and $\hat{\mathbf{h}}$ are the time-invariant ground-truth and the estimated $\overleftrightarrow{RIR}$, L is the length of the ground truth $\overleftrightarrow{RIR}$,

n is the time index and $[\cdot]^T$ is the transpose operator.

5.2 Relative Acoustic Features Extraction

In this section, the rationale for using C_{50} , T_{60} , DRR , SRR , and S for the distance estimation problem is explained and justified. Conventionally, C_{50} , T_{60} , and DRR are derived from RIRs. However, in most smart home applications, direct measurements of RIR are not typically available. Since acoustic features such as C_{50} and DRR are inherently related to the source-receiver distance, traditional acoustic ranging systems rely on blind estimates obtained from reverberant signals [317]. Unfortunately, blind estimation methods often lack precision, resulting in suboptimal distance estimations.

To address this, relative acoustic features ($\overleftrightarrow{\cdot}$) are introduced, which are calculated from the $\overleftrightarrow{RIR}$ estimate, denoted as $\hat{\mathbf{h}}$, using the following expressions:

$$\overleftrightarrow{C}_{50} = 10 \cdot \log_{10} \left(\frac{\sum_0^{n_{50}} \hat{\mathbf{h}}^2(n)}{\sum_{n_{50}}^{\infty} \hat{\mathbf{h}}^2(n)} \right) \text{ dB}, \quad (5.1)$$

$$\overleftrightarrow{DRR} = 10 \cdot \log_{10} \left(\frac{\sum_{n_d-n_0}^{n_d+n_0} \hat{\mathbf{h}}^2(n)}{\sum_{n_d+n_0}^{L-1} \hat{\mathbf{h}}^2(n)} \right) \text{ dB}, \quad (5.2)$$

where n_d and n_0 represent the sample index corresponding to the peak of the direct sound arrival and the number of samples within a small temporal window (2.5 ms in this case), and n_{50} is the sample index for 50 ms. The relative reverberation time $\overleftrightarrow{T}_{60}$, defined as the time it takes for the energy in the $\overleftrightarrow{RIR}$ to decay from -5 dB to -60 dB, is calculated from the relative energy decay curve:

$$\overleftrightarrow{E}(n) = 10 \cdot \log_{10} \left(\frac{\int_n^{\infty} \hat{\mathbf{h}}^2(\xi) d\xi}{\int_0^{\infty} \hat{\mathbf{h}}^2(\xi) d\xi} \right) \text{ dB}. \quad (5.3)$$

The value of $\overleftrightarrow{T}_{60}$ is obtained by performing linear interpolation of $\overleftrightarrow{E}(n)$: from the $\overleftrightarrow{E}(n)$ plot (in dB), a linear region is selected for estimation. This is typically the part of the curve after the initial transient response (early reflections) and before the noise floor is

reached. A common practice is to consider the decay from -5 dB to -35 dB below the initial level, as use in all subsequent experiments. Therefore,

$$\overleftrightarrow{T}_{60} = \frac{-60 \text{ dB}}{A} \text{ s}, \quad (5.4)$$

where A is the slope coefficient of the interpolated line (in dB/s). Additionally, the sparseness coefficient $\overleftrightarrow{S}(n)$ of the estimated filter $\hat{\mathbf{h}}$, defined in [318], is exploited as follows:

$$\overleftrightarrow{S}(n) = \frac{L}{L - \sqrt{L}} \left(1 - \frac{\|\hat{\mathbf{h}}(n)\|_1}{\sqrt{L}\|\hat{\mathbf{h}}(n)\|_2} \right),$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ are the l_1 -norm and l_2 -norm, respectively. The sparseness $\overleftrightarrow{S}(n)$ is used because it is dependent on both room acoustics and the source-receiver distance. Finally, the signal-to-reverberation ratio (SRR) is computed following the method in [319]:

$$SRR = 10 \cdot \log_{10} \left(\frac{\|\mathbf{x}(n)\|_2^2}{\|\mathbf{y}(n)\|_2^2} \right) \text{ dB}. \quad (5.5)$$

5.3 Improved Proportionate Normalized Least Mean Square Algorithm

In this Section, a brief review of the improved proportionate normalized least mean square (IPNLMS) adaptive filter, originally introduced in [320], is provided. This filter is now employed for the estimation of the $\overleftrightarrow{RIR}$ in smart home scenarios. The purpose of the IPNLMS filter is to generate the $\overleftrightarrow{RIR}$ estimate $\hat{\mathbf{h}}(n)$, which characterizes the acoustic path between two devices and captures the effects of sound propagation, reflections, and reverberation in the environment. This estimation is foundational to inter-device distance estimation and other acoustic sensing applications, such as sound source localization, dereverberation, and privacy-aware voice processing since all acoustic features presented in section 5.2 rely on the estimate $\hat{\mathbf{h}}(n)$. The adaptive filter iteratively minimizes the error signal:

$$e(n) = y(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n). \quad (5.6)$$

The update for $\hat{\mathbf{h}}(n)$ is performed using the following equation:

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu \frac{e(n)\mathbf{K}(n-1)\mathbf{x}(n)}{\mathbf{x}(n)^T\mathbf{K}(n-1)\mathbf{x}(n) + \delta}, \quad (5.7)$$

where μ represents the adaptation step, and δ is a regularization factor. The matrix $\mathbf{K}$ is responsible for updating the filter taps in proportion to their relative magnitude, defined as:

$$\mathbf{K}(n-1) = \text{diag}(k_0(n-1), \dots, k_{L-1}(n-1)), \quad (5.8)$$

where each element $k_l(n)$ is calculated as:

$$k_l(n) = \frac{1-\alpha}{2L} + (1+\alpha) \frac{|\hat{h}_l(n)|}{2\|\hat{\mathbf{h}}(n)\|_1 + \epsilon}, \quad l = 0, \dots, L-1. \quad (5.9)$$

In this expression, α is a real number in the range $[-1, +1]$, and ϵ is a small value introduced to prevent division by zero.

While the mathematical formulation of the IPNLMS filter describes its adaptive update mechanism, its core significance lies in its ability to efficiently estimate $\overleftarrow{RIR}$ in sparse acoustic environments, which is common in smart home scenarios. An $\overleftarrow{RIR}$ is often characterized by a few dominant coefficients corresponding to the direct path and strong reflections, with the rest being low-energy reverberation [321]. Standard adaptive filters, such as the NLMS algorithm, treat all coefficients equally, which can result in slow convergence when the system is sparse. In contrast, the IPNLMS filter prioritizes the adaptation of the more significant coefficients, resulting in faster convergence and improved estimation accuracy [322]. This behavior is achieved through the gain matrix $\mathbf{K}$ in equation [5.8], which scales the update for each coefficient based on its magnitude. Larger coefficients, representing the strongest acoustic paths, receive larger updates, while smaller

coefficients are adjusted more conservatively. The α parameter in equation 5.9 controls the degree of this proportional adaptation, with positive values favoring large coefficients and $\alpha = -1$ reducing the filter to the standard NLMS approach. In practice, $\alpha > 0$ is preferred for sparse $\overleftarrow{RIR}$ estimation.

5.4 Distance Estimation and Explainability

The distance estimation task was carried out using the extreme gradient boosting algorithm XGBoost [314], with a decision tree regressor as the base model. XGBoost was chosen for its superior performance on tabular data compared to neural networks [323], and its ability to provide interpretability for the input features. A commonly used algorithm for decision tree training, known as Classification and Regression Trees (CART) [324], was employed. CART works by splitting the training set into two subsets using a single feature f and a threshold t_f , chosen to create the purest possible subset. In a regression problem, CART constructs decision trees by recursively splitting the data based on input features to minimize the variance in the target variable within each subset. At each node, CART evaluates potential splits by selecting a feature f and a threshold t_f that minimize the sum of squared errors (SSE) across the left and right child nodes. The optimal split is determined by minimizing the criterion

$$J(f, t_f) = \sum_{i \in \text{left}(f, t_f)} (y_i - \bar{y}_{\text{left}})^2 + \sum_{i \in \text{right}(f, t_f)} (y_i - \bar{y}_{\text{right}})^2, \quad (5.10)$$

where y_i is the target value for the i -th instance (the inter-device distance), $\bar{y}_{\text{left}}$ and $\bar{y}_{\text{right}}$ are the mean target values in the left and right subsets, respectively. This criterion ensures that the data is partitioned into increasingly homogeneous subsets with respect to the target variable. Once the tree is constructed, each leaf node represents a region in which the predicted value is the mean of the target values of the training samples within that region. The splitting process is repeated recursively until the maximum depth of the tree (an hyperparameter) is reached. The predicted distance for any new sample falling into a leaf node is the average distance of the samples in that leaf.

CART is considered a greedy algorithm, as it seeks optimal splits level by level, without considering the lowest possible impurity deeper in the tree. This approach does not guarantee an optimal solution, as finding such a solution is an NP-complete problem. Despite this limitation, greedy algorithms like CART produce reasonably good solutions for most practical purposes [324].

Ensemble learning [325] is the technique of combining several weaker models to create a more accurate and robust system. Suppose several classifiers, each achieving roughly 80% accuracy, are trained. By aggregating their predictions through a majority-vote system, also known as hard voting, the resulting classifier often achieves a higher accuracy than any of the individual models. This occurs even if each model is a weak learner, as long as the ensemble contains a sufficient number of weak learners that are diverse in their predictions. Boosting [326] is an ensemble method that sequentially trains models, with each new model improving upon the errors of its predecessor. Two common boosting methods are Adaptive Boosting (AdaBoost) [327] and Gradient Boosting [328]. AdaBoost focuses on the misclassified instances of the previous model, while Gradient Boosting fits a new model to the residual errors left by the earlier models. In this study, decision trees were used as base models, leading to the use of Gradient Boosted Regression Trees (GBRT) [329]. Figure 5.2 shows an example of a GBRT trained on a noisy quadratic dataset, illustrating how each successive tree improves the ensemble's predictions. XGBoost [314] is an optimized implementation of Gradient Boosting, designed for fast, scalable, and portable machine learning tasks.

In the experiments described in Section 5.5, the distance estimation task was formulated as a supervised learning problem, where ground truth values for the distance between the smart speaker and the wearable device were provided during training. The model was trained on standardized input features using the MSE loss function. The number of epochs was dynamically determined through an early stopping criterion, which halted training after five epochs without improvement in the validation loss. A grid search was employed to identify the optimal model hyperparameters. For comparison, a DeNN with five layers, also trained using the MSE loss function and the same early stopping criterion,

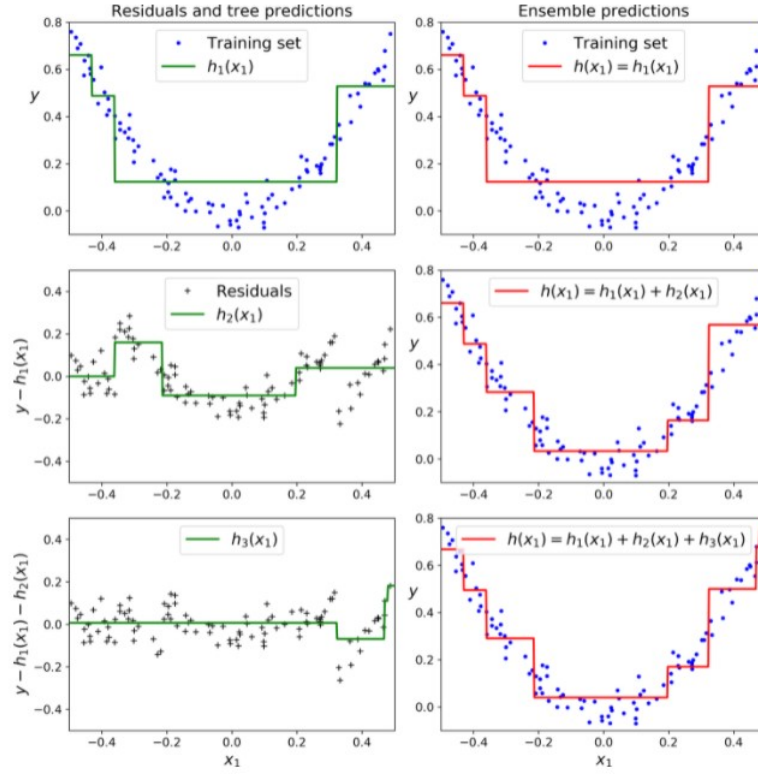


Figure 5.2: Gradient Boosted Regression Trees. Left column shows predictions of three trees; the right column shows the ensemble's combined predictions. Adapted from [330].

was evaluated against the XGBoost model. Additionally, a baseline estimate was computed using ToF with GCC-PHAT. Finally, SHapley Additive exPlanations (SHAP) [316] were used to evaluate the significance of input features in the predictions for both XGBoost and DeNN models.

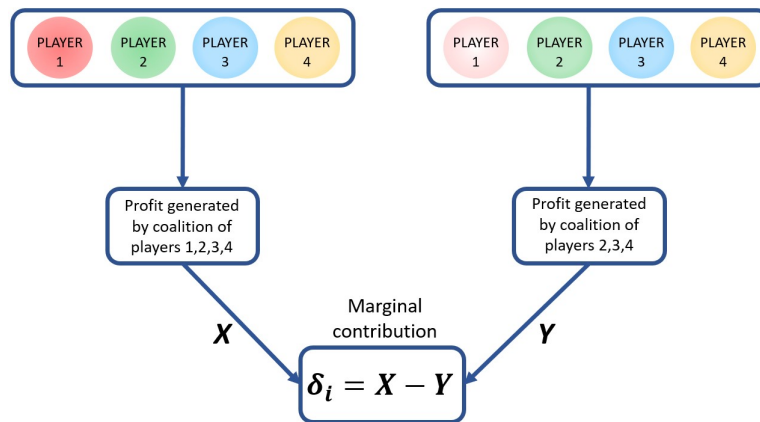


Figure 5.3: Example of marginal contribution computation for *player 1*.

Shapley values, introduced by Lloyd Shapley in 1953 [331], originate from cooperative game theory, where their main purpose is to quantify each player's contribution to a collaborative game. In this framework, N players work together to achieve a collective reward p , which is then distributed among the players based on their individual contributions, known as Shapley values. For example, in the case of a group of four players, as illustrated in Fig. 5.3, Shapley values would allocate the reward proportionally to each player's impact on the outcome. The Shapley value for a player is determined by calculating the difference between the profit generated when the player is included in the game (left-hand side of Fig. 5.3) and the profit when the player is excluded (right-hand side of Fig. 5.3). This difference is referred to as the *marginal contribution* of the player to the current coalition (δ_i , Fig. 5.3). To compute the Shapley value, the marginal contribution is calculated for all possible coalitions involving the player. The average of these differences, known as the *mean marginal contribution*, represents the Shapley value. Figure 5.4 illustrates the process of calculating the Shapley value for *player 1* in the example presented in Fig. 5.3.

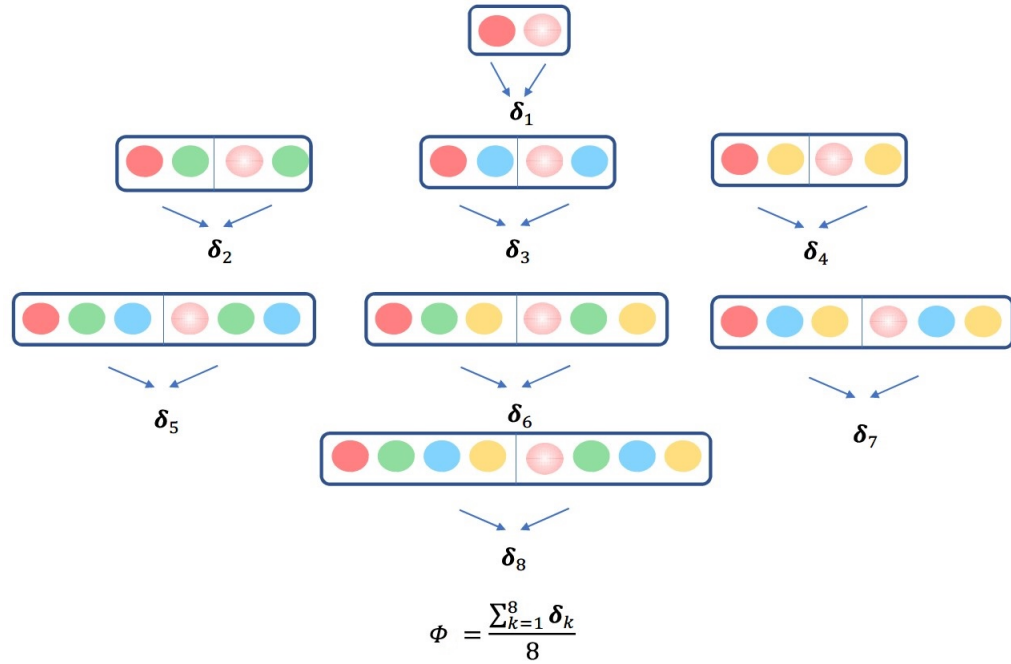


Figure 5.4: Computation example of the Shapley value associated with *player 1*. First, all the possible coalitions with *player 1* need no be computed. Once all the 8 coalitions have been formed, the next step consists of computing all the marginal contributions δ_i , $i = 1, 2, \dots, 8$. Finally, the mean marginal contribution ϕ (or Shapley value) for *player 1*, is computed by averaging all the 8 marginal contributions.

In the context of machine learning (ML), the game is represented by the ML model, the players are the input features, and the profit is the model's prediction. However, calculating Shapley values for each feature and observation is an NP-hard problem, which becomes computationally infeasible even for small datasets. To address this challenge, the authors of [316] proposed Kernel Shap (KS), an approximation method that calculates Shapley values using a reduced number of coalition samples. KS achieves this by employing a weighted linear regression where the coefficients represent the Shapley values. Figure 5.5 illustrates an example of Shapley values computed using KS. For instance, let's consider an observation $\mathbf{x}$, which consists of features f_1, f_2, f_3 , and f_4 , Fig. 5.5, right-hand side. The objective is to explain the relative importance of each feature f_i in producing the output $\hat{y}_{i+2}$: intuitively, there might be features more important than others for the machine learning model (ML, Fig. 5.5) to produce an accurate prediction $\hat{y}_i$. The KF process begins by generating a set of coalition samples $z_i, i : [0, 1, \dots, N]$ for a specific feature f . Next, for each coalition i , the model's prediction ($\hat{y}_i$) is obtained (right-hand side of Fig. 5.5) and the marginal contribution δ_i is calculated by employing the ML loss function $l(z_i, \hat{y}_i) = \delta_i$. The weights for each coalition, w_{δ_i} , are then computed using the following equation:

$$w_{\delta_i} = \frac{\# \text{features} - 1}{N \cdot (\# \text{features excluded in } z_i) \cdot (\# \text{features in } z_i)} \quad (5.11)$$

Finally, the linear model is solved as:

$$\phi_f = \sum_{i=1}^N \delta_i \cdot w_{z_i} \quad (5.12)$$

where ϕ_f represents the Shapley value of a given feature f . This process efficiently estimates the Shapley value for each feature in a computationally tractable manner. Finally, it is important to highlight that KS is the only model-agnostic method for computing Shapley values. This characteristic allows KS to interpret any machine learning model, ranging from traditional algorithms like support vector machines to more complex and widely-used architectures such as DNNs.

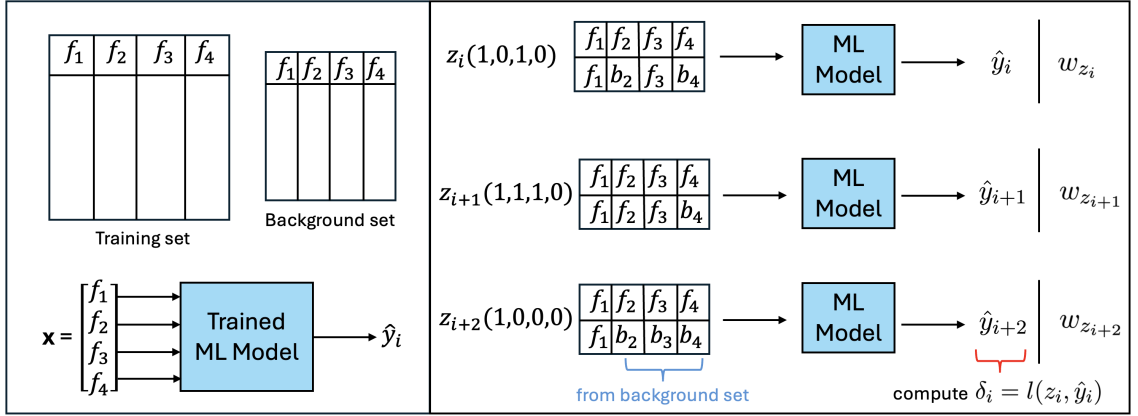


Figure 5.5: Example of KS computation of Shapley values. Note that when computing the prediction for coalition $(1, 0, 1, 0)$, f_1 and f_3 are taken from the instance $\mathbf{x}$ and, since ML models can not omit features at inference time, the original values of f_2 and f_4 of $\mathbf{x}$ are replaced with values drawn from the background set. Once selected a finite number N of coalitions z_i , the marginal contributions δ_i can be computed from the ML model output $\hat{y}_i$ by employing the training loss function $l(z_i, \hat{y}_i) = \delta_i$. Finally, the weights w_{z_i} can be computed following equation 5.11 and can be used in the linear model 5.12 to compute the Shapley values ϕ .

5.5 Experiments

Results are presented for simulated data across three room sizes:

- Small: $[5, 3.5, 2.5]$ m, $T_{60} = 0.5$ s
- Medium: $[7, 4.5, 2.5]$ m, $T_{60} = 0.7$ s
- Large: $[8, 6.5, 2.5]$ m, $T_{60} = 0.9$ s

Broad-band white noise and speech input signals were used to evaluate the system. The performance of distance estimation based on relative features was investigated by simulating reverberant rooms using the image source model [332]. In each simulated room, the wearable device was randomly placed in 30,000 different positions, while the position of the smart speaker remained fixed. For each position of the wearable device, the reverberant sound signal recorded by both the smart speaker and the wearable device was simulated, and the

$\overleftarrow{RIR}$ was estimated using the IPNLMS method, as described in Section 5.3. Two types of sound signals were used in the simulations.

First, broad-band white noise was employed to provide optimal full-band spectral excitation. Second, speech signals of varying duration, randomly selected from the LJSpeech dataset [264], were tested. These simulated conditions reflect real-life scenarios for different operational modes of the smart speaker. For instance, when the speaker is in silent mode (not executing commands or playing sounds), broad-band white noise could be used to acoustically locate the wearable device. This approach may present some challenges, as the signal could be perceptible to human listeners in the environment and potentially cause discomfort. However, while broad-band white noise is primarily a theoretical construct, it remains relevant for experimental validation and robustness testing. To address this issue, an alternative approach could involve the use of inaudible band-shaped noise. Specifically, spectrally shaped noise within the 15–20 kHz frequency range would be suitable, as it minimizes auditory disturbance for human listeners while avoiding interference with the speaker and microphone operations [333]. With this band-filtered signal, full-band excitation can be maintained by resampling the audio signals prior to $\overleftarrow{RIR}$ estimation [334]. In contrast, during active interaction with the environment, such as in a dialogue, acoustic speech signals would be used for the ranging task.

5.5.1 Acoustic Channel Estimation

Figure 5.6 shows the estimation convergence and results for a relative room impulse response estimation with the speaker operating in silent mode. The ground truth $\overleftarrow{RIR}$ shown in Fig. 5.6, panel C, was calculated as

$$\overleftarrow{RIR} = \mathcal{F}^{-1} \left(\frac{RIR_{far}}{RIR_{close}} \right), \quad (5.13)$$

where RIR_{close} , RIR_{far} , are the room impulse responses from the source to the smart speaker and wearable device microphones respectively, calculated for an inter-device distance of 1.55 m in the frequency domain. Figure 5.6A shows the evolution of the error across iterations for an IPNLMS filter with $\alpha = 0$, $\mu = 0.3$ for the first 4.5 s and $\mu = 0.1$

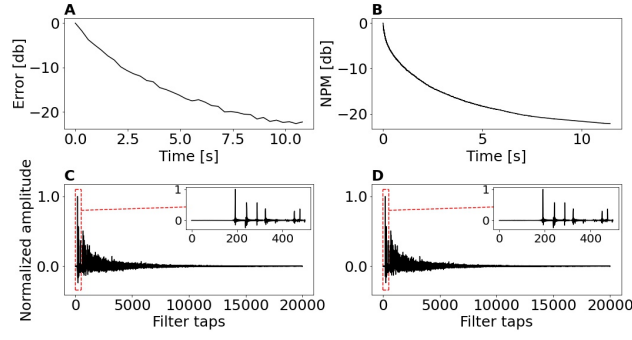


Figure 5.6: Acoustic channel estimation. A: Estimation error, B: Normalized Projection Misalignment (NPM), C: Ground truth relative RIR, D: Estimated relative RIR.

subsequently, white noise input with $\text{SNR} = 30$ dB and a ground truth $\overleftarrow{RIR}$ length of $L = 20000$. The similarity between ground truth and estimated filter has been assessed by computing the normalized projection misalignment [335], [336] as shown in Fig. 5.6B.

5.5.2 Distance Estimation for White Noise Inputs

After extracting the $\overleftarrow{RIR}$ and computing the features introduced in Section 5.2, the correlation matrix among all relative features and the inter-device distance was calculated.

	$\overleftarrow{T}_{60}$	$\overleftarrow{C}_{50}$	$\overleftarrow{DRR}$	SRR	$\overleftarrow{S}$	distance
$\overleftarrow{T}_{60}$	+1.00	-0.32	-0.43	+0.18	-0.50	+0.49
$\overleftarrow{C}_{50}$	-0.32	+1.00	+0.83	-0.27	+0.91	-0.56
$\overleftarrow{DRR}$	-0.43	+0.83	+1.00	-0.29	+0.88	-0.79
SRR	+0.18	-0.27	-0.29	+1.00	-0.32	+0.29
$\overleftarrow{S}$	-0.50	+0.91	+0.88	-0.32	+1.00	-0.78
distance	+0.49	-0.56	-0.79	+0.29	-0.78	+1.00

Table 5.1: Cross correlation matrix for white noise input signal

Table 5.1 shows strong linear correlations between all extracted relative features and the speaker-wearable distance, particularly for $\overleftarrow{DRR}$ and the sparseness coefficient, indicating that these features effectively capture the inter-device distance information. Distance estimation between the speaker and the wearable device was conducted using the gradient boosting decision trees model, as described in Section 5.4. The data was split into 80% for training, 10% for validation, and the remaining 10% for testing. After training, the

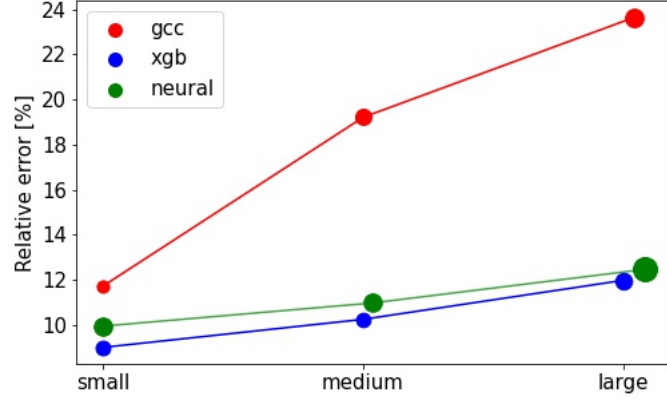
relative error metric (RE) on the test set was calculated using the following equation:

$$RE[\%] = \frac{\sqrt{(d_{tar} - d_{pred})^2}}{d_{tar}} \times 100 , \quad (5.14)$$

where d_{tar} and d_{pred} represent the ground truth and predicted distances, respectively. An average RE of 9.01% was obtained, indicating that the estimated distance error is approximately 9% of the actual inter-device distance. A comparison was made with a ToF algorithm based on GCC-PHAT and a DeNN using the same input features as XGBoost. For the GCC-PHAT estimation, the wearable device and smart speaker audio signals were divided into 300 ms time windows, and the cross-correlation across all frames was computed. The time index corresponding to the maximum correlation value in each window was extracted, and the distance was estimated by multiplying this value by the speed of sound from the room simulation. The final distance estimate was obtained by averaging the distances across all frames. This method resulted in an RE of 11.7%, which is more than 25% higher than the proposed method's error. Figure 5.7 presents a detailed comparison of XGBoost's performance in rooms of different sizes and reverberation times against GCC-PHAT and the DeNN.

The relative error values for each room were obtained by averaging results from 10 Monte Carlo simulations. The diameter of each dot in the figure represents the variance across simulations. Additionally, the relative importance of the input features for both the DeNN and XGBoost models was analyzed. Figure 5.8 displays a comparison of SHAP values for the two algorithms.

Interestingly, both the DeNN and the XGBoost models show a similar ranking of feature importance, with $\overleftrightarrow{S}$ and $\overleftrightarrow{T}_{60}$ being the most influential for inter-device distance estimation. However, $\overleftrightarrow{C}_{50}$ and SRR appeared to be less useful for distance estimation. To test this, new models were trained excluding SRR from the input features, and the relative error distributions were compared with models including SRR . Student T-tests on XGBoost predictions showed no statistically significant differences in relative error distributions for any room (small: p-value = 0.34, medium: p-value = 0.31, large: p-value = 0.68). Similar results were observed for the DeNN. Conversely, when the models were retrained without



Relative error [%]: mean(variance)			
Algorithm	Small	Medium	Large
XGBoost	9.01(0.24)	10.25(0.23)	11.98(0.32)
Neural Net	9.96(0.37)	10.98(0.38)	12.47(0.66)
GCC-PHAT	11.70(0.18)	19.20(0.29)	23.70(0.38)

Figure 5.7: Relative error comparison for GCC-PHAT (gcc), XGBoost (xgb), and a DeNN (neural) for broad-band white noise inputs.

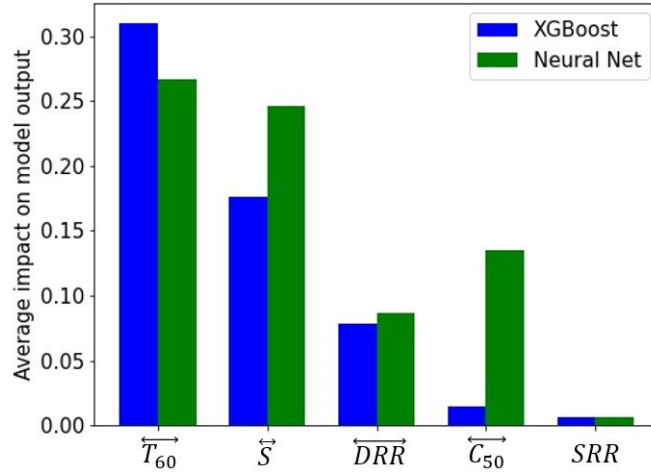
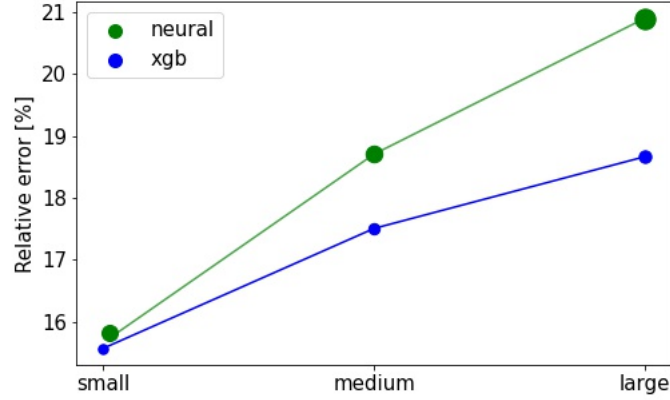


Figure 5.8: SHAP values for XGBoost and DeNN calculated for the small room and white noise inputs.

$\overleftrightarrow{C_{50}}$, statistically significant differences in error distributions were found for all rooms and both algorithms, suggesting that $\overleftrightarrow{C_{50}}$ is a useful feature for distance estimation (Student T-tests for XGBoost predictions: small: p-value = 0.015, medium: p-value = 0.001, large: p-value < 0.001. Similar results were obtained for the DeNN).

5.5.3 Distance Estimation with Speech Signals

This section presents the distance estimation results obtained using speech input signals from the LJSpeech corpus [264].



Relative error [%]: mean(variance)			
Algorithm	Small	Medium	Large
XGBoost	15.57(0.22)	17.51(0.28)	18.67(0.35)
Neural Net	15.82(0.58)	18.71(0.62)	20.90(0.94)

Figure 5.9: Relative error comparison for XGBoost (xgb) and a DeNN (neural) for speech input signals.

Speech utterances with a minimum length of 10 seconds were used, obtained by processing the original audio files with the WebRTC VAD from Google. The WebRTC VAD was selected due to its computational efficiency, making it suitable for on-board applications in smart devices with limited processing power. For each utterance, 30 ms speech segments were extracted and then combined into a single audio file. If the final audio signal exceeded 10 seconds, it was used for room acoustic simulations. Results for the same three rooms as in the case of broad-band white noise inputs are presented in Fig. 5.9. The cross-correlation baseline was excluded, as it resulted in RE values exceeding 100%. Figure 5.10 presents the SHAP values for the case of speech inputs. Interestingly, for speech utterances, all extracted features were statistically significant for distance estimation. Student T-tests for both XGBoost and DeNN predictions produced p-values < 0.05 for all rooms and feature subsets, including $\overleftrightarrow{C}_{50}$, $\overleftrightarrow{DRR}$, and SRR .

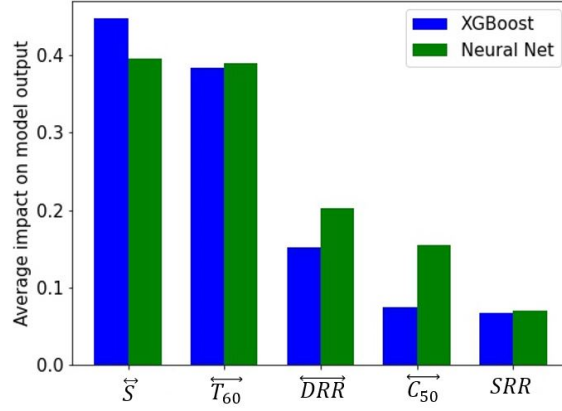


Figure 5.10: SHAP values for XGBoost and the DeNN calculated for the small room and speech inputs.

5.6 Conclusions

This chapter investigated a novel approach for estimating the distance between devices in smart home environments, focusing on acoustic localization using relative room impulse responses ($\vec{RIR}$). The primary contributions include the extraction of relative acoustic features such as clarity index ($\vec{C}_{50}$), direct-to-reverberant ratio ($\vec{DRR}$), reverberation time ($\vec{T}_{60}$), signal-to-reverberation ratio (SRR), and sparseness coefficient ($\vec{S}$) from the $\vec{RIR}$, bypassing the need for traditional RIR measurements. The IPNLMS adaptive filter was employed for $\vec{RIR}$ estimation, accurately modeling the acoustic relationship between a smart speaker and wearable device. The system's distance estimation relied on XGBoost, which demonstrated superior performance in handling tabular data and provided explainability through SHAP. Experimental validation across small, medium, and large rooms showed that the method achieved an average relative error of 9% for broad-band white noise and approximately 15% for speech inputs. Compared to traditional ToF methods, the proposed approach significantly reduced error rates. The analysis also revealed that features like sparseness and reverberation time were crucial for accurate distance estimation. Additionally, the method proved to be computationally efficient, making it suitable for real-time applications such as distributed microphone array geometry reconstruction and acoustic scene analysis.

Chapter 6

Discussion and Conclusions

This thesis presents an exploration of voice conversion and text-to-speech technologies for privacy protection, focusing on the preservation of privacy in speech signals while maintaining their utility for various downstream tasks.

The primary objective has been to develop techniques that obscure personal identifiable information in voice recordings, leveraging both conventional and state-of-the-art machine learning methodologies. Results demonstrate the feasibility of anonymizing speech while preserving its usefulness in tasks such as ASR and SD, which are critical in privacy-conscious environments. Novel voice anonymization techniques were introduced, including a two-stage voice anonymization process that combines x-vector anonymization with zero-shot voice conversion in Chapter 2. This method offers promising results in terms of privacy and utility trade-offs, making it suitable for compliance with stringent data protection regulations, such as the EU-GDPR.

Additionally, the use of accent conversion to augment training data and enhance privacy preservation have been explored in Chapter 3. By simulating accents through a zero-shot voice conversion model, ASR systems were enriched, improving their robustness and fairness across diverse linguistic backgrounds. This method not only augments data for low-resource languages but also enhances privacy measures by obscuring the speaker's original voice characteristics.

Another line of work was the development of efficient privacy-preserving features,

including low-complexity methods suited for devices with limited computational resources as outlined in Chapter 4. Feature extraction methods, such as spectral smoothing and dimensionality reduction, were shown to contribute to privacy preservation without compromising the effectiveness of ASR and SD systems. These findings provide practical solutions for privacy protection in the expanding landscape of smart home technologies.

In addition, Chapter 5 extends the definition traditional acoustic features used in inter-device distance estimation by proposing novel relative acoustic features derived from the relative room impulse response. This approach facilitated accurate distance estimation between smart home devices using audio recordings. The development of this efficient inter-device ranging system provides a means to enhance privacy by preventing unauthorized activation of smart speakers in proximity to wearable devices.

6.1 Future Work

While this thesis has made significant strides in voice privacy protection and inter-device acoustic estimation, several avenues for future research remain. First, further refinement of voice anonymization methods is needed to enhance their performance across different acoustic environments and languages. The integration of advanced deep learning techniques, could potentially improve the accuracy and robustness of these systems. Moreover, exploring the application of neural networks for direct inter-device distance estimation could eliminate the need for traditional feature extraction and enhance real-time processing capabilities. Future work may also focus on creating a holistic privacy framework that integrates voice anonymization techniques with other privacy-enhancing technologies, such as data encryption and differential privacy, to provide a comprehensive privacy solution for smart homes. Additionally, expanding the applications of accent simulation for improving fairness and inclusivity in voice-based systems can be further explored. This would involve more comprehensive datasets representing various linguistic and cultural backgrounds, ensuring equitable performance for all users.

In conclusion, the methods and findings presented in this thesis pave the way for future developments in privacy-preserving speech processing, particularly in smart environments where voice data is omnipresent. The techniques developed here offer practical solutions to the growing concerns over privacy in digital communication, setting the stage for more secure, accessible, and inclusive voice-driven technologies.

Bibliography

- [1] K. J. Kohler, “Speech communication in human interaction,” in *Communicative Functions and Linguistic Forms in Speech Interaction* (Cambridge Studies in Linguistics), Cambridge University Press, 2017, vol. 156, pp. 1–17.
- [2] K. Gunderson, “The imitation game,” *Mind*, vol. 73, no. April, pp. 234–45, 1964.
- [3] M. Katore and M. R. Bachute, “Speech based human machine interaction system for home automation,” in *IBSS*, 2015, pp. 1–6.
- [4] M. Vacher, D. Istrate, F. Portet, *et al.*, “The sweet-home project: Audio technology in smart homes to improve well-being and reliance,” in *Annual Int. Conf. of the IEEE Eng. in Medicine and Biology Society*, 2011, pp. 5291–5294.
- [5] A. Arabo, I. Brown, and F. El-Moussa, “Privacy in the age of mobility and smart devices in smart homes,” in *2012 Int. Conf. on Privacy, Security, Risk and Trust and 2012 Int. Conf. on Social Computing*, 2012, pp. 819–826.
- [6] J. L. Kröger, O. H.-M. Lutz, and P. Raschke, “Privacy implications of voice and speech analysis – information disclosure by inference,” in *Privacy and Identity Management. Data for Better Living: AI and Privacy: 14th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2.2 Int. Summer School, Windisch, Switzerland, August 19–23, 2019, Revised Selected Papers*, M. Friedewald, M. Önen, E. Lievens, S. Krenn, and S. Fricker, Eds. Springer Int. Publishing, 2020, pp. 242–258.
- [7] European Parliament and Council, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 apr. 2016 on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/ec (General Data Protection Regulation)*, 2016.
- [8] A. Smith, L. Goffman, and R. E. Stark, “Speech motor development,” in *Seminars in Speech and Language*, © 1995 by Thieme Medical Publishers, Inc., vol. 16, 1995, pp. 87–99.
- [9] H. T. Luong, “Deep learning based voice cloning framework for a unified system of text-to-speech and voice conversion,” Ph.D. dissertation, Sep. 2020.
- [10] G. Fant, *Acoustic Theory of Speech Production*. Berlin, Boston: De Gruyter Mouton, 1971.
- [11] E. Douglas-Cowie, R. Cowie, and M. Schroeder, “The description of naturally occurring emotional speech,” in *Proceedings of 15th Int. Congress of Phonetic Sciences, Barcelona*, 2003.

- [12] A. Lotto and L. Holt, "Psychology of auditory perception," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 5, pp. 479–489, 2011.
- [13] NIH. "How do we hear?" (2022), [Online]. Available: <https://www.nidcd.nih.gov/health/how-do-we-hear>.
- [14] S. M. Vljakovic and P. R. Thorne, "Purinergic signalling in the cochlea," *Int. Journal of Molecular Sciences*, vol. 23, no. 23, 2022.
- [15] A. Mohamed, H. Y. Lee, L. Borgholt, *et al.*, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, 2022.
- [16] R. Yuste, "From the neuron doctrine to neural networks," *Nature reviews neuroscience*, vol. 16, no. 8, pp. 487–497, 2015.
- [17] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [18] E. W. Weisstein, "Heaviside step function," <https://mathworld.wolfram.com/>, 2002.
- [19] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.
- [20] S. Löwel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity," *Science*, vol. 255, no. 5041, pp. 209–212, 1992.
- [21] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Int. Journal of Engineering Applied Sciences and Technology*, 2020.
- [22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [23] S. I. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.
- [24] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012. ArXiv: [1212.5701](https://arxiv.org/abs/1212.5701).
- [25] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.
- [26] D. H. Hubel, "Single unit activity in striate cortex of unrestrained cats," *The Journal of physiology*, vol. 147, no. 2, p. 226, 1959.
- [27] J. Guibas and T. Virdi, *Convolutional neural networks explained*, <https://twopointseven.github.io/2017-10-29/cnn/>, 2017.
- [28] B. H. Liu, P. Li, Y. T. Li, *et al.*, "Visual receptive field structure of cortical inhibitory neurons revealed by two-photon imaging guided recording," *Journal of Neuroscience*, vol. 29, no. 34, pp. 10 520–10 532, 2009.
- [29] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," 2016. ArXiv: [1609.03499](https://arxiv.org/abs/1609.03499).
- [30] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

- [31] G. Philipp, D. Song, and J. G. Carbonell, “The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions,” 2018. ArXiv: [1712.05577](#).
- [32] A. Rehmer and A. Kroll, “On the vanishing and exploding gradient problem in gated recurrent units,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1243–1248, 2020.
- [33] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [34] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [35] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2014. ArXiv: [1409.0473](#).
- [36] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [37] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Int. Conf. on Machine Learning*, pmlr, 2015, pp. 448–456.
- [38] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016. ArXiv: [1607.06450](#).
- [39] L. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [40] T. B. Amin and I. Mahmood, “Speech recognition using dynamic time warping,” in *2008 2nd Int. Conf. on advances in space technologies*, IEEE, 2008, pp. 74–79.
- [41] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.
- [42] T. Kudo, “Subword regularization: Improving neural network translation models with multiple subword candidates,” 2018. ArXiv: [1804.10959](#).
- [43] Y. Bharath, S. Veena, K. Nagalakshmi, M. Darshan, and R. Nagapadma, “Development of robust vad schemes for voice operated switch application in aircrafts: Comparison of real-time vad schemes which are based on linear energy-based detector, fuzzy logic and artificial neural networks,” in *2016 2nd Int. Conf. on Applied and Theoretical Computing and Communication Technology (iCATccT)*, IEEE, 2016, pp. 191–195.
- [44] S. J. Lee and H. Y. Kwon, “A preprocessing strategy for denoising of speech data based on speech segment detection,” *Applied Sciences*, vol. 10, no. 20, p. 7385, 2020.
- [45] R. Singh, U. Bhattacharjee, and A. K. Singh, “Performance evaluation of normalization techniques in adverse conditions,” *Procedia Computer Science*, vol. 171, pp. 1581–1590, 2020.
- [46] D. Gabor, “Theory of communication,” *Journal of the Institution of Electrical Engineers*, vol. 93, pp. 429–457, 1946.

- [47] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “Wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 12 449–12 460.
- [48] Z. Liu, N. Venkateswaran, E. Le Ferrand, and E. Prud’hommeaux, “How important is a language model for low-resource ASR?” In *Findings of the Association for Computational Linguistics ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand and virtual meeting: Association for Computational Linguistics, Aug. 2024, pp. 206–213.
- [49] S. Newatia and R. Aggarwal, “Convolutional neural network for asr,” in *2018 Second Int. Conf. on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 638–642.
- [50] T. Hori, J. Cho, and S. Watanabe, “End-to-end speech recognition with word-based rnn language models,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 389–396.
- [51] A. Graves, A. rahman Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” 2013. ArXiv: [1303.5778](#).
- [52] A. Graves, “Sequence transduction with recurrent neural networks,” 2012. ArXiv: [1211.3711](#).
- [53] S. R. Eddy, “Hidden markov models,” *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.
- [54] D. W. Mount, “Using hidden markov models to align multiple sequences,” *Cold Spring Harbor Protocols*, vol. 2009, no. 7, 2009.
- [55] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” in *Int. Conf. on Machine Learning*, 2006, pp. 369–376.
- [56] T. Zenkel, R. Sanabria, F. Metze, *et al.*, “Comparison of decoding strategies for ctc acoustic models,” *ArXiv*, 2017. eprint: [1708.04469](#).
- [57] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.
- [58] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
- [59] Y. Wang, X. Deng, S. Pu, and Z. Huang, “Residual convolutional ctc networks for automatic speech recognition,” 2017. ArXiv: [1702.07793](#).
- [60] A. Hannun, A. Lee, Q. Xu, and R. Collobert, “Sequence-to-sequence speech recognition with time-depth separable convolutions,” 2019. ArXiv: [1904.02619](#).
- [61] J. Li, V. Lavrukhin, B. Ginsburg, *et al.*, “Jasper: An end-to-end convolutional neural acoustic model,” 2019. ArXiv: [1904.03288](#).
- [62] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, “Fully convolutional speech recognition,” 2019. ArXiv: [1812.06864](#).

- [63] W. Han, Z. Zhang, Y. Zhang, *et al.*, “Contextnet: Improving convolutional neural networks for automatic speech recognition with global context,” 2020. ArXiv: [2005.03191](#).
- [64] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.
- [65] L. Dong, S. Xu, and B. Xu, “Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition,” in *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5884–5888.
- [66] L. D. Consortium *et al.*, “The darpa timit acoustic-phonetic continuous speech corpus,” *NIST Speech CD*, pp. 1–1, 1990.
- [67] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, “Listen, attend and spell,” 2015. ArXiv: [1508.01211](#).
- [68] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” in *2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.
- [69] D. B. Paul and J. M. Baker, “The design for the wall street journal-based csr corpus,” in *Human Language Technology - The Baltic Perspectiv*, 1992.
- [70] Q. Zhang, H. Lu, H. Sak, *et al.*, “Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss,” in *2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 7829–7833.
- [71] A. Gulati, J. Qin, C.-C. Chiu, *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” 2020. ArXiv: [2005.08100](#).
- [72] R. Naika, “An overview of automatic speaker verification system,” in *Intelligent Computing and Information and Communication: Proceedings of 2nd Int. Conf., ICICC 2017*, Springer, 2018, pp. 603–610.
- [73] Y. Tu, W. Lin, and M.-W. Mak, “A survey on text-dependent and text-independent speaker verification,” *IEEE Access*, vol. 10, pp. 99 038–99 049, 2022.
- [74] B. S. Atal, “Automatic speaker recognition based on pitch contours,” *The Journal of the Acoustical Society of America*, vol. 52, no. 6B, pp. 1687–1697, 1972.
- [75] G. R. Doddington, “A method or speaker verification,” *The Journal of the Acoustical Society of America*, vol. 49, no. 1A.Supplement, pp. 139–139, 1971.
- [76] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, “Support vector machines for speaker and language recognition,” *Computer Speech & Language*, vol. 20, no. 2-3, pp. 210–229, 2006.
- [77] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted gaussian mixture models,” *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.
- [78] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, “Speaker and session variability in gmm-based speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1448–1460, 2007.

- [79] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.
- [80] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.
- [81] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, 2005.
- [82] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI communications*, vol. 30, no. 2, pp. 169–190, 2017.
- [83] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.
- [84] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, ISCA, 2020.
- [85] Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, and L. R. Dai, "Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system.," in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019, pp. 361–365.
- [86] S. Ioffe, "Probabilistic linear discriminant analysis," in *Computer Vision – ECCV*, Springer Berlin Heidelberg, 2006, pp. 531–542.
- [87] T. Hastie and R. Tibshirani, "Discriminant analysis by gaussian mixtures," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 155–176, 1996.
- [88] S. Cumani, S. Sarni, *et al.*, "Impostor score statistics as quality measures for the calibration of speaker verification systems.," in *Odyssey*, 2022, pp. 25–32.
- [89] C. H. Coker, "A model of articulatory dynamics and control," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 452–460, 1976.
- [90] J. Allen, S. Hunnicutt, R. Carlson, and B. Granstrom, "Mitalk-79: The 1979 mit text-to-speech system," *The Journal of the Acoustical Society of America*, vol. 65, no. S1, S130–S130, 1979.
- [91] H. Valbret, E. Moulines, and J. P. Tubach, "Voice transformation using psola technique," *Speech Communication*, vol. 11, no. 2-3, pp. 175–187, 1992.
- [92] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *2000 IEEE Int. Conf. on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*, IEEE, vol. 3, 2000, pp. 1315–1318.
- [93] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.
- [94] A. M. Liberman, F. Ingemann, L. Lisker, P. Delattre, and F. S. Cooper, "Minimal rules for synthesizing speech," *The Journal of the Acoustical Society of America*, vol. 31, no. 11, pp. 1490–1499, 1959.

- [95] D. H. Klatt, "Review of text-to-speech conversion for english," *The Journal of the Acoustical Society of America*, vol. 82, no. 3, pp. 737–793, 1987.
- [96] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," 2021. ArXiv: [2106.15561](#).
- [97] R. A. Khan and J. S. Chitode, "Concatenative speech synthesis: A review," *Int. Journal of Computer Applications*, vol. 136, pp. 1–6, 2016.
- [98] A Indumathi and E Chandra, "Survey on speech synthesis," *Signal Processing: An Int. Journal (SPIJ)*, vol. 6, no. 5, p. 140, 2012.
- [99] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai, "An adaptive algorithm for mel-cepstral analysis of speech.," in *1992 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 92, 1992, pp. 137–140.
- [100] S. Imai, K. Sumita, and C. Furuichi, "Mel log spectrum approximation (mlsa) filter for speech synthesis," *Electronics and Communications in Japan (Part I: Communications)*, vol. 66, no. 2, pp. 10–18, 1983.
- [101] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. Inf. Syst.*, vol. 99-D, pp. 1877–1884, 2016.
- [102] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. D. Richards, "Normalization of non-standard words," *Computer speech & language*, vol. 15, no. 3, pp. 287–333, 2001.
- [103] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.
- [104] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine Learning*, vol. 34, pp. 177–210, 1999.
- [105] N. Beuck, A. Köhn, and W. Menzel, "Decision strategies in incremental pos tagging," in *Nordic Conference of Computational Linguistics*, 2011.
- [106] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (dnn) for parametric tts synthesis," in *2014 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3829–3833.
- [107] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2014.
- [108] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015. ArXiv: [1511.07122](#).
- [109] S. Ö. Arık, M. Chrzanowski, A. Coates, *et al.*, "Deep voice: Real-time neural text-to-speech," in *Int. Conf. on Machine Learning*, PMLR, 2017, pp. 195–204.
- [110] W. Wang, S. Xu, B. Xu, *et al.*, "First step towards end-to-end parametric tts synthesis: Generating spectral parameters with neural attention.," in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2016, pp. 2243–2247.
- [111] Y. Ren, C. Hu, X. Tan, *et al.*, "Fastspeech 2: Fast and high-quality end-to-end text to speech," 2020. ArXiv: [2006.04558](#).

- [112] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, “Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone,” in *Int. Conf. on Machine Learning*, PMLR, 2022, pp. 2709–2720.
- [113] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, “Grad-tts: A diffusion probabilistic model for text-to-speech,” in *Int. Conf. on Machine Learning*, PMLR, 2021, pp. 8599–8608.
- [114] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, “Diff-tts: A denoising diffusion model for text-to-speech,” 2021. ArXiv: [2104.01409](#).
- [115] Z. Borsos, R. Marinier, D. Vincent, *et al.*, “Audiolm: A language modeling approach to audio generation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [116] C. Wang, S. Chen, Y. Wu, *et al.*, “Neural codec language models are zero-shot text to speech synthesizers,” 2023. ArXiv: [2301.02111](#).
- [117] Y. Zhang, L. Deng, and Y. Wang, “Unified mandarin tts front-end based on distilled bert model,” 2020. ArXiv: [2012.15404](#).
- [118] J. Pan, X. Yin, Z. Zhang, *et al.*, “A unified sequence-to-sequence front-end model for mandarin text-to-speech synthesis,” in *2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 6689–6693.
- [119] Y. Ren, Y. Ruan, X. Tan, *et al.*, “Fastspeech: Fast, robust and controllable text to speech,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [120] R. Liu, B. Sisman, F. Bao, G. Gao, and H. Li, “Modeling prosodic phrasing with multi-task learning in tacotron-based tts,” *IEEE Signal Processing Letters*, vol. 27, pp. 1470–1474, 2020.
- [121] I. Vallés-Pérez, J. Roth, G. Beringer, R. Barra-Chicote, and J. Droppo, “Improving multi-speaker tts prosody variance with a residual encoder and normalizing flows,” 2021. ArXiv: [2106.05762](#).
- [122] T. Hayashi, S. Watanabe, T. Toda, K. Takeda, S. Toshniwal, and K. Livescu, “Pre-trained text embeddings for enhanced text-to-speech synthesis,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019, pp. 4430–4434.
- [123] R. Liu, B. Sisman, and H. Li, “Graphspeech: Syntax-aware graph attention network for neural speech synthesis,” in *2021 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 6059–6063.
- [124] S. Bhatt, A. Jain, and A. Dev, “Acoustic modeling in speech recognition: A systematic review,” *Int. Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, 2020.
- [125] N. Adiga and S. Prasanna, “Acoustic features modelling for statistical parametric speech synthesis: A review,” *IETE Technical Review*, vol. 36, no. 2, pp. 130–149, 2019.
- [126] Y. Wang, R. Skerry-Ryan, D. Stanton, *et al.*, “Tacotron: Towards end-to-end speech synthesis,” 2017. ArXiv: [1703.10135](#).
- [127] J. Kim, J. Kong, and J. Son, “Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,” in *Int. Conf. on Machine Learning*, PMLR, 2021, pp. 5530–5540.

- [128] D. Griffin and J. Lim, “Signal estimation from modified short-time fourier transform,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.
- [129] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” 2015. ArXiv: [1505.00387](#).
- [130] J. Shen, R. Pang, R. J. Weiss, *et al.*, “Natural tts synthesis by conditioning wavenet on mel spectrogram predictions,” in *2018 IEEE Int. Conf. on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 4779–4783.
- [131] Y. Wang, D. Stanton, Y. Zhang, *et al.*, “Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis,” in *Int. Conf. on Machine Learning*, PMLR, 2018, pp. 5180–5189.
- [132] R. Skerry-Ryan, E. Battenberg, Y. Xiao, *et al.*, “Towards end-to-end prosody transfer for expressive speech synthesis with tacotron,” in *Int. Conf. on Machine Learning*, PMLR, 2018, pp. 4693–4702.
- [133] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with transformer network,” in *Proceedings of the AAAI Conf. on Artificial Intelligence*, vol. 33, 2019, pp. 6706–6713.
- [134] I. Elias, H. Zen, J. Shen, *et al.*, “Parallel tacotron 2: A non-autoregressive neural tts model with differentiable duration modeling,” 2021. ArXiv: [2103.14574](#).
- [135] Y. Ren, C. Hu, X. Tan, *et al.*, “Fastspeech 2: Fast and high-quality end-to-end text to speech,” 2020. ArXiv: [2006.04558](#).
- [136] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” 2015. ArXiv: [1506.07503](#).
- [137] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, “Dynamic convolution: Attention over convolution kernels,” 2020. ArXiv: [1912.03458](#).
- [138] J. Shen, Y. Jia, M. Chrzanowski, *et al.*, “Non-attentive tacotron: Robust and controllable neural tts synthesis including unsupervised duration modeling,” 2020. ArXiv: [2010.04301](#).
- [139] W. Ping, K. Peng, A. Gibiansky, *et al.*, “Deep voice 3: 2000-speaker neural text-to-speech,” *International Conference on Learning Representations (ICLR)*, pp. 214–217, 2018.
- [140] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [141] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in *2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3617–3621.
- [142] S. Hihi and Y. Bengio, “Hierarchical recurrent neural networks for long-term dependencies,” *Advances in Neural Information Processing Systems*, vol. 8, 1995.
- [143] S. Mehri, K. Kumar, I. Gulrajani, *et al.*, “SAMPLERNN: An unconditional end-to-end neural audio generation model,” 2016. ArXiv: [1612.07837](#).
- [144] J.-M. Valin and J. Skoglund, “Lpcnet: Improving neural speech synthesis through linear prediction,” in *2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 5891–5895.

- [145] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *Int. Conf. on Machine Learning*, PMLR, 2015, pp. 1530–1538.
- [146] G. G. Pires and M. A. Figueiredo, “Variational mixture of normalizing flows,” 2020. ArXiv: [2009.00585](#).
- [147] A. Oord, Y. Li, I. Babuschkin, *et al.*, “Parallel wavenet: Fast high-fidelity speech synthesis,” in *Int. Conf. on Machine Learning*, PMLR, 2018, pp. 3918–3926.
- [148] S. Kim, S. G. Lee, J. Song, J. Kim, and S. Yoon, “Flowwavenet: A generative flow for raw audio,” 2018. ArXiv: [1811.02155](#).
- [149] Y. Cai and C. Daskalakis, “On minmax theorems for multiplayer games,” in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2011, pp. 217–234.
- [150] M. Bińkowski, J. Donahue, S. Dieleman, *et al.*, “High fidelity speech synthesis with adversarial networks,” 2019. ArXiv: [1909.11646](#).
- [151] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” 2018. ArXiv: [1802.04208](#).
- [152] J. Kong, J. Kim, and J. Bae, “Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.
- [153] J. Yang, J. Lee, Y. Kim, H. Cho, and I. Kim, “Vocgan: A high-fidelity real-time vocoder with a hierarchically-nested adversarial network,” 2020. ArXiv: [2007.15256](#).
- [154] J. Su, Z. Jin, and A. Finkelstein, “Hifi-gan-2: Studio-quality speech enhancement via generative adversarial networks conditioned on acoustic features,” in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2021, pp. 166–170.
- [155] K. Kumar, R. Kumar, T. De Boissiere, *et al.*, “Melgan: Generative adversarial networks for conditional waveform synthesis,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [156] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.
- [157] D. Watson, J. Ho, M. Norouzi, and W. Chan, “Learning to efficiently sample from diffusion probabilistic models,” 2021. ArXiv: [2106.03802](#).
- [158] S.-g. Lee, H. Kim, C. Shin, *et al.*, “Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior,” 2021. ArXiv: [2106.06406](#).
- [159] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, “Soundstream: An end-to-end neural audio codec,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.
- [160] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, “High fidelity neural audio compression,” 2022. ArXiv: [2210.13438](#).
- [161] K. Lakhotia, E. Kharonov, W.-N. Hsu, *et al.*, “On generative spoken language modeling from raw audio,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1336–1354, 2021.
- [162] C. Wang, S. Chen, Y. Wu, *et al.*, “Neural codec language models are zero-shot text to speech synthesizers,” 2023. ArXiv: [2301.02111](#).

- [163] A. Radford, “Improving language understanding by generative pre-training,” 2018.
- [164] N. J. Shah and H. A. Patil, “A novel approach to remove outliers for parallel voice conversion,” *Computer Speech & Language*, vol. 58, pp. 127–152, 2019.
- [165] S. H. Mohammadi and A. Kain, “An overview of voice conversion systems,” *Speech Communication*, vol. 88, pp. 65–82, 2017.
- [166] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, “A first step towards text-independent voice conversion,” in *Int. Conf. on Speech and Language Processing (ICSLP)*, 2004.
- [167] T. Kaneko and H. Kameoka, “CycleGAN-vc: Non-parallel voice conversion using cycle-consistent adversarial networks,” in *2018 26th European Signal Processing Conf. (EUSIPCO)*, IEEE, 2018, pp. 2100–2104.
- [168] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, “Autovc: Zero-shot voice style transfer with only autoencoder loss,” in *Int. Conf. on Machine Learning*, PMLR, 2019, pp. 5210–5219.
- [169] T. Toda, A. W. Black, and K. Tokuda, “Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.
- [170] A. T. Liu, P.-c. Hsu, and H.-Y. Lee, “Unsupervised end-to-end learning of discrete linguistic units for voice conversion,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, ISCA, 2019, 1108–1112.
- [171] K. Kobayashi, W. C. Huang, Y. C. Wu, P. L. Tobing, T. Hayashi, and T. Toda, “Crank: An open-source software for nonparallel voice conversion based on vector-quantized variational autoencoder,” in *2021 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 5934–5938.
- [172] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, “CycleGAN-vc2: Improved cycleGAN-based non-parallel voice conversion,” in *2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 6820–6824.
- [173] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros, “Learning dense correspondence via 3d-guided cycle consistency,” in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 117–126.
- [174] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, “Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2018, pp. 266–273.
- [175] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, “Stargan-vc2: Rethinking conditional methods for stargan-based voice conversion,” 2019. ArXiv: [1907.12279](#).
- [176] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, “Diffusion-based voice conversion with fast maximum likelihood sampling scheme,” 2021. ArXiv: [2109.13821](#).
- [177] H.-Y. Choi, S.-H. Lee, and S.-W. Lee, “Dddm-vc: Decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion,” in *Proceedings of the AAAI Conf. on Artificial Intelligence*, vol. 38, 2024, pp. 17 862–17 870.

- [178] H. Wang, T. Thebaud, J. Villalba, *et al.*, “Duta-vc: A duration-aware typical-to-atypical voice conversion approach with diffusion probabilistic model,” 2023. ArXiv: [2306.10588](#).
- [179] Y. Y. Lin, C. M. Chien, J. H. Lin, H. Y. Lee, and L. S. Lee, “Fragmentvc: Any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention,” in *2021 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 5939–5943.
- [180] W. C. Huang, S. W. Yang, T. Hayashi, H. Y. Lee, S. Watanabe, and T. Toda, “S3prl-vc: Open-source voice conversion framework with self-supervised speech representations,” in *2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 6552–6556.
- [181] H. S. Tsai, H. J. Chang, W. C. Huang, *et al.*, “Superb-sg: Enhanced speech processing universal performance benchmark for semantic and generative capabilities,” 2022. ArXiv: [2203.06849](#).
- [182] Z. Wang, Y. Chen, L. Xie, Q. Tian, and Y. Wang, “Lm-vc: Zero-shot voice conversion via speech generation based on language models,” *IEEE Signal Processing Letters*, 2023.
- [183] Z. Wang, Y. Chen, X. Wang, L. Xie, and Y. Wang, “Streamvoice: Streamable context-aware language modeling for real-time zero-shot voice conversion,” 2024. ArXiv: [2401.11053](#).
- [184] C. H. Chan, K. Qian, Y. Zhang, and M. Hasegawa-Johnson, “Speechsplit2. 0: Unsupervised speech disentanglement for voice conversion without tuning autoencoder bottlenecks,” in *2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 6332–6336.
- [185] Z. Du, B. Sisman, K. Zhou, and H. Li, “Disentanglement of emotional style and speaker identity for expressive voice conversion,” 2021. ArXiv: [2110.10326](#).
- [186] Z. Wang, W. Ge, X. Wang, *et al.*, “Accent and speaker disentanglement in many-to-many voice conversion,” in *12th Int. Symposium on Chinese Spoken Language Processing (ISCSLP)*, IEEE, 2021, pp. 1–5.
- [187] X. Zhao, F. Liu, C. Song, *et al.*, “Disentangling content and fine-grained prosody information via hybrid asr bottleneck features for voice conversion,” in *2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 7022–7026.
- [188] S. Yang, M. Tantrawenith, H. Zhuang, *et al.*, “Speech representation disentanglement with adversarial mutual information learning for one-shot voice conversion,” 2022. ArXiv: [2208.08757](#).
- [189] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, “Voice conversion from non-parallel corpora using variational auto-encoder,” in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conf. (APSIPA)*, IEEE, 2016, pp. 1–6.
- [190] S. Liu, J. Zhong, L. Sun, X. Wu, X. Liu, and H. Meng, “Voice conversion across arbitrary speakers based on a single target-speaker utterance,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2018, pp. 496–500.

- [191] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, “Autovc: Zero-shot voice style transfer with only autoencoder loss,” in *Int. Conf. on Machine Learning*, PMLR, 2019, pp. 5210–5219.
- [192] Z. Chen and P. Zhang, “Tqvqc: Transformer based vector quantized variational autoencoder with ctc loss for voice conversion,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2021, pp. 826–830.
- [193] A. Lukács, “What is privacy? the history and definition of privacy,” 2016.
- [194] D. Solove, “Nothing to hide: The false tradeoff between privacy and security,” *Nothing to Hide: The False Tradeoff Between Privacy and Security*, May 2011.
- [195] D. J. Solove, “Conceptualizing privacy,” *California Law Review*, pp. 1087–1155, 2002.
- [196] L. Brandeis and S. Warren, “The right to privacy,” *Harvard Law Review*, vol. 4, no. 5, pp. 193–220, 1890.
- [197] A. Nautsch, C. Jasserand, E. Kindt, M. Todisco, I. Trancoso, and N. Evans, “The gdpr & speech data: Reflections of legal and technology communities, first steps towards a common understanding,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, Sep. 2019, pp. 3695–3699.
- [198] H. Nissenbaum, “Protecting privacy in an information age: The problem of privacy in public,” *Law and Philosophy*, pp. 559–596, 1998.
- [199] D. J. Dubois, R. Kolcun, A. M. Mandalari, M. T. Paracha, D. Choffnes, and H. Haddadi, “When speakers are all ears: Characterizing misactivations of iot smart speakers,” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 4, pp. 255–276, 2020.
- [200] F. Nespoli, D. Barreda, J. Bitzer, and P. A. Naylor, “Two-Stage Voice Anonymization for Enhanced Privacy,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2023, pp. 3854–3858.
- [201] N. Tomashenko, X. Wang, E. Vincent, *et al.*, “The voiceprivacy 2020 challenge: Results and findings,” *Computer, Speech and Language*, vol. 74, p. 101362, 2022.
- [202] A. Nelus, J. Ebbers, R. Haeb-Umbach, and R. Martin, “Privacy-Preserving Variational Information Feature Extraction for Domestic Activity Monitoring versus Speaker Identification,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019, pp. 3710–3714.
- [203] P. Thaine and G. Penn, “Extracting mel-frequency and bark-frequency cepstral coefficients from encrypted signals,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019.
- [204] B. M. L. Srivastava, A. Bellet, M. Tommasi, and E. Vincent, “Privacy-Preserving Adversarial Representation Learning in ASR: Reality or Illusion?” In *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019, pp. 3700–3704.
- [205] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, “Towards efficient and privacy-preserving federated deep learning,” in *2019 IEEE Int. Conf. on Communications (ICC)*, 2019, pp. 1–6.
- [206] Z. Zhao, M. Luo, and W. Ding, “Deep leakage from model in federated learning,” 2022. ArXiv: [2206.04887](https://arxiv.org/abs/2206.04887).

- [207] B. M. L. Srivastava, N. Tomashenko, X. Wang, *et al.*, “Design choices for x-vector based speaker anonymization,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2020.
- [208] A. S. Shamsabadi, B. M. L. Srivastava, A. Bellet, *et al.*, “Differentially private speaker anonymization,” in *Privacy Enhancing Technologies Symposium*, 2022.
- [209] N. Tomashenko, X. Wang, X. Miao, *et al.*, “The voice privacy 2022 challenge evaluation plan,” 2022. eprint: [2203.12468](#).
- [210] H. Zen, V. Dang, R. Clark, *et al.*, “LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2019, pp. 1526–1530.
- [211] J. Yamagishi, C. Veaux, and K. MacDonald, *CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92)*, 2019.
- [212] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: A large-scale speaker identification dataset,” 2017. ArXiv: [1706.08612](#).
- [213] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2018.
- [214] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, “The kaldi speech recognition toolkit,” in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.
- [215] M. Ravanelli, T. Parcollet, P. Plantinga, *et al.*, “SpeechBrain: A general-purpose speech toolkit,” 2021, arXiv:2106.04624. ArXiv: [2106.04624](#).
- [216] M. Maouche, B. M. L. Srivastava, N. Vauquier, A. Bellet, M. Tommasi, and E. Vincent, “Enhancing speech privacy with slicing,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2022.
- [217] Z. Chen, A. Rosenberg, Y. Zhang, G. Wang, B. Ramabhadran, and P. J. Moreno, “Improving speech recognition using gan-based speech synthesis and contrastive unspoken text selection,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2020.
- [218] J. Li, R. Gadde, B. Ginsburg, and V. Lavrukhin, “Training neural speech recognition systems with synthetic speech augmentation,” *arXiv*, 2018. arXiv: [1811.00707](#).
- [219] D. J. Hirst, “A praat plugin for momel and intsint with improved algorithms for modelling and coding intonation,” in *16th Int. Congress of Phonetic Sciences*, Saarbrücken, Germany, Aug. 2007.
- [220] Y. Adi, N. Zeghidour, R. Collobert, N. Usunier, V. Liptchinsky, and G. Synnaeve, “To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition,” in *2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3742–3746.
- [221] T. Kohonen and T. Kohonen, *Self-Organizing Maps*. Springer, 2001, ch. Learning vector quantization, pp. 245–261.
- [222] P. Champion, D. Jouvet, and A. Larcher, “Are disentangled representations all you need to build speaker anonymization systems?,” 2022. ArXiv: [2208.10497](#).

- [223] S. Meyer, P. Tilli, F. Lux, P. Denisov, J. Koch, and N. T. Vu, “Cascade of phonetic speech recognition, speaker embeddings gan and multispeaker speech synthesis for the voiceprivacy 2022 challenge,” in *2nd Symposium on Security and Privacy in Speech Communication*, 2022.
- [224] X. Miao, X. Wang, E. Cooper, J. Yamagishi, and N. Tomashenko, “Speaker anonymization using orthogonal householder neural network,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [225] Ü. E. Gaznepoglu, A. Leschanowsky, and N. Peters, “Voiceprivacy 2022 system description: Speaker anonymization with feature-matched f0 trajectories,” 2022. ArXiv: [2210.17338](https://arxiv.org/abs/2210.17338).
- [226] R. Khamsehashari, Y. Sinha, J. Hintz, *et al.*, “Voice privacy-leveraging multi-scale blocks with ecapa-tdnn se-res2next extension for speaker anonymization,” in *2nd Symposium on Security and Privacy in Speech Communication*, 2022, pp. 43–48.
- [227] K. Hiroto, T. Shinnosuke, S. Sayaka, and K. Hitoshi, “Robustness of Signal Processing-Based Pseudonymization Method Against Decryption Attack,” in *The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 287–293.
- [228] J. Kim, S. Kim, J. Kong, and S. Yoon, “Glow-tts: A generative flow for text-to-speech via monotonic alignment search,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 8067–8077, 2020.
- [229] H. S. Heo, B.-J. Lee, J. Huh, and J. S. Chung, “Clova baseline system for the voxceleb speaker recognition challenge 2020,” *arXiv*, 2020, DOI: <https://doi.org/10.48550/ARXIV.2009.14153>.
- [230] S. Team, *Silero vad: Pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier*, <https://github.com/snakers4/silero-vad>, 2021.
- [231] C. O. Mawalim, S. Okada, and M. Unoki, “Speaker anonymization by pitch shifting based on time-scale modification,” in *2nd Symposium on Security and Privacy in Speech Communication joined with 2nd VoicePrivacy Challenge (SPSC 2022)*, 2022.
- [232] S. Meyer, F. Lux, P. Denisov, J. Koch, P. Tilli, and T. Vu, “Speaker anonymization with phonetic intermediate representations,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, Sep. 2022, pp. 4925–4929.
- [233] F. Nespola, D. Barreda, and P. A. Naylor, “Zero shot text to speech augmentation for automatic speech recognition on low-resource accented speech corpora,” in *57th Asilomar Conf. on Signals, Systems, and Computers, ACSSC 2023, Pacific Grove, CA, USA, October 29 - Nov. 1, 2023*, IEEE, 2023, pp. 1080–1084.
- [234] I. C. Kuo, “Addressing the issue of teaching english as a lingua franca,” *ELT journal*, vol. 60, no. 3, pp. 213–221, 2006.
- [235] Y. Zhang, D. S. Park, W. Han, *et al.*, “Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1519–1532, 2022.
- [236] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Int. Conf. on Machine Learning*, PMLR, 2023, pp. 28 492–28 518.

- [237] M. E. Sadeghi, H. Sheikhzadeh, and M. J. Emadi, “A proposed method to improve the wer of an asr system in the noisy reverberant room,” *Journal of the Franklin Institute*, vol. 361, no. 1, pp. 99–109, 2024.
- [238] A. DiChristofano, H. Shuster, S. Chandra, and N. Patwari, “Global performance disparities between english-language accents in automatic speech recognition,” 2022. ArXiv: [2208.01157](#).
- [239] H. J. Na and J. S. Park, “Accented speech recognition based on end-to-end domain adversarial training of neural networks,” *Applied Sciences*, vol. 11, no. 18, 2021.
- [240] S. Liu, D. Wang, Y. Cao, *et al.*, “End-to-end accent conversion without using native utterances,” in *2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6289–6293.
- [241] P. Klumpp, P. Chitkara, L. Sari, *et al.*, “Synthetic cross-accent data augmentation for automatic speech recognition,” *ArXiv*, 2023.
- [242] J. L. Martin, “Spoken corpora data, automatic speech recognition, and bias against african american language: The case of habitual’be’,” in *Proceedings of the 2021 ACM Conf. on fairness, accountability, and transparency*, 2021, pp. 284–284.
- [243] K. Kulkarni, S. Sengupta, V. Ramasubramanian, J. G. Bauer, and G. Stemmer, “Accented indian english asr: Some early results,” in *2008 IEEE Spoken Language Technology Workshop*, IEEE, 2008, pp. 225–228.
- [244] Y. Zhang, Y. Zhang, B. M. Halpern, T. Patel, and O. Scharenborg, “Mitigating bias against non-native accents,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, vol. 2022, 2022, pp. 3168–3172.
- [245] D. Graddol, “The decline of the native speaker,” *AILA review*, vol. 13, no. 1, pp. 57–68, 1999.
- [246] R. Cumbal, B. Moell, J. Lopes, and O. Engwall, “You don’t understand me!: Comparing asr results for l1 and l2 speakers of swedish,” 2024. ArXiv: [2405.13379](#).
- [247] H. Zhu, L. Wang, P. Zhang, and Y. Yan, “Multi-accent adaptation based on gate mechanism,” 2020. ArXiv: [2011.02774](#).
- [248] N. Das, S. Bodapati, M. Sunkara, S. Srinivasan, and D. Chau, “Best of both worlds: Robust accented speech recognition with adversarial transfer learning,” 2021. ArXiv: [2103.05834](#).
- [249] J. Li, V. Manohar, P. Chitkara, *et al.*, “Accent-robust automatic speech recognition using supervised and unsupervised wav2vec embeddings,” 2021. ArXiv: [2110.03520](#).
- [250] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, “Domain-adversarial neural networks,” 2014. ArXiv: [1412.4446](#).
- [251] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, pp. 41–75, 1997.
- [252] X. Yang, K. Audhkhasi, A. Rosenberg, S. Thomas, B. Ramabhadran, and M. A. Hasegawa-Johnson, “Joint modeling of accents and acoustics for multi-accent speech recognition,” *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2018.
- [253] A. Jain, M. Upreti, and P. Jyothi, “Improved accented speech recognition using accent embeddings and multi-task learning,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2018, pp. 2454–2458.

- [254] X. Liu, F. Zhang, Z. Hou, *et al.*, “Self-supervised learning: Generative or contrastive,” *IEEE transactions on knowledge and data engineering*, vol. 35, no. 1, pp. 857–876, 2021.
- [255] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “Wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.
- [256] K. Deng, S. Cao, and L. Ma, “Improving accent identification and accented speech recognition under a framework of self-supervised learning,” 2021. ArXiv: [2109.07349](#).
- [257] M. A. T. Turan, E. Vincent, and D. Jouviet, “Achieving multi-accent asr via unsupervised acoustic model adaptation,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2020.
- [258] E. Yilmaz, H. v. d. Heuvel, and D. A. van Leeuwen, “Acoustic and textual data augmentation for improved asr of code-switching speech,” 2018. ArXiv: [1807.10945](#).
- [259] G. Karakasidis, N. Robinson, Y. Getman, *et al.*, “Multilingual tts accent impressions for accented asr,” in *Int. Conf. on Text, Speech, and Dialogue*, Springer, 2023, pp. 317–327.
- [260] S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, “Data augmentation for asr using tts via a discrete representation,” in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, IEEE, 2021, pp. 68–75.
- [261] C. T. Do, S. Imai, R. Doddipatla, and T. Hain, “Improving accented speech recognition using data augmentation based on unsupervised text-to-speech synthesis,” 2024. ArXiv: [2407.04047](#).
- [262] Z. Chen, A. Rosenberg, Y. Zhang, *et al.*, “Semi-supervision in asr: Sequential mixmatch and factorized tts-based augmentation,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2021, pp. 736–740.
- [263] X. Shi, F. Yu, Y. Lu, *et al.*, “The accented english speech recognition challenge 2020: Open datasets, tracks, baselines, results and methods,” *ArXiv*, 2021.
- [264] K. Ito and L. Johnson, *The LJ Speech Dataset*, <https://keithito.com/LJ-Speech-Dataset/>, 2017.
- [265] E. Casanova, A. C. Junior, C. Shulby, *et al.*, “TTS-portuguese corpus: A corpus for speech synthesis in brazilian portuguese,” *Language Resources and Evaluation*, vol. 56, no. 3, pp. 1043–1055, 2022.
- [266] I. Solak, *The m-ailabs speech dataset*. <https://www.caito.de/2019/01/the-m-ailabs-speechdataset/>, 2017.
- [267] D. S. Park, W. Chan, Y. Zhang, *et al.*, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, ISCA, Sep. 2019.
- [268] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, “A study on data augmentation of reverberant speech for robust speech recognition,” *2017 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5220–5224, 2017.
- [269] A. Vyas, B. Shi, M. Le, *et al.*, “Audiobox: Unified audio generation with natural language prompts,” 2023. ArXiv: [2312.15821](#).

- [270] M. Łajszczak, G. Cámbara, Y. Li, *et al.*, “Base tts: Lessons from building a billion-parameter text-to-speech model on 100k hours of data,” 2024. ArXiv: [2402.08093](#).
- [271] J. Tremblay, A. Prakash, D. Acuna, *et al.*, “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [272] C. M. de Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins, “Next-generation deep learning based on simulators and synthetic data,” *Trends in Cognitive Sciences*, vol. 26, no. 2, pp. 174–187, 2022.
- [273] Y. Abdullin, D. Molla-Aliod, B. Ofoghi, J. Yearwood, and Q. Li, “Synthetic dialogue dataset generation using llm agents,” 2024. ArXiv: [2401.17461](#).
- [274] F. Borra, C. Savelli, G. Rosso, A. Koudounas, and F. Giobergia, “Malto at semeval-2024 task 6: Leveraging synthetic data for llm hallucination detection,” 2024. ArXiv: [2403.00964](#).
- [275] G. Pallotti *et al.*, “Conversation analysis: Methodology, machinery and application to specific settings,” *Conversation Analysis and language for Specific Purposes*, pp. 37–67, 2007.
- [276] F. Nespoli, J. Pohlhausen, P. A. Naylor, and J. Bitzer, “Long-term conversation analysis: Exploring utility and privacy,” in *Speech Communication; 15th ITG Conf.*, VDE, 2023, pp. 26–30.
- [277] M. VanDam, S. E. Ambrose, and M. P. Moeller, “Quantity of Parental Language in the Home Environments of Hard-of-Hearing 2-Year-Olds,” *Journal of Deaf Studies and Deaf Education*, vol. 17, no. 4, pp. 402–420, Oct. 2012.
- [278] J. Heritage, “Conversation analysis as social theory,” in *The New Blackwell Companion to Social Theory*, B. Turner, Ed., 2008, pp. 300–320.
- [279] K. Smeds, F. Wolters, and M. Rung, “Estimation of Signal-to-Noise Ratios in Realistic Sound Scenarios,” *Journal of the American Academy of Audiology*, vol. 26, no. 02, pp. 183–196, Feb. 2015.
- [280] P. Segerdahl, “Scientific studies of aspects of everyday life: The example of conversation analysis,” *Language & Communication*, vol. 18, no. 4, pp. 275–323, 1998.
- [281] D. Wyatt, T. Choudhury, and J. A. Bilmes, “Conversation detection and speaker segmentation in privacy-sensitive situated speech data.,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2007, pp. 586–589.
- [282] J. Bitzer, S. Kissner, and I. Holube, “Privacy-aware acoustic assessments of everyday life,” *Journal of the Audio Engineering Society*, vol. 64, no. 6, pp. 395–404, 2016.
- [283] Y. Nakajima, M. Matsuda, K. Ueda, and G. B. Remijn, “Temporal Resolution Needed for Auditory Communication: Measurement With Mosaic Speech,” *Frontiers in Human Neuroscience*, vol. 12, 2018.
- [284] F. Adolfi, J. S. Bowers, and D. Poeppel, “Successes and critical failures of neural networks in capturing human-like speech recognition,” *Neural Networks*, vol. 162, pp. 199–211, 2023.

- [285] J. Patino, N. Tomashenko, M. Todisco, A. Nautsch, and N. Evans, “Speaker Anonymisation Using the McAdams Coefficient,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, ISCA, 2021, pp. 1099–1103.
- [286] U. Kowalk, S. Franz, H. Groenewold, I. Holube, P. v. Gablenz, and J. Bitzer, “olMEGA: An open source android solution for ecological momentary assessment,” *GMS Zeitschrift für Audiologie - Audiological Acoustics*, vol. 2, pp. 1–9, 2020.
- [287] D. O’Shaughnessy, “Linear predictive coding,” *IEEE Potentials*, vol. 7, no. 1, pp. 29–32, 1988.
- [288] D. Chicco, M. J. Warrens, and G. Jurman, “The matthews correlation coefficient (mcc) is more informative than cohen’s kappa and brier score in binary classification assessment,” *IEEE Access*, vol. 9, pp. 78 368–78 381, 2021.
- [289] D. Chicco and G. Jurman, “The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation,” *BMC genomics*, vol. 21, pp. 1–13, 2020.
- [290] N. Dawalatabad, M. Ravanelli, F. Grondin, J. Thienpondt, B. Desplanques, and H. Na, “ECAPA-TDNN embeddings for speaker diarization,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2021, pp. 3560–3564.
- [291] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, “The rich transcription 2006 spring meeting recognition evaluation,” in *Machine Learning for Multimodal Interaction*, S. Renals, S. Bengio, and J. G. Fiscus, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 309–322.
- [292] F. Landini, J. Profant, M. Diez, and L. Burget, “Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: Theory, implementation and analysis on standard tasks,” *Computer Speech & Language*, vol. 71, 2022.
- [293] M. Marzinzik and B. Kollmeier, “Speech pause detection for noise spectrum estimation by tracking power envelope dynamics,” *IEEE transactions on speech and audio processing*, vol. 10, no. 2, pp. 109–118, 2002.
- [294] J. Carletta, “Unleashing the killer corpus: Experiences in creating the multi-everything AMI meeting corpus,” *Language Resources and Evaluation*, vol. 41, no. 2, pp. 181–190, 2007.
- [295] T. Cord-Landwehr, T. von Neumann, C. Boeddeker, and R. Haeb-Umbach, “MMS-MSG: A Multi-Purpose Multi-Speaker Mixture Signal Generator,” in *2022 Int. Workshop on Acoustic Signal Enhancement (IWAENC)*, Bamberg, Germany: IEEE, 2022, pp. 1–5.
- [296] F. Nespola, D. Barreda, and P. Naylor, “Relative acoustic features for distance estimation in smart-homes,” in *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, 2022, pp. 724–728.
- [297] M. J. Taghizadeh, “Enabling speech applications using ad-hoc microphone arrays,” in *PhD dissertation, Lausanne: EPFL*, 2015.
- [298] T. Yoshioka, D. Dimitriadis, A. Stolcke, *et al.*, “Meeting transcription using asynchronous distant microphones,” *Proceedings of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, vol. 2019-Sept, pp. 2968–2972, 2019.

- [299] S. Pasha, “Analysis and enhancement of spatial sound scenes recorded using ad-hoc microphone arrays,” *PhD dissertation, School of Electrical, Computer and Telecommunications Engineering, University of Wollongong*, 2017.
- [300] S. Gergen and R. Martin, “Estimating source dominated microphone clusters in ad-hoc microphone arrays by fuzzy clustering in the feature space,” in *Speech Communication; 12. ITG Symposium*, 2016, pp. 1–5.
- [301] S. Miyabe, N. Ono, and S. Makino, “Blind compensation of inter-channel sampling frequency mismatch with maximum likelihood estimation in STFT domain,” *2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, pp. 674–678, 2013.
- [302] Z. Liu, Z. Zhang, L.-W. He, and P. Chou, “Energy-based sound source localization and gain normalization for ad hoc microphone arrays,” in *2007 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 2007, pp. 761–764.
- [303] Y. Liu, Z. Yang, X. Wang, and L. Jian, “Location, Localization, and Localizability,” *Journal of Computer Science Technology*, vol. 25, pp. 274–297, Mar. 2010.
- [304] H. Ding, Y. Huang, J. Bi, S. L. Gay, M. Si, and H. Qi, “Indoor Positioning Method Using WiFi RTT Based on S Identification and Range Calibration,” *ISPRS Int. Journal of Geo-Informatio*, vol. 9, no. 11, 2020.
- [305] D. Porcino and W. Hirt, “Ultra-wideband radio technogy: Potential and challenges ahead,” *IEEE Commun. Mag.*, vol. 41, no. 7, pp. 66–74, 2003.
- [306] Q. Wang, H. Ding, A. Men, *et al.*, “Light positioning: A high-accuracy visible light indoor positioning system based on attitude identification and propagation model,” *Int. J. of Distributed Sensor Networks*, vol. 14, no. 2, 2018.
- [307] S. Sheng, M. Gowda, and R. Roy Choudhury, “Closing the Gaps in Inertial Motion Tracking,” in *Annual Int. Conf. on Mobile Computing and Networking*, 2018, pp. 429–444.
- [308] S. Yun, Y.-C. Chen, and L. Qiu, “Turning a mobile device into a mouse in the air,” in *Proceedings of the 13th Annual Int. Conf. on Mobile Systems, Applications, and Services*, ser. MobiSys ’15, Florence, Italy: Association for Computing Machinery, 2015, 15–29.
- [309] L. Zhang, K. Liu, Y. Jiang, X.-Y. Li, Y. Liu, and P. Yang, “Montage: Combine frames with movement continuity for realtime multi-user tracking,” in *IEEE INFOCOM 2014 - IEEE Conf. on Computer Communications*, 2014, pp. 799–807.
- [310] M. Vajgl, P. Hurtik, and T. Nejezchleba, “Dist-YOLO: Fast Object Detection with Distance Estimation,” *Applied Sciences*, vol. 12, no. 3, 2022.
- [311] F. Zafari, A. Gkelias, and K. K. Leung, “A survey of indoor localization systems and technologies,” *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.
- [312] K. Qian, C. Wu, Z. Yang, Y. Liu, and K. Jamieson, “Widar: Decimeter-Level Passive Tracking via Velocity Monitoring with Commodity Wi-Fi,” in *ACM Int. Symp. on Mobile Ad Hoc Networking and Computing*, 2017.
- [313] H. Ding, J. Han, C. Qian, *et al.*, “Trio: Utilizing Tag Interference for Refined Localization of Passive RFID,” *IEEE Conf. on Comput. Commun.*, pp. 828–836, 2018.

- [314] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [315] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 4, pp. 320–327, 1976.
- [316] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” *arXiv*, 2017.
- [317] P. Peso Parada, D. Sharma, J. Lainez, D. Barreda, T. Waterschoot, and P. A. Naylor, “A single-channel non-intrusive ₅₀ estimator correlated with speech recognition performance,” *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 4, pp. 719–732, 2016.
- [318] P. Loganathan, A. W. H. Khong, and P. A. Naylor, “A sparseness controlled proportionate algorithm for acoustic echo cancellation,” *Proc. Eur. Signal Process. Conf. (EUSIPCO)*, pp. 1–5, 2008.
- [319] P. A. Naylor, N. D. Gaubitch, and E. Habets, “Signal-based performance evaluation of dereverberation algorithms,” *J. Elect. and Comput. Eng.*, vol. 2010, 2010.
- [320] J. Benesty and S. L. Gay, “An improved PNLMS algorithm,” in *2002 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 2002.
- [321] K. Kowalczyk, E. A. P. Habets, W. Kellermann, and P. A. Naylor, “Blind system identification using sparse learning for tdoa estimation of room reflections,” *IEEE Signal Processing Letters*, vol. 20, no. 7, pp. 653–656, 2013.
- [322] J. Arenas-Garcia and A. R. Figueiras-Vidal, “Adaptive combination of ipnlms filters for robust sparse echo cancellation,” in *2008 IEEE Workshop on Machine Learning for Signal Processing*, 2008, pp. 221–226.
- [323] R. Shwartz-Ziv and A. Armon, “Tabular Data: Deep Learning is Not All You Need,” *arXiv*, 2021.
- [324] R. J. Lewis, “An introduction to classification and regression tree (cart) analysis,” in *Annual meeting of the society for academic emergency medicine in San Francisco, California*, Citeseer, vol. 14, 2000.
- [325] R. Polikar, “Ensemble learning,” *Ensemble machine learning: Methods and applications*, pp. 1–34, 2012.
- [326] I. D. Mienye and Y. Sun, “A survey of ensemble learning: Concepts, algorithms, applications, and prospects,” *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.
- [327] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [328] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [329] J. Elith, J. R. Leathwick, and T. Hastie, “A working guide to boosted regression trees,” *Journal of animal ecology*, vol. 77, no. 4, pp. 802–813, 2008.
- [330] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* ” O’Reilly Media, Inc.”, 2022.

- [331] L. S. Shapley, “17. a value for n-person games,” in *Contributions to the Theory of Games (AM-28), Volume II*, H. W. Kuhn and A. W. Tucker, Eds. Princeton University Press, 2016, pp. 307–318.
- [332] R. Scheibler, E. Bezzam, and I. Dokmanic, “Pyroomacoustics: A Python package for audio room simulations and array processing algorithms,” in *2018 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.
- [333] I. Dokmanić, R. Parhizkar, A. Walther, Y. M. Lu, and M. Vetterli, “Acoustic echoes reveal room shape,” *of the Nat. Academy of Sci.*, vol. 110, no. 30, 2013.
- [334] P. P. Vaidyanathan, *Multirate systems and filter banks*, 1993.
- [335] Y. Huang and J. Benesty, “A class of frequency-domain adaptive approaches to blind multichannel identification,” *IEEE Trans. Signal Process.*, vol. 51, no. 1, pp. 11–24, 2003.
- [336] D. R. Morgan, J. Benesty, and M. M. Sondhi, “On the evaluation of estimated impulse responses,” *IEEE Signal Process. Lett.*, pp. 174–176, 1998.

Appendix A

Feature Correlation and Optimization Convergence

Definition A.0.1 (condition number). The **condition number** of a matrix is the ratio of the largest singular value to the smallest singular value. A matrix is ill-conditioned if the condition number is large, usually indicating that (i) the lowest singular value is orders of magnitude smaller than the highest one, and (ii) columns/rows of the matrix are highly correlated.

Definition A.0.2 (Hessian matrix). The **Hessian** matrix (**H**) is constituted by the second order derivatives of a function with respect to all pairs of variables. Given $X = f(x_1, x_2, \dots, x_N)$ the gradient $\nabla_X f$ is an N -dimensional vector and the Hessian is an $N \times N$ matrix.

A.1 Benefits of Decorrelated Features in Optimization

Decorrelated features improve the conditioning of the Hessian matrix, making optimization algorithms more stable and efficient. To illustrate this, let's consider the role of feature correlation in matrix conditioning. Let $\mathbf{X} \in \mathbb{R}^{N \times p}$ be the matrix of p features for N data points. The covariance matrix $\mathbf{C} \in \mathbb{R}^{p \times p}$ of the features is given by:

$$\mathbf{C} = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

The covariance matrix $\mathbf{C}$ describes the correlations between the features. If the features are highly correlated, some off-diagonal elements in $\mathbf{C}$ will be large, leading to a poorly conditioned matrix. In optimization problems, the Hessian matrix $\mathbf{H}$ often includes the covariance of the input features. For example, in the case of a simple quadratic loss function $\mathcal{L}(\mathbf{W}) = \frac{1}{2} \mathbf{W}^T \mathbf{C} \mathbf{W}$, where $\mathbf{W}$ is the parameter matrix, the Hessian is directly related to the covariance matrix:

$$\mathbf{H} = \mathbf{C}$$

Thus, if the covariance matrix $\mathbf{C}$ is ill-conditioned (due to high feature correlation), the Hessian matrix $\mathbf{H}$ will also be ill-conditioned.

A.2 Instability in ill-conditioned problems

In second-order optimization the weights update rule can be formulated as

$$\mathbf{W}^{k+1} = \mathbf{W}^k - \eta \mathbf{H}^{-1} \nabla_{\mathbf{W}} \text{loss}(\mathbf{W}) \quad (\text{A.1})$$

therefore, first-order optimization methods assume $\mathbf{H} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. If, on one hand, this means that the problem is ideally conditioned for gradient descent-like methods, then, on the other, a poorly conditioned Hessian is problematic. By considering the Singular Value Decomposition (SVD) decomposition

$$\mathbf{H}^{-1} = \mathbf{U} \mathbf{\Sigma}^{-1} \mathbf{V}^T, \quad (\text{A.2})$$

$\mathbf{\Sigma}$ is a diagonal matrix with singular values on the diagonal. If the Hessian is ill-conditioned, the inverse can be numerically unstable since the smallest singular values blow up on inversion. Thus, the more ill-conditioned the Hessian is, the more numerically unstable its inverse. Any noise in computing the Hessian such as that introduced by using stochastic versions of gradient descent updates or using mini-batches amplifies tremendously when the Hessian is inverted. Therefore, by decorrelating the features, the covariance matrix $\mathbf{C}$ (and hence the Hessian $\mathbf{H}$) becomes more diagonal, reducing the off-diagonal terms. This

leads to a smaller condition number, as the singular values become more evenly distributed.

In an ideal case of fully decorrelated features, $\mathbf{C}$ becomes a diagonal matrix:

$$\mathbf{C} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2)$$

Here, the condition number simplifies to:

$$\kappa(\mathbf{H}) = \frac{\max(\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2)}{\min(\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2)}$$

This diagonal form ensures a much better-conditioned Hessian, leading to more stable and efficient optimization steps.

Appendix B

Probabilistic Linear Discriminant Analysis

In PLDA, the formulation of the likelihood function is central to understanding how the model is fitted to data. This section extends the discussion on the complete data likelihood and the process of maximizing this likelihood, incorporating more detailed mathematical formulations and an illustrative example.

B.1 Mathematical Formulation of Complete Data Likelihood

The PLDA model posits that each observed data point $\mathbf{x}$ is generated from a continuous latent variable $\mathbf{y}$, which represents the class-specific characteristics, and

$$\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \Phi_b)$$

is drawn from a Gaussian distribution with mean $\mathbf{m}$ and covariance Φ_b . The the class-conditional distributions are written as

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{y}, \Phi_w)$$

where Φ_w is a common covariance matrix. The requirement is for Φ_w to be positive definite and Φ_b to be positive semi-definite. A known result from linear algebra enables simultaneous diagonalization of Φ_w and Φ_b : $\mathbf{V}^T \Phi_b \mathbf{V} = \Psi$ and $\mathbf{V}^T \Phi_w \mathbf{V} = \mathbf{I}$, where Ψ is a diagonal matrix and $\mathbf{V}$ is a non-singular matrix determined by solving a generalized eigenproblem. By defining $\mathbf{A} = \mathbf{V}^{-T}$, it follows that $\Phi_w = \mathbf{A} \mathbf{A}^T$ and $\Phi_b = \mathbf{A} \Psi \mathbf{A}^T$. The model is then expressed as:

$$\mathbf{x} = \mathbf{m} + \mathbf{A} \mathbf{u}$$

where

$$\mathbf{u} \sim \mathcal{N}(\mathbf{v}, \mathbf{I}) \quad \text{and} \quad \mathbf{v} \sim \mathcal{N}(0, \Psi).$$

Here, $\mathbf{v}$ represents the class, and $\mathbf{u}$ denotes an instance of that class in the projected space. Analogously, $\mathbf{y} = \mathbf{m} + \mathbf{A} \mathbf{v}$ and $\mathbf{x} = \mathbf{m} + \mathbf{A} \mathbf{u}$ in the data space.

B.1.1 Learn model parameters

The generative model for each data point is defined as the joint probability of $\mathbf{x}$ and $\mathbf{y}$

$$P(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = \mathcal{N}(\mathbf{y}, \Phi_w) \mathcal{N}(\mathbf{m}, \Phi_b)$$

Integrating out $\mathbf{y}$, we obtain the marginal likelihood of $\mathbf{x}$:

$$P(\mathbf{x}) = \int \mathcal{N}(\mathbf{y}, \Phi_w) \mathcal{N}(\mathbf{m}, \Phi_b) d\mathbf{y} = \mathcal{N}(\mathbf{m}, \Phi_w + \Phi_b)$$

Given a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$, the goal is to find parameters $\mathbf{m}, \Phi_b, \Phi_w$ that maximize the likelihood of $\mathbf{X}$. The EM algorithm [80] is particularly suited for this task.

B.1.2 Example

Suppose $\mathbf{X}$ contains one example from each of M classes. At test time, a probe example $\mathbf{x}_p$ is given. It is known that the probe belongs to one of the M classes. To classify $\mathbf{x}_p$ across the M classes, likelihood maximization can be performed in the latent space. The

transformation $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{m})$ is applied to all $(\mathbf{x}_1, \dots, \mathbf{x}_M)$. Now, consider an example $\mathbf{u}_g$ from $(\mathbf{x}_1, \dots, \mathbf{x}_M)$. The probability $P(\mathbf{u}_p|\mathbf{u}_g)$, which represents the likelihood of the probe example $\mathbf{x}_p$ belonging to the same class $g \in [1, 2, \dots, M]$, is computed by performing inference on the class variable:

$$P(\mathbf{v}|\mathbf{u}) = \mathcal{N}\left(\frac{\Psi}{\Psi + \mathbf{I}}\mathbf{u}, \frac{\Psi}{\Psi + \mathbf{I}}\right)$$

Since $\mathbf{u}_p$ and $\mathbf{u}_g$ are conditionally independent given $\mathbf{v}$, the probability $P(\mathbf{u}_p|\mathbf{u}_g)$ is:

$$P(\mathbf{u}_p|\mathbf{u}_g) = \mathcal{N}\left(\frac{\Psi}{\Psi + \mathbf{I}}\mathbf{u}_g, \mathbf{I} + \frac{\Psi}{\Psi + \mathbf{I}}\right)$$

To classify a probe example, $P(\mathbf{u}_p|\mathbf{u}_g)$ is computed for $g = 1, \dots, M$, and the class with the maximum probability is selected.