LEARNING SPEAKER-SPECIFIC CHARACTERISTICS WITH DEEP NEURAL ARCHITECTURE

A thesis submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2012

Ahmad Salman

School of Computer Science
# Table of Contents

List of Figures .............................................................. 6  
List of Tables ............................................................... 8  
Abstract .................................................................... 11  
Declaration ................................................................. 12  
Copyright Statement ................................................... 13  
Acknowledgements ....................................................... 15  
Abbreviations & Acronyms ............................................. 16  

## Chapter 1  Introduction  .................................................. 19  
  1.1 Motivation ............................................................. 19  
  1.2 Aims and Objectives .............................................. 21  
  1.3 Contributions ....................................................... 23  
    1.3.1 A Dedicated Architecture for Speaker Information Extraction 23  
    1.3.2 Robust Speaker Verification and Segmentation 23  
  1.4 Thesis Structure .................................................... 24  

## Chapter 2  Background  ................................................... 27  
  2.1 Speech Information Processing ............................... 27  
    2.1.1 Speech Production ........................................... 28  
    2.1.2 Speaker Recognition ........................................ 29  
  2.2 State-of-the-art in Speaker Recognition .................... 34  
  2.3 Exploring Generic Speaker Information ..................... 40  
    2.3.1 Explicit Features for Speaker-Specific Information .... 40  
    2.3.2 Implicit Features for Speaker-Specific Information ... 42  
  2.4 Deep Learning ...................................................... 46  
    2.4.1 Inspiration ..................................................... 46
List of Figures

Figure 1.1: Thesis structure 25

Figure 2.1: Human vocal apparatus 28
Figure 2.2: A general overview of SR system 29
Figure 2.3: Overview of deep learning 47
Figure 2.4: Comparison between shallow and deep architecture 49
Figure 2.5: Greedy layer-by-layer training 51
Figure 2.6: Training hierarchy of Deep belief network 54
Figure 2.7: Deep neural network with autoassociators 56
Figure 2.8: Convolutional neural networks 59
Figure 2.9: Deep autoencoder 62
Figure 2.10: Deep Siamese network 63

Figure 3.1: Deep neural architecture (DNA) 69
Figure 3.2: Pre-training with denoising autoencoders 75
Figure 3.3: 2-D projection for DNA representations and MFCCs 82

Figure 4.1: SC with DNAs for various depths 97
Figure 4.2: SC with DNAs for various widths 98
Figure 4.3: SC with DNAs for various code layer dimensions 100
Figure 4.4: SC with DNAs for various regularisations 102
Figure 4.5: SC with MFCCs, DAE, DS, DNA, DNA-R and PitchCon 105
Figure 5.1: Typical SV system  

Figure 5.2: Two-layered convolutional DBN  

Figure 5.3: Intra-corpus SV on TIMIT, NTIMIT, WKING and NKING  

Figure 5.4: SV performance on three DNAs trained on TIMIT, TIMIT + corruptTIMIT, TIMIT + corruptTIMIT + NTIMIT and TIMIT + corruptTIMIT + NTIMIT + VAM-audio  

Figure 5.5: SV performance for DNAs on MASC dataset  

Figure 5.6: Inter-corpus SV on WKING, NKING, CHN and RUS  

Figure 5.7: Inter-corpus SV on MASC dataset  

Figure 5.8: Inter-corpus SV on various test utterance durations  

Figure 6.1: Typical distance-based SS process  

Figure 6.2: Peak detection in distance-based SS  

Figure 6.3: Metric curves for SS  

Figure 6.4: ROC curves for SS
List of Tables

Table 2.1: Performance comparison of various SR methods in [109] 36
Table 2.2: Performance comparison of various SR methods on NIST SRE’10 for 9 core conditions 38
Table 2.3: SS performance of various methods on RT03 dataset. 39

Table 4.1: Information on various speech corpora 90
Table 4.2: Information on various speech corpora for sample size in SC evaluation 95
Table 4.3: Statistical significance (% confidence interval and z-scores) on SC results for DNAs trained with $K = 3, 4, 5$ 97
Table 4.4: Statistical significance (% confidence interval and z-scores) on SC results for DNAs trained with $|h_k| = 50, 100, 200$ 98
Table 4.5: Statistical significance (% confidence intervals and z-scores) on SC results for DNAs trained with $|h_k| = 100, 200, 300$ 100
Table 4.6: Statistical significance (% confidence intervals and z-scores) on SC results for DNAs trained with $\alpha = 0.01, 0.2, 0.3$ 102
Table 4.7: Statistical significance (% confidence intervals and z-scores) on SC results for various features 106

Table 5.1: Enrolment and test utterance lengths 120
Table 5.2: Parameter values for DNA, GMM-UBM and CDBN used
Table 5.3: Information on various speech corpora for sample size in SV evaluation

Table 5.4: Statistical significance (% confidence interval and z-scores) on intra-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN

Table 5.5: Statistical significance (% confidence interval and z-scores) on SV results for DNAs incrementally trained on various corpora. Evaluation is done on WKING, NKING, CHN and RUS corpora

Table 5.6: Statistical significance (% confidence interval and z-scores) on SV results for DNAs incrementally trained on various corpora. Evaluation is done on MASC corpus with five emotional states

Table 5.7: Statistical significance (% confidence interval and z-scores) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN. Evaluation is done on WKING, NKING, CHN and RUS corpora

Table 5.8: Statistical significance (% confidence intervals and z-scores) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN. Evaluation is done on MASC Corpus

Table 5.9: Statistical significance (% confidence interval and z-scores) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN evaluated on WKING and NKING for 3s and 1s test utterance length
Table 6.1: SS Results in terms of %EER and F1 measure.
Abstract

Robust Speaker Recognition (SR) has been a focus of attention for researchers since long. The advancement in speech-aided technologies especially biometrics highlights the necessity of foolproof SR systems. However, the performance of a SR system critically depends on the quality of speech features used to represent the speaker-specific information. This research aims at extracting the speaker-specific information from Mel-frequency Cepstral Coefficients (MFCCs) using deep learning.

Speech is a mixture of various information components that include linguistic, speaker-specific and speaker’s emotional state information. Feature extraction for each information component is inevitable in different speech-related tasks for robust performance. However, almost all forms of speech representation carry all the information as a whole, which is responsible for the compromised performances by SR systems. Motivated by the complex problem solving ability of deep architectures by learning high-level task-specific information in the data, we propose a novel Deep Neural Architecture (DNA) to extract speaker-specific information (SI) from MFCCs, a popular frequency domain speech signal representation. A two-stage learning strategy is adopted, which is based on unsupervised training for network initialisation followed by regularised contrastive learning.

To train our network in the 2nd stage, we devise a contrastive loss function to discriminate the speakers on the basis of their intrinsic statistical patterns, distributed in the representations yielded by our deep network. This is achieved in the contrastive pair-wise comparison of these representations for similar or dissimilar speakers. To improve the generalisation and reduce the interference of environmental effects with the speaker-specific representation, we regulate the contrastive loss with the data reconstruction loss in a multi-objective optimisation.

A detailed study has been done to analyse the parametric space in training the proposed deep architecture for optimum performance. Finally we compare the performance of our learned speaker-specific representations with several state-of-the-art techniques in speaker verification and speaker segmentation tasks. It is evident that the representations acquired through learned DNA are invariant and comparatively less sensitive to the text, language and environmental variability.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.
Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis,
the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

I am thankful to God Almighty for giving me the strength to overcome all the hurdles in completing my studies. I am extremely thankful to my parents who have invested their energies in my academic carrier. I am also grateful to Nabahat who is always there to support me in all the ups and downs.

I acknowledge the contribution and effort of my supervisor (Ke Chen) in our research. He has always been a helping hand and untiring source of guidance and motivation throughout my PhD study era.

I have made number of great friends during my stay in Manchester. Our everlasting bond will always be the source of sweet nostalgia. I thank all my colleagues and the university Staff who helped towards getting my research done.

Last but not least, I want to thanks NUST (National University of Sciences and Technology), Pakistan. They provided me with the finances throughout the studies.
# Abbreviations & Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian Information Criteria</td>
</tr>
<tr>
<td>CASA</td>
<td>Computational Auditory Scene Analysis</td>
</tr>
<tr>
<td>CD-1</td>
<td>Contrastive Divergence-1</td>
</tr>
<tr>
<td>CDBN</td>
<td>Convolutional Deep Belief Network</td>
</tr>
<tr>
<td>CMS</td>
<td>Cepstral Mean Subtraction</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CRBM</td>
<td>Convolutional Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>DA</td>
<td>Deep Architecture</td>
</tr>
<tr>
<td>DACBIC</td>
<td>Divide and Conquer Bayesian Information Criteria</td>
</tr>
<tr>
<td>DAE</td>
<td>Deep Autoencoder</td>
</tr>
<tr>
<td>DAN</td>
<td>Deep Autoassociator Network</td>
</tr>
<tr>
<td>DBN</td>
<td>Deep Belief Network</td>
</tr>
<tr>
<td>DET</td>
<td>Detection Error Trade-off</td>
</tr>
<tr>
<td>DISTBIC</td>
<td>Distance-based Bayesian Information Criteria</td>
</tr>
<tr>
<td>DNA</td>
<td>Deep Neural Architecture</td>
</tr>
<tr>
<td>DSN</td>
<td>Deep Siamese Network</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>EBM</td>
<td>Energy-based Model</td>
</tr>
<tr>
<td>EER</td>
<td>Equal Error Rate</td>
</tr>
<tr>
<td>FA</td>
<td>Factor analysis</td>
</tr>
<tr>
<td>FAR</td>
<td>False Alarm Rate</td>
</tr>
<tr>
<td>FDA</td>
<td>Fisher’s Discriminant Analysis</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>GSI</td>
<td>Generic Speaker Information</td>
</tr>
<tr>
<td>GSL</td>
<td>Gaussian Mixture Model Supervector Linear Kernel</td>
</tr>
</tbody>
</table>
HMM  Hidden Markov Model
ICA  Independent Component Analysis
JFA  Joint Factor Analysis
JSF  Joint Factor Analysis Speaker-Factor
JSV  Joint Factor Analysis SuperVector
KL  Kullback-Leibler
LDA  Linear Discriminant Analysis
LDC  Linguistic Data Consortium
LI  Linguistic Information
LLR  Log-Likelihood Ratio
LPC  Linear Prediction Coding
LPCC  Linear Prediction Cepstral Coefficient
MDR  Miss Detection Rate
MFCC  Mel-frequency Cepstral Coefficient
MKM  Multi-layer Kernel Machine
ML  Machine Learning
MLP  Multi Layer Perceptron
NAP  Nuisance Attribute Projection
NIST  National Institute of Standards and Technology
PCA  Principle Component Analysis
RBF  Radial Basis Function
RBM  Restricted Boltzmann Machine
ROC  Receiver Operating Characteristics
RT  Rich Transcription
SC  Speaker Comparison
SFA  Symmetric Factor Analysis
SGD  Stochastic Gradient Descent
SI  Speaker-specific Information
SNR  Signal to Noise Ratio
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>Speaker Recognition</td>
</tr>
<tr>
<td>SRE</td>
<td>Speaker Recognition Evaluation</td>
</tr>
<tr>
<td>SS</td>
<td>Speaker Segmentation</td>
</tr>
<tr>
<td>STFT</td>
<td>Short-term Fourier Transform</td>
</tr>
<tr>
<td>SV</td>
<td>Speaker Verification</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>t-SNE</td>
<td>t-distributed Stochastic Neighbourhood Embedding</td>
</tr>
<tr>
<td>UBM</td>
<td>Universal Background Model</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantisation</td>
</tr>
</tbody>
</table>
Chapter 1  Introduction

This chapter gives a brief introduction to the research work presented in this thesis.

1.1  Motivation

Speech is one of the most important ways of human communication. Like fingerprints, it carries the identity of the speaker as *voice print*. Speech perception marks a challenging area for the signal processing community, as it is desired to extract the information of interest from the speech signal that is a mixture of various information components. This information varies from linguistic information (LI) to speaker-specific information (SI). Linguistic or lexical information carries the message from the speaker and is the focus of attention in speech recognition systems. On the other hand, SI conveys the information about the speaker and is dependent on the structure of the vocal apparatus in speech production. Such kind of speaker-related information plays a key role in various speaker recognition (SR) tasks.

Although effortless for humans, modern speech information processing systems find it difficult to sift out specific information for a particular task. Numerous efforts have been made to extract the task-specific features from speech using either time-domain signal or its spectral representation [1], [2], [3]. Nevertheless, nearly all forms of speech representation carry all the information as a whole and the interference of these information...
components makes the speech and speaker recognition systems compromise on their performance [1], [3], [4], [5], [6], [7]. For the last two decades, the problem of exploring the task-specific information e.g., SI for SR has been a crux of attention by researchers. In the pursuit of feature extraction that can universally characterise the speaker, various attempts have been made to yield better and acceptable performance in SR systems. These systems in turn depend on the efficiency and effectiveness of machine learning (ML) methodologies that are related to the task of modelling each speaker’s unique characteristics.

So far, a few efforts have been made in terms of data component analysis, i.e., Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Isomap, to separate the speaker-dependent components from a general speech representation e.g., spectrogram [1], [2], [95]. Assuming the invariance for SI among different utterances of a speaker, these techniques undergo multivariate analysis of speech data to span the correlated input variables to uncorrelated output variables or divide the data into statistically independent components. These techniques are based on the optimisation of an objective defined on underlying data distribution as whole rather than specific information hidden in the data. Therefore due to the non-stationary nature of the SI and knowing the fact that it is minor information component along with dominant LI, unsupervised learning techniques like PCA, Isomap or ICA are less successful in delivering promising outcome in SR tasks. Similarly supervised learning techniques like LDA and neural networks have been used in extracting speaker dependent features but without significant success in various SR tasks [94], [96], [97], [98]. In general, modern state of the art SR systems either rely on generative modelling of speaker’s data distribution or discriminative modelling by mapping the input speech data to some target space (classification, regressing etc.) [8], [9], [10], [11], [12].
any case, by far the problem of segregating speech data into its information components remains unresolved [4].

Not until recently, deep learning has emerged as one of the strong candidates for learning highly complex functions capable of representing intrinsic invariance from the information manifolds of the data [10], [11]. Inspired by the learning hierarchy of the brain, deep architecture (DA) is a parametric model with multiple levels of nonlinear operations e.g. a neural network with many hidden layers [11], [13], [54]. Through flexible multi-objective optimisation, DA can be designed to learn high-level abstract representation with repeated transformation of the input data at different levels of inference. Through well-defined propositional training, these abstract representations of the input data can efficiently encode the unique characteristics of each information component with better separation and discrimination.

Speech information processing is yet another AI task which involves complex problem solving to learn to discriminate among various information components of speech. SI is one such component that is unique for every speaker. Inspired by the success of deep learning in various vision tasks [13], [14], [15], we are interested in extracting speaker-specific characteristics from speech using a deep neural architecture which may eventually lead to better performance in various SR tasks.

### 1.2 Aims and Objectives

The performance of SR systems is greatly influenced by the variability present in utterances of a speaker. As a result, SR systems may not exhibit good generalisation capability towards classification of speakers [88], [89], [90]. Modern sophisticated SR systems can give promising results for a particular speech dataset with certain conditions but due to the lack of generic speaker-specific features, the performance of these approaches
produces unstable results especially when there is a mismatch in training and test data due to noise, channel, session and emotional variability in speech.

We aim to propose such a deep learning strategy, which will provide a way to implicitly learn the speaker-dependent characteristics from a general speech spectral representation. We will explore the parametric space in designing a DA to separate speaker related information from the non-speaker related information.

The strength of the speaker-specific features as a result of proposed DA will be judged in tasks like speaker comparison, verification and segmentation using several speech corpora with different environmental and emotional variability. The detailed comparison of the proposed architecture will be carried out with the state-of-the-art SR techniques.

The flexibility to learn a high-level abstraction of the data through discriminative training in DA can be a way to objectify speaker-specific features. Therefore, we are interested in investigating the performance of acquired features/representations as a result of our proposed DA for following criteria:-

- The features should be generic in terms of SI. In other words they should be effective for all speakers even if the speech data for those speakers is not involved in training i.e., open set SR.
- The features should be insensitive to intra-speaker variability, e.g. session, age and emotional variability and also environmental factor like channel effects and noise.
- Text and language should not influence the features.
- Duration of speaker’s speech should have minimal effects on features.
- The features should be used to create a speaker model with relative ease.
• The features should provide good performance in various SR tasks.

1.3 Contributions

1.3.1 A Dedicated Architecture for Speaker Information Extraction

We propose a deep neural architecture (DNA) to extract the SI from Mel-frequency Cepstral Coefficient (MFCCs). Deep learning has been used to produce promising results in different vision tasks but, to our knowledge, this is the first attempt for speech component analysis in discriminative style neural networks. The proposed approach utilise two-stage unsupervised and regularised contrastive learning to exploit the statistical pattern in speech that is sensitive to speaker variation.

A multi-objective loss function is devised to train the proposed DNA. The loss function is driven by the pair-wise contrastive comparison of the 1st and 2nd order statistics between the representations corresponding to the similar or different speakers. The reconstruction loss is also incorporated to improve generalisation and suppress the non-speaker related information. The final output speech representation produced by DNA is overcomplete and distributed in terms of SI and non-SI components.

1.3.2 Robust Speaker Verification and Segmentation

Although the purpose of this research is not to implement a sophisticated SR system, we compare the performance of the representations yielded by our DNA with state-of-the-art techniques in speaker verification (SV) and speaker comparison (SC). Using a simple Gaussian classifier, our DNA representations outperform the sophisticated SR systems on various wide-band, narrow-band and emotional speech corpora. The SI-rich representations are robust against variation in speech and show better generalisation. To detect a speaker change point in an audio conversation is called speaker segmentation (SS). This is yet another challenging SR task
that involves decision making for discriminating speakers on the basis of usually very short utterances. We investigate the potential of generic speaker-specific representations by our DNA in these tasks and show that the performance is better than the state-of-the-art.

1.4 Thesis Structure

The thesis has seven chapters. A brief introduction to our research has been presented already in Chapter 1. In Chapter 2, the general background about SR, speaker modelling and speech feature extraction is discussed, which highlights the need for speech information processing dedicated to capture generic SI. The chapter also presents an overview about deep learning and various DAs that have been proposed in the literature. These architectures and their successful application in various classification tasks motivate our research. Chapter 3 gives the structural details of new deep neural architecture (DNA) that is specially designed to capture speaker similarity/dissimilarity in a two-stage unsupervised and supervised learning. For training DNA, a multi-objective contrastive loss is proposed. Chapter 4 is the empirical evaluation about optimum parameter settings of DNA. Furthermore the comparison between representations acquired through various DAs (including DNA) and the base line MFCCs is judged in terms of their ability to represent SI. For the justification of the proposed architecture and the comparative study, speaker comparison (SC) experiments are performed. Chapter 5 is the application of the DNA representation in an important SR tasks i.e., speaker verification (SV). The chapter begins with the introduction of popular speaker modelling techniques for SV. The experimental settings for SV for various speech corpora are presented followed by the detailed results and comparisons. Chapter 6 covers another SR application i.e., speaker segmentation (SS). Starting with the overview of the current state-of-the-art approaches in SS,
various experiments are reported for the comparison of DNA representation with the other popular methods in SS on different speech corpora. Chapter 7 summarises our research with suggestions about the prospective improvements.

Figure 1.1 shows the structural flow of the thesis. Chapter 3 and Chapter 4 are related as they provide detailed design strategy for DNA and its optimum parameter settings through empirical evaluation respectively, hence boxed together. Similarly Chapter 5 and Chapter 6 cover the
comparative study of DNA representations with various techniques in two SR tasks i.e., SV and SS. Therefore, these two chapters provide the application of DNA representations. The symbol (o) at the beginning of some arrows implies that they are linked with the preceding arrow.
Chapter 2  Background

This chapter reviews the background knowledge on speech information processing with emphasis on exploring speaker-specific information (SI). This ranges from the past attempts for feature extraction to characterise the speakers by exploiting human speech production mechanism to various speaker modelling techniques used in SR. Deep architectures have the ability to produce high-level abstract representation of targeted information component in the data. With this motivation, deep learning can be a prospective application for extracting speaker-dependent information from speech. Therefore, a detailed literature review about several DAs with their training strategies is also presented. This chapter also discusses the limitations of current SR systems and various speech representations that generate a need to extract generic speaker-specific representation from speech.

2.1  Speech Information Processing

The goal of this research is to explore the possibility of representing speech in such a way that the representation should help the given SR system to have more discriminative power for classifying the speakers. The acquired speech representation is supposed to encode the SI, which is one of the several information components conveyed by speech signal. Therefore it is necessary to first understand the structure of speech, process of speech production and the mechanism of various SR systems.
2.1.1 Speech Production

Speech has a special production mechanism in human vocal apparatus, which discriminates it from the rest of the audio signals in general. The human vocal apparatus, as shown in Figure 2.1 [99], is divided into three major parts, i.e. lungs, larynx and vocal tract. These further contain many different organs and each organ has a specific function.

The process of speech production starts from lungs. Lungs generate airflow, which after passing through the windpipe, resembles with a band-limited white noise. At the vocal folds in larynx, the air pressure is built up, which results in a glottal opening and vibration of vocal folds. The vocal folds vibrate at certain frequency called fundamental frequency, $f_0$ and generate the series of air puffs or pulses separated by the pitch period (inverse of fundamental frequency). These pulses of air after passing through the nasal, oral and mouth cavities create a perceivable sound called speech. It is due to the specific structural properties of human vocal apparatus, speech carries information related to the person as his/her intrinsic characterisation.
2.1.2 Speaker Recognition

The goal of speaker recognition is to recognise the speaker from his/her voice. Figure 2.2 shows the general structure of SR system. A speaker recognition system is a biometric system that involves speech feature extraction (pre-processing), characterisation (speaker modelling) and finally recognition of the information in the utterance of the speaker, about his/her identity (decision making) [1], [14]. There is a variety of speech-aided applications that involve SR e.g., speaker identification (who is speaking?), speaker verification (is he/she the one who he/she claims to be?), speaker segmentation (who is speaking when?).

In this section, a brief overview is presented about the steps and methodologies involved in SR.

Feature Extraction

The aim of the feature extraction step in SR is to collect and process the necessary information about the speaker, while discarding the irrelevance and redundancy present in the signal. Using the processed information, the
SR system exploits the discrimination between the speakers. In a broad description, features extracted from the speech signal can be divided into two categories, high-level features and low-level features. High-level features include clarity, roughness, animation and also the prosodic features [15], [16]. Low-level features usually include spectral representation of speech signal that convey the information about the vocal tract in general. Other low-level features include explicit information extraction about individual vocal apparatus organs [14], [18]. We will only review low-level features that can be broadly categorised as spectral and non-spectral features.

Spectral features include the frequency domain analysis of speech signal and encode much of the information conveyed by the speech in general. This property along with the computational ease, make them suitable for both speech and speaker recognition. Spectral envelope, Mel-frequency Cepstral Coefficient (MFCCs), Linear Prediction Cepstral Coefficients (LPCCs), Sub-Cepstra and spectrograms are the ways of representing spectral features.

The process of extracting these features, usually involves short-time Fourier transform (STFT) on small frames of speech signal (20ms to 30ms long). The STFT magnitude envelope characterises the vocal tract resonance and anti-resonance frequencies, hence giving the information about vocal tract dimension, which is speaker-specific. Similarly, the whole vocal tract can be modelled by an all-pole filter in linear predictive coding, yielding the LPCCs that can be used as input features for SR systems [20]. On the other hand, MFCCs are amongst the most popular spectral features used in speech and speaker recognition [4], [19] [21]. In MFCCs, the spectral energies are arranged on the Mel-frequency scale, using a series of band-pass filters. Mel-scale mimics the auditory perceptual frequency scale of human ear. Many SR applications utilise raw spectrograms as features.
However the information content in spectrogram is proportional to the resolution of the STFT used for each frame. High resolution results in high dimensional features that make them less appropriate to use in ML algorithms. Spectrograms are usually further processed with dimensionality reduction techniques like PCA.

Non-spectral features explicitly characterise the speech production sources. Fundamental frequency f0, intensity, vocal excitation signals, glottal flow derivatives and source onset timing are examples that fall in the category of non-spectral features [22], [14], [23]. There are well-established algorithms to extract these features but generally they are computationally very expensive which make them less popular as compared to the spectral features.

**Speaker Modelling**

After feature extraction and pre-processing, the next step in SR is speaker classification. Based on the feature vectors of speech, a model is created for each speaker. The main quality criteria for a speaker model are its ability to generalise to a data that is not involved in training those models, the ability to discriminate the speakers, the resistance against environmental effects and disguise and also computational efficiency [4]. Based on the method of evaluation, the speaker modelling can be viewed in terms of either text-dependent mode or text-independent mode. Text-independent SR implies that the enrolment and test utterances of the speaker may differ in lexical content without any constraint on the utterance duration. Popular methods used for text-independent SR are as follows: -

*Vector Quantization (VQ):* Speaker models are created by pooling all the training feature vectors of each speaker to form codebooks [29], [30]. So if there are S speakers, there will be S codebooks. For the test utterance
sequence \( X^{ts} = \{x_{1}^{ts}, x_{2}^{ts}, ..., x_{N}^{ts} \} \) and the \( s^{th} \) \((s = 1, ..., S)\) codebook, the average distortion is calculated by averaging the minimum distances acquired for each test feature vector \( x_{i}^{ts} \) when compared with all the training feature vectors \( X^{tr} = \{x_{1}^{tr}, x_{2}^{tr}, ..., x_{M}^{tr} \} \) in a codebook i.e.,

\[
\mathcal{D}^s = \frac{1}{N} \sum_{i=1}^{N} \min_{1 \leq j \leq M} d(x_{i}^{ts}, x_{j}^{tr}),
\]

(2.1)

where \( d(\cdot) \) is any distance metric e.g. Euclidean distance. In the recognition phase, the speaker whose codebook produces minimum average distortion is selected i.e.,

\[
S^* = \arg \min_{1 \leq s \leq S} \mathcal{D}^s.
\]

(2.2)

Another approach is to compare the test feature vector in (2.2) with the \( k \)-nearest neighbour training feature vectors instead of all feature vectors in the codebook.

**Gaussian Mixture Models (GMM):** The speaker models can be characterised by a mixture of Gaussians [26], [27], [28]. GMMs are similar to codebooks in VQ in a sense that the clusters are created for each speaker where the mean as well as the covariance of the training feature vectors is estimated. GMM therefore can capture more detailed statistical information from the speaker’s speech. Gaussian mixtures can estimate a probability density function, which is a weighted sum of \( Mg \) mono-Gaussians, with an arbitrary precision. For a \( \delta \)-dimensional test feature vector \( \{x_{i}^{ts}\}_{i=1}^{N} \in X^{ts} \), the mixture density is given as

\[
p(x_{i}^{ts} | S_{M}) = \sum_{j=1}^{Mg} w_{j} g_{j}(x_{i}^{ts}),
\]

(2.3)

where

\[
g_{j}(x_{i}^{ts}) = \frac{1}{(2\pi)^{\delta/2} |\Sigma_{j}|^{1/2}} \exp \left\{ -\frac{1}{2} (x_{i}^{ts} - \mu_{j})^{T} \Sigma_{j}^{-1} (x_{i}^{ts} - \mu_{j}) \right\},
\]

(2.4)

with \( \mu_{j} \) and \( \Sigma_{j} \) are component mean vector and component covariance matrix respectively that are estimated using the training feature vectors.
The mixture weights $w_j$ are constrained such that $\sum_{j=1}^{Mg} w_j = 1$. Collectively, a speaker model for a speaker can be represented as $S_M = \{w_j, \mu_j, \Sigma_j\}_{j=1}^{Mg}$.

Equation (2.3) is used as score in SR.

**Support Vector Machine (SVM):** A binary classifier that optimally separates the data points belonging to the two classes by constructing a linear decision boundary or hyperplane, usually in a transformed high dimensional feature space [100], [101]. For linearly separable data, the hyperplane is given as $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is perpendicular to the plane, $\mathbf{x}$ is a data vector and $b$ is constant offset. For linearly separable data $\{\mathbf{x}_j^{tr}\}_{j=1}^{M} \in X^{tr}$ with binary label $y_j \in \{-1,1\}$, the optimal hyperplane (the parameters $\mathbf{w}$ and $b$) is chosen by using maximum margin criteria i.e., the Euclidean distance between the data vectors on either side of the hyperplane should be maximum. For the trained SVM, the score on the test data vectors $\{\mathbf{x}_i^{ts}\}_{i=1}^{N} \in X^{ts}$ is given as,

$$
S_{svm} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w} \cdot \mathbf{x}_i^{ts} + b),
$$

(2.5)

which can be used in SR for decision making.

**Neural Networks:** There are various topologies to train the neural networks to create speaker models. One way is to train a separate neural network, like MLP for each registered speaker with supervised learning using labels as speaker identity, e.g. one of $n$-coding [32], [33]. The other way is to train a single neural network, e.g. RBF or MLP by taking the data from all the speakers as a training data [35].

In text-dependent recognition, the speaker models are created on the basis of fixed phrases or words spoken by registered speakers. For recognition, the test speaker is required to speak the same phrase for comparison with the registered speakers. Some examples for text-dependent SR are discussed below: -
Template Matching: A speaker model contains a template, which is a sequence of feature vectors from a fixed spoken text. Dynamic Time Warping (DTW) is used to align a text utterance with the template of a registered speaker and classification is done on the basis of some distance metric [82], [83].

Hidden Markov Models (HMM): Speaker model is created by encoding the statistical variation in the time-domain speech signal using HMMs [36]. Testing is performed by comparing the likelihood of the test speech against the registered speaker model.

In general, GMM, SVM, HMM or their combination and variants are the best performing speaker modelling techniques [26], [27], [36], [37].

2.2 State-of-the-art in Speaker Recognition

In this section, we present a quantitative overview on the performance of several state-of-the-art systems for different SR tasks.

For the last two decades, systems based on generative modelling of the speakers are generally credited as the most successful in applications like speaker verification, identification and detection [26], [27], [28], [40], [44], [82]. In addition to their ability of modelling speaker-specific statistical variation from speech utterances, their variants involving discriminative classifiers further boosted the performances and are the best to date. Similarly in speaker segmentation/diarisation, which is yet another challenging SR task, systems based on Bayesian Information Criteria (BIC) [84] are considered as the most favourable for good performance [85], [86], [87]. However, the performance of any SR system is critically dependent not only on the specific application but most importantly the experimental protocol, speech dataset and its environment. Therefore practically it is impossible to attribute the term ‘the best’ for a particular SR system. Due to this reason, it is hard to find a solid work in literature that compares
various popular methods on a particular dataset on same experimental settings.

*National Institute of Standard and Technology* (NIST) organises a competition once every two years for *speaker detection* in Speaker Recognition Evaluation (SRE) and for *speaker diarisation* in Rich Transcription (RT). In the NIST competition, various algorithms are judged for their performance on given datasets and experimental settings. However it is not mandatory for the participants to make their methodology public. On the available literature, to our knowledge, we can compare several systems in the latest SRE 2010 competition. It is worth mentioning that NIST does not claim any system by any participant to be the best or state-of-the-art in general as the performance of these systems may vary drastically for different experimental protocols or speech datasets with different noise, channel, recording equipment or session variabilities. The NIST SRE systems mentioned in this section are published works and we call them state-of-the-art in relative sense. However, such systems or their variants appear repeatedly in literature claiming significant gains in SR.

Speaker detection is a process of making a decision whether the target speaker is present in an audio stream involving various speakers. This is similar to speaker verification, however instead of comparing a speech utterance of single speaker, we compare a whole stream with the reference speaker model [108].

In [109], SVM and several variants of GMM are studied. The baseline system is universal background model trained via Gaussian mixture model (GMM-UBM) [82]. It is compared with the following approaches:

- GMM Supervector Linear kernel (GSL). A combination of GMM-UBM and SVM, where the statistics of speaker model adapted from UBM are subjected to further classification using SVM. See [109], [110] for details.
<table>
<thead>
<tr>
<th>Method</th>
<th>%EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM-baseline</td>
<td>8.47</td>
</tr>
<tr>
<td>GSL-NAP</td>
<td>4.33</td>
</tr>
<tr>
<td>SFA</td>
<td>4.78</td>
</tr>
<tr>
<td>FA-GSL</td>
<td>3.97</td>
</tr>
<tr>
<td>Fusion</td>
<td>3.95</td>
</tr>
</tbody>
</table>

Table 2.1: Performance comparison of various SR methods in [109].

- Symmetric Factor Analysis (SFA), where statistics learned via GMM-UBM are factorised in speaker-dependent and environment-dependent components to compensate for session and channel variability [109]. Classification is done using log-likelihood ratio (LLR).
- GSL used with Nuisance Attribute Projection (GSL-NAP). NAP is a compensation procedure for environmental variability used in SVM framework [111], [112].
- FA-GSL, a combination of factor analysis (FA) [113] and GSL.
- A fusion system of SFA and GLS-NAP [109].

All the above systems are trained on NIST SRE 2004 and 2005 datasets [73], while evaluation is done on NIST SRE 2006 dataset [73] using the same experimental settings. Table 2.1 shows the performance of various speaker detection systems [109] in terms of percentage equal error rate (%EER). The best score is shown in bold.

Now we discuss various state-of-the-art systems that appeared in NIST SRE 2010 competition. Some of the participants with the best performing systems are HKCUPU (Chinese University of Hong Kong and the Hong Kong Polytechnic University) [114], I3A (Aragon Institute for Engineering Research, University of Zaragoza, Spain) [115], TUBITAK UEKAE-SABANCI University (Turkey) [116], LOQENDO-POLITECNICO (Italy) [119]. Each participant submitted one or more than one system to SRE 2010. The systems submitted by HKCUPU are as follows:
• JFA (Joint Factor Analysis): A system based on joint factor analysis [117] that uses GMM-UBM supervector approach [109], [114]. MFCCs are used as input features. The classification is done using LLR.

• JSV (JFA SuperVector): A combination of JFA and GSL. MFCCs are used as input features.

• JSF (JFA Speaker-Factor): Similar to JSV but with Perceptual Linear Prediction (PLP) coefficients as input [118].

• GSV (GMM-SVM): A GSL-NAP system (as in Table 2.1) with MFCCs as input [114].

• FSH (JFA-FiSHervoice): A system that uses MFCC-based GMM-UBM supervectors with Fisher’s Discriminant Analysis (FDA) as a classifier using cosine distance [114].

The systems submitted by I3A are as follows:

• I3A-1: A fusion of JSV and JFA. MFCCs are used as input features [115].

• I3A-2: A simple JSV system trained on MFCCs [115].

• I3A-3: A simple JFA system trained on MFCCs [115].

TUBITAK UEKAE-SABANCI University submitted one system based on GSL-NAP trained on MFCCs [116].

LOQUNDO-POLITECNICO submitted one system based on GMM-UBM supervectors with JFA and Total Variability [119] approaches for session and environment compensation. Again MFCCs are used as input features.

It can be seen that all the above-mentioned systems are based on generative GMM-UBM modelling with some variants including SVM and session/environmental variability compensation techniques like JFA or NAP. Apart from this commonality, all the systems differ in either input features with the most using MFCCs of various dimensions or the number of Gaussian components in GMM-UBM [114], [115], [116], [119]. The
<table>
<thead>
<tr>
<th>Method/Cond.</th>
<th>cc1</th>
<th>cc2</th>
<th>cc3</th>
<th>cc4</th>
<th>cc5</th>
<th>cc6</th>
<th>cc7</th>
<th>cc8</th>
<th>cc9</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFA</td>
<td>3.88</td>
<td>8.04</td>
<td>4.53</td>
<td>5.79</td>
<td>4.52</td>
<td>7.17</td>
<td>7.52</td>
<td>2.01</td>
<td>3.45</td>
</tr>
<tr>
<td></td>
<td>(0.63)</td>
<td>(0.84)</td>
<td>(0.66)</td>
<td>(0.76)</td>
<td>(0.47)</td>
<td>(0.82)</td>
<td>(0.74)</td>
<td>(0.46)</td>
<td>(0.39)</td>
</tr>
<tr>
<td>JSV</td>
<td>4.55</td>
<td>9.11</td>
<td>7.59</td>
<td>7.14</td>
<td>8.31</td>
<td>8.79</td>
<td>2.68</td>
<td>2.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.73)</td>
<td>(0.81)</td>
<td>(0.79)</td>
<td>(0.68)</td>
<td>(0.84)</td>
<td>(0.90)</td>
<td>(0.55)</td>
<td>(0.46)</td>
<td></td>
</tr>
<tr>
<td>JSF</td>
<td>7.51</td>
<td>13.55</td>
<td>11.32</td>
<td>11.11</td>
<td>5.77</td>
<td>9.14</td>
<td>8.63</td>
<td>3.69</td>
<td>3.79</td>
</tr>
<tr>
<td></td>
<td>(0.82)</td>
<td>(0.89)</td>
<td>(0.83)</td>
<td>(0.83)</td>
<td>(0.60)</td>
<td>(0.83)</td>
<td>(0.77)</td>
<td>(0.44)</td>
<td>(0.41)</td>
</tr>
<tr>
<td>GSV</td>
<td>4.40</td>
<td>7.39</td>
<td>6.28</td>
<td>5.58</td>
<td>4.76</td>
<td>7.75</td>
<td>9.15</td>
<td>2.57</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>(0.67)</td>
<td>(0.77)</td>
<td>(0.70)</td>
<td>(0.68)</td>
<td>(0.57)</td>
<td>(0.79)</td>
<td>(0.75)</td>
<td>(0.47)</td>
<td>(0.39)</td>
</tr>
<tr>
<td>FSH</td>
<td>7.10</td>
<td>11.32</td>
<td>8.26</td>
<td>6.96</td>
<td>4.09</td>
<td>6.09</td>
<td>8.35</td>
<td>1.68</td>
<td>4.14</td>
</tr>
<tr>
<td></td>
<td>(0.73)</td>
<td>(0.86)</td>
<td>(0.90)</td>
<td>(0.85)</td>
<td>(0.54)</td>
<td>(0.80)</td>
<td>(0.85)</td>
<td>(0.28)</td>
<td>(0.49)</td>
</tr>
<tr>
<td>I3A-1</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td></td>
</tr>
<tr>
<td>I3A-2</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>(0.72)</td>
<td>(0.48)</td>
<td>(0.41)</td>
<td>(0.62)</td>
<td>(0.40)</td>
<td>(0.77)</td>
<td>(0.69)</td>
<td>(0.17)</td>
<td>(0.40)</td>
</tr>
<tr>
<td>I3A-3</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>(0.75)</td>
<td>(0.57)</td>
<td>(0.49)</td>
<td>(0.67)</td>
<td>(0.50)</td>
<td>(0.73)</td>
<td>(0.75)</td>
<td>(0.22)</td>
<td>(0.44)</td>
</tr>
<tr>
<td></td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
</tr>
<tr>
<td>LOQUNDO-POLITECNICO</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>2.40</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(0.28)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
<td>(---)</td>
</tr>
</tbody>
</table>

Table 2.2: Performance comparison of various SR methods on NIST SRE 2010 for 9 core conditions. Scores are given in %EER and (minDCF).

training dataset for all systems include NIST SRE 2004, NIST SRE 2005 and NIST SRE 2006 [73]. The systems are evaluated on the pre-defined core test conditions of NIST SRE 2010. The performance is measured either in %EER or minimum Detection Cost Function minDCF [120], whichever is given in a tabular form in authors’ documents. Table 2.2 shows the comparative results on nine core conditions (cc1-cc9) of NIST SRE 2010 [120] in terms of %EER and minDCF (in parentheses). These conditions are based on training and testing on same or different channels e.g., various microphones or telephone channels.

Remaining within the scope of this thesis, we also discuss the state-of-the-art in speaker segmentation (SS). SS is a process of dividing long audio streams (conversation of multiple speakers) into homogeneous segments such that each segment belongs to single speaker. In other words, to find out the speaker change points in the audio stream. In [86], various SS...
### Table 2.3: SS performance of various methods on NIST RT03 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>%EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinGrow</td>
<td>MDR: 17.79</td>
</tr>
<tr>
<td></td>
<td>FAR: 16.59</td>
</tr>
<tr>
<td>DISTBIC</td>
<td>MDR: 22.30</td>
</tr>
<tr>
<td></td>
<td>FAR: 21.08</td>
</tr>
<tr>
<td>SeqDACDec</td>
<td>MDR: 16.44</td>
</tr>
<tr>
<td></td>
<td>FAR: 15.95</td>
</tr>
<tr>
<td>FixSlidHAC</td>
<td>MDR: 23.19</td>
</tr>
<tr>
<td></td>
<td>FAR: 24.88</td>
</tr>
</tbody>
</table>

MDR is miss detection rate and FAR is false alarm rate.

...algorithms based on MFCCs as input features are evaluated on conversational speech from NIST RT03 dataset [73].

In SS, state-of-the-art systems are mostly based on Bayesian information criteria [84] to detect the speaker change points. Table 2.3 compares WinGrow, DISTBIC, SeqDACDEC and FixSlidHAC in terms of %EER with these names as appear in [86]. For details on all these algorithms see, [84], [85], [86] and Chapter 6 of this thesis.

It is again emphasised that the experimental settings for the systems presented in Table 2.1 and 2.3 have been set for optimal performance for given datasets, i.e., the Switchboard, MIXER and RT corpora [73], which are used to generate the NIST corpora. The performance and the conclusion may change significantly while evaluating the above-mentioned systems on different corpora/environment.

Most of the state-of-the-art systems in Table 2.1 are based on generative GMM speaker modelling. In this approach, the main drawback is that the GMM models the statistics of the speaker by taking all the information in the data as a whole without making any discrimination on task-specific information, i.e., SI for SR. Also the ability to learn speaker-independent UBM is critical as the speaker population needed to learn such a UBM should be large enough, covering various aspects of variability. Similarly environmental variability compensation using techniques like FA and NAP require large number of various examples for each speaker with speech recorded in different sessions and environments [109], which is less pragmatic. The lack of ability to extract SI and the limitation of training...
As mentioned earlier, Chapter 5 and Chapter 6 of this thesis provide a comparative study of our DNA representations with several popular systems in SV and SS. However, due to unavailability of NIST SRE and RT speech corpora, direct quantitative comparison of the state-of-the-art systems given in this section with ours cannot be established.

2.3 Exploring Generic Speaker Information

The performance of a SR system is heavily dependent on how well the input speech representation captures the SI. For feature extraction, several efforts have been made to model the speech production system and distil the acoustic features that are sensitive to the speaker variations. In contrast, on a higher level, numerous SR systems have been designed to generate speaker models that exploit the statistical information related to the speakers. In addition, various feature selection and data component analysis techniques have also been applied to sift out the SI. This section reviews various features that can be extracted implicitly or explicitly to explore SI and hence favourable in various SR tasks.

2.3.1 Explicit Features for Speaker-Specific Information

Speech conveys various information including LI, SI and speaker’s emotional state information (EI). SI depends on physiology of human speech production system. Therefore, features that carry SI also depend on factors involved in speech production. These features can be divided into two categories, source related features and prosodic features.

Source-Related Features

There are various factors that are related to the structure and organs of
vocal apparatus and they affect the SI. In other words, every human has a unique vocal apparatus that is responsible for a distinct natural sound for each speaker.

The structure of vocal folds in the larynx is speaker-specific. Therefore the fundamental frequency $f_0$ required to vibrate the vocal folds is also speaker-specific. Another feature is loudness or intensity that is the underlying characteristic of the vocal folds i.e., the magnitude of vocal fold vibration. Features like $f_0$ and loudness can be estimated using short-term signal analysis of speech, e.g. linear prediction coding [23], [102]. In addition, glottal flow derivative characterising the functioning of glottal opening, spectral tilt and source onset timings are also speaker dependent features [38].

Although $f_0$ and loudness are dependent on the larynx [14], the structure of the vocal tract indirectly influences their effect in audible speech. The dimensions of cavities in the vocal tract are highly speaker-dependent. Most commonly, vocal tract is modelled by an all-pole filter. The filter transfer function holds unique SI. Therefore the filter coefficients serve as speaker-related features. Linear prediction analysis of speech is a popular algorithm to model the vocal tract as a filter [23], [102]. The estimated filter coefficients serve as input features to many algorithms [41], [42].

**Prosodic Features**

Prosodic information depends on the long-term effect of the source-related features. For example, the variation in $f_0$, loudness and articulation rate are called prosodic features. Prosodic information can be speaker or language dependent. A speaker may have specific style and rate of speaking that is yet another SI. On the other hand, different languages may have different rules of prosody [45]. The emotional state of the speaker greatly influence the speech production mechanism, e.g. the speech of a speaker may sound
very different in anger or depression. Source-related and prosodic features can encompass SI but at the cost of computational burden in general. Extracting such features for each and every utterance of every speaker in a large population is a huge hindrance in their pragmatic use, e.g., in online SR. Also being highly sensitive to the environmental corruption such as noise, media variation or imitation makes them less favourable in a real world SR tasks. In contrast, the spectral features e.g., spectrograms, MFCCs or LPCCs are more common in modern SR systems with relatively high success. These features represent most of the information components of speech, which makes them generic for both speech and speaker recognition problems.

2.3.2 Implicit Features for Speaker-Specific Information

Apart from the feature extraction by modelling the sources of speech production, efforts have been made to implicitly extract the speaker dependent features from the general speech representation, e.g. MFCCs or spectrogram. Now, a brief overview on some methodologies is given.

Data Component Analysis

Statistical procedures like PCA or ICA are the examples of data component analysis techniques that factorise the data on the basis of different information content [46], [47]. PCA is an Eigen-value based multivariate analysis technique that uses the orthogonal transformation to split the data into uncorrelated variables (principal components). Various information components of speech are assumed to be uncorrelated with SI. The first few principal components try to cover the speaker variability as much as possible. Under this assumption, PCA and the variants of PCA have been used to extract the speaker-specific features from a raw spectral representation of speech [2], [46]. ICA is yet another kind of factor analysis
of data which separates the multivariate signal into additive components assuming that the components are statistically independent [47]. In speech a typical example of ICA is the *Cocktail Party Problem*, where a mixture of different speech signals is decomposed into original signals based on the speakers’ identity, hence the SI. This is the blind source separation where the sources are discriminated on the basis of SI.

PCA and ICA have been studied in Computational Auditory Scene Analysis (CASA). The aim of CASA is to mimic the blind source separation ability of human brain. CASA has been used in SR tasks with some success [46]. Features, in the transformed space, acquired through PCA or ICA are used in various SR tasks mostly in conjunction with other speaker modelling techniques like GMM [46],[48].

The data component analysis techniques, in general, fail to produce significance performance for robust SR. The reason behind this is the lack of task-specific discriminative learning setup. Speech is heavily doped with LI. In addition to this, the interference of extra-speech factors like environmental effects further makes the SR tasks challenging. Techniques like PCA or ICA [46],[47] perform unsupervised objective optimisation on the data from different speakers and cannot guarantee the distribution of output components to be sensitive to the inter-speaker variability. In fact they might get more sensitive to the variability in LI and environment [89],[90].

**Discriminative Features**

A few attempts have been made in the past, exploring the discriminative nature of speech features (mostly spectral) prior to speaker modelling in SR. One such attempt is using discriminative transformation of data in feature space to maximise the inter-speaker variability and intra-speaker similarity. In this motivation, Linear Discriminant Analysis (LDA) [103] is
applied on conventional spectral features like MFCCs to enhance separability among speakers. LDA is a dimensionality reduction approach based on the linear transformation of input features where the supervised discriminative objective optimisation is carried out to maximise the inter-class variance and minimising the intra-class variance. As a result, the lower dimensional features in the transformed space are used to create speaker models. In [94], LDA is applied on MFCCs and speaker models are created using GMM trained via the EM algorithm. The performance improvement is observed on closed set speaker identification. Similarly, Heteroscedastic LDA (HLDA), a generalisation of LDA, is applied to enhance the discriminative nature of MFCCs to train a GMM-based universal background model (UBM) [82] on large amount of speaker and session independent speech data [97]. The speaker models are then created using Bayesian adaptation on data for each reference speaker to perform speaker verification. The results show advantage of using features transformed by LDA as compared to baseline MFCCs. In doing so, the motivation is to decorrelate intra-speaker, session and channel variability with speaker specific information in the data.

In the same spirit, multilayer perceptrons (MLP) have been successfully used in nonlinear data transformation to project input speech feature to a more discriminative output space prior to speaker modelling. In [96], a feedforward neural network with multiple hidden layers is trained by pooling all the data (as MFCCs) of various speakers. Standard error backpropagation is used in training with fixed class (speaker) targets. In this approach, the hidden layer representation is used as new features. After training, the MLP is used to project the speech data of speakers to the new transformed space, which are used to input GMM to create speaker models for speaker identification. A Similar approach is used in [98] and [104] for speaker verification. Inspired by the Linear Predictive Coding
(LPC) of speech, a nonlinear extension of LPC is proposed as Neural Predictive Coding in [105] where speaker-dependent speech features are discriminatively learned in a nonlinear autoregressive model by minimising the prediction error. The network connectionist weights act as new features and speaker models are created using covariance matrices of acquired features that are compared using Arithmetic-Harmonic Sphericity [130]. The speaker models can also be estimated using GMMs as stated in [105].

In addition to the above, a few efforts have been made for SI extraction [58], [106], [107], based on the assumption that the dominant LI in speech is easily available to be extracted. Therefore, a nonlinear neural mapping approach is developed to extract the SI and suppress the LI in a new speaker-specific representation via discriminative learning. It is either assumed that the LI is only present in lower frequency bands of speech or the network learns to minimise the difference of phones uttered by different speakers. For speaker verification their performance is comparable to GMM.

Feature extraction techniques like LDA or its variants are highly dependent on speaker population size and are suitable for closed set SR tasks [94], [103]. In such a case, ideally for an SR task, training data should cover all the examples of intra-speaker, session and different environmental variabilities to effectively estimate the speaker dependent statistics. Failure to do so results in overfitting to specific dataset and its environment with poor performance for the examples unseen to the training. Similarly, nonlinear discriminative feature mapping using MLP is either constrained to the closed set SR [96], [104] or fails to achieve good generalisation due to lack of task-specific information extraction, i.e., SI, that requires highly complex and nonlinear objective optimisation [89], [90].

The proposed DNA in our work is also a discriminative model but ours
differs with all the above-mentioned techniques in motivation, hypothesis and goals, as there is a need to extract highly speaker-specific representations from the speech that directly implies the segregation of underlying speech information components at the output representation or feature space to encode generic SI.

**Deep Architectures**

Recently, Convolutional Deep Belief Network (CDBN) has been employed to extract speech representations from speech spectrograms through unsupervised learning [48]. Using the data reconstruction and sparsity constraints on learning objective, the acquired output representations are used in various speech and speaker related tasks like speaker identification, gender and phone classification with favourable results. However, the aim is to extract a generic speech representation rather than generic speaker representation. The details about CDBN are given in a later section. Also the comparative performance of CDBN is investigated for speaker verification in chapter 5.

### 2.4 Deep Learning

**2.4.1 Inspiration**

The idea of deep learning is based on the fact that AI tasks that require complex problem solving process can be tackled in an effective way by efficiently learning complicated functions with unique and abstract representations [10]. Good inference about the subject of interest and its environment is a key task of intelligent learning systems. In other words, from the ML perspective, a clairvoyant AI system should deal with the real world problems from perception to decision making by capturing maximum variation about specific information in the data [10], [11].

Inspired by the information mining and problem solving capability of
humans and mammals, the aim of deep learning by machines is to divide the complicated problems into sub-problems at various levels of perception. Eventually, each level provides an insight about the fundamental intrinsic characteristics of the given observation. It is assumed that such high-level representations of the observed variable may distribute the information content of its physical structure, state or environment in the target space [49], [50]. Therefore by utilising specific information for a specific task, these high-level distributive representations facilitate robust decision making from ML point of view. Figure 2.3 hypothetically illustrates the internal state of a deep architecture for a general problem of image or speaker classification.

2.4.2 Deep vs Shallow Architectures

The context of being deep or shallow for architecture is relative. There is no
hard theoretical and parametric boundary that separates these two kinds. In fact it is the specific ML task and the underlying data that identifies how deep (or shallow) the architecture should be to achieve an objective with good generalisation over large class variability [11], [54]. In contrast, less deep (more shallow) architectures may not efficiently capture the task-specific information from the data that pools multiple information. For high exclusivity and separability among the information components, the architecture should have enough computational elements or parameters to handle the complexity or nonlinearity of the function that uniquely describes the information of interest in the data. Failing to do so, as the case with shallow architectures, may result in the output representation that is more sensitive to the variability in the data [10], [11].

Most of the work in deep learning, till now, has been done in neural networks [50], [51], [52], [53]. Increasing the depth of a network means more nonlinear hidden layers. Each hidden layer in this case serves as a higher-level abstraction as compared to the layer below. Conventional architectures like neural networks with one or two hidden layers, RBF networks or SVM are categorized as shallow architectures. From a mathematical perspective, a large number of functions can be well approximated by such shallow architectures, provided they have sufficient number of elements in their hidden layers [10]. However, the number of elements needed to estimate a highly nonlinear function can grow exponentially [11], [50]. On the other hand, the same function may be represented efficiently with addition of another nonlinear hidden layer. In other words, increasing the depth of a network by one layer may exponentially reduce the number of elements/units/neurons in the preceding layer. Another major issue with shallow networks is their inability to show good generalisation in the case of highly variable functions [11], [50]. This means, the training data should cover all the
examples related to each variation, e.g., intensity, rotation, background effect in case of images/vision and noise, channel effects, language expressions, session and speaker variability in case of speech. Even though, the outcome can result in overfitting to the training data. Figure 2.4 highlights the general structures of shallow and deep architectures. Figure 2.4 (a) is a neural network with one hidden layer parameterised by weights $W_1$ and $W_2$, Figure 2.4 (b) is a typical SVM with the kernel function $\Lambda(x,x), i = 1,2, \ldots, N$ and weight matrix $W$. Figure 2.4 (c) is a deep neural network with $K$ hidden layers with weights $W_k, k = 1,2,\ldots,K$.

2.4.3 Challenges for Deep Learning

The idea of deep learning is not new but until 2006, there was no solid approach to train the networks with many hidden layers. Practically, it is hard to get a good learned solution to a problem for the neural network of more than two layers. As per convention, when these networks are initialised with random weights and biases and gradient descent optimisation is used to minimise the objective function, it is highly likely
that the training may get stuck in the local minima of highly a non-convex
cost function [49], [54]. In this case, a network with one or two layers may
produce better results as compared to the deeper ones, at least when
generalisation is not an issue and training is confined to the limited data. A
few attempts have been made to overcome this problem by systematically
adding layers and neurons in a neural network [55], [56] but in general
these techniques either result in overfitting to the training data or fail to
achieve the optimum goal [50].

In 2006, Hinton in his pioneering work proposed a systematic greedy layer-
by-layer training of a deep network [52]. The idea is to divide the training
of successive layers of a deep network in the form of small sub-networks
and use unsupervised learning to minimise input reconstruction error. This
will be discussed in detail later. This technique successfully eliminates the
shortcomings of the gradient based learning by averting the local minima.

2.5 Strategies and Methodologies for Deep Learning

As mentioned earlier, since efficient and reliable training procedures are
devised [50], [52], [10], deep learning gains the attention of the ML
community especially in several vision and language tasks. The most
popular and benchmark training strategies are reviewed in this section.

2.5.1 Greedy Layer-by-Layer Training

It is a known fact that in many semi-supervised scenarios, the
unsupervised learning helps in getting better generalisation [11], [13], [45],
[50]. The ability of an algorithm to learn from unlabeled data facilitates in
finding a better parametric solution for a supervised objective optimisation
[10], [11]. With random initialisation, especially in deep neural networks,
the optimisation solution usually gets stuck in local minima for a highly
non-convex cost function. This was the major hindrance in the use of such
Figure 2.5: Greedy layer-by-layer training of $K$-layered network.

deep networks until a systematic unsupervised procedure was adopted for training each layer of the network one by one [52]. This helps in better initialisation of a deep network and the learning trajectory with respect to a supervised objective may end up in the vicinity of global optimum solution. This way the unsupervised pre-training acts like a regularisation constraint for the network parameters in supervised fine-tuning. However this is different from the conventional regularisation of typical semi-supervised learning where an explicit constraint like L1 and L2 regularisation is added to the cost function [50]. Deep networks usually undergo two separate phases of unsupervised and supervised learning. In the pre-training of network parameters (weights and biases), each hidden layer of neural network is separately trained with some unsupervised algorithm. The first layer acquires a representation that is a nonlinear transformation of the input. After training, this representation is taken as an input to the second layer, which repeats the same learning process and so on. In this way, each layer acts as a high-level nonlinear transformation of the layer below and tries to capture many factors of variation in its input. This stage is followed by supervised learning (fine-tuning). Each separately trained layer in the pre-training phase is stacked over one
another to create a deep network and all the parameters are tuned with respect to a global objective, usually using gradient based optimisation. Figure 2.5 illustrates a general idea of two-stage learning of deep network with $h_k$ represents the $k^{th}$ hidden layer and $W_k$ are the weight matrices connecting input and hidden layers. $k = 1, 2, \ldots, K$ is the number of hidden layers. The left three networks shows greedy layer-by-layer unsupervised training while the right most network created by stacking the hidden layers is the final supervised training. $o_k(x), k = 1, 2, \ldots, K$ are the unsupervised objective functions while the supervised objective is given by $o_s(x)$.

Apart from unsupervised pre-training, deep network can be initialised in layer-by-layer training using supervised criteria. So each layer in Figure 2.5 is separately trained with a supervised local objective function and finally stacked over each other in the fine-tuning stage. For both learning stages, in this case, the local and global objective functions can be the same. Although the feasibility of employing either unsupervised or supervised pre-training can be application specific, empirical evaluation on various ML tasks suggests that initialising deep networks in an unsupervised way better captures the possible variation in the input at each layer as compared to supervised pre-training [10], [11], [50].

We now discuss in detail, the pioneering methodologies to train deep neural networks as they are somehow related to our DNA. These techniques mainly differ in their building blocks to create a deep network.

### 2.5.2 Learning by Abstract Probabilistic Inference

To train a deep network in a greedy layer wise manner, Hinton proposed the pre-training procedure by training each layer of the network through a Restricted Boltzmann Machine (RBM) [51], [52]. RBM is a two-layered neural network, in which the hidden layer infers the probability
distribution of the input data. To create a deep network, the hidden layers of several RBMs are stacked over each other. This arrangement is called Deep Belief Network (DBN). Each RBM network is trained independently by taking the hidden layer representation of the previous RBM as its input and minimising the difference between the actual and estimated input data. Hence, at each RBM, more complex high-level features representing the input data are learned.

The RBM is an energy-based model (EBM) that defines the probability distribution of data through an energy function \[50\], \[57\]. Therefore for input \(x\)

\[ p(x) = \frac{e^{-E(x)}}{Z}, \tag{2.6} \]

where \(E(\cdot)\) is the energy function over the input variable \(x\) and \(Z\) is the normalisation term to keep \(p(x)\) in the range \([0,1]\), i.e.

\[ Z = \sum_x e^{-E(x)}. \tag{2.7} \]

From (2.6), it is evident that in training, the EBM maximises the probability for a desirable configuration of \(x\) by minimising the energy function \(E(x)\). In other words the numerator of (2.6) is maximised for a particular configuration of \(x\) while in the denominator all other configurations are minimised. For an RBM with one hidden layer, the input \(x_i, i = 1, ..., |x|\) and hidden layer variable \(h_j, j = 1, ..., |h|\) can be modelled using conditional probability density,

\[ p(x|h) = \prod_i p(x_i|h), \quad p(h|x) = \prod_j p(h_j|x) \tag{2.8} \]

where

\[ p(x_i|h) = \left(\sigma \left( b_i + \sum_{j=1}^{[h]} W_{ij} h_j \right) \right)^{|x|}, \tag{2.9} \]

assuming that \(x\) is binary, i.e., \(x \in [0,1]\) and \(\sigma(\cdot)\) is a sigmoid neural
where the energy \( E(x, h) \) for binary \( x \) and \( h \) is given as

\[
E(x, h) = -x^T W h + b^T x + c^T h
\]
Algorithm 2.1: RBM Training

**Input:** Input data $\mathbf{x}$, weights $W_{ij}$, biases $b_j$ and $b_i$ and learning rate $\epsilon$

**Initialise:** $W_{ij} \in \text{normal}(0,0.1)$

**Initialise:** $b_j, b_i \in (0)_{i,j}$

**Start:** For iteration $I = 1, 2, ..., N$

$$p(h_j|x) = \left( \sigma \left( b_j + \sum_{i=1}^{[x]} W_{ij} x_i \right) \right)_{j=1}^{[h]},$$

$h_j^{(r)} \sim p(h_j|x)^*$,

$$x_i^{(1)} = p(x_i|h^{(r)}) = \left( \sigma \left( b_i + \sum_{j=1}^{[h]} W_{ij} h_j^{(r)} \right) \right)_{i=1}^{[x]},$$

$$h_j^{(1)} = p(h_j|x^{(1)}) = \left( \sigma \left( b_j + \sum_{i=1}^{[x]} W_{ij} x_i^{(1)} \right) \right)_{j=1}^{[h]}.$$

**Update:**

$$W_{ij} \leftarrow W_{ij} - \epsilon \frac{\partial \log p(x)}{\partial W_{ij}},$$

$$b_j \leftarrow b_j - \epsilon \frac{\partial \log p(x)}{\partial b_j},$$

$$b_i \leftarrow b_i - \epsilon \frac{\partial \log p(x)}{\partial b_i}.$$

**End**

* $a^{(r)} \sim b$ implies: $a$ is a random sample of $b$

$a^{(1)}$ implies: one step contrastive divergence

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{i=1}^{[x]} b_i x_i - \sum_{j=1}^{[h]} b_j h_j - \sum_{i=1}^{[x]} \sum_{j=1}^{[h]} x_i h_j W_{ij}. \quad (2.13)$$

And for real valued input, (2.13) becomes

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{i=1}^{[x]} \frac{(x_i - b_i)^2}{2 \rho_i^2} - \sum_{j=1}^{[h]} b_j h_j - \sum_{i=1}^{[x]} \sum_{j=1}^{[h]} \frac{x_i}{\rho_i} h_j W_{ij}. \quad (2.14)$$

The network parameters i.e. weights and biases are updated using gradient descent of the negative log-likelihood of (2.13) using one step contrastive divergence (CD-1) as explained in [52]. For $K$ RBMs (with $K$ hidden layers), The DBN as shown in Figure 2.6 is created by placing the hidden layers of the trained RBMs over each other while $\mathbf{x}$ being the input to the network $W_k$, $k = 1, 2, ..., K$ are connection weights between the successive
Such a network is to represent the data in a form of encoding. The objective of unsupervised learning, the aim of autoassociators or autoencoder on the other hand, are a class of models that can represent the data in a series of nonlinear transformations \[10\], \[50\]. Generally, an autoassociator is a neural network with multiple linear or non-linear hidden layers. Based on the principle of unsupervised learning, the aim of such a network is to represent the data in a form of encoding. The objective of learning is to minimise the data reconstruction error. Autoassociators are created by stacking the hidden layers of several RBMs. The unsupervised learning of RBMs in a greedy layer-by-layer training procedure provides an efficient way to initialise a deep neural network. Such a network is created by stacking the hidden layers of several RBMs. Each layer is thus a higher-level abstract characterisation of the input data distribution. The learning objective in RBM is achieved by minimising the difference between the actual and reconstructed data \[52\].

**2.5.3 Learning by Encoding**

The unsupervised learning of RBMs in a greedy layer-by-layer training procedure provides an efficient way to initialise a deep neural network. Such a network is created by stacking the hidden layers of several RBMs. Each layer is thus a higher-level abstract characterisation of the input data distribution. The learning objective in RBM is achieved by minimising the difference between the actual and reconstructed data \[52\]. Autoassociators or autoencoder on the other hand, are a class of models that can represent the data in a series of nonlinear transformations \[10\], \[50\]. Generally, an autoassociator is a neural network with multiple linear or non-linear hidden layers. Based on the principle of unsupervised learning, the aim of such a network is to represent the data in a form of encoding. The objective of learning is to minimise the data reconstruction error. Autoassociators are

---

**Figure 2.7: Deep neural network with autoassociators.**
mainly used as dimensionality reduction tools in the bottle-neck networks [58], [59], but the flexibility in choosing their structural topology makes them an alternative to the RBM in the pre-training phase of constructing deep networks [10], [50]. In this case, the training hierarchy is similar to that of stacked RBMs as shown is Figure 2.7. Like RBMs, several autoassociator networks are trained in the pre-training phase. Each autoassociator is a two-layered neural network with a single nonlinear hidden layer. In greedy layer-wise training as in Figure 2.5, the representation learned by one autoassociator serves as an input to the next level autoassociator and so on. Therefore for input $x$, the hidden layer representation is given as

$$h_j(x) = \left( \mathcal{F} \left( b_j + \sum_{i=1}^{[x]} W_{ij}^{(1)} x_i \right) \right)_{j=1}^{[h]}, \quad (2.15)$$

where $W_{ij}^{(1)}$ are the connectionist weights between input $x_i$ and $h_j$, while $b_j$ is the hidden layer bias vector. $\mathcal{F}(\cdot)$ is a neural network activation function. The choice of the activation function varies depending upon the statistical properties of the data [52]. The input reconstruction at the output layer is

$$\hat{x}_i = \left( b_i + \sum_{j=1}^{[h]} W_{ij}^{(2)} h_j(x) \right)_{i=1}^{[x]}, \quad (2.16)$$

In (2.15) and (2.16) $W_{ij}^{k}, k = 1, 2$ are the weights corresponding to the hidden layer $(k = 1)$ and output layer $(k = 2)$. Here, (2.16) is shown as a linear transformation of hidden layer to output layer whereas in practice it can be nonlinear transformation like (2.15). The objective function for training autoassociator networks can be mean square error between actual and reconstructed input,

$$o(x, \hat{x}) = \frac{1}{[x]} \sum_{i=1}^{[x]} (x_i - \hat{x}_i)^2 = \|x - \hat{x}\|_2^2. \quad (2.17)$$
Algorithm 2.2: Autoassociator Training

**Input:** Input data $x$, weights $W_{ij}^{(1)}$, $W_{ij}^{(2)}$, biases $b_j$, $b_i$ and learning rate $\epsilon$

** Initialise:** $W_{ij}^{h} \in N(0,0,1)$

** Initialise:** $b_j, b_i \in (0)_{i,j}$

**Start:** For iteration $I = 1, 2, \ldots N$

$$h_j(x) = \left(\mathcal{F} \left( b_j + \sum_{i=1}^{|x|} W_{ij}^{(1)} x_i \right) \right)_j,$$

$$\hat{x}_i = \left( b_i + \sum_{j=1}^{|h|} W_{ij}^{(2)} h_j(x) \right)_i.$$

**Update:**

$$W_{ij}^{(1)} \leftarrow W_{ij}^{(1)} - \epsilon \frac{\partial o(x, \hat{x})}{\partial W_{ij}^{(2)}}, W_{ij}^{(2)} \leftarrow W_{ij}^{(1)^T},$$

$$b_i \leftarrow b_i - \epsilon \frac{\partial o(x, \hat{x})}{\partial b_i}, b_j \leftarrow b_j - \epsilon \frac{\partial o(x, \hat{x})}{\partial b_j}.$$

**End**

Same learning principle is applied for higher layers. The parameter optimisation usually involves gradient descent error backpropagation. Autoassociator networks can be used to reconstruct the input with arbitrary precision. However, if the dimension of the hidden layer is equal to or more than the input data dimension, the system can learn an identity function, where it just copies the input at the output layer. In this case, the encoded representation at the hidden layer may not convey any information about the input. One possible solution to this problem is proposed in [24], where the weights between the hidden and output layer are constrained to be in the proximity of the weights between the input and the hidden layer, i.e. $W_{ij}^{(2)} = W_{ij}^{(1)^T}$. Another idea to avoid learning an identity function is to corrupt the actual input to the network and train the network to reconstruct the clean input [60]. The mismatch between the desired target and actual input forces the network to concentrate on the structural and statistical details of the input data in the hidden layer. Such
Figure 2.8: Convolutional neural networks. (a) One feature map in a hidden layer. (b) LeNets.

an autoassociator is called denoising autoencoder, which not only eliminates the possibility of learning identity weights, it also increases the robustness of the network model against unwanted redundancies in the data. The details of denoising autoencoder will be discussed in Chapter 3.

2.5.4 Learning by Preserving Spatial Correlation

Convolutional neural networks (CNN) are the oldest in the deep architecture family [54]. Inspired by the visual cortex of a cat, CNN are variants of simple multi-layer perceptrons (MLP). The neurons in the network layer are connected to the fixed sub-region called receptive field or input map in the input space. The same principle can be repeated for upper layers, in which each neuron is linked with the specific sub-region in the layer below. Each layer, as a whole, covers its entire input region and acts as a higher-level feature detector of the input. Figure 2.8 (a) illustrates the connectionist arrangement in one layer of CNN. Each neuron in the hidden
layer is connected to the fixed set of input variables (input map). The weights for each input map are shared among few hidden layer neurons. The neurons that share the same weights are collectively called a feature map. The input can be divided into several input maps. Similarly a hidden layer may have multiple feature maps. CNN exploits a special correlation in the input space and using several layers of non-linear transformation, the output representation encodes the unique pattern of input data. The spatially locked feature maps detect the features irrespective of their position in the input field of view [61], [12]. Proposed by Yann LeCun, LeNets are the effective implementation of CNN, which incorporates several feature selection (sub-sampling) layers along with the typical convolutional layers as shown in Figure 2.8 (b). The resulting architecture is not only computationally efficient but characterises the intrinsic structure of the input in lower dimension. LeNets have been used in various vision related tasks, ranging from hand-written digit recognition to object recognition [61], [63], [12] with promising results.

The input maps are connected with the feature maps through local sets of weights called kernels. Each output map is a combination of multiple convolutions between input maps and kernels. The convolution layer is the valid convolution of feature maps of the previous layer and the kernels through a nonlinear activation function. The \( m^{th} \) feature map in layer \( k \), where \( k = 0, ..., K \) is given as

\[
 h_j^m = \left( F \left( \sum_{i=N_j} h_{ij}^{m-1} \ast \Gamma_{ij}^m + b_j^m \right) \right)_{j=1}^{|h^m|}, \tag{2.18}
\]

where \( |h^m| \) is the number of neurons in the \( m^{th} \) feature map of layer \( k \). \( k = 0 \) represents the input layer with \( N_j \) input maps. \( \Gamma_{ij}^m \) is the kernel function connecting the \( i^{th} \) neuron in a map of a layer \( k-1 \) with the \( j^{th} \) neuron in the map of layer \( k \). In LeNets, the convolutional layer is usually
Algorithm 2.3: LeNet Training

**Input:** Input data $\mathbf{x}$, kernels $\Gamma^m_{ij}$, weights $W^K_{ij}$, biases $b^m_j$, $b^K_j$, learning rate $\epsilon$

**Initialise:** $\Gamma^m_{ij}, W^K_{ij} \in \mathcal{N}(0,0.1)$

**Initialise:** $b^m_j \in (0)^m, b^K_j \in (0)^K$

**Start:** For iteration $I = 1, 2, \ldots, N$

\[
h^m_j = \left( \mathcal{F} \left( \sum_{i=\mathcal{N}_j} h^{m-1}_i \ast \Gamma^m_{ij} + b^m_j \right) \right)_{j=1}^{|h^m|}
\]

\[
\delta^m_j = \left( \mathcal{F} \left( \beta^m_j d(h^m_j) + b^m_j \right) \right)_{j=1}^{|h^m|}, k \neq 0,1
\]

\[
h^K_j = \left( \mathcal{F} \left( W^K_{ij} h^K_{j-1} + b^K_j \right) \right)_{j=1}^{h^K}
\]

**Update:**

\[
W^K_{ij} \leftarrow W^K_{ij} - \epsilon \frac{\partial o(x)}{\partial W^K_{ij}}
\]

\[
\Gamma^m_{ij} \leftarrow \Gamma^m_{ij} - \epsilon \frac{\partial o(x)}{\partial \Gamma^m_{ij}}
\]

\[
b^m_j \leftarrow b^m_j - \epsilon \frac{\partial o(x)}{\partial b^m_j}
\]

\[
b^K_j \leftarrow b^K_j - \epsilon \frac{\partial o(x)}{\partial b^K_j}
\]

**End**

followed by a sub-sampling (down sampling) layer. Instead of kernel or weights, the sub-sampling layer is connected to the convolutional layer with a bias $\beta$. Sub-sampling layer reduces the dimension of the input feature maps but keeps the number of maps same. The sub-sampling operation is given by

\[
\delta^m_j = \left( \mathcal{F} \left( \beta^m_j d(h^m_j) + b^m_j \right) \right)_{j=1}^{|h^m|}, k \neq 0,1
\]  

(2.19)

where $d(\cdot)$ is down sampling operation. The last two layers of LeNet are standard fully connected MLPs linked through the weight matrix $W_{ij}$ as

\[
h^K_j = \left( \mathcal{F} \left( W^K_{ij} h^K_{j-1} + b^K_j \right) \right)_{j=1}^{h^K},
\]  

(2.20)

where $K$ denotes the output layer.
2.5.5 Deep Autoencoder

Autoencoders have been used in numerous tasks ranging from dimensionality reduction (bottle-neck networks) to image and signal denoising [10], [50], [54]. Other applications involve SR using the reconstruction error distribution [58], [59].

In the pre-training phase, network weights and biases of the \( K \) autoassociators or RBMs are trained in a greedy layer-by-layer training. The deep network is created by stacking the \( K \) hidden layers of the corresponding autoassociators or RBMs. Deep autoencoder is the unfolding of these layers with an objective to reconstruct the input \( x \) in the global unsupervised learning. The resulting deep network has \( 2K \) layers.

For the first \( K \) layers, the weight matrices are \( W_1, \ldots, W_K \) and for the unfolded layers, \( W_{K+k} = W^{T}_{K-k+1} \) to initialise \( W_{K+1}, \ldots, W_{2K} \). The loss
The function for training the deep autoencoder is same as (2.17). The general structure of a deep autoencoder is shown is Figure 2.9. After training, \( h_K(x) \) is the output representation, while \( h_{2K}(x) \) is the reconstruction of the input \( x \).

### 2.5.6 Deep Siamese Network

The Siamese network is used for a pair-wise comparison of the same or different input classes. In this setup, two identical networks (sub-nets) with same parameters are linked together with a shared compatibility measure. The concept of using Siamese network goes back to [64] for the images of hand-written signature verification. Similarly, the deep CNN and DBN have been used in the similar contrastive learning for face and digit recognition [12], [65]. Figure 2.10 illustrates a general Siamese architecture. Here, again for the \( K \) layers, the weights of the two sub-nets with inputs \( x_i, i = 1,2 \) are initialised with pre-training. For fine-tuning, the loss function driven by the compatibility measure or metric between the output representations \( h_K(x_i) \) is minimised with respect to the different inputs \( x_i \) but the same parameters \( W_k, k = 1,2, ... K \), for the two sub-nets.

### 2.5.7 Other Variants

So far in the literature, the milestone development in the evolution of deep
architectures can be attributed to DBN, deep autoassociative networks and CNN. However, several variants have been presented that are based on the structural and learning hierarchy of these three benchmark architectures. These variant architectures have successfully produced better or state-of-the-art in different vision, audio and language related tasks [67], [51], [48], [69], [70]. In this section a brief overview about such deep architectures is presented.

**Unsupervised CNN**

As mentioned earlier, unsupervised learning has been successfully used to initialise the parameters of deep networks in case of DBN and deep autoassociative networks. Previously random initialisation of such deep networks was responsible for poor optimisation solution and hence was the biggest obstacle in practical application of deep learning in real world problems.

The ability to extract useful information from unlabelled data makes unsupervised learning attractive to learn generic and invariant features, especially when used in conjunction with additional regularisation or supervised constraints. With the similar motivation, the extension to the LeNet with unsupervised learning in an encoder-decoder setup is presented [67]. Employed in an image classification task, the architecture behaves like a conventional LeNet, with standard convolution layers followed by sub-sampling layers alternately. The encoder may contain multiple sets of convolution and sub-sampling layers. The encoder produces a sparse representation of input that is then forced to reconstruct the input in gradient descent error backpropagation optimisation at the decoder. In general several modules can be trained in this way with each module capable of representing more abstract and transformation invariant features. A deeper network is created by stacking these learned encoders
and finally subjecting them to supervised learning with a global objective. In the proposed technique the unsupervised training helps in alleviating the problem of overfitting with better generalisation as compared to standard LeNet.

**Convolutional DBN (CDBN)**

The RBM is credited for its potential to extract the intrinsic distribution pattern from the data. Stacking RBM in DBN further boosts the capacity of the architecture to discover meaningful and complex nonlinear representation of data. On the other hand CNN exploits the spatial structure of the object of interest in the data to yield its shift invariant features.

Convolutional DBN (CDBN) provides a way to combine a high-level generative modelling of DBN and shift invariance property of CNN [51], [61]. Mainly feasible for object and image recognition, CDBN has been used in several speech related tasks like phone and speaker recognition. [48].

**Mean-Covariance RBM (mcRBM)**

Although DBNs are generative models and each hidden layer captures the statistical relationship between the neurons of the lower layer, estimating the true mean and covariance structure of the data is a challenging problem. Mean-Covariance RBM (mcRBM) is proposed to tackle such issues in modelling natural images [68]. Moreover this technique proves to be the state-of-the art in phone recognition [69]. The fundamental aim of mcRBM is feature extraction by effective acoustic modelling of phones in speech data. These features are then taken as an input to the standard DBN. Previously DBN has been used for phone recognition using general spectral representation of speech like MFCCs as input. The mcRBM uses learning steps of the conventional RBM but the hidden layer units are
divided into two sub-layers. One sub-layer models the mean while the other sub-layer models the covariance of the input data. After training, the features represented by the hidden layer serve as input to the DBN, which can be trained in a greedy layer-by-layer procedure.

**Multi-layer Kernel Machines (MKM)**

Inspired by the hierarchical learning of deep networks, the kernel PCA approach is proposed in multi-stage learning that mimics the functional style of deep neural networks like DBN and deep autoassociator networks [70]. The input data is subjected to the kernel PCA decomposition and the most favourable principal components are then selected using some feature selection technique. The selected features are then further decomposed into the next stage kernel PCA. After several stages, the output features are categorised using a supervised classifier like Large Margin Nearest Neighbour classifier [71]. The main contribution of this work is the design of a sophisticated kernel function for kernel PCA. The results in hand-written digit classification with MKMs are comparable with the standard DBN.

### 2.6 Chapter Summary

This chapter starts with the literature review about the structure of speech signals and speech production systems. Then the background knowledge on various SR systems and their functional hierarchy is presented. This is followed by the quantitative review of various state-of-the-art SR systems in terms of their recently published performances. Different past attempts to extract the speaker-specific features from speech have also been reviewed which range from basic signal processing on sources of speech production, unsupervised data component analysis techniques to various discriminatively learned features used in SR. Deep architectures are
potential candidates to extract task-specific information from the data. Therefore, a theoretical study is carried out on deep learning and the advantages and challenges involved in training deep architectures. Various deep learning strategies and deep architectures have been reviewed. This includes Deep Belief Networks (DBN), Deep Autoencoders (DAE), Deep Siamese Networks (DSN), Convolutional Neural Networks (CNN) and their variants like Convolutional DBN and Mean-Covariance RBM.
Chapter 3 Design Hierarchy of Deep Neural Architecture

This chapter presents the design strategy for the proposed Deep Neural Architecture (DNA). Starting with pre-training we investigate the role of two-stage unsupervised and regularised contrastive learning to explore favourable representation for SI. To avoid the environmental corruption and detrimental interference of various speech information components with SI, it is inevitable to extract generic speaker-specific features for robust SR performance. To achieve this we propose a DNA and a regularised contrastive loss function for its training. The proposed scheme ensures the segregation of SI from dominant LI and other non-speaker related information by exploiting the similarity/dissimilarity among the speakers at the DNA’s output feature space. This marks the main contribution of our work and differentiates ours with other discriminative feature extraction approaches. Therefore, in this chapter, detailed training hierarchy of the DNA is presented.

3.1 Model Description

In this section, we propose a DNA that is specifically designed to extract the SI by learning the statistical compatibility among the speakers. The two-stage training procedure is presented that involves pure unsupervised network initialisation (pre-training) followed by regularised contrastive learning (fine-tuning) to minimise a contrastive loss function.
3.1.1 Structure of DNA

The structure of proposed DNA is shown in Figure 3.1. The network consists of two sub-nets with $2K$ hidden layers for $K > 1$. Each subnet is a fully connected multi-layered feedforward neural network with $W_k$, $k = 1, 2, ..., 2K$ being the connection weights between the layers. The input to the subnet $i$, $i = 1, 2$ is a pair of MFCC feature vectors (corresponding to speech frames) $x_{it} \in X_i$ at time $t$, where $X_i = \{x_{it}\}_{t=1}^{TB}$ is a batch of $T_B$ vectors which represents a speech segment. After designing an appropriate loss function, the system is trained using Stochastic Gradient Descent (SGD) [121] with the batch training. Each batch comprises a pair of $T_B$ MFCC vectors $(X_1, X_2)$.
as input to the two sub-nets while the corresponding top layers of the sub-nets produce the reconstruction \( \hat{X}_1, \hat{X}_2 \). Let \( h_{k,j}(x_{it}) \) denote the output of the \( j^{th} \) neuron in layer \( k \) for \( k = 0, 1, \ldots, K, \ldots, 2K \). Therefore the collective notation for all the neurons in layer \( k \) is \( h_k(x_{it}) = (h_{k,j}(x_{it}))^{l_{hk}}_{j=1} \) for the sub-net \( i, i = 1,2 \) where, \( |h_k| \) represents the number of neurons in layer \( k \). The layer with \( k = 0 \) means the input layer, i.e., \( h_0(x_{it}) = x_{it} \), while \( k = 2K \) is the top layer of each sub-net so \( h_{2K}(x_{it}) = \hat{x}_{it} \). The layer \( K \) in both sub-nets is divided into two parts (set of blue and red neurons). The red neurons, denoted collectively as \( CS \) in both sub-nets, are the subset of the \( K^{th} \) layer encoding that is responsible to capture SI while the blue neurons denoted as \( \overline{CS} \) in the \( K^{th} \) layer accommodate the non-speaker related information which may include linguistic content or environmental corruption. The combined hidden layer representation at layer \( K \) is denoted as \( h_K(x_{it}) = [CS(x_{it}), \overline{CS}(x_{it})] \), where \( CS(x_{it}) = (h_{K,j}(x_{it}))^{CS}_{j=1} \) and \( \overline{CS}(x_{it}) = (h_{K,j}(x_{it}))^{l_{hk}}_{j=|CS|+1} \) for \( i = 1,2 \). Both sub-nets are linked together with an incompatibility measure \( D(CS(X_1), CS(X_2); \theta) \) at layer \( K \) where \( \theta \) is the collective notation for the DNA parameters i.e., weights and biases shared by both sub-nets. All the neurons in the layer \( K \) for both sub-nets are fully connected to the neurons in layer \( K + 1 \) but the contrastive measure \( D \) is only dependent on the \( CS \) representations of both sub-nets. After training, the network output is the representation at layer \( K \) that is called code layer as it encodes the final output. Both sub-nets have the same hyper parameters i.e., weights and biases, so either of the sub-nets can produce an output representation. The final network after training is shown in the boxed region in Figure 3.1. Now for \( k = 1,2, \ldots, 2K - 1 \) the output of the layer \( k \) is given as

\[
    h_k(x_{it}) = \sigma(u_k(x_{it})),
\]

(3.1)
where
\[ u_k(x_{it}) = W_k h_{k-1}(x_{it}) + b_k. \] (3.2)
and \( \sigma(z) = (1 + e^{-z})^{-1} \) is the sigmoid neural activation function. For the top layer i.e., \( k = 2K \)
\[ h_{2k}(x_{it}) = u_{2k}(x_{it}) = W_{2k} h_{2k-1}(x_{it}) + b_{2k}. \] (3.3)
Eq. 3.3 is the linear transformation of \( h_{2k-1}(x_{it}) \) as it is the reconstruction of the real valued MFCC input \( x_{it} \).

### 3.1.2 Loss Function Design

For the batch of input pair \( X_i, i = 1,2 \), the output SI representation at the code layer of both sub-nets is denoted as \( CS(X_i) = \{CS(x_{it})\}_{t=1}^{TB} \). Therefore \( CS(X_i) \) is the collection of the representations for \( TB \) input MFCC frames.

Inspired by the fact that statistics can capture the speaker-dependent characteristics [1], the compatibility measure \( D \) is based on the first and second order statistics of the output representations, i.e.,
\[ D[CS(X_1), CS(X_2); \Theta] = \|\mu^1 - \mu^2\|_2^2 + \|\Sigma^1 - \Sigma^2\|_F^2, \] (3.4)
where
\[ \mu^i = \frac{1}{TB} \sum_{t=1}^{TB} CS(x_{it}), i = 1,2, \] (3.5)
and
\[ \Sigma^i = \frac{1}{TB} \sum_{t=1}^{TB} [CS(x_{it}) - \mu^i][CS(x_{it}) - \mu^i]^T, i = 1,2. \] (3.6)
Therefore, (3.5) and (3.6) are the mean and covariance matrices of the \( CS(X_i) \) representation of the corresponding sub-net. In (3.4) \( \|\cdot\|_2 \) and \( \|\cdot\|_F \) are the \( L_2 \) norm and Frobenius norm respectively. It is evident from (3.4) that the compatibility measure should be small if the input pair \( X_i, i = 1,2 \) represents the speech segments from the same speaker. In contrast, it should be large if the input pair belongs to two different speakers. In the case of same speakers, the pair \( X_i, i = 1,2 \) is called genuine pair with label
$\mathcal{I} = 1$ and for different speakers, it is called *imposter pair* with label $\mathcal{I} = 0$.

The overall loss function to train the DNA is given as

$$L(X_1, X_2; \Theta) = \alpha[L_R(X_1; \Theta) + L_R(X_2; \Theta)] + (1 - \alpha)L_D(X_1, X_2; \Theta),$$

where the reconstruction loss is

$$L_R(X_i; \Theta) = \frac{1}{T_R} \sum_{t=1}^{T_R} \|	ilde{x}_{it} - \tilde{x}_{it}\|_2^2, i = 1, 2,$$

and the discriminative contrastive loss is

$$L_D(X_1, X_2; \Theta) = \mathcal{I} \mathcal{D} + (1 - \mathcal{I}) \left(e^{-\frac{\mathcal{D}_m}{\lambda_m}} + e^{-\frac{\mathcal{D}_s}{\lambda_s}}\right).$$

In (3.9), $\mathcal{D}_m = \|\mu^1 - \mu^2\|_2^2$ and $\mathcal{D}_s = \|\Sigma^1 - \Sigma^2\|_F^2$. $\lambda_m$ and $\lambda_s$ are the upper bounds on $\mathcal{D}_m$ and $\mathcal{D}_s$. The measure $\mathcal{D}$ is divided into two separate metrics in the case of an imposter pair. This is the result of our empirical evaluation that $\mathcal{D}_m$ and $\mathcal{D}_s$ should be tuned separately in the loss function because of the difference in their order. Since $CS(X_i) \in [0, 1]$, the negative exponential of $\mathcal{D}_s$, which involves higher-order covariance, vanishes during training. This could result in a biased learning towards the mean value in $\mathcal{D}_m$ in the case of imposter pairs. We dropped explicit parameters from (3.9) for simplicity.

The loss function (3.7) is a multi-objective function based on the reconstruction losses $L_R(\cdot)$ with respect to the input pair $X_i, i = 1, 2$ and the supervised contrastive loss $L_D(\cdot)$. The contrastive loss, as given in (3.9), is the combination of two cost functions to minimise the objective with respect to the genuine and imposter pair, as the absence of either one may create a dangerous plateau in the overall loss function. This could result in a *collapsed* solution in which the system either ends up in a trivial solution or fails to learn any useful information [12]. The parameter $\alpha(0 < \alpha < 1)$ in (3.7) is a regularisation constant to balance the effect both losses with optimum performance. Smaller values of $\alpha$ result in powerful discriminative learning with a small influence of the unsupervised
reconstruction constraint, while with the larger values of $\alpha$, the network behaves like an autoencoder without any contrastive learning. For optimum performance, tuning the value of $\alpha$ plays an important role.

### 3.2 DNA Training

The training of DNA involves two-stage learning of the network parameters. The first stage, i.e., pre-training, is unsupervised learning using unlabeled MFCC data and provides the initialisation for the DNA parameters. The second stage, i.e., fine-tuning stage involves hybrid training which depends on the regularisation (unsupervised) and contrastive (supervised) losses as given by (3.7), (3.8) and (3.9).

#### 3.2.1 Network Initialisation (Pre-training)

As discussed earlier, unsupervised training, especially in deep learning, provides a way towards better generalisation by capturing the invariant information in the data. Stacked RBMs and stacked autoassociators are the two approaches used to initialise the parameters of deep neural networks. Such initialisation helps in averting the poor solution in gradient descent based optimisation problems in the networks with many hidden layers. Previously, the depth of a multi-layer neural network was unable to go beyond one or two hidden layers, when the aim was to minimise highly nonlinear and non-convex loss functions. For the unsupervised deep network initialisation, RBM minimises the input data reconstruction error by generating the data using the estimate of its distribution. On the other hand autoassociator, a fully connected neural network with one hidden layer, achieves the data reconstruction in an encoder-decoder style. The reconstruction error in this case can be minimised at any desired level. The hidden layer representation in both RBM and autoassociator serves as an input to the next level RBM or autoassociator respectively, which repeats
the same learning cycle. The process continues until a desired network depth is reached. For the initialisation of DNA in pre-training, a variant of autoassociative network called denoising autoencoder is used.

**Denoising Autoencoder**

The training protocol of a conventional autoassociator is already presented in Section 2.5.3. One major limitation of an autoassociator network was highlighted i.e., the network may learn the identity function if the number of neurons in the hidden layer exceeds the dimension of the input. A strategy was proposed to constrain the weights connecting the hidden and output layer to be in the numerical vicinity of the weights connecting the input and hidden layer [50]. This constraint together with the stochastic gradient descent for parameter optimisation proved to be successful in encoding the useful statistical pattern of the input data.

Another approach is to use *Denoising Autoencoder* to avoid learning the identity function [60]. Denoising autoencoder is similar to the conventional autoassociator, except the fact that it uses the input that is partially destroyed or corrupted. Such network is forced to reconstruct the clean version of the corrupted input. The advantages of this simple yet effective approach are twofold; one advantage is that, it prevents the network to learn the identity function. As the goal of an autoassociator is to reconstruct the input, so the target is the input itself. In case of identify in parametric space, the network just copies the input at the output layer and the hidden layer representation cannot encode useful information about the input. However, the mismatch between the target and input avoids such situation. The second advantage is the fact that the network learns to recover the missing information contents in the input. Provided nonlinear hidden layers, this helps in capturing the intrinsic input data distribution at the hidden layer. The ability to extract the full information from partially
Figure 3.2: Pre-training with denoising autoencoders.

Observable data increases the robustness of the system against unknown harmful variability such as noise. This is motivated by the human ability to recognise and recall images and sounds with the observable information being occluded or partially present [50], [60].

Denoising autoencoders have been used in various learning tasks to denoise the corrupted images [50], [60]. Also they were used for enhancing speech corrupted by additive noise [72]. Inspired by the above-mentioned advantages, we use denoising autoencoders as building blocks to construct the deep architecture.

**Greedy Layer-Wise Pre-training**

Stacked denoising autoencoders are used to construct successive layers in the deep network. Therefore the weights and biases learned in the pre-training stage will be used to initialise the corresponding parameters in the fine-tuning stage. Figure 3.2 depicts the greedy layer-wise pre-training
using denoising autoencoders. Like a conventional autoassociator, denoising autoencoder is a three-layered neural network for which the input $\tilde{x}$ is the corrupted version of the target $x$. Data corruption can be carried out in a number of ways depending upon the numerical structure of the input, e.g. the salt and pepper noise is often used to corrupt the images with binary values [60]. In our case, for the first autoencoder, MFCCs are used as inputs while the inputs to high-level autoencoders are the representation learned in the hidden layers of the corresponding lower-level autoencoders. All the inputs to the autoencoders are continuous-valued, therefore we used additive Gaussian noise for the data corruption. For a training example $(\tilde{x}_t, x_t)$ corresponding to the frame at time $t$, the denoising autoencoder estimates the clean input as $\hat{x}_t$. So for the $k^{th}$ layer, $k = 0, 1, \ldots, K$, the hidden layer encoding is given as

$$ h_k(x_t) = \sigma(W_k h_{k-1}(x_t) + b_{hk}), \quad (3.10) $$

where $h_k(x_t) = \left( h_{kj}(x_t) \right)_{j=1}^{h_k}$, $W_k$ is the connection weight matrix between layer $k - 1$ and $k$, while $b_{hk}$ is the bias vector for the corresponding hidden layer. Again the layer $k = 0$ refers to the input and output layer of the first autoencoder with $h_0(x_t) = x$ and $\hat{h}_0(x_t) = \hat{x}$. The reconstructed input at the output layer of the first autoencoder is the linear transformation of its hidden layer representation as MFCCs are real valued input. Thus the output reconstruction is given as

$$ \hat{h}_{k-1}(x_t) = W_k^T h_k(x_t) + b_{ok}, \quad k = 1, \quad (3.11) $$

and for the higher-level autoencoders, the transformation is non-linear,

$$ \hat{h}_{k-1}(x_t) = \sigma(W_k^T h_k(x_t) + b_{ok}), \quad k = 2, \ldots, K. \quad (3.12) $$

In (3.11) and (3.12), $b_{ok}$ is the bias vector of the autoencoder output layer. The objective function for all autoencoders is the mean squared error between the clean input and its reconstruction, i.e.,

$$ L_{dec}(x_t) = \|h_{k-1}(x_t) - h_{k-1}(x_t)\|_2^2. \quad (3.13) $$
Algorithm 3.1: DNA Pre-training

Input: Input data $x$, weights $W_k$, biases $b_{hk}, b_{ok}$ for layer $k = 1, 2, ..., K$ and learning rate $\epsilon$

Initialise: $W_k \in \mathcal{N}(0,0.1)$
Initialise: $b_{hk}, b_{ok} \in (0)$

For layer $k = 1, 2, ..., K$

Start: For iteration $I = 1, 2, ... N$

$h_k(x_t) = \sigma(W_k h_{k-1}(x_t) + b_k)$,

$h_{k-1}(x_t) = W_k^T h_k(x_t) + b_{ok}, k = 1$

and

$h_{k-1}(x_t) = \sigma(W_k^T h_k(x_t) + b_{ok}), k = 2, ..., K.$

$L_{dec}(x_t) = \|h_{k-1}(x_t) - h_{k-1}(x_t)\|_2^2.$

Update (see Appendix A):

$W_k \leftarrow W_k - \epsilon \frac{\partial L_{dec}(x_t)}{\partial W_k},$

$b_{hk} \leftarrow b_{hk} - \epsilon \frac{\partial L_{dec}(x_t)}{\partial b_{hk}}, b_{ok} \leftarrow b_{ok} - \epsilon \frac{\partial L_{dec}(x_t)}{\partial b_{ok}}.$

End

SGD is used to minimize (3.13). The training algorithm for denoising autoencoder in pre-training is given in Appendix A. The data corruption process is only confined to the pre-training. After that the hidden layers of the denoising autoencoders are stacked to create the deep network where the input layer receives the clean data $x$. The summary of DNA pre-training is given in Algorithm 3.1.

3.2.2 Regularised Contrastive Learning (Fine-tuning)

For contrastive learning, we minimise the loss function in (3.7) to update the weights and biases of our DNA in a global unsupervised-supervised training scenario using SGD. As the function (3.7) consists of two losses i.e., reconstruction loss (3.8) and contrastive loss (3.9), the gradients for these two losses can be found separately. Since the speech data are divided into
short segments of $T_B$ frames (MFCCs) and input to the DNA, the parameters cannot be updated unless the code layer pools the representations corresponding to the $T_B$ frames to measure the 1st and 2nd order statistics in (3.4). For the reconstruction loss in (3.8), the gradients are found with respect to the input frame at time $t$. So for the top layer $k = 2K$,

$$\frac{\partial L_R}{\partial u_{2K}(x_{it})} = -2(x_{it} - \hat{x}_{it}), i = 1,2. \quad (3.14)$$

For layers $k = 2K - 1, ..., 1$,

$$\frac{\partial L_R}{\partial u_k(x_{it})} = \left( \frac{\partial L_R}{\partial h_{kj}(x_{it})} h_{kj}(x_{it}) \left( 1 - h_{kj}(x_{it}) \right) \right)_{j=1}^{h_K}, \quad (3.15)$$

$$\frac{\partial L_R}{\partial h_{kj}(x_{it})} = \left[ W_{k+1}^{(i)} \right]^T \frac{\partial L_R}{\partial u_{k+1}(x_{it})}. \quad (3.16)$$

As the contrastive loss $L_D(X_1, X_2; \Theta)$ is only dependent on the neurons of $CS(X_i) = \{CS(x_{it})\}_{t=1}^{T_B}, i = 1,2$ in the code layer for both sub-nets, the gradients can be for the first $K$ layers. Therefore for $k = K$,

$$\frac{\partial L_D}{\partial u_k(x_{it})} = \left( \left( \psi_j(x_{it}) \left[ x - \lambda_m^{-1} (1 - x) e^{-D_m \lambda_m^{-1}} \right]_{j=1}^{CS} \right)_{j=1}^{CS} \right) + \left( \left( \xi_j(x_{it}) \left[ x - \lambda_s^{-1} (1 - x) e^{-D_s \lambda_s^{-1}} \right]_{j=1}^{CS} \right)_{j=1}^{CS} \right)$$

$$\left( 0 \right)_{j=CS+1}^{h_K} \quad (3.17)$$

where

$$\psi_j(x_{it}) = p_j^{(i)}(CS(x_{it})) \left[ 1 - CS(x_{it}) \right], \quad (3.18)$$

$$\xi_j(x_{it}) = q_j^{(i)}(x_{it})(CS(x_{it})) \left[ 1 - CS(x_{it}) \right]. \quad (3.19)$$

Here, $p^{(i)} = \frac{2}{T_B} \text{sign}(1.5 - i)(\mu^{(1)} - \mu^{(2)})$ and $q^{(i)}(x_{it}) = \frac{4}{T_B} - 1(\Sigma^{(1)} - \Sigma^{(2)})(CS(x_{it}) - \mu^{(i)})$ and $(CS(x_{it}))$ is the output of the $j^{th}$ neuron in $CS$ for $x_{it}$. The derivation of the derivative in (3.17) is given in Appendix B. For layers $k = K - 1, ..., 1$,
Thus for layer 7 as possible if the input examples are from two different speakers. speech segments by same speaker, while their difference should be as large way that they tend to get as similar as possible if the example are the representations produced by the architecture. Using the labelled example, similarity/dissimilarity in the statistical modalit ies of the output discriminated in a pair-wise contrastive learning by exploiting the The training of DNA is a multi-objective optimisation. The speakers are the parameters of the two sub-nets to be identical.

Now all the DNA parameters are updated using the gradients in (3.14-3.21) for the training example i.e., the labelled speech segment \((x_{1t}, x_{2t}, \mathcal{Z})^T\). Thus for layer \(k = K + 1, ..., 2K\), the weights and biases are updated as

\[
W_k^{(i)} \leftarrow W_k^{(i)} - \frac{\varepsilon}{T_B} \sum_{t=1}^{T_B} \sum_{r=1}^{2} \alpha \frac{\partial L_R}{\partial u_k(x_{rt})} [h_{k-1}(x_{rt})]^T,
\]

and for layer \(k = 1, ..., K\),

\[
W_k^{(i)} \leftarrow W_k^{(i)} - \frac{\varepsilon}{T_B} \sum_{t=1}^{T_B} \sum_{r=1}^{2} \left( \frac{\alpha \partial L_R}{\partial u_k(x_{rt})} + \frac{(1 - \alpha) \partial L_D}{\partial u_k(x_{rt})} \right) [h_{k-1}(x_{rt})]^T,
\]

\[
b_k^{(i)} \leftarrow b_k^{(i)} - \frac{\varepsilon}{T_B} \sum_{t=1}^{T_B} \sum_{r=1}^{2} \left( \frac{\alpha \partial L_R}{\partial u_k(x_{rt})} + \frac{(1 - \alpha) \partial L_D}{\partial u_k(x_{rt})} \right).
\]

The parameter \(\varepsilon\) in (3.22-3.25) is the learning rate. The two sub-nets in the DNA are linked together at layer \(k\) via contrastive loss \(L_D\), so the gradients with respect to \(L_D\) and \(L_R\) are added to update the weights and biases for the layers \(k = 1, ..., K\) as evident in (3.24-3.25). Eq. (3.22-3.25) guarantees the parameters of the two sub-nets to be identical.

The training of DNA is a multi-objective optimisation. The speakers are discriminated in a pair-wise contrastive learning by exploiting the similarity/dissimilarity in the statistical modalities of the output representations produced by the architecture. Using the labelled example, the 1\(^{st}\) and 2\(^{nd}\) order statistics of the representations are matched in such a way that they tend to get as similar as possible if the example are the speech segments by same speaker, while their difference should be as large as possible if the input examples are from two different speakers.
Algorithm 3.2: DNA Regularised Contrastive Learning (Fine-tuning)

**Input:** Input data \( \mathbf{x} \), weights \( W_k^{(i)} \), biases \( b_k^{(i)} \), for layer \( k = 1, 2, \ldots, K \), sub-net \( i = 1, 2 \) and learning rate \( \epsilon \).

**Initialise:** \( W_k^{(i)} \) from DNA Pretraining.

**Initialise:** \( b_k^{(i)} \) from DNA Pretraining.

**Start:** For iteration \( I = 1, 2, \ldots, N \)

\[
\mathbf{h}_k(\mathbf{x}_{it}) = \sigma \left( W_k^{(i)} \mathbf{h}_{k-1}(\mathbf{x}_{it}) + b_k^{(i)} \right), k = 1, 2, \ldots, 2K - 1,
\]

\[
\mathbf{h}_{2K}(\mathbf{x}_{it}) = W_k^{(i)} \mathbf{h}_{2k-1}(\mathbf{x}_{it}) + b_k^{(i)}.
\]

\[
L_R = \frac{1}{T_B} \sum_{t=1}^{T_B} ||\mathbf{x}_{it} - \hat{\mathbf{x}}_{it}||^2, i = 1, 2, \text{ from (3.8)},
\]

\[
L_D = \mathcal{X} D + (1 - \mathcal{X}) \left( e^{-\frac{\alpha_m}{\lambda_m}} + e^{-\frac{\alpha_s}{\lambda_s}} \right) \text{ from (3.9)}.
\]

**Update** (see Appendix B):

For \( k = K + 1, \ldots, 2K \),

\[
W_k^{(i)} \leftarrow W_k^{(i)} - \epsilon \frac{\partial L_R}{\partial W_k^{(i)}}, i = 1, 2,
\]

\[
b_k^{(i)} \leftarrow b_k^{(i)} - \epsilon \frac{\partial L_R}{\partial b_k^{(i)}}, i = 1, 2.
\]

For \( k = 1, \ldots, K \),

\[
W_k^{(i)} \leftarrow W_k^{(i)} - \epsilon \left[ \frac{\partial L_R}{\partial W_k^{(i)}} + \frac{\partial L_D}{\partial W_k^{(i)}} \right], i = 1, 2,
\]

\[
b_k^{(i)} \leftarrow b_k^{(i)} - \epsilon \left[ \frac{\partial L_R}{\partial b_k^{(i)}} + \frac{\partial L_D}{\partial b_k^{(i)}} \right], i = 1, 2.
\]

**End**

Furthermore minimising the reconstruction losses helps the system to discriminate only those components in the output representation that characterise the invariant SI while suppressing the non-SI. After training, we can visualise the effects of speaker separation in terms of the incompatibility learned by the DNA. The summary of regularised contrastive learning (fine-tuning) of DNA is given in Algorithm 3.2.
3.3 Visualisation

Vowels are believed to be the main carrier of SI and convey most of the information present in speech formants [43], [44]. Hence vowels are more likely to encode the SI by exploiting the structures of various organs that are speaker-specific. TIMIT is a phonetically transcribed speech dataset that contains 20 English vowels [73]. There are total of 630 speakers involved in TIMIT and each speaker recorded 10 different phonetically rich sentences. As all the vowels may be present in 10 different utterances of each speaker, up to 200 vowel segments in length of 0.1-0.5s each can be available for any speaker. This enables us to investigate the behaviour of DNA representations on a phonetic level for different speakers. Figure 3.3 is a two-dimensional visualisation for the SI and non-SI DNA representations i.e., $C^S$ and $\overline{C^S}$ respectively. Also the visualisation of MFCCs is presented. For dimensionality reduction, t-distributed Stochastic Neighbourhood Embedding (t-SNE) algorithm is used [74] which reflects intrinsic distribution manifold by projecting the data on 2-D plane. Each speaker’s data in Figure 3.3 is represented by a different colour. For better visualisation we used randomly selected five speakers from TIMIT. Each point in Figure 3.3 (a, c, e) is the 2-D projection of mean representation corresponding to specific vowel frames in an utterance. On the other hand each point in Figure 3.3 (b, d, f) is 2-D projection of the average of the rows of covariance matrices of the representation corresponding to a set of vowel frames. Despite the fact that some vowels are highly correlated, the $C^S$ representation for mean value are well clustered in terms of speakers, while these clusters are highly inter-mingled in the case of $\overline{C^S}$ and MFCCs where phonetically correlated vowels are co-located as they are shown in circles. Therefore $C^S$ representations are highly speaker-specific even on very short phonetic segments. In addition, representations
Figure 3.3: 2-D projection for mean and covariance of MFCCs, $\overline{CS}$ and $CS$ representations. Mean and covariance for (a, b) MFCCs (c, d) $\overline{CS}$ (e, f) $CS$.

For covariance values are less informative as compared to the mean values but there is still some cluster pattern for $CS$ representations. In contrast, MFCCs and $\overline{CS}$ fail to categorise speakers on just covariance. This behaviour is expected as the separation between the speakers is mainly
attributed towards mean, which specifies the separation between the speaker distributions. Covariance, which indicates the spread of distribution, is less illustrative on SI by knowing the fact that DNA representations and MFCCs show limited spread in overall data distribution. With this observation, we put more emphasis on mean values in our discriminative loss function (3.8) by tuning the parameters $\lambda_m$ and $\lambda_s$ i.e., the trade-off between the influence of mean and covariance.

### 3.4 Discussion

The proposed DNA is related to the alternative deep networks i.e., deep autoencoders [10], [50], [54] and deep Siamese network [12], [64], [65], yet the motivation is to learn the discrimination among the speakers using speaker-specific representation. It is shown empirically in the next two chapters that the representation produced by the DNA is indeed superior in capturing the SI.

Each sub-net of the DNA can be seen as a separate deep autoencoder that utilises the speech data irrespective of the identity (label) to minimise the reconstruction error. The overcomplete representation produced by either of the sub-net deep autoencoder is believed to distribute various speech information components at the output space. However the lack of learning the discrimination among those information components cannot exclusively sift out specific information from the output representation. The fact that SI is the minor information component conveyed by speech along with dominant LI and knowing that correlated LI also exhibits the statistical pattern (Figure 3.3), a deep autoencoder cannot deliver what is desired in a challenging SR task. In addition the environmental and other speaker dependent variabilities further worsen the representational capability of autoencoder.

DNA, if modified by removing all the layers above code layer, takes the
form of a typical deep Siamese network. With a strong contrastive learning using labelled data, it tries to maintain a separation among various speech information components irrespective of the fact that which component to focus on. There is a possibility to employ non speaker-specific components in a discrimination that could either result in useless output representation or overfitting to a particular dataset. Deep Siamese network has been successfully used in facial recognition and other vision tasks [12], [64], where the information of interest is dominant in the input space.

The DNA, on the other hand, is a regularised Siamese network where the autoencoding helps in distribution of the SI in the output representation and normalising the redundant information, while contrastive learning maintain a statistical separation among the speaker related information components. The splitting of the code layer to accommodate the non-speaker information proves to be beneficial in overall system performance. A separate part of the output representation is kept for a dedicated pairwise contrastive learning while the other half part normalises the redundancy through reconstruction constraint.

To sum up, the structure of DNA in terms of building blocks is inspired by the other architectures as in [12], [50], [51], [65] however, our motivation for speech component analysis results in a novel and specialised loss function design to train the DNA, which in turn leads to a totally different justification about its architectural hierarchy that is different from the other deep networks.

3.5 Chapter Summary

In this chapter, we have presented the detailed design of the proposed Deep Neural Architecture (DNA). We employed the architecture to extract speaker-specific features in a form of overcomplete representations. First, the classical two-stage training procedure for DNA is discussed. This
includes unsupervised layer-by-layer pre-training that utilises denoising autoencoder as a building block of the deep network. For supervised fine-tuning, a multi-objective loss function is designed which is based on contrastive losses and data reconstruction losses. The contrastive losses employ $1^{st}$ and $2^{nd}$ order statistics of the DNA representation to exploit the similarity/dissimilarity among the speakers. Furthermore, the data reconstruction losses are used as regularisation to suppress the non-SI components in speech and improve generalisation. Using MFCCs as input, a regularised discriminative pair-wise comparison of speakers is carried out to train the DNA. The high-level distributed representation acquired through DNA is judged on phonetic level speaker separation.
Chapter 4  Model Selection and Justification: Empirical Studies

This chapter explores the parametric space of the proposed DNA for optimum performance. The overview of speech corpora used in our research, pre-processing and speaker modelling for DNA representations is presented. Afterwards, the design of DNA is evaluated and justified in Speaker Comparison (SC) task for the range of parameters like network depth, width and regularisation terms in the loss function. Also the DNA is compared in SC against the two baseline deep networks, i.e., deep autoencoder network and deep Siamese network. The representations acquired through our trained architecture show outperforming results as compared to MFCCs and representations yielded by a deep autoencoder and deep Siamese network. While evaluating the parametric space, certain issues are discussed regarding the DNA design, specialised for speech information processing.

4.1  Speech Corpora

To justify the DNA structure, loss function and different parameter settings, various speaker comparison performances are presented using the representations acquired by training the DNA with a wide range of parameters. Although different in motivation, DNA is somehow related to the two alternative deep networks i.e., deep autoassociator (DAE) and deep Siamese network (DSN). Therefore comparison of the representations
produced by these deep nets with the DNA is also reported in SC using different speech corpora.

The robustness of features in SR can be judged on various conditions in which the speech has been recorded. Therefore, for the critical evaluation of speech features, the speech dataset should cover various factors of variation that pose a challenge in recognising the speakers in any SR task. These factors of variation include noise, channel effects through which the speech signal is passed, effects of recording equipments, session variability due to the time difference in recording sessions and emotional state of the speaker. Other factors include the difference of language in training and testing speech for SR system.

In this research, nine different speech corpora have been used in the detailed evaluation of the DNA representation. Out of eight, four benchmark LDC corpora are used in this chapter for parametric evaluation of DNA. The salient features of various speech corpora used in this research are overviewed as follows:

**TIMIT**

TIMIT is wide-band clean speech English dataset especially designed for acoustic-phonetic studies [73]. There are total of 630 speakers (438 male and 192 female) involved in recording phonetically rich sentences. Each speaker recorded 10 different speech clips of approximately 2-2.5s each. There are no session gaps in between the recordings of the speaker. The dataset is collected from eight different regions of USA with different dialects.

**Network TIMIT (NTIMIT)**

NTIMIT is a narrow-band noisy dataset that is created by passing all the TIMIT speech for all speakers through 10 long-distance fixed line
telephonic channels [73]. Therefore this dataset involves noise distortion and channel variability.

Wide-band KING (WKING)

This is also a wide-band English dataset [73]. A total of 51 speakers are involved in recording in 10 different sessions. Each utterance is approximately 27s in duration. This dataset exhibits session variability that ranges from weeks to months. The variability is more prominent in the last five sessions as compared to the first five sessions. All speakers in this dataset are male.

Narrow-band KING (NKING)

Like NTIMIT, NKING is created by passing all the speech of WKING through fixed line telephonic channel and recording it back. The speech in NKING is corrupted by the noise with varying intensity and also by the adverse channel effects. There is another anomaly in this dataset called Great Divide, which is the degradation of speech in the last five sessions due to the change in recording equipment. This is one of the most challenging datasets created for SR [73].

Chinese Language Dataset (CHN)

This dataset involves 59 speakers. Each speaker recorded 10 different utterances in three sessions. This is a wide-band clean speech dataset [76].

MASC

This is also a Chinese language wide-band clean speech dataset [78]. This dataset is especially created to judge the performance of SR systems under speaker emotional variability. Total number of speakers involved is 68 (23 female and 45 male). Each speaker has recordings in five different emotional states i.e., neutral, anger, elation, panic and sadness. In each
emotional state, each speaker recorded 15 short phrases of about 1s and 60 utterances of about 2.5-3s each. In addition, in neutral emotion, each speaker recorded two long paragraph readings of about 30-40s each.

**VAM-Audio**

This is a German language dataset that is collected from a TV talk show. It involves 47 speakers (11 males and 36 females). This is a wide-band clean speech dataset that is also emotionally variable. There are total 947 short sentences of variable lengths (2-10s on average), which are categorised in three emotional states, i.e., valance, activation and dominance [77].

**A Russian Language Dataset (RUS)**

This is also wide-band clean speech dataset that involve 50 speakers (21 males and 29 females). There is one long utterance of approximately 3-4 minutes for each speaker [75].

**Artificially Generated Datasets**

TIMIT, NITIMIT and CHN datasets have been used in this research to artificially generate new datasets for the experimental evaluation of DNA. One dataset is created artificially by corrupting the TIMIT to see the effect of environmental variability. In addition, three datasets are generated by concatenating the speech of TIMIT, NTIMIT and CHN to mimic audio conversations in SS tasks.

Table 4.1 summarizes the salient features of the above-mentioned speech corpora.

**4.2 Pre-Processing**

We use Mel-frequency cepstral coefficients (MFCCs) as raw speech representation to input DNA. MFCCs are widely used representation in numerous speech related tasks. They compactly encode most of the
<table>
<thead>
<tr>
<th>Corpus</th>
<th>No. Of Speakers</th>
<th>Bandwidth</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>630</td>
<td>0-8 kHz</td>
<td>------</td>
</tr>
<tr>
<td>NTIMIT</td>
<td>630</td>
<td>0.3-3.3 kHz</td>
<td>Noise, Channel</td>
</tr>
<tr>
<td>WKING</td>
<td>51</td>
<td>0-4 kHz</td>
<td>Session</td>
</tr>
<tr>
<td>NKING</td>
<td>51</td>
<td>0.3-3.3 kHz</td>
<td>Session, Noise,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Channel</td>
</tr>
<tr>
<td>CHN</td>
<td>59</td>
<td>0-8 kHz</td>
<td>Session</td>
</tr>
<tr>
<td>MASC</td>
<td>68</td>
<td>0-4 kHz</td>
<td>Emotional</td>
</tr>
<tr>
<td>VAM-Audio</td>
<td>47</td>
<td>0-4 kHz</td>
<td>Emotional</td>
</tr>
<tr>
<td>RUS</td>
<td>50</td>
<td>0-4 kHz</td>
<td>------</td>
</tr>
</tbody>
</table>

Table 4.1: Information on various speech corpora.

information components in speech [27]. To extract MFCCs from the speech signal, a standard procedure is followed [27], [28], which involves the following steps; (i) silence removal from speech using an energy-based method, (ii) pre-emphasis the speech signal using filter \( H(z) = 1 - 0.96z^{-1} \), (iii) divide the speech into frames of 20 ms using Hamming windows with an overlap of 10 ms, (iv) calculate the magnitude spectrum of each frame with short-term Fourier transform (STFT) using 24 Mel-scale triangular filters, (v) extract MFCCs for each frame by applying cosine transform on the magnitude spectrum. We use 20 dimensional MFCCs excluding the 0th coefficient.

Cepstrums, especially Mel-frequency cepstrums are one of the most commonly used forms of speech signal representation. They effectively and compactly encode various information in speech including LI and SI hence, used in both speech and speaker recognition [26], [27], [28]. In addition to this, MFCCs also characterise the human auditory system. These claims can be justified by studying the structural components of MFCCs that are closely related to human speech production and perception systems. The SI is directly dependent on the physiology of human vocal tract and cavity system. Therefore, it is always useful to use speech parameterisation that depends on them. The speech signal is the outcome of vocal fold excitation signal \( x[n] \) (that contains another speaker-specific feature, i.e., fundamental frequency \( f_0 \)) convolved with vocal tract filter response \( Y(z) = X(z)V(z) \).
For SI, it is desirable to separate out the vocal tract response \( v[n] \) from \( x[n] \) [122]. When the spectrum \( Y(z) \) is mapped on logarithmic scale, its components become additive due to the property of logarithm, i.e.,

\[
\log_{10}(X(z)V(z)) = \log_{10}X(z) + \log_{10}V(z).
\]

Afterwards it is easy to sift out the components using filtering. This is achieved in MFCCs using log-spectral magnitudes. These spectral magnitudes are then mapped on Mel-frequency scale using series of triangular bandpass filters. Each filter represents a band of frequencies hence, results in dimensionality compaction on frequency scale. These filters are linearly arranged below 1 kHz and logarithmically arranged above 1 kHz. This nonlinear frequency mapping mimics the perceptual behaviour of human ear, i.e.,

\[
m(f) = 2595 \log_{10} \left( 1 + \frac{f}{700} \right),
\]

where \( f \) is the linear frequency and \( m \) is Mel-scaled frequency. The cepstral coefficients are created by applying the discrete cosine transform (DCT) to the log-spectral magnitudes. For \( M \) frequency channels (bandpass filters) and log-magnitude \( a_1 \) to \( a_M \), the cepstral coefficients \( c_k \) are given as

\[
c_k = \sum_{m=1}^{M} a_m \cos \left( k \frac{\pi}{M} (m - 0.5) \right), k = 1, ..., M
\]

The DCT rearranges the spectral bins in order of higher energy, representing strong formants in speech. As a result, the first few coefficients represent about 95% of whole spectral energy and therefore enriched with SI [122]. This compaction makes MFCCs favourable speech features in terms of fewer burdens on computation for pattern classification. This property together with the perceptual inspiration from human auditory system makes MFCCs more attractive as compared to other forms of cepstral coefficient [123]. Equation (4.2) implies that as \( k \) increases, \( c_k \) captures fine spectral details i.e., first, overall spectral shape, general formants and then more detailed spectral structure between the formants [122]. Higher values of \( k \) can capture most of the excitation (wide-
band signal). Therefore, first few coefficients compactly represent the vocal tract behaviour (narrow-band signal) and the phonetic information, whereas later coefficients can also include information about excitation signal. This is the main reason that for speech recognition, where phonetic information is important, only the first few MFCCs are used, whereas for speaker recognition, it is desirable to use more coefficients as, in addition to speaker-specific vocal tract information, the spectrum at higher frequencies may also convey important speaker dependent characteristics.

4.3 Speaker Modelling

After pre-processing, a set of MFCC frames representing a speech segment \(\{\mathbf{x}_t\}_{t=1}^{T_M}\) with \(T_M\) frames is applied to the trained DNA as input. The corresponding output \(\{CS(\mathbf{x}_t)\}_{t=1}^{T_M}\) is used to create a speaker model \(S_M\) with the mean and covariance matrix, i.e., \(S_M = \{\mu, \Sigma\}\), where \(\mu = \frac{1}{T_M} \sum_{t=1}^{T_M} CS(\mathbf{x}_t)\) and \(\Sigma = \frac{1}{T_M-1} \sum_{t=1}^{T_M} [CS(\mathbf{x}_t) - \mu][CS(\mathbf{x}_t) - \mu]^T\).

Speaker comparison (SC) is a process to measure the discrimination between two speech segments that may belong to the same or two different speakers. Quantitatively, the discrimination is measured using some discriminant function or distance metric. We use symmetric Gaussian log-likelihood to measure the compatibility between two speaker models [81].

Consider a speech segment representation \(\{CS(\mathbf{x}_t^{ts})\}_{t=1}^{T_N}\) for a registered speaker with a model \(S_M = \{\mu^{tr}, \Sigma^{tr}\}\). The likelihood that a test utterance vector \(CS(\mathbf{x}_t^{ts}) \in \{CS(\mathbf{x}_t^{ts})\}_{t=1}^{T_N}\) belongs to the speaker model \(S_M\) is given as

\[
l(\text{CS}(\mathbf{x}_t^{ts}), S_M) = \frac{1}{2\pi |\Sigma^{tr}|^{\frac{3}{2}}} e^{-\frac{1}{2} (\text{CS}(\mathbf{x}_t^{ts}) - \mu^{tr})(\Sigma^{tr})^{-1}(\text{CS}(\mathbf{x}_t^{ts}) - \mu^{tr})^T}. \tag{4.3}
\]

The average negative log-likelihood for the \(T_N\) test utterance vectors is

\[
\mathcal{L}(\text{CS}(\mathbf{x}_t^{ts}), S_M) = -\frac{1}{T_N} \sum_{t=1}^{T_N} \log l(\text{CS}(\mathbf{x}_t^{ts}), S_M). \tag{4.4}
\]

Similarly the average negative log-likelihood that a reference speech
segment belongs to test speaker model \( S_N = \{ \mu^{ts}, \Sigma^{ts} \} \) is given as

\[
\mathcal{L}(CS(x_i^{tr}), S_N) = \frac{1}{T_M} \sum_{t=1}^{T_M} \log[l(CS(x_i^{tr}), S_N)].
\] (4.5)

The symmetric likelihood outcome is given as

\[
\mathcal{L}_s(S_N, S_M) = \frac{T_M \mathcal{L}(CS(x_i^{ts}), S_M) + T_N \mathcal{L}(CS(x_i^{tr}), S_N)}{T_M + T_N}.
\] (4.6)

The symmetric version of the Gaussian likelihood (4.6) is useful especially when the utterances involved in speaker registration or testing are of short duration [81].

### 4.4 DNA Training Protocol

In this chapter, the experiments on SC are reported for four LDC benchmark datasets i.e., TIMIT, NTIMIT, WKING and NKING.

For TIMIT, we randomly selected 600 speakers and for each speaker, we concatenated the first five utterances to create a longer speech segment and include that in the *training set*. In the similar way, the *testing set* is created using the last five utterances of the same 600 speakers. From the total of 630 speakers, all the speech of the remaining 30 speakers is used in *validation set* to monitor the training. For each speaker in the testing set, multiple test segments are generated using the procedure in [28]. The set of MFCC frames representing a full test utterance are divided into overlapping segments and each segment comprises of \( T_N \) frames.

\[
\overline{X_1, X_2, \ldots, X_{T_N}, X_{T_N+1}, \ldots, \overline{X_1, X_2, \ldots, X_{T_N}, X_{T_N+1}, \ldots,}
\]

In the similar way the training, testing and validation sets are created for NTIMIT. For WKING and NKING datasets, the utterances from session-1 and session-2 for all speakers were involved in training set. Session-3 was reserved for validation set while all the remaining sessions were used to
create testing set. Like TIMIT and NTIMIT, the test segments for each speaker are created using the frame-by-frame overlapping window.

For the performance evaluation we used Detection Error Trade-off (DET) curves to observe the errors in the complete span of false alarm and miss detection probability [79].

For the training of DNA, all the utterances in the training set of all four datasets are concatenated in the form of MFCC frames and applied as input for the unsupervised pre-training on each dataset. For the fine-tuning, we generated the genuine and imposter pairs by the exhaustive combination of the MFCC frames. For genuine pairs, the set of MFCC frames corresponding to a speech segment for a speaker in the training set is split into numerous smaller non-overlapping chunks and pairs are created using inter-chunk combinations. On the other hand, to create imposter pairs, the chunks are taken from the MFCC vectors corresponding to the speech segments from two different speakers. There is no repetition of any genuine and imposter pair in the training set. For TIMIT and NTIMIT, there are 10,000 genuine and 10,000 imposter pairs for each dataset, while there are 25,000 genuine and 25,000 imposter pairs in the case of WKING and NKING datasets. In the training, the number of genuine and imposter pairs was kept equal to avoid a biased learning towards one class. Batch style SGD is used in both pre-training and fine-tuning. In fine-tuning, the batch size is kept equal to the chunk size in the genuine and imposter pairs. To be exact, each batch in pre-training is a set of MFCC vectors equivalent to 1s of the speech while each batch in fine-tuning corresponds to 5s of speech. The larger batch size in fine-tuning is necessary to keep the covariance matrix non-singular in (3.4).

4.5 Statistical Significance

We employ z-score testing [124] to validate the claim that the proposed
<table>
<thead>
<tr>
<th>Corpus</th>
<th>No. of speakers in test set</th>
<th>No. of true trials</th>
<th>No. of imposter trials</th>
<th>Total no. of trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>600</td>
<td>60,000</td>
<td>1,797,000</td>
<td>1,857,000</td>
</tr>
<tr>
<td>NTIMIT</td>
<td>600</td>
<td>60,000</td>
<td>1,797,000</td>
<td>1,857,000</td>
</tr>
<tr>
<td>WKING</td>
<td>51</td>
<td>5,100</td>
<td>12,750</td>
<td>17,850</td>
</tr>
<tr>
<td>NKING</td>
<td>51</td>
<td>5,100</td>
<td>12,750</td>
<td>17,850</td>
</tr>
</tbody>
</table>

Table 4.2: Information on various speech corpora for sample size in SC evaluation.

DNA is indeed statistically significantly better than the other methods. In the next Sections, the SC performance of various DNAs trained with different parameter values and comparison with various features are presented on four speech corpora i.e., TIMIT, NTIMIT, WKING and NKING. According to the experimental settings of Section 4.4, the number of trials for each corpus is summarised in Table 4.2. For TIMIT and NTIMIT, each speaker contributes 100 speech segments for true tests, so total number of true tests involved in an experiment is (100 × 600 = 60,000). Similarly there are 100 speech segments reserved for each speaker for true tests in WKING and NKING (100 × 51 = 51,000). On the other hand, each speaker contributes 5 speech segments for imposter tests in all four corpora therefore, total number of imposter tests are (5 × 599 × 600 = 1,797,000) for TIMIT/NTIMIT and (5 × 50 × 51 = 12,750) for WKING/NKING respectively.

The z-score is calculated over equal error rate (EER) for each method in an experiment. The standard deviation of the sample estimate of the difference is given in [124], [131] as

$$
\sigma_d = \sqrt{\left(\frac{\text{EER}_1 (1 - \text{EER}_1)}{4} + \frac{\text{EER}_2 (1 - \text{EER}_2)}{4}\right) \left(\frac{1}{n_1} + \frac{1}{n_2}\right).}
$$

(4.7)

The z-score is given as

$$
z = \frac{d}{\sigma_d},
$$

(4.8)

Where, $d = |\text{EER}_1 - \text{EER}_2|$ with EER$_1$ and EER$_2$ are EERs produced by two methods to be compared. The variables $n_1$ and $n_2$ are the total number of
true and imposter trials respectively used to evaluate two methods to achieve EER (in our case the values are same for a given corpora). The value of \(z\) is checked against standard normal probability table to calculate the confidence interval, i.e., the confidence that a given result is statistically significantly different from the other.

### 4.6 Evaluation of DNA with Speaker Comparison

Although there is a huge parametric space to explore for the optimum performance of a given deep architecture through experimental evaluation, we investigate different structural parameters of DNA that include the depth and width of the network and the dimension of the code layer. There is no established study to have prior knowledge about the settings of these parameters and it is believed that such settings are dependent on the application and dataset [10], [50].

#### 4.6.1 DNA Depth

The depth of a network implies the number of hidden layers \(K\) involved in creating a deep network. On the other hand the width of a network is related to the number of neurons in each hidden layer. For DNA, we investigate for \(K = 3, 4, 5\) while keeping the width of the network constant. In other words the dimension or the number of neurons in each hidden layer is kept 100 i.e., \(|h_k| = 100, k = 12, \ldots, K - 1\). The dimension of the code layer is kept 200, which is equally divided into \(CS\) and \(\overline{CS}\), i.e., 100 neurons for each, SI and non-SI representations. So \(|CS| = |\overline{CS}| = 100\).

Figure 4.1 shows the DET plots for SC performance by the DNA of various depths. The performance is judged on four LDC datasets. To create speaker models, the reference utterance duration in all datasets is 5s while the test utterance duration is also 5s. For all datasets the reference utterances are taken from the training sets. The discriminant function of (4.6) is used to
Table 4.3: Statistical significance (z-score and % confidence interval) on SC results for DNAs trained with $K = 3, 4, 5$.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA $K = 3$</th>
<th>DNA $K = 4$</th>
<th>DNA $K = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>(NAN) 0.00%</td>
<td>(NAN) 0.00%</td>
<td>(NAN) 0.00%</td>
</tr>
<tr>
<td>NTIMIT</td>
<td>(33.17) 100%</td>
<td>(6.75) 100%</td>
<td>(26.48) 100%</td>
</tr>
<tr>
<td>WKING</td>
<td>(6.36) 100%</td>
<td>(3.88) 100%</td>
<td>(2.47) 99%</td>
</tr>
<tr>
<td>NKING</td>
<td>(7.03) 100%</td>
<td>(0.60) 45%</td>
<td>(6.43) 100%</td>
</tr>
</tbody>
</table>

measure the speaker compatibility. For all the DNAs the training and testing protocols are same. From Figure 4.1, it is evident that the depth $K = 4$ is more appropriate in SR tasks as the DNA with $K = 4$ captures the SI more effectively as compared to the DNA with $K = 3$ which is relatively shallow to encode the characteristic speaker information. Also DNA with
Figure 4.2: SC performance for DNA width $|h_k| = 50, 100, 200$. DET curves on (a) TIMIT (b) NTIMIT (c) WKING (d) NKING.

| Corpus    | DNA $|h_k| = 50$           | DNA $|h_k| = 100$           | DNA $|h_k| = 200$           |
|-----------|-------------------------|-------------------------|-------------------------|
| TIMIT     | (NAN) 0.00%             | (NAN) 0.00%             | (NAN) 0.00%             |
| NTIMIT    | (44.77) 100%            | (7.79) 100%             | (7.80) 100%             |
| WKING     | (9.30) 100%             | (1.90) 94%              | (9.50) 100%             |
| NKING     | (10.13) 100%            | (0.43) 33%              | (10.56) 100%            |

Table 4.4: Statistical significance (z-score and % confidence interval) on SC results for DNAs trained with $|h_k| = 50, 100, 200$.

$K = 5$ performs slightly worse than DNA with $K = 4$ which is consistent with the observation in [24] that increasing the number of layer beyond certain point may not improve the performance of the network due to the difficulty in optimising the tuneable parameters. From this observation, we can conclude that DNA with four layers is better than the other
DNAs in all datasets expect clean speech TIMIT, where all the three DNAs produce no errors. Table 4.3 shows the confidence intervals and z-score for SC results on four corpora. For each dataset in the table, there are two rows, the first row compared the DNA with $K = 3$ with $K = 4$ and $K = 5$ while second row compares DNA with $K = 4$ with $K = 5$. The z-scores are shown in parenthesis and NAN stands for “Not A Number”, i.e., the case when all the DNAs produce same EERs. It is evident that DNA trained with $K = 4$ is significantly superior as compared to the DNAs trained with $K = 3$ and $K = 5$.

4.6.2 DNA Width

Next, the similar SC experiments are reported for different dimensions of hidden layers $|h_k|$, $k = 1, 2, ..., K − 1$ by keeping the network depth $K = 4$ and the dimension of the code layer $|CS| = |C̅S| = 100$ as before. Figure 4.2 shows the optimum performance for three different DNAs with $|h_k| = 50, 100, 200$. The DNA with $|h_k| = 100$ seems the better option in SC as it provides enough neurons in each layer for the best optimisation solution. On one hand, increasing number of neurons increases the number of parameters while on the other hand, it also increases the redundancy with the extra neurons that fail to capture any useful information. Also the DNA with $|h_k| = 50$ fails to perform better due to inadequate network width. Table 4.4 tabulates the confidence interval and z-scores for SC results by DNAs trained with $|h_k| = 50, 100, 200$.

4.6.3 DNA Code Layer Dimension

Finally, the effect of the dimension of the code layer is also investigated in SC experiments with $|h_k| = 100, 200, 300$. In other words the dimension of $|CS|$ and $|C̅S|$ is 50, 100 and 150 each time. The network depth is kept at $K = 4$ and network width is taken to be $|h_k| = 100$ for $k = 1, 2, ..., K − 1$. 
Figure 4.3: SC performance on code layer dimension $|h_K| = 100, 200, 300$. DET curves on (a) TIMIT (b) NTIMIT (c) WKING and (d) NKING.

| Corpus   | DNA $|h_K| = 100$ | DNA $|h_K| = 200$ | DNA $|h_K| = 300$ |
|----------|----------------|----------------|----------------|
| TIMIT    |                | (NAN) 0.00%    | (NAN) 0.00%    |
|          | DNA $|h_K| = 200$    | (NAN) 0.00%    | (NAN) 0.00%    |
| NTIMIT   | DNA $|h_K| = 100$    | (48.93) 100%   | (14.76) 100%   |
|          | DNA $|h_K| = 200$    | (34.35) 100%   |                |
| WKING    | DNA $|h_K| = 100$    | (4.65) 100%    | (1.23) 78%     |
|          | DNA $|h_K| = 200$    | (5.88) 100%    | (6.37) 100%    |
| NKING    | DNA $|h_K| = 100$    | (7.24) 100%    | (0.87) 62%     |
|          | DNA $|h_K| = 200$    | (5.71) 100%    | (0.86) 62%     |

Table 4.5: Statistical significance ($z$-score and % confidence interval) on SC results for DNAs trained with $|h_K| = 100, 200, 300$.

Figure 4.3 shows the performance corresponding to the different code layer dimension with $|h_K| = 200$ being the most suitable. The code layer with $|h_K| = 100$ is insufficient to distribute or characterise the SI especially in the presence of interference like noise and channel distortion in NTIMIT.
and NKING. Code layer with $|\mathbf{h}_K| = 300$ adds the redundancy to the SI features which degrades the performance.

In all the above experiments, the early stopping criteria using the validation set is applied to get the optimum performance by various DNAs.

Our other experimental evaluation involves varying dimensions of hidden layers and $|\overline{C^S}|$. It is observed that keeping the dimension of the hidden layer the same provides better performance. This effect is also observed in [24]. Similarly it is found advantageous to take the dimension of $|\overline{C^S}|$ to be equal to $|C^S|$. However there is no solid evidence to justify this phenomenon except the hypothesis that fewer number of neurons in $|\overline{C^S}|$ may fail to segregate the non-SI components from the SI which leads to relatively worse performance. In contrast if the dimension of $|\overline{C^S}|$ is greater than $|C^S|$, no change in performance is observed. Table 4.5 shows the confidence intervals and z-scores for the SC results produced by DNAs trained with $|\mathbf{h}_K| = 100, 200, 300$.

### 4.6.4 DNA Loss Function Parameters

The loss in (3.11-3.13) has two tuneable parameters, i.e., the regularisation constant $\alpha$ and the tolerance bounds $\lambda_m$ and $\lambda_s$ on the compatibility measure $D$. The parameter $\alpha$ is the trade off factor between the reconstruction losses $L_R$ and the contrastive loss $L_D$. Therefore it regulates the influence of unsupervised and supervised criteria. The selection of parameter $\alpha$ is very critical in extracting the SI. Choosing lower value of $\alpha$ drives the DNA towards deep autoencoder while higher value makes DNA resembles with deep Siamese network. Using cross-validation and monitoring the training and validation error $\alpha = 0.2$ is selected for the best performance. Figure 4.4 shows the SC performance by DNA for $\alpha = 0.01, 0.2, 0.3$. Statistical significance in terms of confidence intervals on z-
Figure 4.4: SC performance for $\alpha = 0.01, 0.2, 0.3$. DET curves on (a) TIMIT (b) NTIMIT (c) WKING (d) NKING.

Table 4.6: Statistical significance ($z$-score and % confidence interval) on SC results for DNAs trained with $\alpha = 0.01, 0.2, 0.3$.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA $\alpha = 0.01$</th>
<th>DNA $\alpha = 0.20$</th>
<th>DNA $\alpha = 0.30$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>(NAN) 0.00%</td>
<td>(NAN) 0.00%</td>
<td>(NAN) 0.00%</td>
</tr>
<tr>
<td>NTIMIT</td>
<td>DNA $\alpha = 0.01$</td>
<td>(61.28) 100%</td>
<td>(24.58) 100%</td>
</tr>
<tr>
<td></td>
<td>DNA $\alpha = 0.20$</td>
<td>(37.07) 100%</td>
<td>(37.07) 100%</td>
</tr>
<tr>
<td>WKING</td>
<td>DNA $\alpha = 0.01$</td>
<td>(11.69) 100%</td>
<td>(3.65) 100%</td>
</tr>
<tr>
<td></td>
<td>DNA $\alpha = 0.20$</td>
<td>(8.05) 100%</td>
<td>(8.05) 100%</td>
</tr>
<tr>
<td>NKING</td>
<td>DNA $\alpha = 0.01$</td>
<td>(12.32) 100%</td>
<td>(0.93) 65%</td>
</tr>
<tr>
<td></td>
<td>DNA $\alpha = 0.20$</td>
<td>(11.33) 100%</td>
<td>(11.33) 100%</td>
</tr>
</tbody>
</table>

scores for this case is given in Table 4.6. As discussed before, two main information components of speech are LI and SI. LI, which is in fact lexical or spoken text information, dominates the speech information space. Therefore its interference with the SI, which is a minor information component, is a major hindrance in the information inter-component.
separability. The purpose of employing unsupervised learning to increase the generalisation capability of the DNA as it helps in extracting the invariant features in speech. In a text-independent SR task, the only thing purely invariant among various utterances of a particular speaker is the SI. So ideally the unsupervised learning should have a greater influence to capture the SI, as is the case with several vision tasks [12], [65], [50], [53], where the main invariant information is held by the object of interest in the image. However, despite the high variability in the text of speech, LI is highly correlated. The voiced phonemes that are the building blocks of the perceivable speech show high correlation and mostly occupy the spectrum of speech at lower frequencies [14], [17]. This high correlation may have adverse effects on DNA representation in the case of strong unsupervised learning. Therefore the output features may have more influence of LI than SI. Also other factors like intra-speaker variability and environmental effects also influence the SR but such variability is mostly random in nature and can be subsided by unsupervised learning that is responsible for good generalisation.

Strong supervised constraint on the other hand, fails to suppress the non-speaker related information or force the system towards overfitting on a particular dataset or environment, especially when the labelled data is limited. Therefore a careful selection of $\alpha$ is critical in training the DNA. The other parameters include the bounds $\lambda_m$ and $\lambda_s$ on the mean and covariance driven compatibility measure (3.13). We choose $\lambda_m = 100$ and $\lambda_s = 2.5$ to balance the effect of mean and covariance on the loss for imposter pair with $\mathcal{I} = 0$. Since $\mathcal{C}(X_i) \in [0,1]$, the negative exponential of $\mathcal{D}_s$, which involve higher order covariance, vanishes during training. This could results in a biased learning towards the mean value in $\mathcal{D}_m$ in the case of imposter pairs.
4.7 Feature Comparison

In this section, SC experiments are reported using the representations acquired through deep architectures. These include deep autoencoder networks (DAE), deep Siamese networks (DSN) and DNA. In addition, performances on two other features are also presented which include MFCCs, pitch contours and representation by DNA when it is trained with random parameter initialisation. Training with random initialisation does not employ any pre-training, instead, the DNA is training is in fact only fine-tuning stage with all the network parameters randomly initialised. DAE and DSN networks are trained using standard two-stage training (pre-training and fine-tuning). Like DNA, DAE and DSN networks are trained for each of the four datasets i.e., TIMIT, NTIMIT, WKING and NKING. Each DAE is trained by pooling all the data included in the training set of each dataset. The same pooled data is used in both pre-training and fine-tuning. The training protocol for DSN network is similar to that of DNA in both training stages. SR using pitch contours of speech utterances is a classical approach for explicit speaker-specific feature extraction [80]. The method involves short-term speech analysis where the speech signal is divided into small overlapping frames using windowing. Typically, each frame is 20-30ms in duration with an overlap of 10ms. Pitch period for each frame is calculated via autocorrelation method [41]. The pitch contour is created by combining all the pitch periods for a speech signal. Afterwards, PCA and LDA are applied to create a feature vector for each speech clip. The details are given in [80]. For TIMIT and NTIMIT we create the training/reference feature vectors using the first five utterances of each speaker. The remaining five utterances are used to extract test feature vectors. Similarly for WKING and NKING, the training feature vectors for each speaker are created using session-1 and session-2, while all
Figure 4.5: SC performance on MFCCs, deep autoencoder (DAE), deep Siamese network (DSN), DNA with random initialisation (DNA-R), Pitch Contour (PitchCon) and DNA. DET curves on (a) TIMIT (b) NTIMIT (c) WKING (d) NKING.

the other sessions are used to generate test set. The SC is carried out by comparing the reference and test feature vectors in Euclidean space as given in [80]. Figure 4.5 shows the SC performances for the representation yielded by DNA, DAE and DSN networks. The results for MFCCs and pitch contours are also included. The representation acquired through DNA clearly outperforms all the other representations and features in all the four datasets. The poor performance of DAE and DSN are due to similar reasons as were discussed in Section 4.6.4. On the other hand the unsatisfactory performance of randomly initialised DNA strongly highlights the importance of unsupervised pre-training in achieving better solution to a complex problem of training a deep network. It is also evident
Table 4.7: Statistical significance (z-score and % confidence interval) on SC results for various features.

evident from Figure 4.5 that pitch contours fail to characterise SI, hence result in poor performance. Pitch along with several other features can be extracted from the speech signal explicitly, as discussed in detail in
Chapter 2, may convey some speaker-dependent information but in general, none of such features are universal in representing the SI. These features or their combination have been used in several SR attempts but the lack of generality and computational load make them relatively less feasible for SR tasks as compared to spectral representation of speech [17], [18], [23]. Table 4.7 shows the confidence intervals and $z$-scores for SC results produced with DNA and various features.

### 4.8 Chapter Summary

This chapter presents the empirical evaluation of various DNA parameters. The high-level distributed representation acquired through optimum DNA is judged on speaker comparison (SC) task using four LDC benchmark speech corpora, i.e., TIMT, NTIMIT, WKING and NKING. In addition, the salient features of various speech corpora used in this research are overviewed. The optimum parametric space in DNA design is investigated through several experiments on SC. Similarly, various representations yielded by DNA, deep autoencoder and deep Siamese network are compared in SC. From the thorough experimental evaluation on base-level SR, it is evident that SI representation from DNA outperforms the representations produced by other deep networks and also MFCCs. It can be concluded that DNA representation are robust against different environmental variability like, noise and channel effects and capable of characterising speakers even at phonetic levels with extremely short utterances. This makes the proposed architecture favourable in conducting more challenging SR tasks. In the next chapter, comprehensive experimentation in speaker verification and speaker segmentation is presented on various speech corpora in more difficult SR scenarios. This will include the comparison of DNA representation with the state-of-the-art SR techniques.
Chapter 5  Speaker Verification

This chapter covers the detailed experimental evaluation of DNA representations in an important SR task i.e., Speaker Verification (SV). Although in the previous chapter, various empirical studies have been made to justify our proposed architecture through several SC experiments, SV can scrutinise the SR performance of DNA via different yet more challenging test protocols. This includes cross-corpus and cross-language testing with various environmental and speaker variability and also investigating the effect of short duration utterances on the performance. Furthermore a comprehensive comparison with the state-of-the-art techniques in SV is presented. The chapter ends with the discussion on the outcome of the experiments.

5.1  Speaker Verification

There are numerous measures and techniques that have been proposed for modern biometric systems. Usually, these include fingerprints, facial and retina scans. Speech has also been used as an alternative individual’s identity as it provides an audio signature or voice print of speaker. Despite the advantages and disadvantages of every biometric measure, there are few applications e.g., online telephonic transactions and talker recognition for forensic purposes, where effectiveness, robustness and accuracy of SR system is detrimental to compromise. As compared to other SR tasks, SV has been more a focus of attention in real world practical application.
In an automatic system, SV is a process of verifying a claim of an unknown speaker about a speaker who is already known to the machine. The known speaker is also called enrolled, registered or target speaker. Based on the methodology of verification, SV can be categorised into two forms. One is text-dependent SV, where the unknown is prompted to speak a particular word or phrase. The other is text-independent SV, where the speaker is classified regardless of the text spoken. The later being the more challenging task and a step forward in machine automation in recognising speaker based on their unconstrained natural utterances.

The general scheme of SV system is given in Figure 5.1. As shown, the SV system has two distinct phases i.e., training or enrolling phase, the other is testing phase. For enrolling, speech signal of a speaker is pre-processed for a speech parameterisation. This step is in fact responsible for speech representation with the parameters that are speaker related. This is however a common step in every SR system as discussed in earlier chapters. Therefore speech parameterisation is actually acquiring spectral or non-spectral speech representation. The next step after this is speaker modelling. Based on the extraction of various speaker-dependent statistical characteristics from speech representation, this step assigns a dedicated
model to a speaker. Speaker modelling may involve sophisticated ML algorithms that learn and measure the compatibility between the speakers based on the parameters/representations of the speaker’s speech. The state-of-the-art systems utilise statistical modelling, as statistics better characterise the SI [1], [27].

The testing phase compares the speech representation of an unknown speaker with the claimed registered model to make a decision. Before decision, the speech representation is acquired in a similar way as is done in enrolling phase of SV.

5.1.1 Speech Representation

Typically for any SR system, time domain speech signal is transformed into feature vectors using some spectral or non-spectral signal analysis. The resulting speech representation should be compact and suitable to provide statistical information about the speaker [4]. Speech spectrograms and cepstral features are among the popular choices for speech representation. As mentioned earlier, MFCCs are the most commonly used speech features in both speech and speaker recognition as they efficiently and compactly present most of the information required in speech. The procedure involved in calculating MFCCs is already described in Section 4.2.

DNA is a parametric model that is taken as an alternative speech parameterisation model to extract a novel representation from the baseline MFCCs. The new representation characterises the SI more effectively in the output feature space. So applying MFCCs as input to the DNA to get a new representation falls under the category of feature extraction. The new DNA representations are used to create speaker-specific model. Therefore using (3.1) and (3.2), for an MFCC feature vector $x_t$, the DNA representation is given as

$$CS(x_t) = h_K(x_t) = \sigma(W_K h_{K-1}(x_t) + b_K).$$

(5.1)
where $W_K$ and $b_K$ are the weight and biases for the output layer $K$. The speaker model is created with 1st and 2nd order statistics of $\{CS(x_i)\}_{t=1}^{T_M}$ corresponding to $T_M$ MFCC frames as given in Section 4.3.

CDBN is a convolutional extension of the conventional DBN that utilises RBMs as the building blocks of deep network [67]. Although in general, CDBN can have multiple hidden layers, the original presented in [48] for several audio classification task is a two-layered network. Working on the principal of DBN to estimate high-level inference about the probability distribution of the data, CDBN also manages to extract the pattern based on the spatial correlation of the data. Figure 5.2 shows the CDBN with two hidden layer and one max-pooling layer. Using (2.8-2.10) for a given input data vector $x$ (not necessary MFCCs), the hidden layer representation of the $m^{th}$ feature map in convolution RBM (CRBM) is given by the relation

$$p(h_{kj}^m = 1|x) = \left(\sigma(W_{1j}^m * _x + b_{1j}^m)\right)^{h_{kj}}_{j=1}$$

(5.2)

and the corresponding input reconstruction is

$$p(x_i = 1|h) = \left(\sigma\left(\sum_m (W_{1i}^m * h^m_i) + b_{1i}^m\right)\right)^{|x|}_{i=1}.$$  

(5.3)

For real valued inputs (5.3) becomes

$$p(x_i|h) = \left(\mathcal{N}\left(\sum_m (W_{1i}^m * h^m_i) + b_{1i}^m\right), 1\right)^{|x|}_{i=1}.$$ 

(5.4)
In the above equation (*v*) and (*r*) are the valid and full convolution respectively. The representation (5.2) is highly overcomplete, therefore an additional max-pooling layer is used to pick the neuron with maximum activation to be the output of fixed neuron neighbourhood in a feature map [48]. This step not only keeps the high dimensional representation sparse but also helps in avoiding the spurious information in the data. This is similar to the sub-sampling layer in Figure 2.8 for CNN. After max-pooling, the next level CRBM uses the pooled representation of the first hidden layer as input to repeat the process of (2.18) and (2.19) with \(W^m_2\) and \(b^m_2\) as weight and biases for the second layer, hence getting a higher level inference about the data distribution.

CDBN has been used in various speech and speaker related tasks including phone recognition, speaker identification, gender and music classification [48]. In this chapter, the performance of the CDBN representations are compared in SV. As in [67], spectrograms of speech signals are used to train CDBN in a purely unsupervised way. The CDBN training resembles with the pre-training stage of DBN and DNA. The representations yielded by CDBN are used to create speaker model based on the 1\(^{st}\) and 2\(^{nd}\) order statistics as is done for creating speaker models in DNA.

### 5.1.2 Speaker Modelling

After feature extraction, speaker models are created to register a speaker in a SV system database. A speaker model is a parsimonious collection of modalities that represent the intrinsic information about the speaker. These modalities directly or indirectly depend on the statistics of the speech utterances of a particular speaker. Modern SV systems mostly objectify these statistics with either mono-Gaussian or mixture of Gaussian modelling [81], [27].
Mono-Gaussian Model (MGM)

This speaker model is a statistical model with an assumption about the speech data to be Gaussian with single state [81]. MGM is not only easy to implement and computationally efficient, it can well represent SI. The given speech feature vectors or representations can be summarised by the mean vector and the covariance matrix as discussed in Section 4.3. For our SV experiments MFCCs, DNA and CDBN representations are modelled through MGMs. The scores are calculated using the symmetric log likelihood (4.6) that is based on MGMs for enrolling/reference and test speech.

Gaussian Mixture Model (GMM)

GMM is a parametric generative model that represents a probability density function of a data as a weighted sum of several mono-Gaussian densities. Each density component of GMM is characterised by the mean and covariance. Using the combination of mono-Gaussians, GMM can estimate any density function with arbitrary precision [27]. It is believed that speech spectral features hold semi-Gaussian distribution, therefore can be modelled accurately with GMMs. Each Gaussian component can be considered to model broad phonetic sound for a particular speaker. The phonemes like vowels, nasals and fricatives carry speaker-dependent information, which in turn dependent on a speaker-specific configuration of vocal apparatus. Therefore each phonetic class can be individually modelled by mono-Gaussian parameters $\mu$ (mean) and $\Sigma$ (covariance). Hence the overall speaker-dependent distribution is the weighted collection of individual phonetic Gaussians. Mathematically a speaker model by GMM is given as $S_M = \{w_j, \mu_j, \Sigma_j\}_{j=1}^{M_g}$. The likelihood that the unknown speech data belongs to the speaker with model $S_M$ is given by (2.3) and (2.4). Model parameters for GMM are learned through Expectation
Maximization (EM) algorithm, which is iterative in nature [27], [28]. In each iteration, the likelihood that the observation (speaker data for enrolment) belongs to the estimated model is increased i.e., for training data \( X^{tr} = \{x_{t}^{tr}\}_{t=1}^{T_{M}} \) and iteration \( n \), \( p(X^{tr} | S_{M}^{n}) \geq p(X^{tr} | S_{M}^{n-1}) \).

GMM proves to be state-of-the-art system in SR [27], [28], [1]. However one main disadvantage of GMM learned through EM is that it requires large amount speaker’s data to estimate the model. This limitation bars GMM in some important online SR tasks, where the amount of training data is limited. In fact in some situations, MGM models can better model the statistics of a speaker [81]. The remedy to this problem is a variant of GMM called GMM-MAP, where the speaker model is estimated using Bayesian adaption or maximum a posteriori (MAP) estimation [82] using a large background GMM called universal background model (UBM). For UBM a large speaker-independent GMM is trained via EM algorithm by pooling all the available data of a large speaker population. The number of Gaussian components \( M_{g} \) in UBM is usually of higher order e.g., 512 to 2048. The UBM model is denoted as \( S_{UBM} = \{w_{j}^{UBM}, \mu_{j}^{UBM}, \Sigma_{j}^{UBM}\}_{j=1}^{M_{g}} \). A speaker model is created through MAP adaptation from UBM by aligning the speaker data with the UBM density components as

\[
\text{Pr}(j|x_{t}^{tr}) = \frac{w_{j}^{UBM} g_{j}^{UBM}(x_{t}^{tr})}{\sum_{i=1}^{M_{g}} w_{i}^{UBM} g_{i}^{UBM}(x_{t}^{tr})},
\]

Now the statistics for weights, mean and variance is given as

\[
\eta_{j} = \left( \sum_{t=1}^{T_{M}} \text{Pr}(j|x_{t}^{tr}) \right)^{M_{g}},
\]

\[
E_{j}(x) = \left( \frac{1}{\eta_{j}} \sum_{t=1}^{T_{M}} \text{Pr}(j|x_{t}^{tr}) x_{t}^{tr} \right)^{M_{g}},
\]
The parameters for new speaker model are calculated by updating by UBM parameters as

\[
\hat{\omega}_j = \left( \left( a_j \frac{\eta_j}{T_M} + (1 - a_j) \omega_{jUBM} \right) \right)^{M_g},
\]
(5.9)

\[
\hat{\mu}_j = \left( a_j \mu(x) + (1 - a_j) \mu_{jUBM} \right)^{M_g},
\]
(5.10)

\[
\hat{\sigma}_j^2 = a_j \mu(x) + (1 - a_j) \left( \sigma_{UBM}^2 + \mu_{jUBM}^2 \right) - \mu_{jUBM}^2.
\]
(5.11)

The scalar \( \gamma \) ensures that \( \sum_{j=1}^{M_g} \hat{\omega}_j = 1 \), while \( a_j = \frac{\eta_j}{\eta_j + r} \) balances the new and old estimates with \( r \), the relevance factor. The variance \( \sigma_j^2 \) constitute the diagonal covariance matrix. The adapted speaker model is \( S_M = \{ \hat{\omega}_j, \hat{\mu}_j, \hat{\sigma}_j \}_{j=1}^{M_g} \). After model creation for all registered speakers in SV system, the likelihood score for a test utterance \( X^{ts} = (x_t^{ts})_{t=1}^{T_N} \) is calculated using (2.3) and (2.4)

\[
L_{UBM}(X^{ts}) = \log_{10} p(X^{ts} | S_M) - \log_{10} p(X^{ts} | S_{UBM}).
\]
(5.12)

In the SV experiments reported in this chapter the performance of CDBN and GMM-UBM is compared with DNA on eight different speech corpora reported in Chapter 4.

### 5.2 Experimental Settings

The experiments are divided into two phases with respect to the testing protocols (i) intra-corpus testing (ii) inter-corpus testing. In intra-corpus four LDC corpora are used for SV i.e., TIMIT, NTIMIT, WKING and N KING. DNA, CDBN and GMM-UBM are trained on each of the four datasets separately and the testing is also done on the four datasets separately. For example if training is done on TIMIT, testing only includes TIMIT dataset and so on. The inter-corpus tests include training DNA,
CDBN, and GMM-UBM on different datasets, while testing is done on totally different datasets. For training, TIMIT, NTIMIT and VAM-audio datasets are reserved while WKING, NKING, CHN, MASC and RUS datasets are kept for testing. Inter-corpus testing judges the performance of the three techniques to produce stable and generic results when the training and testing is done on totally different conditions. We are interested especially to monitor the capability of DNA to extract generic SI, irrespective of the fact that testing is done on datasets with severe environmental, session and emotional variability, the condition unseen by DNA in training (see Table 4.1). Inter-corpus testing also indirectly includes the cross-language testing, as some datasets are involved in training and testing are different in terms of spoken language. The training and testing condition for the two test scenarios is now discussed.

5.2.1 Training and Testing Protocol

In intra-corpus SV, experimental settings for DNA is similar to the SC experiments in Chapter 4, i.e., training and test sets are created in the same manner for all four LDC datasets by extracting the MFCCs. For training GMM-UBM, the first five utterances of each of 600 speakers in TIMIT are used to create a training set. UBM is trained by pooling all the data in the training set. For speaker modelling, the first five utterances of each speaker are combined to adapt the speaker models. The test set is created by using the remaining five utterances of each speaker. In the similar way, the training and testing sets are created for NTIMIT. For WKING and NKING, the utterances of all the speakers in first two sessions are used to train UBMs while speaker models are adapted using Session-3 of each dataset. The test sets are created using all the utterances in Session 4 to 10. CDBN is purely unsupervised training setup, so the training is done using
the same setting as is done for pre-training the DNA for all four datasets. The test sets are similar to those used for DNA and GMM-UBM.

For inter-corpus SV, three separate DNAs are trained with three different training sets. For first DNA i.e., DNA-1 all the clean speech that constitutes all the ten utterances by each of the 600 randomly selected speakers is used to pre-train the DNA. Out of 600 speakers, 100 speakers are randomly selected to generate the training set for fine-tuning. All the utterances of the remaining 30 speakers out of the population of 630 are used to create validation set to monitor the performance of DNA during training. Second DNA (DNA-2) is trained using TIMIT and artificially generated dataset from TIMIT dubbed, corruptTIMIT. In corruptTIMIT, the utterances of all the speakers involved in training set of the DNA-1 are corrupted using Additive White Gaussian Noise (AWGN) channel with the SNR of 10dB and Rayleigh fading channel with 5 Hz Doppler shift [93]. The pre-training of DNA-2 is similar to that of DNA-1 except now two datasets are involved with all the utterances of 600 speakers each. For fine-tuning the DNA-2, the training set is created by randomly selecting 50 speakers from clean TIMIT and 50 speakers corruptTIMIT from the pool of 600 speakers for each dataset. Similarly the validation set for DNA-2 has remaining 60 (30+30) speakers for both datasets. The DNA-3 utilises NTIMIT and VAM-audio as two additional datasets in training. For pre-training the DNA-3, all the utterances by 600 randomly selected speakers from each of TIMIT, corruptTIMIT and NTIMIT are used, while all the utterances from all 45 speakers in VAM-audio are used. Fine-tuning involves all the utterances of 50 speakers from each of the TIMIT, corruptTIMIT and NTIMIT datasets and all the utterances of 30 randomly selected speakers from VAM-audio. So there are 170 speakers involved in fine-tuning. The validation set includes 30 speakers from each of TIMIT, corruptTIMIT and NTIMIT and 17 speakers from VAM-audio. Therefore the validation set covers the total
of 107 speakers. For the three DNAs, the testing is done on WKING, NKING, CHN, RUS and MASC datasets. For WKING and NKING, the test sets are created by taking the utterances of all 51 speakers from Session 2 to 10. Session 1, for both WKING and NKING, is reserved to generate speaker models. Test set for CHN is created using the last five utterances of each of 59 speakers, while the first five utterances of each speaker kept for speaker modelling. RUS dataset contains 50 speakers, where each speaker recorded one long sentence of about 3-4 minutes. Each utterance is divided into two parts to create the speaker models and test set. In MASC dataset, the speaker models are created using the paragraph portion of natural recording section, while testing is performed separately for each emotional state by creating multiple testing sets, each for each emotional state.

The three separately trained DNA are used in testing to judge the effect of incrementally increasing the training data with different variation. The corrupt TIMIT and NTIMIT increase the environmental variability in the training data by degrading the speech with noise and channel effects. Also VAM-audio introduces the emotional variability to the training data.

The next round of tests includes the comparison of best performing DNA (out of DNA-1, DNA-2 and DNA-3) with GMM-UBM and CDBN. For inter-corpus/inter-language tests, CDBN is trained with all the speech data of TIMIT, corrupt TIMIT, NTIMIT and VAM-audio. On the other hand three UBMs are created with each one trained by using all the data from TIMIT, NTIMIT and VAM-audio. The UBN trained on TIMIT, which is a wide-band clean speech dataset, is used in testing of WKING, CHN and RUS, which are also wide-band clean speech datasets. The UBM trained on NTIMIT (narrow-band) is used to test NKING (narrow-band) dataset, while for testing on MASC (emotional), UBM trained on VAM-audio (emotional) is used. The reason behind using three different UBMs instead of one is to avoid the spectral mismatch between training and testing by
GMM-UBM, as failing to do so deteriorate the GMM-UBM performance. All the data in TIMIT, NTIMIT and VAM-audio is used to train the three UBMs. The speaker modelling and test protocols for DNA, CDBN and GMM-UBM are same.

5.2.2 Parameter Settings and Pre-processing

In all the SV experiments, like SC in chapter 4, MFCCs are used as basic input features to train the DNA. The trained DNA is applied on MFCCs to produce the DNA representations that are used to create models and test speech representations. Similarly MFCCs are used to train and test the GMM-UBM. To train the CDBN, however, the speech pre-processing step involves calculating the high dimensional speech spectrograms. The spectrograms are further processed with PCA whitening to reduce the data dimension [48]. Therefore the speaker modelling and testing is also done with the CDBN representations produced by applying spectrogram PCA as input features to the trained the CDBN.

In the GMM-UBM, NKING dataset has a special case of extra pre-processing. Since the dataset is noisy with adverse channel and session variability, Cepstral Mean Subtraction (CMS) is used to neutralise the linear channel effects due to noise and mismatch in spectral properties of training and testing speech in different sessions. CMS is not used in other datasets, as they don’t have such variability. Doing so may reduce the performance [26].

In all experiments, speaker models for TIMIT, NTIMIT, CHN and RUS are created using reference speech of 5s duration while it is 10s for WKING, NKING and MASC. The testing is always done with 5s utterance duration in intra-corpus SV, while it varies in inter-corpus SV. For MASC, tests are performed using natural utterance lengths that range from 2s to 3s. Table 5.1 summarises the settings for utterance duration in SV experiments. The
Table 5.1: Enrolment and test utterance lengths.

<table>
<thead>
<tr>
<th>SV Phase</th>
<th>TIMIT</th>
<th>NTIMIT</th>
<th>WKing</th>
<th>NKING</th>
<th>CHN</th>
<th>RUS</th>
<th>MASC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrolment</td>
<td>5s</td>
<td>5s</td>
<td>10s</td>
<td>10s</td>
<td>5s</td>
<td>5s</td>
<td>10s</td>
</tr>
<tr>
<td>Testing</td>
<td>5s</td>
<td>5s</td>
<td>1-5s</td>
<td>1-5s</td>
<td>5s</td>
<td>5s</td>
<td>Actual/5s</td>
</tr>
</tbody>
</table>

DNA is trained using the same parameter settings as were selected in SC experiments for optimum performance i.e., the network depth $K = 4$, the dimension of hidden layers $|h_k| = 100, k = 1, 2, ..., K$, the dimension of code layer is 200 with $|cS| = |cS| = 100$. The regularisation parameter $\alpha = 0.2$ in (3.7) while the bounds $\lambda_m = 100$ and $\lambda_s = 2.5$ in (3.9).

For GMM-UBM, the background models UBMs are trained with 2048 Gaussian components using EM algorithm with 50 iterations. Speaker models are adapted using MAP. The relevance factor $r = 16$ is used. Also in testing, the test feature vectors are aligned with the top $c$ Gaussian components of the speaker model $s_M$ and UBM model $s_{UBM}$ [82]. These $c$ Gaussian components produce the best likelihood scores. In SV experiments $c = 5$ is used. For model adaption, only mean is adapted as suggested in [82].

For training CDBN, we strictly followed the experimental settings suggested in [48]. However, few parameters are further tunes for the best performance. These include the learning rate of 0.01, the sparsity parameter of 0.02 and the sparsity regularisation constant of 0.1. The CDBN is architecture with two hidden layers. We investigated the performance of first hidden layer output, second hidden layer output or their combination as the final representation. The output of the first hidden layer is used as the final representation in our SV experiments as it produces the best performance. This is consistent with the original findings in [48]. Table 5.2 tabulates the parameter values for DNA, GMM-UBM and CDBN. The threshold range values to get DET curve in the case of MFCCs is similar to that of DNA, CDBN and GMM-UBM. The symmetric log likelihood
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DNA</strong></td>
<td></td>
</tr>
<tr>
<td>Network depth, $K$</td>
<td>4</td>
</tr>
<tr>
<td>Hidden layer dimension $</td>
<td>h_4</td>
</tr>
<tr>
<td>Code layer dimension $</td>
<td>C</td>
</tr>
<tr>
<td>Regularisation constant in loss function, $\alpha$</td>
<td>0.2</td>
</tr>
<tr>
<td>Upper bounds in loss function, $\lambda_m, \lambda_v$</td>
<td>100, 2.5</td>
</tr>
<tr>
<td>Threshold range for DET curve in SV</td>
<td>0 : 0.002 : 20</td>
</tr>
<tr>
<td><strong>GMM-UBM</strong></td>
<td></td>
</tr>
<tr>
<td>No. of Gaussians per mixture</td>
<td>2048</td>
</tr>
<tr>
<td>Relevance factor, $r$</td>
<td>16</td>
</tr>
<tr>
<td>No. of top Gaussians, $C$</td>
<td>5</td>
</tr>
<tr>
<td>No of EM iterations</td>
<td>50</td>
</tr>
<tr>
<td>Threshold range for DET curve in SV</td>
<td>0 : 0.002 : 20</td>
</tr>
<tr>
<td><strong>CDBN</strong></td>
<td></td>
</tr>
<tr>
<td>Sparsity parameter</td>
<td>0.02</td>
</tr>
<tr>
<td>Sparsity regularisation constant</td>
<td>0.1</td>
</tr>
<tr>
<td>Threshold range for DET curve in SV</td>
<td>0 : 0.002 : 20</td>
</tr>
</tbody>
</table>

**Table 5.2:** Parameter values for DNA, GMM-UBM and CDBN used in SV.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>No. of speakers in test set</th>
<th>No. of true trials</th>
<th>No. of imposter trials</th>
<th>Total no. of trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>600</td>
<td>60,000</td>
<td>1,797,000</td>
<td>1,857,000</td>
</tr>
<tr>
<td>NTIMIT</td>
<td>600</td>
<td>60,000</td>
<td>1,797,000</td>
<td>1,857,000</td>
</tr>
<tr>
<td>WKING</td>
<td>51</td>
<td>5,100</td>
<td>12,750</td>
<td>17,850</td>
</tr>
<tr>
<td>NKING</td>
<td>51</td>
<td>5,100</td>
<td>12,750</td>
<td>17,850</td>
</tr>
<tr>
<td>CHN</td>
<td>59</td>
<td>5,900</td>
<td>17,110</td>
<td>23,010</td>
</tr>
<tr>
<td>RUS</td>
<td>50</td>
<td>5,000</td>
<td>12,250</td>
<td>17,250</td>
</tr>
<tr>
<td>MASC</td>
<td>68</td>
<td>4,080</td>
<td>22,780</td>
<td>26,860</td>
</tr>
<tr>
<td>MASC*</td>
<td>68</td>
<td>6,800</td>
<td>22,780</td>
<td>29,580</td>
</tr>
</tbody>
</table>

**Table 5.3:** Information on various speech corpora for sample size in SV evaluation.

The statistical significance of the SV results is reported using $z$-score confidence interval as given in Section 4.5. Table 5.3 tabulates the number of trials for each corpus in SV experiments. In Table 5.3, MASC* indicates the experiments on MASC dataset where test utterance of 5s for each speaker is used in SV while MASC means the trials with natural test
utterance lengths. The later case has fewer true trials.

5.3 Experiments for Speaker Verification

5.3.1 Intra-Corpus Speaker Verification

As mentioned in the previous section, intra-corpus SV results are reported for TIMIT, NTIMIT, WKING and NKING data sets. Three models are compared i.e., DNA, CDBN and GMM. In addition, the performance of MFCCs is also presented on the same test settings using the discriminant function (4.6) as a score measure. Figure 5.3 shows the SV performance of all the four models on intra-corpus SV, while Table 5.4 lists the statistical significance using z-score confidence interval. It is evident from the results that DNA representations outperform CDBN representation for all datasets and GMM-UBM in TIMIT and NTIMIT. For TIMIT, MFCCs and DNA representations produce zero error. In contrast, GMM-UBM produces best results for WKING and comparable with DNA for NKING dataset. This gives a clue that GMM-UBM better adapts to the condition of WKING and NKING datasets by avoiding session and channel variability. The results for WKING and NKING are the average performance on the Sessions 4 to 10 as first two sessions are used in training and session 3 is used in speaker modelling. This rule prevails even for MFCCs i.e., the speaker models for MFCCs are created from Session 3, while Sessions 4 to 10 are used in testing. In (5.12) the likelihood with respect to the background model acts as a background score normalisation [82]. This helps in better performance for the datasets with higher environmental variability by reducing the variation in the test scores. However, for clean speech dataset like TIMIT, this reduces the performance due to averaging effect of background normalisation. This is the reason for the worse performance of GMM-UBM on TIMIT. CDBN representation on the other hand, fails to show good
Figure 5.3: Intra-corpus SV. DET curves for SV on (a) TIMIT (b) NTIMIT (c) WKING and (d) NKING.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>MFCCs</th>
<th>GMM-UBM</th>
<th>CDBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT</td>
<td>DNA (NAN) 0.00% (53.11) 100% (140) 100%</td>
<td>(140) 100%</td>
<td>(110) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs (53.11) 100%</td>
<td>(53.11) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTIMIT</td>
<td>DNA (15.91) 100% (33) 100% (30.95) 100%</td>
<td>(420) 100%</td>
<td>(354) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs (30.95) 100%</td>
<td>(33) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WKING</td>
<td>DNA (16.01) 100% (7.74) 100% (23.46) 100%</td>
<td>(38.89) 100%</td>
<td>(23.27) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs (23.46) 100%</td>
<td>(7.74) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NKING</td>
<td>DNA (13.57) 100% (1.43) 85% (12.14) 100%</td>
<td>(45.69) 100%</td>
<td>(31.67) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs (12.14) 100%</td>
<td>(1.43) 85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Statistical significance (z-score and % confidence interval) on intra-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN.

performance on any dataset because of the poor discriminative ability for speakers.
5.3.2 Inter-Corpus Speaker Verification

Inter-corpus SV, which also include inter-language experiments, is divided into two parts. In first part, three DNAs (DNA-1, DNA2 and DNA-3) trained incrementally on different datasets are investigated for the effects of increasing data variation in the training. Figure 5.4 illustrates the performance of DNAs for four datasets i.e., WKING, NKING, CHN and RUS. Figure 5.5 is the similar comparison for MASC data for five different emotions. In almost all cases, DNA-3 shows better performance and is statistically significantly better than DNA-1 and DNA-2.

From all results, it is evident that increasing the variability in the training provides the DNA chance to learn to extract SI by suppressing the harmful redundancies like noise and channel degradation effects on speech. DNA-3, which is trained on four datasets with environmental and speaker emotional variability has more filtering power to extract SI enriched representation. Clearly this effect is more visible on test datasets with high variability like WKING, NKING and MASC. This concept is similar to the idea of self-taught learning [83] except the fact that the data with increasing variability is used in both unsupervised (pre-training) and regularised contrastive learning (fine-tuning), while it is mainly discussed as a benefit in unsupervised learning [83]. Figure 5.5 also shows the influence of individual emotional state on SV, i.e., the highest variability is observed in panic, elation and anger state as shown by the relatively worse results. The performance of DNA trained on clean speech TIMIT is similar to the DNA trained on TIMIT + corruptTIMIT. However the inclusion of NTIMIT, which contains natural narrow-band degraded speech and emotional dataset VAM-audio significantly increase the performance on all the datasets. For MASC dataset in Figure 5.5, actual test utterance lengths of speakers are used. Table 5.5 and 5.6 show statistical significance for the
Figure 5.4: SV performance of three DNAs trained on TIMIT (DNA-1), TIMIT + corruptTIMIT (DNA-2) and TIMIT + corruptTIMIT + NTIMIT + VAM-audio (DNA-3). DET curves for SV on (a) WKING (b) NKING (c) CHN and (d) RUS.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA-3</th>
<th>DNA-1</th>
<th>DNA-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKING</td>
<td>(9.11) 100%</td>
<td>(1.62) 93%</td>
<td></td>
</tr>
<tr>
<td>NKING</td>
<td>(10.87) 100%</td>
<td>(1.44) 85%</td>
<td></td>
</tr>
<tr>
<td>CHN</td>
<td>(2.24) 97%</td>
<td>(2.12) 97%</td>
<td></td>
</tr>
<tr>
<td>RUS</td>
<td>(7.64) 100%</td>
<td>(1.71) 91%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Statistical significance (z-score and % confidence interval) on SV results for DNAs incrementally trained on various corpora. Evaluation is done on WKING, NKING, CHN and RUS corpora.

the results of Figure 5.4 and 5.5 respectively.

In the next set of experiments, only DNA-3, simply called DNA is used. These experiments detail the inter-corpus SV comparison between DNA,
Figure 5.5: SV performance by DNA-1, DNA-2 and DNA-3 on MASC with five different emotional states (a) anger (b) elation (c) neutral (d) panic and (e) sadness.

CDBN, GMM-UBM and MFCCs. From Figure 5.6, it is evident that DNA representations, in general, outperform the MFCCs, CDBN and GMM-UBM in every test scenario and are significantly better than MFCCs, GMM-UBM and CDBN in general (Table 5.7). The performance of DNA
Table 5.6: Statistical significance (z-score and % confidence interval) on SV results for DNAs incrementally trained on various corpora. Evaluation is done on MASC corpus with five emotional states.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA-3</th>
<th>DNA-1</th>
<th>DNA-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASC (Anger)</td>
<td>(13.35) 100%</td>
<td>(8.95) 100%</td>
<td>(4.39) 100%</td>
</tr>
<tr>
<td>MASC (Elation)</td>
<td>(10.02) 100%</td>
<td>(11.40) 100%</td>
<td>(1.62) 89%</td>
</tr>
<tr>
<td>MASC (Neutral)</td>
<td>(7.37) 100%</td>
<td>(9.36) 100%</td>
<td>(1.99) 95%</td>
</tr>
<tr>
<td>MASC (Panic)</td>
<td>(12.01) 100%</td>
<td>(11.75) 100%</td>
<td>(0.26) 21%</td>
</tr>
<tr>
<td>MASC (Sadness)</td>
<td>(6.24) 100%</td>
<td>(7.73) 100%</td>
<td>(1.49) 86%</td>
</tr>
</tbody>
</table>

Figure 5.6: Inter-corpus SV. DET curves for SV on (a) WKING (b) NKING (c) CHN and (d) RUS datasets.

representations and MFCCs are similar in CHN dataset with the %EER of 0.6 and 0.64 respectively. It can be concluded that DNA representations successfully characterise SI, which is responsible for the stable performance.
Table 5.7: Statistical significance (z-score and % confidence interval) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN. Evaluation is done on WKING, NKING, CHN and RUS corpora.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA</th>
<th>MFCCs</th>
<th>GMM-UBM</th>
<th>CDBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKING</td>
<td></td>
<td>(11.08) 100%</td>
<td>(20.15) 100%</td>
<td>(75.89) 100%</td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>(9.16) 100%</td>
<td>(64.73) 100%</td>
<td>(55.25) 100%</td>
</tr>
<tr>
<td>NKING</td>
<td>DNA</td>
<td>(10.78) 100%</td>
<td>(7.44) 100%</td>
<td>(45.60) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs</td>
<td>(3.32) 100%</td>
<td>(34.43) 100%</td>
<td>(37.87) 100%</td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHN</td>
<td>DNA</td>
<td>(0.48) 37%</td>
<td>(11.78) 100%</td>
<td>(30.35) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs</td>
<td>(11.36) 100%</td>
<td>(30.05) 100%</td>
<td>(20.95) 100%</td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUS</td>
<td>DNA</td>
<td>(10.70) 100%</td>
<td>(27.51) 100%</td>
<td>(32.89) 100%</td>
</tr>
<tr>
<td></td>
<td>MFCCs</td>
<td>(21.45) 100%</td>
<td>(27.56) 100%</td>
<td>(7.13) 100%</td>
</tr>
<tr>
<td></td>
<td>GMM-UBM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

in both intra-corpus and inter-corpus SV. Also the results in both cases are nearly equal. The equal error rates (%EER) produced in intra-corpus SV are 6.50% and 18.63% for WKING and NKING respectively. The %EER for inter-corpus SV on the same datasets are 7.20% and 19.06% respectively. Despite the fact that the dataset used in training and testing are different in terms of language, speaker and environmental variability, the good generalisation depicted by DNA representations is attributed towards generic SI extraction. In contrast GMM-UBM fails to produce as promising results as was the case with intra-corpus SV. This clearly shows the overfitting of GMM-UBM to a WKING and NKING datasets and their environments. CDBN on the other hand again fails to show any better performance. CDBN representations have been used to produce favourable results in various speech relate tasks [48]. However, such representations are generic speech representations rather than speaker-specific as the same representations have been used in tasks like phoneme recognition and speaker identification. The performance criteria for both tasks depend on LI and SI that are two important yet totally different speech information components. Figure 5.7 shows the inter-corpus SV results on MASC*
Figure 5.7: Inter-corpus SV on MASC* dataset on five emotional states (a) anger (b) elation (c) neutral (d) panic and (e) sadness.

dataset. Here we take the test utterance duration of 5s for each speaker by concatenating their speech signals. In these results, GMM-UBM is more stable as compared to MFCCs that are more influenced by the spectral variation due to the change in speaker’s emotional state. Hence it is hard to capture the speaker sensitive distribution pattern using a simple modelling
Table 5.8: Statistical significance (z-score and % confidence interval) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN. Evaluation is done on MASC* corpus.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA</th>
<th>MFCCs</th>
<th>GMM-UBM</th>
<th>CDBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASC* (Anger)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>MFCCs</td>
<td>GMM-UBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(37.14) 100%</td>
<td>(3.44) 100%</td>
<td>(65.69) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(33.66) 100%</td>
<td>(27.23) 100%</td>
<td>(62.03) 100%</td>
<td></td>
</tr>
<tr>
<td>MASC* (Elation)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>MFCCs</td>
<td>GMM-UBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(38.37) 100%</td>
<td>(6.12) 100%</td>
<td>(62.81) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(32.16) 100%</td>
<td>(23.53) 100%</td>
<td>(56.34) 100%</td>
<td></td>
</tr>
<tr>
<td>MASC* (Neutral)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>MFCCs</td>
<td>GMM-UBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(15.73) 100%</td>
<td>(8.97) 100%</td>
<td>(121) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(6.96) 100%</td>
<td>(108) 100%</td>
<td>(114) 100%</td>
<td></td>
</tr>
<tr>
<td>MASC* (Panic)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>MFCCs</td>
<td>GMM-UBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(30.89) 100%</td>
<td>(4.63) 100%</td>
<td>(54.71) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(26.20) 100%</td>
<td>(23.01) 100%</td>
<td>(49.85) 100%</td>
<td></td>
</tr>
<tr>
<td>MASC* (Sadness)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DNA</td>
<td>MFCCs</td>
<td>GMM-UBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(14.67) 100%</td>
<td>(6.91) 100%</td>
<td>(81.62) 100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(7.77) 100%</td>
<td>(65.96) 100%</td>
<td>(74.24) 100%</td>
<td></td>
</tr>
</tbody>
</table>

technique like MGMs. In contrast, the speaker specific representations produced by DNA are generic and robust against emotion variability, as ideally, the intrinsic SI should remain invariant. The performance of SV system is greatly influenced by test utterance duration as it is hard to capture the sufficient statistics in speech for shorter utterances. In our experiments, especially inter-corpus SV, WKING and NKING prove to be the most challenging datasets marked by severe environmental corruption and inter-session variability. Figure 5.8 shows the SV results on WKING and NKING for 3s and 1s test utterance duration respectively. This gives us a chance to judge the performance of DNA representations on even more difficult test scenario. For WKING and NKING datasets, DNA representations beat all the other models. GMM-UBM performance is even worse than MFCCs for 3s test utterance duration. However, it is better than MFCCs and CDBN for 1s test utterance duration and also better in NKING for both cases of test utterance duration. Table 5.9 shows the statistical significance of the results presented in Figure 5.8. Overall it can be seen
Figure 5.8: Inter-corpus SV for various test utterance duration. DET curves for (a) WKING on 3s (b) WKING on 1s. (c) NKING on 3s and (d) NKING on 1s test utterance duration.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DNA MFCCs</th>
<th>GMM-UBM</th>
<th>CDBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKING (3s)</td>
<td>(17.64) 100%</td>
<td>(21.74) 100%</td>
<td>(74.25) 100%</td>
</tr>
<tr>
<td></td>
<td>(4.11) 100%</td>
<td>(55.95) 100%</td>
<td>(51.64) 100%</td>
</tr>
<tr>
<td>WKING (1s)</td>
<td>(18.71) 100%</td>
<td>(10.95) 100%</td>
<td>(62.75) 100%</td>
</tr>
<tr>
<td></td>
<td>(7.73) 100%</td>
<td>(42.69) 100%</td>
<td>(50.91) 100%</td>
</tr>
<tr>
<td>NKING (3s)</td>
<td>(13.44) 100%</td>
<td>(6.09) 100%</td>
<td>(49.85) 100%</td>
</tr>
<tr>
<td></td>
<td>(7.34) 100%</td>
<td>(35.79) 100%</td>
<td>(43.44) 100%</td>
</tr>
<tr>
<td>NKING (1s)</td>
<td>(16.75) 100%</td>
<td>(8.06) 100%</td>
<td>(45.03) 100%</td>
</tr>
<tr>
<td></td>
<td>(8.66) 100%</td>
<td>(27.67) 100%</td>
<td>(36.60) 100%</td>
</tr>
</tbody>
</table>

Table 5.9: Statistical significance (z-scores and % confidence interval) on inter-corpus SV results for DNA, MFCCs, GMM-UBM and CDBN evaluated on WKING and NKING for 3s and 1s test utterance length.

in Figure 5.6 and Figure 5.8 that MFCCs are more vulnerable to poor performance for shorter test speech durations as the deterioration in their
performance going from 5s to shorter utterances is more as compared to other models. In contrast the performance by DNA representation, although suffer for shorter test utterances, but stable i.e., the conclusion on the comparative performance of DNA representation with other models remains unchanged for all test utterance lengths.

5.4 Discussion

As mentioned earlier, the aim of this research is not to implement a sophisticated SR system rather propose a strategy to extract and enhance the speaker-dependent characteristics in acoustic feature space. As evident through extensive experimental evaluation on SV, the performance of any SR system is critically dependent on the ability of speech representations to characterise the speakers. We have demonstrated the representation produced by DNA shows promising results in different SV test scenarios. The DNA representation is compared to MFCCs that are a general speech representation and the performance gain in different experiments validates the idea of speech information component analysis. Furthermore the performance of GMM-UBM, which is the state-of-the-art speaker modelling technique, suggests that although it successfully models the data distribution sensitive to the speaker variation, it also models the non-speaker related interference in the data such as environmental conditions that eventually results in overfitting. In addition, the difference in the speaker and environmental variability of training and test data degrades the GMM performance significantly, which further highlights the need of speaker-specific feature extracting prior to speaker modelling. Generic DNA representations using simple mono-Gaussian model, on the other hand, show stable performance irrespective of the nature of training data as shown in the intra and inter-corpus SV. The comparison with CDBN further strengthen our argument that
unsupervised training alone cannot extricate various speech information components unless used with a well balanced supervised discrimination criteria in learning.

Ideally the training data should cover the examples with as much variability as possible to achieve better generalisation [50]. We train the DNA with increasing variation in the data and observe the improvement in the SV performance, which is consistent with the previous studies [50], [83].

5.5 Chapter Summary

This chapter provides the comprehensive experimental evaluation of the proposed DNA representations in SV task. The chapter starts with the brief overview about SV followed by the study of methodologies for feature extraction and speaker modelling used for comparative study. This includes convolutional deep belief nets (CDBN), mono-Gaussian models (MGMs) and Gaussian mixture models (GMM). After stating the experimental settings for training and testing on various speech corpora by DNA, CDBN and GMM-UBM, detailed SV experiments are performed with intra-corpus and inter-corpus/inter-language settings.
Chapter 6 Speaker Segmentation

In this chapter, DNA representations are evaluated in Speaker Segmentation (SS), a challenging SR task that highlights the importance of SI in a setup different from SC and SV. After reviewing various SS methods, comparison of DNA representation with the state-of-the-art in SS is presented on three speech corpora.

6.1 Speaker Segmentation

Speaker segmentation (SS) also referred as speaker change detection is a SR task to detect speaker change points in an audio stream. The purpose of SS is to divide the audio stream into homogeneous segments such that each segment should contain the speech content belonging to a single speaker. SS has it main application in speaker indexing in long audio stream where multiple speakers are involved in a conversation e.g., broadcast news. This can also be a preliminary step to another SR task like speaker identification. On the basis of execution, SS can be broadly divided into two categories i.e., metric-based segmentation and model based-segmentation. Metric-based SS is the comparison of the two short audio streams based on some distance or metric without any prior knowledge about the speaker involved in audio conversation. In contrast, model-based SS involves prior training of speaker models, e.g., GMM, on the offline data of speaker involved in conversation. There is another methodology to find speaker change point that is based on the silence detection between the
speeches of the two speakers. Metric-based SS algorithms are the most popular due to their simplicity and computational efficiency in online SS task.

Generally metric-based SS involves dividing the long parameterised audio stream into short segments and finding out whether two segments belong to the same speaker and therefore, if there is any speaker change point between the two segments. In our SS experiments, we mainly focus on comparing the DNA representation with MFCCs using some distance measures. The comparison is provided with the state-of-the-art SS algorithms that also use MFCCs as inputs. The underlying idea is: the speaker change point can be accurately found, if the two comparing audio segments are discriminated on the basis of features or representations which better characterise SI.

Before experimental strategy, the dataset involved in SS are reviewed.

6.2 Dataset Generation

For SS experiments, three datasets are used i.e., TIMIT, NTIMIT and CHN to artificially create number of audio streams with multiple speaker involved in conversation style.

TIMIT

Using TIMIT dataset, we generate 25 artificial audio conversations. Each conversation includes speech clips from 10 different speakers. So in total, there 250 speakers involved in 25 audio conversations. The length of each audio stream is approximately 40s that involve the speech clip from each speaker with the duration range of 1.6 to 7s. Since TIMIT dataset involves 10 different utterances for each speaker, where the average duration of each utterance is 2.5s, longer duration segments in a conversation are created by concatenating randomly selected 2 or 3 utterances. The audio
stream is parameterized as MFCC feature vectors where each time-domain stream is represented by a sequence of MFCC feature vectors.

**NTIMIT**

NTIMIT is a narrow-band version of TIMIT speech. Therefore like TIMIT, 25 audio conversations are created for NTIMIT dataset with the same settings. However, the speakers and their speech duration might be different from TIMIT but general statistics are similar, e.g., the average duration of each audio stream is 40s and the speech clip duration from single speaker is in the range of 1.6 to 7s.

**CHN**

From CHN dataset, 10 artificial audio conversations are generated where each conversation include 10 different speakers. The speech content from each speaker in a single stream ranges from 3 to 5s. The average length of each audio stream is 45s. Total of 50 speakers are involved in this dataset generation.

### 6.3 Speaker Segmentation Approaches

#### 6.3.1 Distance-Based Approach

The steps involved in the distance-based SS algorithm are as following:

1. The audio stream is converted into the sequence of feature vectors, e.g., MFCCs.
2. The distance between two fixed sized windows $win_1$ and $win_2$ is calculated. Each window comprises of fixed number of feature vectors corresponding to the fixed duration $T_w$.
3. The windows are slide over the audio stream (sequence of feature vectors) with an interval $\tau$ to calculate a new distance point. The process is shown in Figure 6.1.
Figure 6.1: Typical distance-based speaker segmentation process.

4. The distance curve as a result of step 3 is low pass filtered to remove the sharp glitches. The maxima in the distance curve are the speaker change point candidates, while the minima indicate the two sliding windows belong to the same speaker.

5. Speaker change point is detected at the peaks using some threshold \( \beta_{seg} \). If the difference between the candidate maxima and the two adjacent minima on its left and right are greater than \( \beta_{seg} \), it is regarded as a change point as shown in Figure 6.2.

6. If the detected change point falls into a tolerance interval \( T_l \) of a ground truth, the change point is regarded as a correct decision, otherwise either false alarm or miss detection occurs. False alarm is due to the false candidate peak, while miss detection occurs if no peak is detected as a result of step 5 as shown in Figure 6.3 (a).

For DNA, the feature vectors in step 1 are the sequence of DNA representations. The distance measure used in our experiments is given as

\[
\begin{align*}
    d_{seg}(win_1, win_2) &= \text{tr}[(\Sigma_1^{-1} + \Sigma_2^{-1})(\mathbf{\mu}_1 - \mathbf{\mu}_2)(\mathbf{\mu}_1 - \mathbf{\mu}_2)^T].
\end{align*}
\]  

(6.1)
Figure 6.2: Peak detection in distance-based speaker segmentation.

Eq. (6.1) is the modified KL divergence metric [1], where \( \mu_1 \) and \( \Sigma_1 \) are the mean vector and covariance metrics of segments \( \text{win}_1 \), while \( \mu_2 \) and \( \Sigma_2 \) belong to \( \text{win}_2 \). \( \text{tr} (\cdot) \) in (6.1) is matrix trace. We dropped the term that deals with only covariance metrics in the symmetric KL divergence metric. This is based on our observation that covariance matrix show instability for very short utterances, while (6.1) is stable is stable enough irrespective of the utterance length.

### 6.3.2 Bayesian Information Criteria (BIC)

BIC is the benchmark metric-based SS algorithm [84]. BIC and it variants are among the state-of-the-art systems which have shown promising results [85], [86], [87]. The simple yet effective idea behind BIC is based on calculating the likelihood ratio of two short segments in audio stream.

Suppose an audio stream \( X = \{x_1, x_2, ..., x_N\} \) with \( N \) feature vectors. The stream can be divided into two sub-streams \( X_1 = \{x_1, x_2, ..., x_i\} \), \( X_2 = \{x_{i+1}, x_{i+2}, ..., x_N\} \). The likelihood ratio is given as

\[
R(i) = N \log|\Sigma| - N_1 \log|\Sigma_1| - N_2 \log|\Sigma_2|,
\]  

(6.2)
Figure 6.3: Metric curves. (a) Distance curve with circles indicating false alarm (FA) and miss detection (MD) points. (b) BIC likelihood curve.

where $\Sigma$, $\Sigma_1$ and $\Sigma_2$ are the covariance matrices of $X$, $X_1$ and $X_2$ respectively. $N_1$ and $N_2$ are the total number of feature vectors in $X_1$ and $X_2$ respectively. Eq. (6.2) can be seen as a two model-fitting problem, i.e., whether the audio stream $X$ is better represented by a single Gaussian or two Gaussians. The difference between the two models is given as

$$
\Delta BIC(i) = R(i) - \omega \mathcal{P},
$$

(6.3)

where $\mathcal{P} = \frac{1}{2} \left( \frac{1}{2} \delta + \frac{1}{2} \delta (\delta + 1) \right) \log N$ and $\delta$ is the dimension of the feature vector. The point $i$ is regarded as a valid change if

$$
\max_i \Delta BIC(i) > 0.
$$

(6.4)

The speaker change detection using BIC has following steps [84],

1. Suppose an interval $[a, b]$ of duration $T_w$ in the audio stream $X$. Find a change point in $[a, b]$ using (6.3) and (6.4). If the speaker change is detected at point $i$, set $a = i + 1$ and $b = a + \tau$. If no change is detected, set $a = a$ and $b = b + \tau$. The parameter $\tau$, as in distance-based method, is the small increment over sequence of feature vectors.

2. Repeat step 1 for the whole audio stream.
6.3.3 DISTBIC

DISTBIC [85] is the combination of distance-based change point detection and BIC. The main objective of DISTBIC is to reduce the false alarm by keeping the miss detection rate minimum. It is a two-stage process that can be summarised as follows:

1. Find the speaker turn candidate with distance-based algorithm using the threshold $\beta_{seg}$ in such a way that it gives minimum miss detection rate (MDR) at the cost of high false alarm rate (FAR).
2. In second step, each candidate points are validated using (6.4) for the two segments cantered at the candidate peak points $i$. The two windows comprise of the sequence between the immediately left and right candidates of point $i$. This step tries to minimise FAR by judging every possible candidate speaker change point as a result of step 1.
3. If (6.4), for a candidate peak at $i$ is valid, it is regarded as a valid change point otherwise discarded.

6.3.4 Divide and Conquer BIC (DACBIC)

Recently proposed, this technique is one of the few state-of-the-art SS algorithms [86], which is refined version of standard BIC. The algorithm involved following steps:

1. The whole audio sequence is applied to BIC to detect speaker change point candidates. The point with maximum value of (6.3) is selected as an initial candidate. If (6.4) is valid, it is regarded as a change point.
2. Based on the initial candidate, no matter it is a valid change point or not, two segments are generated by dividing the longer sequence at the point of maximum (6.3) and for each segment repeat step 1.
3. Continue dividing the stream using step 1 and 2 until a minimum
window length $T_w^{(\text{min})}$ is achieved for all the divided segments. In the process, the candidate with valid (6.4) is saved as valid change points.

4. When the $T_w^{(\text{min})}$ is achieved for all the segments, the final step checks (6.4) for two windows centered at each candidate points except those that are already validated in step 1, 2 and 3.

5. If for a candidate change point (6.4) is valid, it is regarded as a correct change point otherwise the two windows are merged to form a single segment and repeat step 4.

Using dividing strategy, DACBIC compared only those two segments, which precisely belong to two speakers. This generates a minimum false alarm by avoiding miss detection as much as possible. In contrast, standard BIC in the case of miss detection assign a whole segment to one speaker with single decision step, which degrades the overall performance.

6.4 Experimental Settings

The performance of DNA representation is compared with baseline MFCCs, BIC, DISTBIC and DACBIC. For DNA and MFCCs distance-based SS is employed with parameters as follows; the window length $T_w$ corresponds to 1.5s of audio stream, the threshold $\beta$ is variable for DNA and MFCCs, the values of $\beta$ are tuned to trade-off between FAR and MDR. The tolerance interval $T_t = 0.5s$ and sliding increment $\tau = 0.1s$. MFCCs are also used as speech features for BIC, DISTBIC and DACBIC. For BIC, the initial interval length $T_w = 3s$ and $\omega$ is variable to trade-off between FAR and MDR. The increment $\tau = 1s$. DISTBIC follows the parameter settings of distance-based algorithm and standard BIC. DACBIC has additional parameter $T_w^{(\text{min})} = 3s$.

For performance evaluation by various algorithms, we use the measures i.e., false alarm rate (FAR), miss detection rate (MDR) and $F_1$ measure [85],
[85], [86]. FAR = \( \frac{N_{FA}}{N_{FA} + N_{GC}} \) and MDR = \( \frac{N_{MD}}{N_\text{GC}} \), where \( N_{FA}, N_{MD} \)

and \( N_{GC} \) are number of false alarms, miss detection and genuine speaker change points (ground truth). The \( F_1 \) measure is given as \( F_1 = \frac{2 \cdot PR}{(P + R)} \), where the precision rate \( P = \frac{N_{CFC}}{N_{TDC}} \) and recall rate \( R = \frac{N_{CFC}}{N_{GC}} \). \( N_{CFC} \) and \( N_{TDC} \) are the number of correctly found speaker change points and totally detected change points. For better performance, the FAR and MDR should be small while larger value of \( F_1 \) indicates the good performance.

### 6.5 Experiments for Speaker Segmentation

The ROC curves in Figure 6.4 and numerical performance measures in Table 6.1 show performance of DNA representation compared to the other techniques. The distance based algorithms on DNA and MFCCs are represented as \( \text{dist-DNA} \) and \( \text{dist-MFCC} \) respectively. The parameters \( \beta \) in \( \text{dist-DNA} \) and \( \text{dist-MFCC} \) and \( \omega \) in BIC and DACBIC are tuned to get FAR and MDR as near to equal error rate (%EER) as possible. For all the datasets, DNA representations in a distance-based algorithm outperform the other algorithms, which shows their ability to characterise the speakers at very short utterance level. Indeed the accuracy in detecting the speaker change point is directly related to the speaker-specific information content contained by the two windows comparing the speakers. Apart from DNA, MFCCs also show stable performance in a distance-based algorithm as compared to the BIC and DISTBIC methods in TIMIT and CHN as it is pointed out in [84] that BIC normally performs poor for short utterances (shorter than 7s). On the other hand DACBIC shows advantage over the standard BIC as it finds the compatibility between two exhaustively found segments with a correct duration of speech by each speaker. The
Figure 6.4: SS performance by dist-DNA, dist-MFCCs, BIC, DISTBIC and DACBIC. ROC curves for SS on (a) TIMIT (b) NTIMIT and (c) CHN.

Table 6.1: SS results in terms of %EER and F1 measure.
performance of DACBIC is better than \textit{dist}-MFCCs, BIC and \textit{DISTBIC} in both TIMIT and NTIMIT datasets while it is better than BIC and \textit{DISTBIC} in CHN and comparable to \textit{dist}-MFCC. For TIMIT dataset in Figure 6.4, the performance curve for \textit{dist}-DNA and DACBIC are comparable although the \%EER of \textit{dist}-DNA is superior. Similarly \textit{dist}-DNA outperforms others in NTIMIT and CHN.

6.6 Chapter Summary

This chapter provides comparative study of DNA representation with various MFCC based sophisticated techniques in SS task. As the aim of SS is to segment speech conversation involving numerous speakers into speaker-homogeneous segments, artificial conversations are created using TIMIT, NTIMIT and CHN datasets. After that, the algorithms of various state-of-the-art SS techniques are discussed followed by experimental evaluation and the discussion on the outcome of the results.
Chapter 7 Conclusion and Future Work

This chapter concludes the research work carried out for this thesis and suggests some future improvements.

7.1 Thesis Summary

Speech signal conveys various yet mixed information. The performance of any speech related system heavily depends on its ability to extract and utilise the desired information in speech e.g., speech recognition system is requires LI extraction while speaker recognition depends on the SI. However, the interference of various information components in speech is the major hindrance in achieving better performance by ML algorithms for speech-aided systems. Automatic SR is a process of recognising speakers on the basis of information components that are sensitive to the speaker variation. Therefore to achieve better recognition, it is inevitable to extract SI. Comparatively most of the work has been done in the field of speaker modelling that resulted in numerous sophisticated SR systems but to our knowledge there are only few attempts to segregate the SI from other non-speaker related information.

In the last five years, deep architectures have emerged as potential candidates to learn highly complex functions. Inspired by the learning hierarchy of the brain, deep architectures can extract high-level abstract information in the data. Their successful application in various vision-related task make them attractive to explore SI in speech data, which
requires yet another complex problem solving from ML point of view.

In this research, we propose a deep neural architecture, DNA to implicitly extract SI from the raw MFCC speech representation.

Chapter 1 highlights our motivation to utilise deep learning to acquire a high-level speech representation enriched with SI. The detailed literature review about speech information processing and deep architectures is presented in Chapter 2. This includes feature extraction methods by exploiting the speech production mechanism, the study of spectral and non-spectral speech features and overview of various modern SR systems. The limitations of current SR system lead us to investigate the possible application of deep architectures to learn speaker-dependent characteristic features. In this regard the background knowledge on various deep architectures is presented. This study covers the pioneering work about deep belief networks (DBN), deep autoassociator networks (DAN), deep Siamese networks (DSN) and deep convolutional networks (CNN).

Chapter 3 details the design hierarchy of the proposed DNA. Using the knowledge gained about the deep architectures and their systematic learning protocols, we adopted two-staged network training strategy. This is based on the unsupervised layer-wise training, called pre-training using multiple denoising autoencoders to initialise the deep network and contrastive fine tuning with global objectives. Unsupervised pre-training provides better solution in gradient-based training by averting the local optima, which is the main problem in shallow networks. To learn the intrinsic speaker-dependent statistical pattern in the speech, a regularised contrastive loss function is designed that is minimised with a contrastive pair-wise comparison between the 1st and 2nd order statistics of DNA output representations. In addition, data reconstruction constraint is imposed on the loss function that acts as a regularisation penalty to neutralise the non-speaker information and redundancies like noise and
channel effects. Using visualisation it is shown that the representations produced by the trained DNA are successful in categorising the speakers on the short phonetic level due to the learned speaker-specific characteristics.

Chapter 4 is the empirical study to evaluate the parametric space of DNA. Various parameters that govern the structure of the network and the loss function are judged through experimental study for their optimum values. Furthermore, the proposed DNA is justified using the comparison with related deep architecture i.e., deep autoencoder and deep Siamese networks in speaker comparison (SC) task.

Chapter 5 and chapter 6 are based on the detailed experiments on speaker verification (SV) and speaker segmentation (SS), which are the two applications of the DNA representation on challenging SR tasks. The performance of DNA representations is compared with the state-of-the-art techniques like GMM-UBM in SV and BIC, DISTBIC, and DACBIC in SS using various speech corpora.

The outcome of various comparisons and experiments in Chapter 4, Chapter 5 and Chapter 6 suggests that DNA representation successfully characterise the speakers on the basis of SI. On the other hand, this research highlights the potential of deep learning especially in speech information processing. This provides a new direction for speech community to investigate the discriminative platform for speaker modelling. Previously this has been overshadowed by generative speaker modelling due to the limitation of the former in learning complex non-linear function that can represent unique and invariant pattern in the data.

7.2 Future Work

Throughout the research, while designing and evaluating the DNA, several areas are pointed out for possible improvement in the DNA in terms of
structure and loss function. There is also a chance to further investigate the possibility of using DNA representation in other speech related tasks.

7.2.1 Input Features

So far, we rely on MFCCs as base features to provide SI out of the pool of other information components. This is based on the assumption by previous studies that MFCCs better encode SI as compared to other spectral features [1], [4]. However they are a form of speech representation that depend mainly on the acoustic characterises governed by the human speech production system. There is another class of features that models the human auditory system. The examples include Lyon ear modelling that describes the behaviour of cochlea in the ear, Sneff ear model and Meddis inner hair-cell model [91], [92]. The combination of spectral and auditory features may increase the information range on higher level to investigate the effect of perceptual system in recognising speakers.

7.2.2 DNA Structure

The choice of using denoising autoencoder was made after our preliminary comparison with RBMs as building blocks of DNA. Recently convolutional RBMs enhance the generative modelling of RBM by combining the information about the spatial correlation in the data. Such strategy can be utilised on autoencoders to further enhance their representational power in deep architecture.

7.2.3 DNA Loss Function

There is a vast theoretical space to investigate the effect of different loss functions for training DNA. Among those, higher order, more nonlinear kernel function are attractive candidates which can be used to make speech information components more separable in the high dimensional space.
7.2.4 DNA Representation

DNA representations are overcomplete and higher in dimension as compared to MFCCs. Although distributive in nature for SI, these representations pose curse of dimensionality for extremely short utterances (shorter than 1s) where it is hard to correctly estimate the statistics. This poses performance degradation in SR tasks like SS using BIC that is highly sensitive to the data dimension. We tried to apply DNA representations as input to BIC but results were worse as compared to MFCCs, although DNA representations beat all other methods using simple distance based SS method. Using additional constraint on loss function like sparsity regularisation may help reduce the dimension of the DNA representations.

7.2.5 DNA Application

The main focus of this research is to learn speaker-specific characteristics in a speaker comparison setup with a deep architecture. However with a modification, DNA can be applied to the tasks like speech enhancement using the reconstruction constraints on clean and corrupted speech, while exploring SI side by side with enhanced speech. The other possible investigation is to utilise the non-speaker part of the code layer in DNA for speech/phoneme recognition as it helps in normalising the non-speaker information (mainly LI) from speaker-specific part of the code layer.

7.3 Conclusion

This research has highlighted the need to extract SI from the general speech representation to address the issue that the performance of the SR system is always compromised due to insufficient speaker-specific characterisation in the input feature space. A deep architecture is proposed that works on the regularised contrastive loss function to yield speaker-specific representation. Through various experimental evaluations, it is evident
that DNA representations outperform baseline MFCCs and other sophisticated speaker modelling techniques in SC, SV and SS tasks. This research has concluded with the overview of various suggestions about the prospective improvements in DNA design.
Appendix A

Denoising Autoencoder Training

In reference to Section 3.2.1, this appendix covers the pre-training of DNA using the denoising autoencoder.

Suppose $W$ be the connection weight matrix between the input layer and the hidden layer of the autoencoder, while, $W^T$ be the weights between hidden layer and output layer. Let $b_h$ and $b_o$ denote biases of the hidden and the output layers, respectively. For a distorted input $\tilde{x}$, the output representation of the hidden and the output layers are $h(\tilde{x}) = \sigma[u_h(\tilde{x})]$ and $\hat{x} = u_o(\tilde{x})$ for the first hidden layer (first autoencoder). Similarly reconstruction at output layers for the autoencoder $K, K = 2, ..., K$ is $\hat{x} = \sigma[u_0(\tilde{x})]$, where $u_h(\tilde{x}) = W\tilde{x} + b_h$ and $u_0(\tilde{x}) = W^T h(\tilde{x}) + b_0$. In this appendix, for simplicity, we only use $(x, \tilde{x})$ to represent input and output of each denoising autoencoder with $\tilde{x}$ being the corrupted version of $x$.

Given a training example $(\tilde{x}, x)$, we have the gradient for the cost function in Eq. (3.13) with respect to $u_0(\tilde{x})$:

$$ \frac{\partial L_{dec}(\tilde{x}, x)}{\partial u_0(\tilde{x})} = 2(\tilde{x} - x)\sigma'[u_0(\tilde{x})], \quad (A.1) $$

where $\sigma'[u_0(\tilde{x})]$ is the derivative of sigmoid function $\sigma(\cdot)$ with respect to $u_0(\tilde{x})$. Using (A.1), we have the gradient

$$ \frac{\partial L_{dec}(\tilde{x}, x)}{\partial h(x)} = W \frac{\partial L_{dec}(\tilde{x}, x)}{\partial u_0(\tilde{x})}. \quad (A.2) $$

By using the chain rule, we achieve the gradient

$$ \frac{\partial L_{dec}(\tilde{x}, x)}{\partial u_h(\tilde{x})} = \left( h_j(\tilde{x})[1 - h_j(\tilde{x})] \frac{\partial L_{dec}(\tilde{x}, x)}{\partial h_j(\tilde{x})} \right)_{j=1}^{h}, \quad (A.3) $$

here, $h_j(\tilde{x})$ is the $j^{th}$ element of $h(\tilde{x})$. Similarly, we have the gradient with respect to biases
\[
\frac{\partial L_{dec}(\bar{x}, \bar{y})}{\partial b_0(\bar{x})} = \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_0(\bar{x})}, \quad \frac{\partial L_{dec}(\bar{x}, x)}{\partial b_h(\bar{x})} = \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_h(\bar{x})}.
\] (A.4)

Based on (A.1-A.4), we apply the gradient descent method to achieve update rules as follows:

\[
W \leftarrow W - \epsilon \left( \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_h(\bar{x})} \bar{x}^T + h(\bar{x}) \left[ \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_0(\bar{x})} \right]^T \right),
\] (A.5)

\[
b_0 \leftarrow b_0 - \epsilon \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_0(\bar{x})}
\] (A.6)

\[
b_h \leftarrow b_h - \epsilon \frac{\partial L_{dec}(\bar{x}, x)}{\partial u_h(\bar{x})}
\] (A.7)

The parameters are updated using SGD in a batch style training.
**Appendix B**

**DNA Contrastive Training**

In this appendix we specifically derive the gradient of contrastive loss $L_D(X_1, X_2; \theta)$ with respect to $u_k(x_{it})$ to obtain Eq. (3.17) in Section 3.2.2 of the main text. To simplify the presentation, we first elucidate our notation system that is completely consistent to that used in the main text.

We collectively denote the output of neurons in $CS$ at the coder layer or layer $K$ of subnet $i$ ($i = 1, 2$) as $CS(X_i) = \left\{\left(\left(\left(CS(x_{it})_j\right)_j\right)_{t=1}^{T_B}\right)_{j=1}^{\mid CS\mid}\right\}$ for a speech segment of $T_B$ frames, $X_i = \{x_{it}\}_{t=1}^{T_B}$. Accordingly, we have $\mu^{(i)} = \left(\mu_j^{(i)}\right)_{j=1}^{\mid CS\mid}$ and $\Sigma^{(i)} = \left[\sigma_{tn}^{(i)}\right] (t, n = 1, \ldots, \mid CS\mid)$ where $\sigma_{tn}^{(i)} = \frac{1}{T_B-1} \Sigma_{t=1}^{T_B} \left(\left(CS(x_{it})\right)_t - \mu_t^{(i)}\right)\left(\left(CS(x_{it})\right)_n - \mu_n^{(i)}\right)^T$. In the following derivation, we also drop all explicit parameters in $L_D(X_1, X_2; \theta)$ and rewrite it into $L_D = L_m + L_s$ where $L_m = \mathcal{X}\mathcal{D}_m + (1 - \mathcal{X})e^{-\frac{2m}{\lambda_m}}$ and $L_s = \mathcal{X}\mathcal{D}_s + (1 - \mathcal{X})e^{-\frac{2s}{\lambda_s}}$ are losses with respect to mean and covariance respectively.

Using our notation described above, we immediately achieve

$$\frac{\partial L_D}{\partial u_k(x_{it})} = \frac{\partial L_m}{\partial u_k(x_{it})} + \frac{\partial L_s}{\partial u_k(x_{it})} \quad \text{(B.1)}$$

$$\frac{\partial L_D}{\partial u_k(x_{it})} = \left(\left(\left[\mathcal{X} - \lambda_m^{-1}(1-\mathcal{X})e^{-\frac{2m}{\lambda_m}}\right]\frac{\partial \mathcal{D}_m}{\partial u_{kj}(x_{it})}\right)_{j=1}^{\mid CS\mid}, (0)_{j=\mid CS\mid+1}^{\mid h_k\mid}\right) + \left(\left(\left[\mathcal{X} - \lambda_s^{-1}(1-\mathcal{X})e^{-\frac{2s}{\lambda_s}}\right]\frac{\partial \mathcal{D}_s}{\partial u_{kj}(x_{it})}\right)_{j=1}^{\mid CS\mid}, (0)_{j=\mid CS\mid+1}^{\mid h_k\mid}\right) \quad \text{(B.2)}$$

To facilitate the presentation, we define $\psi_j(x_{it}) = \frac{\partial \mathcal{D}_m}{\partial u_{kj}(x_{it})}$ and $\xi_j(x_{it}) = \frac{\partial \mathcal{D}_s}{\partial u_{kj}(x_{it})}$. Now we simply need to calculate $\psi_j(x_{it})$ and $\xi_j(x_{it})$ for $j = 1, \ldots, \mid CS\mid$ to obtain Eq. (3.17) in the main text. As $\mathcal{D}_m = \left\|\mu^{(1)} - \mu^{(2)}\right\|_2^2$.
Given the fact that the transfer function used for the code layer or layer $K$ is the sigmoid function, collectively, we have

$$
\psi_j(x_{it}) = \frac{\partial D_m}{\partial (CS(x_{it}))} \frac{\partial (CS(x_{it}))}{\partial u_{kj}(x_{it})},
$$

where

$$
p_j = \frac{\partial \Sigma_{i=1}^{CS} (\mu_{i}^{(1)} - \mu_{i}^{(2)})}{\partial (CS(x_{it}))} \frac{2}{T_B} \text{sign}(1.5 - i)(\mu_{j}^{(1)} - \mu_{j}^{(2)}),
$$

and

$$
\frac{\partial (CS(x_{it}))}{\partial u_{kj}(x_{it})} = (CS(x_{it})) \left[ 1 - (CS(x_{it})) \right].
$$

Given the fact that the transfer function used for the code layer or layer $K$ is the sigmoid function, collectively, we have

$$
p = \frac{2}{T_B} \text{sign}(1.5 - i)(\mu^{(1)} - \mu^{(2)}).
$$

Similarly, $D_s = \|\Sigma^{(1)} - \Sigma^{(2)}\|_F = \Sigma_{i=1}^{CS} \Sigma_{n=1}^{\Sigma^{(1)}} (\sigma_{jn}^{(1)} - \sigma_{jn}^{(2)})^2$. Hence, we have

$$
\xi_j(x_{it}) = \frac{\partial D_s}{\partial (CS(x_{it}))} \frac{\partial (CS(x_{it}))}{\partial u_{kj}(x_{it})},
$$

where

$$
q_j(x_{it}) = \frac{4}{T_B - 1} \text{sign}(1.5 - i) \sum_{n=1}^{\Sigma^{(1)}} (\sigma_{jn}^{(1)} - \sigma_{jn}^{(2)}) \left[ (CS(x_{it})) - \mu_{n}^{(1)} \right],
$$

and, collectively, we have $q(x_{it}) = \frac{4}{T_B - 1} \text{sign}(1.5 - i)(\Sigma^{(1)} - \Sigma^{(2)})[CS(x_{it}) - \mu^{(1)}]$. Inserting Eq. (B.3) and (B.7) into Eq. (B.1), we obtain Eq. (3.17) in the
main text as

$$\frac{\partial L_D}{\partial u_k(x_{it})} = \left( \left[ \mathcal{I} - \lambda_m^{-1} (1 - \mathcal{I}) e^{-\frac{v_m}{\lambda_m}} \right] \psi_j(x_{it}) \right)_{j=1}^{\left| c_s \right|}, (0)_{j=\left| c_s \right|+1}^{\left| h_k \right|} +$$

$$\left( \left[ \mathcal{I} - \lambda_s^{-1} (1 - \mathcal{I}) e^{-\frac{v_s}{\lambda_s}} \right] \xi_j(x_{it}) \right)_{j=1}^{\left| c_s \right|}, (0)_{j=\left| c_s \right|+1}^{\left| h_k \right|}.$$
Appendix C

Software Implementation

The software implementation of all the work presented in this thesis is carried out using MATLAB® [125].

The source code for DNA is written using following MATLAB® toolboxes: -

- MATLAB® signal processing toolbox [125].
- MATLAB® stats toolbox [125].
- Netlab [126].
- Dimensionality reduction toolbox, drtool [128].

Mel-frequency cepstral coefficients, MFCCs are extracted from speech files using: -

- MATLAB® signal processing toolbox [125].
- MATLAB® stats toolbox [125].
- Netlab [126].
- Voicebox [127]
- Auditory toolbox [129].

Gaussian Mixture Models, GMM are implemented using: -

- Netlab [126].
- MATLAB® signal processing toolbox [125].
- MATLAB® stats toolbox [125].

Source code for Convolutional Deep Belief Networks, CDBN is provided by the authors of [48] on request. CDBN uses following MATLAB® toolboxes: -

- MATLAB® signal processing toolbox [125].
- MATLAB® stats toolbox [125].

In general, all the experiments presented in Chapter 4, Chapter 5, and Chapter 6 are carried out using following MATLAB® toolboxes: -

- MATLAB® signal processing toolbox [125].
- MATLAB® stats toolbox [125].
• Netlab [126].
• drtool toolbox [128].

Source Code for DNA Pre-training using Denoising Autoassociators

```matlab
%-----------------------------------------------------------------
%                     Denoising Autoassociater Network
%-----------------------------------------------------------------
% Input: Data in batch style
% Output: Autoassociator weights i.e., w1, w2
%-----------------------------------------------------------------

[numcases numdims numbatches] = size(batchdata); % Structure of Data
N = numcases;
Dn = 0.1*randn(numcases,numdims); % Noise generation
maxepoch = 1;

% Main iterations for DNA training
fprintf(1,'------------------Epoch %d Set %d\r',epoch,m);
err1 = 0; err2 = 0;
for batch = 1: numbatches,
    fprintf(1,'Batch %d\r', batch);
    data_1 = batchdata(:, :, batch);
    [w1, w2, f] = Wupdate_pretrain(w1, w2, data_1, Dn, flag);
    data_2 = data_1 + Dn;
    data_2 = [data_2 ones(N,1)];
    w1probs = 1./(1 + exp(-data_2*w1));
    w1probs = [w1probs ones(N,1)];
    if flag == 1
        XXout = w1probs*w2;
    else
        XXout = 1./(1 + exp(-w1probs*w2));
    end
    err1 = err1 + 1/N*sum(sum(data_1 - XXout).^2);
    clear w1probs
end

%-----------------------------------------------------------------
%-----------------------------------------------------------------

function [W1,W2,f_new] = Wupdate_pretrain(W1,W2,XX,Dn,flag)

epsilon = 0.001; % Learning rate
max_iter = 5; % Maximum no. of iteration
N = size(XX,1);
X = XX + Dn;
X = [X ones(N,1)];
w1probs = 1./(1 + exp(-(X*W1)));
w1probs = [w1probs ones(N,1)];
if flag ==1
    XXout = w1probs*W2;
else
    XXout = 1./(1 + exp(-(w1probs*W2)));
end
err1 = err1 + 1/N*sum(sum(data_1 - XXout).^2);
clear w1probs

% Loss function for pre-training
f_old = 1/N*sum(sum((XX-XXout).^2));
for n = 1:max_iter,
```

157
[f_new,dw1,dw2] = grad_pretrain(W1,W2,XX,Dn,flag);
if f_new > f_old
    fprintf(1,'Direction Uphill\r');
    break;
else
    b1 = W1(end,:); W1 = W1(1:end-1,:);
    d1 = dw1(end,:); dw1 = dw1(1:end-1,:);
    b2 = W2(end,:); d2 = dw2(end,:);
    dw2 = dw2(1:end-1,:);
end
end

% Weight update
W1 = W1 - epsilon*(dw1 + dw2');
W2 = W1';
% Bias update
b1 = b1 - epsilon*(d1);
b2 = b2 - epsilon*(d2);
W1 = [W1;b1];
W2 = [W2;b2];
f_old = f_new;

function [f,dw1,dw2] = grad_pretrain(W1,W2,XX,Dn,flag)

N = size(XX,1);
X = XX + Dn;
X = [X ones(N,1)];
w1probs = 1./(1 + exp(-(X*W1))); w1probs = [w1probs ones(N,1)];
if flag ==1
    XXout = w1probs*W2;
else
    XXout = 1./(1 + exp(-(w1probs*W2)));
end
f = 1/N*sum(sum((XX-XXout).^2)); % Loss function

% Gradients
IO = -2/N*(XX-XXout);
Ix2=IO;
dw2 = w1probs'*Ix2; % Gradient w.r.t second layer
Ix1 = (Ix2'*W2').*w1probs.*(1-w1probs);
Ix1 = Ix1(:,1:end-1);
dw1 = X'*Ix1; % Gradient w.r.t first layer

Source Code for DNA Fine-tuning

%-----------------------------------------------------------------
%        DNA Finetuning (Regularised Contrastive Learning)
%-----------------------------------------------------------------

% Input: Load weights/biases for each hidden layer after Pre-training
% Input: Genuine and Imposter pair data (separately generated)

% Output: Final DNA weights
% Note: This code assumes DNA with four hidden layers.

maxepoch = 200; % Maximum epochs

K1 = num_sp(1);
K2 = num_sp(2);

labels = [zeros(K1,1); ones(K2,1)];
Rn1 = randperm(K1+K2);
labels = labels(Rn1);
Rn2 = randperm(K1); Rn3 = randperm(K2);
ep = 1;
for epoch = 1:maxepoch,
    Cases = K1+K2;
k1 = 1; k2 = 1;
    for n = 1:Cases,
        fprintf(1,'Epoch %d Case %d\n',epoch,n);
        if labels(n) == 0; % Load Genuine pair for two DNA sub-nets
            D = load(['Path to genuine... data',num2str(Rn2(k1)),'.mat']);
            data_1 = struct2cell(D); data_1 = data_1{:};
            D = load(['Path to genuine... data',num2str(Rn2(k1)),'.mat']);
            data_2 = struct2cell(D); data_2 = data_2{:};
            k1 = k1+1;
        else % Load Imposter pair for two DNA sub-nets
            D = load(['Path to imposter... data',num2str(Rn3(k2)),'.mat']);
            data_1 = struct2cell(D); data_1 = data_1{:};
            D = load(['Path to imposter... data',num2str(Rn3(k2)),'.mat']);
            data_2 = struct2cell(D); data_2 = data_2{:};
            k2 = k2+1;
        end
        label = labels(n);
        [W1,W2,W3,W4,f] = ... Wupdate_finetune(W1,W2,W3,W4,data_1,data_2,label);
    end
save final_weights W1 W2 W3 W4;
end
save final_weights W1 W2 W3 W4;

% Wupdate_finetune(W1,W2,W3,W4,XX1,XX2,label)

epsilon = 0.001; % Learning rate
max_iter = 5;
N = size(XX1,1);

XX1 = [XX1 ones(N,1)];
w1probs = 1./(1 + exp(-XX1*W1)); w1probs = [w1probs ones(N,1)];
w2probs = 1./(1 + exp(-w1probs*W2)); w2probs = [w2probs ones(N,1)];
w3probs = 1./(1 + exp(-w2probs*W3)); w3probs = [w3probs ones(N,1)];
w4probs = 1./(1 + exp(-w3probs*W4)); w4probs = [w4probs...
ones(N,1));
C_1 = w4probs;
w5probs = 1./(1 + exp(-(w4probs*W5))); w5probs = [w5probs ones(N,1)];
w6probs = 1./(1 + exp(-(w5probs*W6))); w6probs = [w6probs ones(N,1)];
w7probs = 1./(1 + exp(-(w6probs*W7))); w7probs = [w7probs ones(N,1)];
XXout1 = w7probs*W8;

XX2 = [XX2 ones(N,1)];
w1probs = 1./(1 + exp(-(XX2*W1))); w1probs = [w1probs ones(N,1)];
w2probs = 1./(1 + exp(-(w1probs*W2))); w2probs = [w2probs ones(N,1)];
w3probs = 1./(1 + exp(-(w2probs*W3))); w3probs = [w3probs ones(N,1)];
w4probs = 1./(1 + exp(-(w3probs*W4))); w4probs = [w4probs ones(N,1)];
C_2 = w4probs;
w5probs = 1./(1 + exp(-(w4probs*W5))); w5probs = [w5probs ones(N,1)];
w6probs = 1./(1 + exp(-(w5probs*W6))); w6probs = [w6probs ones(N,1)];
w7probs = 1./(1 + exp(-(w6probs*W7))); w7probs = [w7probs ones(N,1)];
XXout2 = w7probs*W8;

dim = 100;
C1 = C_1(:,1:dim);
C2 = C_2(:,1:dim);

% E1+E2 makes discriminative function
E1 = (sum(C1) - sum(C2)).^2;
E2 = cov(C1) - cov(C2);

% Loss function for genuine and imposter pairs
if label == 0
    f_old = 0.2/N*(sum(sum((XX1(:,1:end-1) - XXout1).^2))) +
            0.2/N*(sum(sum((XX2(:,1:end-1) - XXout2).^2))) +
            0.8*(sum(E1) + sum(sum(E2.^2)));
else
    f_old = 0.2/N*(sum(sum((XX1(:,1:end-1) - XXout1).^2))) +
            0.2/N*(sum(sum((XX2(:,1:end-1) - XXout2).^2))) +
            0.8*(sum(exp(-E1/100)) + sum(sum(exp(-(E2.^2)/2.5))));
end

for n = 1:max_iter,
    [f_new,dw1,dw2,dw3,dw4,dw5,dw6,dw7,dw8] = ...
        gradient_finetune(W1,W2,W3,W4,W5,W6,W7,W8,XX1,XX2,label);
    if f_new > f_old
        fprintf(1,'Direction Uphill\r');
        break;
    else
        % Weight/bias update
        W1 = W1 - epsilon*(dw1);
        W2 = W2 - epsilon*(dw2);
        W3 = W3 - epsilon*(dw3);
        W4 = W4 - epsilon*(dw4);
        W5 = W5 - epsilon*(dw5);
        W6 = W6 - epsilon*(dw6);
        W7 = W7 - epsilon*(dw7);
    end
end
```
W8 = W8 - epsilon*(dw8);
end
end

%-----------------------------------------------------------------
%-----------------------------------------------------------------
function [f,dw1,dw2,dw3,dw4,dw5,dw6,dw7,dw8] = ...
    CG_MNIST2(W1,W2,W3,W4,W5,W6,W7,W8,XX1,XX2,label)

N = size(XX1,1);

w1_1probs = 1./(1 + exp(-(XX1*W1))); w1_1probs = ...
    [w1_1probs ones(N,1)];
w2_1probs = 1./(1 + exp(-(w1_1probs*W2))); w2_1probs = ...
    [w2_1probs ones(N,1)];
w3_1probs = 1./(1 + exp(-(w2_1probs*W3))); w3_1probs = ...
    [w3_1probs ones(N,1)];
w4_1probs = 1./(1 + exp(-(w3_1probs*W4))); w4_1probs = ...
    [w4_1probs ones(N,1)];
C_1 = w4_1probs;
w5_1probs = 1./(1 + exp(-(w4_1probs*W5))); w5_1probs = ...
    [w5_1probs ones(N,1)];
w6_1probs = 1./(1 + exp(-(w5_1probs*W6))); w6_1probs = ...
    [w6_1probs ones(N,1)];
w7_1probs = 1./(1 + exp(-(w6_1probs*W7))); w7_1probs = ...
    [w7_1probs ones(N,1)];
XXout1 = w7_1probs*W8;

w1_2probs = 1./(1 + exp(-(XX2*W1))); w1_2probs = ...
    [w1_2probs ones(N,1)];
w2_2probs = 1./(1 + exp(-(w1_2probs*W2))); w2_2probs = ...
    [w2_2probs ones(N,1)];
w3_2probs = 1./(1 + exp(-(w2_2probs*W3))); w3_2probs = ...
    [w3_2probs ones(N,1)];
w4_2probs = 1./(1 + exp(-(w3_2probs*W4))); w4_2probs = ...
    [w4_2probs ones(N,1)];
C_2 = w4_2probs;
w5_2probs = 1./(1 + exp(-(w4_2probs*W5))); w5_2probs = ...
    [w5_2probs ones(N,1)];
w6_2probs = 1./(1 + exp(-(w5_2probs*W6))); w6_2probs = ...
    [w6_2probs ones(N,1)];
w7_2probs = 1./(1 + exp(-(w6_2probs*W7))); w7_2probs = ...
    [w7_2probs ones(N,1)];
XXout2 = w7_2probs*W8;

dim = 100;
C1 = C_1(:,1:dim);
C2 = C_2(:,1:dim);
[r,c] = size(C1);

E1 = (sum(C1) - sum(C2)).^2;
E2 = cov(C1) - cov(C2);

% Loss function for genuine and imposter pairs
if label == 0
    f = 0.2/N*(sum(sum((XX1(:,1:end-1) - XXout1).^2))) +...
        0.2/N*(sum(sum((XX2(:,1:end-1) - XXout2).^2))) +...
        0.8*(sum(E1) + sum(sum(E2.^2)));
end

IO1 = -(2*0.2)/N*(XX1(:,1:end-1) - XXout1);
IO2 = -(2*0.2)/N*(XX2(:,1:end-1) - XXout2);
```
IO3 = 0.8*((2)*repmat((sum(C1)-sum(C2)),r,1).*C1.*(1-C1) + ... 
(4)/(r-1)*((E2*(C1-repmat(mean(C1),r,1))').*C1.*(1-C1));
IO4 = 0.8*((-2)*repmat((sum(C1)-sum(C2)),r,1).*C2.*(1-C2) - ... 
(4)/(r-1)*((E2*(C2-repmat(mean(C2),r,1))').*C2.*(1-C2));
else
f = 0.2/N*(sum(sum((XX1(:,1:end-1) - XXout1).^2))) + ... 
0.2/N*(sum(sum((XX2(:,1:end-1) - XXout2).^2))) + ... 
0.8*(sum(exp(-E1/100)) + sum(sum(exp(-(E2.^2)/2.5))));
IO1 = -(2*0.2)/N*(XX1(:,1:end-1) - XXout1);
IO2 = -(2*0.2)/N*(XX2(:,1:end-1) - XXout2);
IO3 = 0.8*((-2/100)*repmat(exp(-E1/100),r,1).* ... 
(4/2.5)/(r-1)*((exp(-E2.^2).*E2)*(C1-repmat(mean(C1),r,1))'.*C1.*(1-C1));
IO4 = 0.8*((2/100)*repmat(exp(-E1/100),r,1).* ... 
(4/2.5)/(r-1)*((exp(-E2.^2).*E2)*(C2-repmat(mean(C2),r,1))').*C2.*(1-C2));
end

%----------------------------------------------------------------
% Gradient Calculation for all layers
IOR8 = IO1;
dwR8_1 = w7_1probs'*IOR8;
IOR7 = (IOR8*W8').*w7_1probs.*(1-w7_1probs);
dwR7_1 = w6_1probs'*IOR7;
IOR6 = (IOR7*W7').*w6_1probs.*(1-w6_1probs);
dwR6_1 = w5_1probs'*IOR6;
IOR5 = (IOR6*W6').*w5_1probs.*(1-w5_1probs);
dwR5_1 = w4_1probs'*IOR5;
IOR4 = (IOR5*W5').*w4_1probs.*(1-w4_1probs);
dwR4_1 = w3_1probs'*IOR4;
IOR3 = (IOR4*W4').*w3_1probs.*(1-w3_1probs);
dwR3_1 = w2_1probs'*IOR3;
IOR2 = (IOR3*W3').*w2_1probs.*(1-w2_1probs);
dwR2_1 = w1_1probs'*IOR2;
IOR1 = (IOR2*W2').*w1_1probs.*(1-w1_1probs);
dwR1_1 = XX1'*IOR1;

%----------------------------------------------------------------
IOR8 = IO2;
dwR8_2 = w7_2probs'*IOR8;
IOR7 = (IOR8*W8').*w7_2probs.*(1-w7_2probs);
IOR7 = IOR7(:,1:end-1);
dwR7_2 = w6_2probs'*IOR7;

IOR6 = (IOR7*W7').*w6_2probs.*(1-w6_2probs);
IOR6 = IOR6(:,1:end-1);
dwR6_2 = w5_2probs'*IOR6;

IOR5 = (IOR6*W6').*w5_2probs.*(1-w5_2probs);
IOR5 = IOR5(:,1:end-1);
dwR5_2 = w4_2probs'*IOR5;

IOR4 = (IOR5*W5').*w4_2probs.*(1-w4_2probs);
IOR4 = IOR4(:,1:end-1);
dwR4_2 = w3_2probs'*IOR4;

IOR3 = (IOR4*W4').*w3_2probs.*(1-w3_2probs);
IOR3 = IOR3(:,1:end-1);
dwR3_2 = w2_2probs'*IOR3;

IOR2 = (IOR3*W3').*w2_2probs.*(1-w2_2probs);
IOR2 = IOR2(:,1:end-1);
dwR2_2 = w1_2probs'*IOR2;

IOR1 = (IOR2*W2').*w1_2probs.*(1-w1_2probs);
IOR1 = IOR1(:,1:end-1);
dwR1_2 = XX2'*IOR1;

%----------------------------------------------------------------
IOC4 = [IO3 zeros(r,100)];
dw4_1 = w3_1probs'*IOC4;

IOC3 = (IOC4*W4').*w3_1probs.*(1-w3_1probs);
IOC3 = IOC3(:,1:end-1);
dw3_1 = w2_1probs'*IOC3;

IOC2 = (IOC3*W3').*w2_1probs.*(1-w2_1probs);
IOC2 = IOC2(:,1:end-1);
dw2_1 = w1_1probs'*IOC2;

IOC1 = (IOC2*W2').*w1_1probs.*(1-w1_1probs);
IOC1 = IOC1(:,1:end-1);
dw1_1 = XX1'*IOC1;

%----------------------------------------------------------------
IOC4 = [IO4 zeros(r,100)];
dw4_2 = w3_2probs'*IOC4;

IOC3 = (IOC4*W4').*w3_2probs.*(1-w3_2probs);
IOC3 = IOC3(:,1:end-1);
dw3_2 = w2_2probs'*IOC3;

IOC2 = (IOC3*W3').*w2_2probs.*(1-w2_2probs);
IOC2 = IOC2(:,1:end-1);
dw2_2 = w1_2probs'*IOC2;

IOC1 = (IOC2*W2').*w1_2probs.*(1-w1_2probs);
IOC1 = IOC1(:,1:end-1);
dw1_2 = XX2'*IOC1;

%----------------------------------------------------------------
dw1 = dwR1_1 + dwR1_2 + dw1_1 + dw1_2;
dw2 = dwR2_1 + dwR2_2 + dw2_1 + dw2_2;
dw3 = dwR3_1 + dwR3_2 + dw3_1 + dw3_2;
dw4 = dwR4_1 + dwR4_2 + dw4_1 + dw4_2;
dw5 = dwR5_1 + dwR5_2;
dw6 = dwR6_1 + dwR6_2;
dw7 = dwR7_1 + dwR7_2;
dw8 = dwR8_1 + dwR8_2;
Bibliography


pp. 61-64, 1990.


[73] Linguistic Data Consortium (LDC). [online]: www.ldc.upenn.edu


Russian Speech Corpus. [online]: www.repository.voxforge1.org


P. Delacourt and C. J. Wellekens, “DISTBIC: A Speaker-based


[94] Q Jin, A. Waibel, “Application of LDA to speaker verification,” In


[114] M. Man-Wai, R. Wei and H. Meng, “The HKCUU system for NIST


[126] Netlab, [online]: www1.aston.ac.uk.

[127] Voicebox, [online]: www.ee.ic.ac.uk.

[128] drtool, [online]: http://homepage.tudelft.nl.

