UNSUPERVISED DOMAIN ADAPTATION FOR SPEAKER VERIFICATION IN THE WILD

by

Phani Sankar Nidadavolu

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

July, 2021

Abstract

Performance of automatic speaker verification (ASV) systems is very sensitive to mismatch between training (*source*) and testing (*target*) domains. The best way to address domain mismatch is to perform matched condition training – gather sufficient labeled samples from the *target* domain and use them in training. However, in many cases this is too expensive or impractical. Usually, gaining access to unlabeled *target* domain data, e.g., from open source online media, and labeled data from other domains is more feasible. This work focuses on making ASV systems robust to uncontrolled ('wild') conditions, with the help of some unlabeled data acquired from such conditions.

Given acoustic features from both domains, we propose learning a mapping function – a deep convolutional neural network (CNN) with an encoder-decoder architecture – between features of both the domains. We explore training the network in two different scenarios: training on *paired* speech samples from both domains and training on *unpaired* data. In the former case, where the *paired* data is usually obtained *via* simulation, the CNN is treated as a non-

linear regression function and is trained to minimize L_2 loss between original and predicted features from *target* domain. We provide empirical evidence that this approach introduces distortions that affect verification performance. To address this, we explore training the CNN using adversarial loss (along with L_2), which makes the predicted features indistinguishable from the original ones, and thus, improve verification performance.

The above framework using simulated *paired* data, though effective, cannot be used to train the network on *unpaired* data obtained by independently sampling speech from both domains. In this case, we first train a CNN using adversarial loss to map features from *target* to *source*. We, then, map the predicted features back to the *target* domain using an auxiliary network, and minimize a cycle-consistency loss between the original and reconstructed *target* features.

Our unsupervised adaptation approach complements its supervised counterpart, where adaptation is done using labeled data from both domains. We focus on three domain mismatch scenarios: (1) sampling frequency mismatch between the domains, (2) channel mismatch, and (3) robustness to far-field and noisy speech acquired from wild conditions.

Primary Reader and Advisor: Najim Dehak

Secondary Reader: Jesús Villalba

Acknowledgments

First of all, I would like to thank my advisors – Najim Dehak and Jesús Villalba – without whose support this work would not have been possible. Najim gave me the freedom to explore topics of my interest. He always valued knowledge acquisition and efforts over results. He challenged me to broaden my knowledge by giving me the opportunity to work on several projects throughout my study. On a personal front, Najim has been a friend, a well wisher, and an elder brother. He made himself available every time I needed help.

Observing Jesús closely has always inspired me to work hard. Along with the technical expertise I acquired from our collaboration, Jesús taught me good coding practices, the importance of paying attention to details, being persistent with experimentation and several important writing tips. His calm demeanour and friendly nature has made me very comfortable during technical discussions.

I would like to thank Hynek Hermansky for his guidance during the initial years of my PhD. Multi-stream approach to noise robustness work I did with

him inculcated interest in me to explore the topic of domain adaptation for my dissertation. I would like to extend my thanks to Sanjeev Khudanpur for his constant support during my study, for always keeping his door open and for many valuable suggestions he has given me. Special thanks to Daniel Povey for all the support he has given me throughout my study.

I thank Paola for taking me in her team during JSALT'19 summer workshop, and for all the warmth she has given me ever since I met her. I thank Yishay Carmiel, Laureano and Piotr for all their support. I thank Shinji Watanabe, Mark Dredze, Philip Koehn and Dan Naiman for agreeing to be on my GBO panel.

I would like to thank my lab mates Raghu, JJ, Saurabh, Sonal, Bhati, Nanxin and Jeff for all the memorable conversations and fun times we have had over the past several years. I extend my thanks to Vimal, David, Pegah, Ashish for their valuable collaborations. I would cherish my friendship with Bhaskar, Harish and Vijay for the rest of my life. I extend my thanks to Ruth, Debbie, Belinda and Yamese, and Christine Kavanagh for all their efforts and support throughout my study.

With a heavy heart I cannot help but remember the contributions of my family. I express my love to my parents for everything they have done to see me where I am today. I thank my parents-in-law for all the love and support they offered me in the past ten years, and for always making me feel special.

ACKNOWLEDGMENTS

Love to my sister, my wife's siblings and their families. Special love to all my nieces and nephews.

Finally, I cannot express in words how lucky I am to have my wife, Haripriya, in my life. Her support and sacrifices make my heart heavy. All I can offer is to reciprocate her love for the rest of my life. A special hug to my daughters – Lasya and Sruthi – for entering my life and making me feel very special.

Dedication

This thesis is dedicated to my parents, Prasada Rao and Sarojini; to my wife, Haripriya; and my two lovely daughters – Lasya and Sruthi.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of automatic speaker verification (ASV) has made huge strides in the last decade. The invention of i-vector [1] was a break through in the field. In this approach, factor analysis [2] is used as a feature extractor to model a *total variability space*, which contains both channel and speaker variability. With the success in speaker verification (SV), i-vectors soon found their way to many speech applications – language identification [3, 4], speaker adaptation in automatic speech recognition (ASR) [5], far-field robustness of ASR [6], and age estimation [7] to name a few.

Over the same period, machine learning (ML) field has achieved great success and has benefited many real-world applications. The re-emergence of artificial neural network (ANN) with a deeper architecture – termed as deep neural network (DNN) – powered by the advances in computational hardware,

and the availability of large amounts of labeled data has pushed the limits of many tasks. DNNs are now being found in many applications such as image classification [8], acoustic modelling in ASR [9, 10], neural machine translation (NMT) [11, 12] and language models (LM) [13]. Recently, the field of ASV has joined the above club with the introduction of x-vectors [14].

Supervised learning has contributed tremendously to the advances made in almost all the fields dominated by DNNs, and SV using x-vectors is no exception. x-vector speaker embeddings are extracted from a feed-forward DNN with a statistics pooling layer, that is trained discriminatively to classify speakers given acoustic features as input. SV using x-vectors is now an established technology giving better performance than i-vectors on many test conditions, and hence, became the state-of-the-art (SOTA) [15] for SV.

Along with the success, the field is facing its own set of challenges. To develop a robust system that caters for large number of domains, the system needs to be trained on large amounts of annotated data acquired from all the domains. Some of the conditions that test the limits of ASV system include channel variations, acoustic mismatch conditions, room reverberations and language spoken[1]. Acquiring and annotating datasets for every domain is extremely expensive and time-consuming processes. Hence, sufficient training data may not always be available. When available, large amount of data also

[1]All the mentioned variations also affect the performance of ASR systems. In addition, ASR should also be robust to variation in speaker characteristics.

increases the training time, making it hard to scale up. In the absence of training data from a particular domain, the domain shift between the training and testing data can affect the performance. Fortunately, large amounts of unlabeled video/audio data is being uploaded to internet daily, a significant portion of which can be used for adapting ASV models. Since the data is unlabeled, adaptation approach has to be unsupervised.

Domain adaptation is a branch of ML, that makes use of some data acquired from a *target* domain to adapt a system trained on a *source* domain to the *target* domain. In the specific case of unsupervised domain adaptation (UDA), the *target* domain data used for adaptation is unlabeled. Some popular approaches to UDA include mapping feature representations from one domain to the other [16], learning to extract features that are invariant to the domain from which they are extracted [17–19] or learning two features spaces: one component which is private to each domain and one which is shared across domains [20, 21]. In this work, we focus on the first approach – mapping features from one domain to the other.

In speech applications, a conventional feature mapping approach is to train a DNN to map acoustic features from one domain to the other using features extracted from unlabeled *paired* speech samples from both the domains. Paired audio samples refer to two audio samples (each sampled from individual do-

mains) that are recorded simultaneously[2]. Hence, they both have one-to-one correspondence between them i.e. both the samples are of same duration, spoken by same speaker and they have same linguistic content. An example of paired audio samples is close-talk speech and far-field speech recorded simultaneously. Speech recorded using a microphone (typically head-mounted) placed very close to the talker is usually referred as close-talk speech. Far-field speech is recorded using stationary microphones mounted at a distance from the talker[3]. Once the paired samples are acquired from both the domains, the training process of DNN is as follows: the DNN takes as input features from one particular domain and outputs certain features. To ensure the output features are identical to the features in the opposite domain, loss functions like mean squared error (MSE) or L_1 are used. MSE is measured between the output features of the DNN and the features extracted from audio sample (paired with the input sample) in the opposite domain.

Two speech applications that are developed using this approach are speech bandwidth extension (BWE) [22] and speech enhancement [23]. In BWE application, bandwidth of narrowband (NB) speech samples, usually sampled at 8kHz, is extended to match the sampling frequency of wideband (WB) speech (sampling frequency 16kHz). In this application, DNN takes NB features as

[2]Since audio samples from both domains are recorded simultaneously, we also refer to paired data as parallel data in this work. We use both the terms interchangeably.

[3]On the other hand, any two samples each drawn randomly from two different domains are considered *unpaired* with each other – since, there does not exist any one-to-one correspondence between them.

input and predicts WB features. On the other hand, speech enhancement application can be broadly divided into two categories: far-field speech enhancement (also termed as dereverberation) and noisy speech enhancement. In the former case, far-field speech (also termed as reverberant speech) is mapped to close-talk domain, whereas in the later case noisy speech is mapped to clean domain.

The above mentioned non-linear regression approach to train a DNN using *paired* data for both the applications suffers from several disadvantages:

1. Acquiring *paired* audio data is an expensive process, which limits the amount of data that can be made available for training the adaptation DNN. It also limits the amount of natural variations that can be acquired in the speech, since, *paired* data is usually acquired in controlled recording conditions. An alternate approach to acquire *paired* training data is using simulation – for instance, far-field speech simulated from close-talk speech [6], or simulated NB speech is obtained by downsampling WB speech. Though effective, it imposes a restriction on using the real data from the domain of interest for adaptation.

2. Noisy speech enhancement (or dereverberation) via regression mapping is good at improving the perceptual quality of *noisy* (or far-field) speech signal (measured by quantitative and qualitative evaluations) [23]. However, it has not proven to be very effective at improving the noise (or far-

field) robustness of speech applications like ASR [24] or SV.

3. In limited number of cases where speech enhancement seemed to help downstream applications like ASR [25] or SV [26,27], the testing was limited to simulated conditions. Simulated data, in nature, can be different from the real data which is usually acquired from uncontrolled environments (considered wild in this work). Hence, the enhancement networks trained and tested on simulated data may not generalize well to unseen conditions usually encountered in real data.

The main focus of this thesis is to address the above mentioned limitations of the non-linear regression approach using *paired* data. Firstly, we focus on training the DNNs on *unpaired* speech samples acquired from both domains. This framework enables us to use real data from both domains for training, thus avoiding the need for simulation. Use of *unpaired* samples for training prevents us from using MSE or L_1 loss functions, since the linguistic content in both the samples are different. Our training procedure is inspired from the cycle consistent generative adversarial network (CycleGAN) framework proposed for unpaired image-to-image translation. This framework uses a combination of adversarial [28] and cycle-consistency [16] loss functions to train the adaptation DNN. Similar to the non-linear regression approach using *paired* data, the *unpaired* approach using CycleGAN also does not require labels for speech samples from both domains.

Secondly, we address the other limitation of the non-linear regression approach using *paired* data – loss in performance on a downstream task like SV. The loss in performance can be attributed to distortions introduced by the loss functions used in the regression mapping approach [29]. To circumvent the distortion problem, we employ adversarial [28] loss to train the DNN along with MSE.

Finally, unlike previous work, we demonstrate the effectiveness of our proposed techniques on a wide range of simulated and real test sets sampled from challenging wild (uncontrolled) conditions.

We experiment with three domain mismatch scenarios encountered by ASV systems: channel mismatch [30, 31], acoustic mismatch between degraded vs clean conditions [31–33], and sampling frequency mismatch scenarios [34, 35]. We present results on several simulated and real data sets acquired from uncontrolled 'wild' conditions. We present evidence that our choice of loss functions reduces the distortion introduced in the regression approach, and thus, improve SV performance. Our work also addresses the over-fitting problem experienced by feature mapping networks when trained on limited amounts of data, and we present a simple technique to regularize training. Our UDA approach complements SOTA SV systems trained using data augmentation – a supervised adaptation approach. More importantly, since our approach does not require labeled data from both domains, the adapted features can be used

for any speech application like ASR.

The main contributions of this work are as follows:

1. Our work is the first to demonstrate the effectiveness of CycleGAN framework on two domain mismatch scenarios commonly encountered by ASV systems: channel mismatch [31,35], and acoustic mismatch between clean and wild testing conditions [31–33].

2. Unlike previous speech enhancement works, which focus on either denoising or dereverberation, our framework demonstrates improvements on both noisy and far-field conditions [32, 33].

3. Our work is the first to address the over fitting scenario observed by the feature mapping DNNs, when trained on limited amount of data. We propose a simple strategy to regularise the training [31].

4. Unlike previous approaches which present results on simulated testing conditions, we present results on both simulated and wild testing conditions [32, 33].

5. Unlike previous works which focus on either *paired* [23] or *unpaired* [25] feature mapping approaches, in our work we provide a comparison of both the approaches [33].

Rest of the thesis is organized as follows: in Chapter 2, we present the training details of current SOTA ASV system and give an overview of the several

mismatch scenarios considered in this work. In Chapter 3, we discuss the problem of sampling frequency mismatch and the adaptation approaches in detail. In Chapter 4, we address the channel mismatch scenario where we train an adaptation DNN on *unpaired* data acquired from telephone and microphone domains. Finally, in Chapter 5, we present adaptation techniques to make the ASV system more robust to far-field and noisy testing conditions.

Chapter 2

Background

2.1 Introduction

The field of automatic speaker recognition can be broadly divided into two categories: *speaker identification* and *speaker detection*. The task of *speaker identification*, as the name suggests, involves identifying a person from his/her voice. This is closely aligned with the natural way humans approach the recognition task. The task is well poised when trying to identify a speaker from a known set of speakers, also known as 'closed set identification'. To automate this process would involve making hard design considerations like determining the number of test speakers and their distribution. The accuracy of the task also depends on the number of speakers to be evaluated. More realistic approach to the problem would be to identify an *unknown* speaker, an 'open set

identification' task. Building and evaluating 'open set' tasks remain an active area of research.

The limitations in the design of *speaker identification* task – determining the number of test speakers, and their distribution – can be overcome by reformulating the speaker recognition task as a *detection* problem. Formally, given two speech recordings, each spoken by a single speaker, the task of *speaker detection* system is to find out whether both the recordings were spoken by same speaker or not. Speaker verification (SV) can be achieved from *detection* by considering one recording as spoken by a test speaker, and the other as spoken by an enrolled speaker.

The content spoken by the speakers being evaluated also determines the form of recognition system. If the speech content is known to the evaluator in advance, the process becomes *text-dependent*. If no restrictions are posed on the content, the task is *text-independent*. In this work, we experiment with automatic *speaker recognition* systems that are designed to perform *text-independent speaker verification* task. We refer to such systems as automatic speaker verification (ASV) systems. In the next section, we present the main design components involved in a typical ASV pipeline.

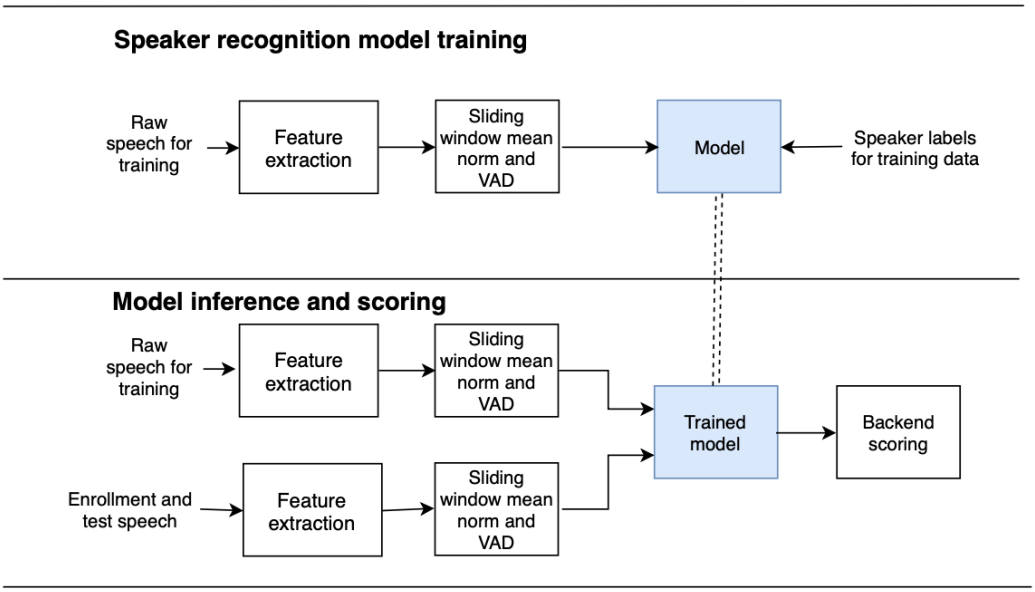


Figure 2.1: Speaker verification model pipeline

2.2 Speaker Verification Pipeline

The outline of a conventional ASV system is shown in Figure 2.1. It comprises of four major stages: (1) acoustic feature extraction, (2) feature normalization and voice activity detection (VAD), (3) training an embedding extractor model, and (4) speaker modelling and scoring.

2.2.1 Acoustic Feature Extraction

The process of transforming raw speech samples to spectral vector representations is termed as feature extraction. Short-term spectral features are extracted by dividing the raw speech signal into a window of 25 milli seconds (msec). chunks with 10 msec. shift between successive windows, and perform-

ing spectral analysis on each window.

Mel-frequency cepstral coefficients (MFCC), a form of short-term spectral feature extraction, are the most widely used in ASV and ASR systems. MFCC feature extraction pipeline is outlined in Figure 2.2. The time domain signal is first pre-emphasized to amplify high-frequencies. The intensity of high-frequencies is lower than the low-frequencies, and this operation brings them both to equilibrium. The signal is then divided into 25 msec. windows as explained above. To mitigate the effect of finite duration of the window on Fourier transform [36], a windowing operation is performed. Most common choices of window are Hamming or *Povey* [37]. Magnitude spectrum is computed with the fast Fourier transform (FFT). To integrate the energy present in several bands, triangular Mel filters are used, the output of which is compressed by performing a logarithm operation. A Discrete Cosine Transform (DCT) is applied to decorrelate the features and the first 12-40 coefficients are retained.

From the MFCC extraction pipeline, two intermediate features can be extracted: (1) log power spectrogram (LPS) features, and (2) log Mel filter-banks. The extraction of Mel filter-banks is motivated by the way human being's auditory system perceive sounds. The last two steps – log and DCT, are introduced for algorithmic convenience. Decorrelating filter-bank coefficients using DCT makes MFCC a very good choice of features for Gaussian mixture models (GMM) with diagonal covariance [38]. However, since decorrelation is not a

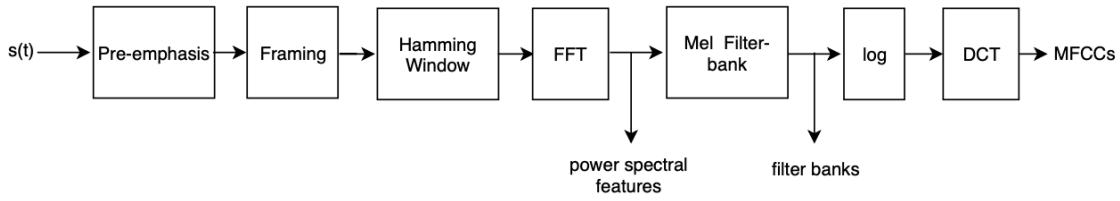


Figure 2.2: MFCC feature extraction pipeline

requirement for DNNs, both the intermediate representations are used in several applications. Several speech enhancement applications using DNNs use LPS features [23], since a speech signal can be reconstructed (retrieved) back from the power spectral features. Mel-filter banks are used for training DNNs in ASV systems [14] and ASR applications [39].

2.2.2 Feature Normalization and Voice Activity Detection

Following the acoustic feature extraction, features are normalized to reduce the acoustic mismatch between training and evaluation features, which improves robustness of the features and increases accuracy of the systems [40]. A simple form of feature normalization technique, termed as cepstral mean normalization (CMN) [41], is widely used. Specifically, a mean vector is computed from several frames of acoustic feature vectors. The mean vector is then subtracted from each frame, which makes the long term average of the normalized features zero. CMN makes the features robust to linear filtering on

the cepstral features, which might be caused by different microphones, room reverberation, and varying distance from mouth to microphone. All these operations are convoluted in time domain which makes it additive in cepstral domain, which explains the effectiveness of CMN in increasing the robustness. VAD is then applied to remove the unvoiced frames from the features. Energy based VAD implemented in Kaldi [37] is widely used.

2.2.3 Embedding Extractors

Different speech signals can have different durations, making it hard to compare two speech signals to verify the speaker identity. To overcome this disadvantage, the field of SV embraces the use of embedding extractors, which convert acoustic features of variable duration to fixed dimensional vector representations (termed as embeddings). Below we explain two most commonly used frameworks for extracting speaker embeddings – i-vectors [1] and x-vectors [14]. The former uses a factor-analysis [2] framework to extract the embeddings, whereas the later uses a deep learning [42] framework.

2.2.3.1 i-vector

A feature extraction approach based on factor analysis, was proposed to extract speaker embeddings [1], termed as identity vectors (i-vectors). This approach uses a total variability space that contains both speaker and channel

variability. The GMM super-vector for a given utterance is defined as follows

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\phi \tag{2.1}$$

where $\mathbf{m}$ is the mean of universal background model (UBM), $\mathbf{T}$ is a low-rank matrix defining the total variability space, and ϕ is a standard normal distributed vector. ϕ is referred as total variability factors or *identity vectors* (*i-vectors*), which are used as features (embeddings). In [1], linear discriminant analysis (LDA) and within-class covariance normalization (WCCN) compensate the channel distortion in the i-vectors. Then, scoring is produced by cosine distance or support vector machines (SVM).

2.2.3.2 x-vector

Authors in [14] proposed a DNN architecture that was discriminately trained to classify speakers given variable length acoustic frames as input. Input acoustic features pass through a time-delay neural network (TDNN) [10], which operate at the frame level. This is followed by a statistics pooling layer which computes the mean and standard deviation, the concatenation of which gets passed to two affine layers. The output of last affine layer is fed to a final softmax layer, which is of the dimension of number of speakers present in the

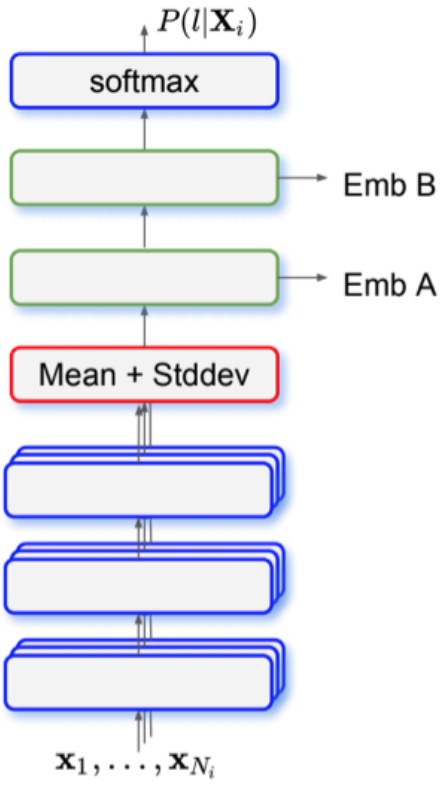


Figure 2.3: DNN embedding extractor architecture [15]

training corpora. The non-linearity used is rectified linear unit (ReLU). The network is trained with cross entropy loss.

An overview of network architecture is shown in Figure 2.3. Once trained, the output of the first affine layer that follows the statistics pooling layer is used as speaker embedding (typically of 512 dimension). Speaker embedding is shown as 'Emb A' in the Figure.

2.2.4 Speaker Modelling and Scoring

Speaker embeddings are modelled by probabilistic linear discriminant analysis (PLDA) [43]. In this approach, an embedding ϕ_{ij} from session j of speaker i is represented as

$$\phi_{ij} = \mu + \mathbf{V}\mathbf{y}_i + \mathbf{U}\mathbf{x}_{ij} + \epsilon_{ij} \tag{2.2}$$

where μ is a speaker independent term, $\mathbf{V}$ is a low-rank matrix of eigenvoices, $\mathbf{y}_i$ is the speaker factor vector, $\mathbf{U}$ is a low-rank matrix of eigen-channels, $\mathbf{x}_{ij}$ is the channel factor vector and ϵ_{ij} is an offset that accounts for rest of channel variability. The prior distribution on ϵ is diagonal Gaussian. Simplified PLDA (SPLDA) model puts aside the eigen-channels term and assumes a full covariance Gaussian for the ϵ prior.

PLDA is scored by computing the ratio between the likelihood of the trial x-vectors given the target hypothesis and the corresponding likelihood given the non-target hypothesis. If the speaker in the enrollment and test embeddings is the same M_1, both embeddings share the same speaker factor $\mathbf{y}$ but have different channel offsets. On the other hand, if they belong to different speakers M_0 they have different speaker factors. The ratio is expressed as below

$$\mathbf{LLR} = \frac{P(\mathbf{w}_1, \mathbf{w}_2/\mathbf{M}_1)}{P(\mathbf{w}_1, \mathbf{w}_2/\mathbf{M}_0)} = \frac{\int P(\mathbf{w}_1, \mathbf{w}_2/y_1)P(\mathbf{y}_1)\, d\mathbf{y}_1}{\int P(\mathbf{w}_1/\mathbf{y}_1)P(\mathbf{y}_1)\, d\mathbf{y}_1 \int P(\mathbf{w}_2/\mathbf{y}_2)P(\mathbf{y}_2)\, d\mathbf{y}_2} \tag{2.3}$$

In the **LLR** computation, speaker identity variables are integrated out instead of computing point estimates. To take into account the uncertainity about the value of y and to compare the enrollment and test embeddings, the likelihood is computed under the assumption that both are generated by the same y regardless of its exact value.

The speaker verification decision is taken by comparing the score provided by the classifier with a threshold. If the score is higher than the threshold the target speaker is accepted, otherwise it is rejected. The choice of the optimum threshold is troublesome in speaker verification due to the score variability between trials. The score variability may be caused by different phenomena. They include phonetic content of the utterances, length, channel type, noise, emotion or any other type of inter-session variability. To reduce that variability, score normalization [44] was applied. The last step before making a decision is calibration [45]. Calibration refers to selecting the optimum decision threshold for verification.

2.2.5 Performance Evaluation

During evaluation, for each of a large set of speaker detection trials an ASV system is required to make a binary decision – to accept or reject an enrolled speaker. There are two types of trails : *target trials*, where the target speaker is present in the input speech; and *non-target trials*, where the target speaker is absent. The decisions are compared against a truth reference. Two types of errors can occur in this scenario: a false reject (FR) (also termed as a miss) or false accept (FA) (also called as false alarm). A FR occurs when a valid target speaker is rejected. A FA happens when an impostor is accepted. Both types of errors depend on the decision threshold. A low decision threshold will produce low FR rate (P_{FR}) and high FA rate (P_{FA}). P_{FR} is the FR count, normalized by the number of target trials. P_{FA} is the FA count, normalized by the number of non-target trials. The pair (P_{FR}, P_{FA}) can be considered evaluation outcome, and would be used to compute a single evaluation metric.

A popular evaluation metric used in verification application is *equal error rate* (EER). It is defined as the error rate at the operating point where $\mathrm{P}_{FR} = \mathrm{P}_{FA}$.

Another popular metric – a cost based one – is the *detection cost function* (DCF) C_{Det}, which is a weighted sum of P_{FR} and P_{FA} and is denoted as

$$\mathrm{C}_{Det} = \mathrm{C}_{FR}\mathrm{P}_T\mathrm{P}_{FR} + \mathrm{C}_{FA}(1-\mathrm{P}_T)\mathrm{P}_{FA} \tag{2.4}$$

where C_{FR} and C_{FA} are the costs of having a miss or a false alarm respectively. P_T is the probability of target prior. C_{FR}, C_{FA} and P_T are application dependent parameters. The optimum operating point of the verification system is the pair (P_{FR}, P_{FA}) at which C_{Det} is minimum.

A normalized version of DCF, termed as C_{Norm}, is popularly used. It is defined as

$$\mathrm{C}_{Norm} = \mathrm{C}_{Det}/min(\mathrm{C}_{FR}\mathrm{P}_T, \mathrm{C}_{FA}(1-\mathrm{P}_T)) \tag{2.5}$$

In this thesis, we refer to C_{Norm} as DCF. Both C_{Norm} and C_{Det} depend on the decision threshold. It is common in the literature to evaluate systems with respect to their actual and minimum DCF. For actual DCF, the cost is computed for a fixed threshold. If the scores are well-calibrated likelihood ratios, the threshold is selected to minimize a Bayes risk [46]. However, if we select the threshold that minimizes the cost on the test dataset, we obtain a minimum DCF (*minDCF*), which allows comparison across the systems regardless of the calibration. $\mathrm{C}_{Norm}>1$ indicates that the system is not appropriate for the

intended application. DCF measures how well a detector actually performed when designed for and tested on a specific application. On the other hand, *minDCF* measures how well it could have performed if the score threshold had been perfect. For more detailed discussion on the topic, refer to [46]

2.3 State-of-the-art Speaker Verification System

During the time this thesis was written, SOTA ASV systems use x-vectors as speaker embeddings [14, 15]. As explained earlier, x-vectors are extracted from a DNN trained discriminatively to classify speakers on large amounts of labeled speaker data. Since, acquiring labeled speaker data is an expensive operation, authors in original x-vector work [14] proposed an approach based on data simulation techniques to increase the amount of labeled training data – method popularly known as data augmentation. Authors applied two types of simulation – artificially adding noise to the speech, and simulation of far-field speech. Noise files from Music, Speech and Noise (MUSAN) [47] corpus were used for additive noise based simulation, and room impulse response (RIR)s used in [48] (and made publicly available at `http://www.openslr.org/26`), which consisted of both simulated and real RIRs, for far-field speech simulation. MUSAN corpora consists of noise files from three different subcate-

gories: *music* files from several genres ($\approx$ 42 hours and 31 minutes), *speech* from twelve languages ($\approx$ 60 hours), and *noise* files consisting of wide assortment of technical and non-technical noises ($\approx$ 6 hours). The simulated RIRs used to create far-field speech were divided into three sets based on the ranges from which width and length of the room were sampled from: *small*, *medium* and *large*. The width and length of *small*, *medium* and *large* rooms sets were uniformly sampled from ranges 1-10m, 10-30m and 30-50m respectively. In all the three sets, room height was sampled uniformly from 2-5m; and absorption coefficient was sampled uniformly from [0.2; 0.8]. In each set, 200 rooms were first sampled and 100 RIRs were sampled in each room based on speaker and receiver position. The distance between the speaker and the receiver was not greater than 5m.

Simulation strategy was as follows: the authors in [14] assumed availability of a labeled training corpus. Three different noise subcategories from MUSAN corpus were used to create three different copies of noisy training corpora, by artificially adding noise to the original training data. An additional copy of far-field speech was created by convolving the simulated RIRs with the original speech from training corpus. A random subset of size typically 2 times the original training data was chosen from the four copies of simulated data, which was then used to augment the original training data. The data augmentation strategy, thus, increases the size of the training data by two fold. This strategy

was proven to be very effective in making x-vector networks a very significant component in SOTA ASV systems. Data augmentation was also used to improve the performance of ASR systems [24, 48–50].

2.4 Domain Adaptation Overview

The main focus of this work is to address the sensitivity of x-vector network to domain mismatch between train and test domains. We refer the distributions from which the corpora used to train and evaluate an ASV system are sampled as *source* and *target* domains respectively. When the domains are distinct enough, the system trained on *source* domain[1] would suffer a loss in performance when tested on the *target* domain. We treat such system as operating in a domain mismatch scenario. Procedure used to address the mismatch scenario, and thus, improve the performance of systems operating under mismatched conditions is called 'domain adaptation'. Depending on the data used to train the adaptation mechanism, domain adaptation techniques can be classified into supervised or unsupervised.

[1]For brevity, we refer to 'trained on data sampled from *source* distribution' as 'trained on *source* domain'. Similarly, 'evaluated on data sampled from *target* distribution' as 'evaluated on *target* domain'

2.4.1 Supervised vs Unsupervised Adaptation

In a supervised domain adaptation (SDA) setting, in order to improve the performance of the system under the mismatch scenario, we gather some labeled data from *target* domain. This labeled *target* domain data, along with the labeled *source* domain data, is used to learn (train) an adaptation mechanism. The labeled *target* domain data used in training should be different from the evaluation data used to test the system. Since acquiring labeled data is expensive, it is further assumed that the labeled data from *target* domain is limited to train the system solely on it, whereas the *source* domain data is abundantly available.

Unsupervised domain adaptation (UDA), on the other hand, refers to the setting where unlabeled data from *target* domain, along with the labeled *source* domain data, is available to learn an adaptation mechanism. Similar to SDA, we assume that the *target* domain used to learn the adaptation mechanism is different from the evaluation data used in the testing stage. Unlike the SDA setting, we assume that acquiring unlabeled data is not very expensive, and is available in abundance for training. UDA opens up opportunities to leverage large amounts of data made available for public on internet to build ML models. Moreover, since this data is usually collected in wild conditions, using this data for adaptation makes the ML models more robust to realistic testing conditions.

2.4.2 Overview of Proposed Domain Adaptation Approach

In this work, we consider several domain mismatch scenarios (discussed in Section 2.5) and propose several adaptation approaches. All the approaches are aimed at improving the verification performance on the *target* domain. As discussed in Section 2.2, ASV system comprises of three main stages – (1) acoustic feature extraction, (2) embedding extraction, and (3) speaker modelling and scoring. Depending on the chosen technique, domain adaptation can be applied in any of these stages. In this work, we explored domain adaptation at the acoustic feature level – the acoustic features of one domain gets mapped to the opposite domain, and the mapped (adapted) features are used either in the training stage or evaluation stage. In our approach, since adaptation is done at the beginning of the ASV pipeline, it makes both the x-vector and PLDA robust to domain mismatch between training and testing domains.

2.4.2.1 Adaptation During Training vs Adaptation During Testing

When the acoustic features of *source* domain are mapped to *target* domain, we train the embedding extractor DNN on the adapted features. During evaluation, the original *target* domain features (without any adaptation) are used

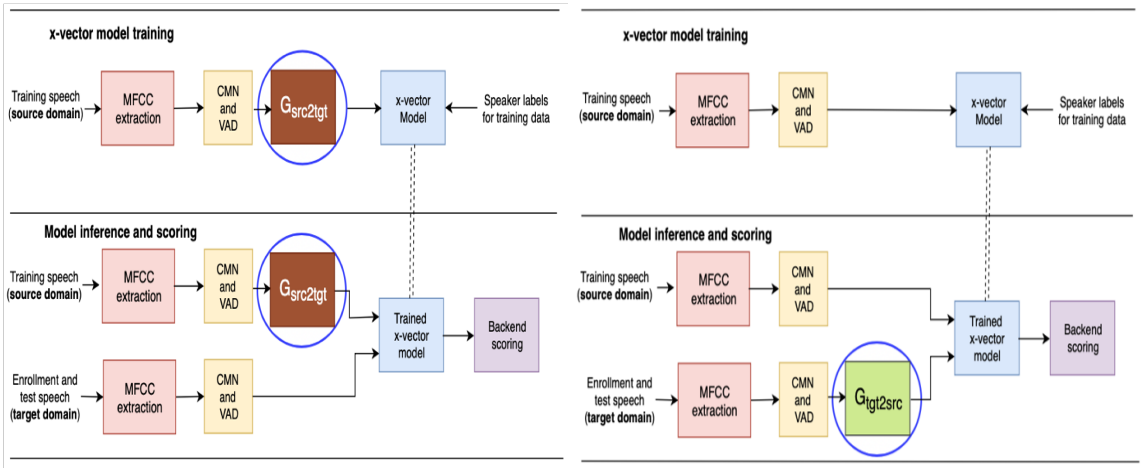
to extract embeddings for the test data. Since, the network was trained on adapted features, we consider this process as 'adaptation during training' (referred in Figure 2.4(a)). The adapted ASV system used in this process can be treated as trained and tested on *target* domain (assuming an ideal feature mapping mechanism).

We also explore 'adaptation during testing' (shown in Figure 2.4(b)). In this scenario, the ASV system is trained on *source* domain features. During evaluation, the *target* domain acoustic features are mapped to *source* domain. These mapped features are used to extract x-vectors used for scoring. This system can be treated as trained and tested on *source* domain. In this scheme, the same ASV system trained on *source* domain can be used to evaluate on multiple *target* domains, whereas, in the previous scheme a separate system has to be trained on adapted features for each individual *target* domain.

2.4.3 Feature Mapping Overview

As discussed above, mapping acoustic features of one domain to the opposite lies at the heart of all domain adaptation approaches in this work. In this approach, feature mapping function, typically a DNN, is first trained on features from both domains. Once trained, the DNN is used to map features to the opposite domain.

Our experimental framework is as follows: we assume the availability of



(a) Adaptation during training pipeline (b) Adaptation during testing pipeline

Figure 2.4: Adaptation during training vs adaptation during testing. In both approaches, adaptation is achieved by mapping acoustic features of one domain to other. (a) In adaptation during training, mapping is done from *source* to *target* domain and x-vector network is trained on mapped features, (b) in adaptation during testing, mapping is done from *target* to *source* domain during evaluation.

unlabeled data from both domains to train the feature mapping DNN. Since we use unlabeled data from *target* domain, the adaptation procedure is unsupervised. We consider two scenarios to train the DNN – (1) train with paired audio data from both domains, and (2) train with unpaired data from both domains.

2.4.3.1 Feature Mapping with Paired Audio Data

Paired audio samples refer to two audio samples (each sampled from individual domains) that are recorded simultaneously[2]. Hence, they both have one-to-one correspondence between them i.e. both the samples are of same length, spoken by same speaker and they have same linguistic content.

An example of paired audio samples is close-talk speech and far-field speech recorded simultaneously. Speech recorded using a microphone (typically head-mounted) placed very close to the talker is usually referred as close-talk speech. Far-field speech, on the other hand, is recorded using stationary microphones mounted at a distance from the talker. Speech recorded using distant microphones is affected by various acoustic conditions, like room reverberation and background noises. Close-talk speech can be considered free from such distortions. Hence, in this work we consider close-talk speech as *clean* speech. Far-field speech is considered as *degraded* speech. In practice, acquiring paired audio samples is not always feasible or economically viable. In this scenario, paired audio data can be obtained via simulation.

For some domain mismatch scenarios, we experimented with using *paired* data to train feature mapping DNN (details in Section 2.5). Given an audio sample from one domain, we create its pair in the opposite domain using simulation. For instance, in robustness to far-field speech scenario, far-field

[2]Since audio samples from both domains are recorded simultaneously, we also refer to paired data as parallel data in this work. We use both the terms interchangeably.

speech was simulated from close-talk speech by convolving with RIRs. Using the paired data, we train a DNN to map features from one domain to the other.

2.4.3.2 Feature Mapping with Unpaired Data

Acquiring paired audio data is an expensive process, which limits the amount of data that can be made available for training ML models. It also limits the amount of natural variations that can be acquired in the speech, since, paired data is usually acquired in controlled conditions or created via simulation. On the other hand, acquiring samples from unconstrained 'wild' conditions that are usually made available from open-source media is relatively less expensive. However, such data is usually made available without any ground truth labels (speaker labels in this scenario).

In this work, for some domain mismatch scenarios, we experimented with using unpaired data from both domains to train the feature mapping network. Typical experimental setup is as follows: we sample unlabeled *target* data from open-source media. For the *source* domain data, we use the same data that we use to train the x-vector network. Since, training data from both the domains are sampled separately from different sources, it is considered unpaired. We train the adaptation DNN using this unpaired data (details in Section 2.5).

In the next section, we describe several domain mismatch scenarios we addressed and the corresponding adaptation approaches. We present an overview

of the training procedure of feature mapping DNNs for each of this task.

2.5 Overview of Domain Mismatch Scenarios and Adaptation Approaches

We experimented with three different domain mismatch scenarios: (1) mismatch in sampling frequency between *source* and *target* domains, (2) mismatch in channel (used to record speech), and (3) mismatch in quality of speech (clean vs degraded) between both the domains. In all the three cases, we assume the ASV system was solely trained on data from *source* domain and tested on *target* domain, except for the sampling frequency mismatch scenario where we assume a limited amount of labeled data from *target* domain is available during training. As explained earlier, in all the three scenarios, domain adaptation was achieved with the help of a feature mapping DNN. The training procedure of DNN (training data and loss functions used) differ between the scenarios. In this section, we describe all three scenarios and present a high level overview of corresponding domain adaptation approaches. We will also present previous work on each of this topics.

2.5.1 Sampling frequency mismatch

Different devices record speech at different sampling rates, and thus, create a mismatch later on while training speech models. When these experiments were performed, a considerable amount of speech data available to train ASV systems was recorded at 8 kHz, mostly telephone conversations. We will refer to these data as NB speech, in this context. On the other hand, a limited amount of speech is recorded at 16 kHz, such as far field microphone speech. We will refer to this as WB speech.

In this scenario, we consider the availability of large portion of NB data and a limited amount of WB data, both labeled, for training the ASV system. We also consider the system would be evaluated on WB data. Since, both the datasets are sampled at different frequencies, the traditional approach to combine both the datasets during training is to downsample the WB data to match the sampling frequency of NB speech and train the ASV system on the combined dataset. During evaluation, the WB speech from test set also gets down sampled to match the training sampling frequency. However, downsampling operation degrades performance of speech models as it throws away information in the upperband (UB) of WB speech (8-16kHz) that could potentially be meaningful later on. Therefore, bridging the gap between sampling mismatch and information loss could increase the quality of training data and potentially improve the performance of ASV system especially when tested on WB data.

In this work, we experimented with two main approaches to address this issue: (1) mixed bandwidth training [34], and (2) extending the bandwidth of NB speech to match the sampling frequency of WB speech [35] – procedure termed as band-width extension (BWE) in the literature. Both techniques upsample the NB speech (to match the sampling frequency of WB speech). The WB training and evaluation data remain unaltered. Hence, no information loss occurs. However, both the techniques differ in the way NB data gets upsampled. In the first approach, the NB speech was upsampled using a basic upsampling technique to match the sampling frequency of WB speech. Basic upsampling would not predict any information in the upperband (UB). The BWE approach overcomes this disadvantage. In this approach, a DNN is trained to map (upsample) the NB speech to WB domain. The network is trained to predict the entire information in the WB band. It, thus, overcomes the disadvantage of basic upsampling. We train the DNN on acoustic features extracted from paired NB-WB data. The paired data was obtained using simulation – by downsampling the WB data. The training of BWE network does not require any speaker labels. Hence, the domain adaption procedure via BWE is unsupervised. Since, the ASV system was trained on adapted (upsampled) features and evaluation data was kept unchanged, the procedure would fall under 'adaptation during training' category (details in Section 2.4.2). We discuss both the BWE approach and basic upsampling approach in detail in Chapter 3.

Both BWE and mixed band-width training, have been explored in the past for training ASR systems. Early work showed that WB spectrum can be predicted from extending NB spectral envelope by a linear model and exciting it with white noise [51]. The linear model poses the assumption that speech is relatively smooth and linear in frequency domain. Most recent work focused on exploiting the capability of DNN, given its success in several tasks in speech processing and analysis, DNN-based speech BWE has demonstrated improvement in ASR. A feed-forward DNN trained on log spectrogram features of NB and WB data was incorporated in an ASR system [22]. The authors were able to show that DNN is capable of extending band-width of a signal, and that these features, when used to train a downstream task like ASR, would give better performance compared to system trained only on NB data. Multi-task learning and transfer learning were explored as a means of assisting multilingual task and cross-lingual task in [52]. Their BWE network was trained on bandlimited WB data and further retrained on NB data and achieves a subsequent 45% relative word error rate (WER) reduction. An alternative approach to BWE is to modify the NB features by applying some transformation on them to match some specific properties of WB data, and then train a neural network, along with the available WB data. Authors in [39] trained a mixed band-width ASR on log-mel filter banks. Authors use 22 and 29 dimensional filter banks for NB and WB data respectively. The filters are designed such that the first 22

filters of WB data are aligned with that of NB data. The NB features are zero padded (transformed) to match the dimension of WB features. The neural connections of the network are optimized to learn from the first 22 filter banks for the NB data and the entire feature vector for the WB data. All the techniques discussed so far are developed for ASR applications. In this work, we focus on exploring mixed bandwidth (mixed BW) training and BWE for improving SV performance (details in Chapter 3).

2.5.2 Channel Mismatch

In this scenario, we approach the channel mismatch between the telephone and microphone speech[3] perspective. We consider the scenario where an ASV system is trained on speech acquired using telephone channel and evaluated on microphone speech (similar to sampling frequency mismatch scenario). The authors in [14] approached this problem by augmenting the telephone data with microphone speech from the VoxCeleb dataset [53] and obtained a significant improvement on Speakers In The Wild (SITW) test set. However, this is a supervised domain adaptation (SDA) approach which requires access to labeled in domain data. Acquiring labeled data is a time consuming and an expensive approach. On the other hand, getting access to unlabeled microphone speech is relatively easy. To avoid mismatch in channels between both

[3]We assume each channel to have a specific transfer function, which would affect the speech characteristics and its acoustic features

domains, we train a feature mapping DNN on unlabeled data from both the domains. During evaluation, we map the microphones features to telephone domain and use those mapped features to extract embeddings using a x-vector network trained on telephone speech [30,31]. Hence, this adaptation procedure falls under 'adaptation during testing' scenario (details in Section 2.4.2).

To train the feature mapping DNN, we assume the availability of unlabeled microphone data (different from evaluation data) during training. Since, it is unlabeled it cannot be used for training the ASV system. This microphone data along with telephone data used to train the x-vector network is used to train the feature mapping DNN. The training data from both domains is sampled from different sources which makes the data *unpaired*. Hence, we cannot train the DNN using the regression approach that we used in the BWE approach. Inspired by the work in *unpaired* image translation [16], we train the network using a combination of adversarial [28,54] and cycle-consistency losses [16] (details in Section 2.5.4). The training procedure does not require speaker labels from any domain. Hence, the domain adaptation approach is unsupervised. The advantage of training the network on *unpaired* data is that we could use real data sampled from both distributions, thus, avoiding the need for using simulated data. Similar to the BWE network, the DNN we used to map features is a deep residual CNN with an encoder-decoder structure. Experiments and training procedure are detailed in Chapter 4.

2.5.3 Robustness to Far-field Speech

In this scenario, we experimented with checking the effectiveness of our feature mapping approach in making ASV system robust to degraded speech. We refer to degraded speech as far-field speech (also termed as reverberant speech) with additional back ground noise acquired in uncontrolled (wild) environments. We experimented with two ASV systems: one trained solely on clean speech and other trained on both clean and degraded speech, referred as multi-condition data. We test both systems on several far-field and noisy data sets created using simulation. Along with simulated test sets, we also test our approach on real degraded speech obtained from wild conditions. Adaptation was done during testing, where far-field speech was mapped to clean domain using a feature mapping DNN. The mapped features were used to extract embeddings used for scoring. Since, the DNN maps acoustic features from far-field to clean domain, the adaptation procedure can be termed as 'feature enhancement' or 'feature dereverberation'.

We experimented with training feature mapping DNNs on both paired and unpaired training data. Paired training data was obtained by simulating far-field speech from clean speech, whereas unpaired training data was sampled from real wild life scenarios. We provide a comparison of both the approaches.

Previously, enhancement techniques have been proposed for improving the noise robustness of ASR and ASV systems. Non-linear regression based ap-

proach using a DNN has been proposed in [23] to improve the noise robustness of ASR systems, with results on simulated datasets. Denoising approach using CycleGAN was proposed by [25] to improve the performance of ASR with results reported on several simulated test conditions. For SV, [27] and [55] have reported improvements on simulated data. As opposed to previous work, where results were only presented on simulated datasets, we demonstrate the effectiveness of our approach on both simulated (far-field and noisy) datasets and real speech sampled from wild conditions.

2.5.4 Related work

As explained earlier, adversarial loss and cycle-consistency loss are at the heart of most feature mapping approaches in this work. Adversarial loss was first proposed in [28], where authors proposed a new frame work for estimating generative models with the help of an adversarial process. The frame work consists of two networks: a generator G and a discriminator D. Generative model G captures a data distribution. Discriminator D is trained to be a binary classifier, that estimates the probability of a sample coming from a training distribution rather than G. The weights of G are updated to maximize the probability of D making a mistake, termed as adversarial loss (procedure detailed in Section 4.3.1.1).

Plethora of work used adversarial loss for various applications. It was used

in super resolution to create photo realistic images [29], for domain adaptation [56, 57], and for multi-domain image translation [16, 58]. The network in [16], termed as CycleGAN, was proposed in computer vision to learn mapping functions of images between domains with non parallel data. The network makes use of cycle-consistency and adversarial loss to learn the mapping functions. The procedure is outlined here and discussed in detail in Section 4.3.1.1. To learn a mapping from domain X to domain Y, a generator G was trained using adversarial loss such that distribution of images from $G(X)$ become indistinguishable from Y. In order to minimize loss of information during this mapping, additional constraint is added to the training process with the help of a network F such that $F(G(X) \approx Y$, which is ensured by minimizing the cycle-consistency loss. Qualitative and quantitative results demonstrate the efficiency of CycleGAN in achieving cross domain image transfer learned without any parallel data. CycleGAN soon found their way to speech research where they were used for adapting automatic speech recognition (ASR) trained on clean speech to noisy speech [25, 59], voice conversion [60–62], and gender adaptation [63].

Authors in [64], proposed a speech enhancement network that operates on the time domain waveform based on the *generative adversarial networks* (*GAN*) framework. The network, termed speech enhancement generative adversarial network (SEGAN), was trained in an end-to-end fashion on 28 speakers and 40

different noises. The benefit of this approach is that a single network, whose parameters are shared across all noise conditions, is capable of enhancing different noise conditions. Subjective and objective evaluations demonstrate the effectiveness of this approach in improving the perceptual quality of the signal.

Unlike the above approach, which focussed on improving the perceptual quality of speech signal, other enhancement techniques have been proposed for improving the noise robustness of ASR and ASV systems. Denoising approach using CycleGAN was proposed by [25] to improve the performance of ASR with results reported on several simulated test conditions. For SV, [27] and [55] have reported improvements on simulated data. For feature denoising in ASV, deep feature loss (DFL) [65] in lieu of feature mapping loss is proposed in [66, 67].

Our feature enhancement method differs from the above in two main aspects: (1) we focus on far-field enhancement as opposed to the denoising task that was addressed in the above discussed methods, (2) we present improvements on both wild and simulated test conditions as opposed to the above methods that present results only on simulated conditions. The test sets we use in this work are much diverse and larger compared to the ones discussed above.

Mismatch Scenario	**Domains**		**Adaptation during**	
Chapter	*Source*	*Target*	training	testing
Mismatch in sampling frequency				
Chapter 3	narrowband	wideband	Y	N
Channel mismatch				
Chapter 4	telephone	microphone	N	Y
Robustness to far-field and noisy speech				
Chapter 5	clean	degraded	N	Y
	multi-condition	degraded	Y	Y

Table 2.1: Summary of domain mismatch scenarios addressed in this work. *Source* and *target* domains represent distributions from which training and evaluation datasets are sampled from. For description of 'adaptation during training' and 'adaptation during testing' refer to Section 2.4.2.1

2.6 Summary

We described three domain mismatch scenarios addressed in this work: (1) mismatch in sampling frequency between *source* and *target* domains, (2) mismatch in channel between both domains, and (3) mismatch in acoustic conditions between domains. Table 2.1 summarizes the *source* and *target* domains used for each of these scenarios. We also gave an over view of adaptation approaches we developed to tackle the mismatch scenarios.

In Chapter 3, we discuss the problem of sampling frequency mismatch and the adaptation approaches in detail. In Chapter 4, we address the channel mismatch scenario. In Chapter 5, we present adaptation techniques to make the ASV system more robust to far-field and noisy testing conditions.

Chapter 3

Bandwidth Extension For Improved Speaker Verification

3.1 Introduction

In this chapter, we consider the scenario of training ASV systems with data gathered from two different domains – telephone and microphone speech. Telephone and microphone speech are usually sampled at 8 kHz and 16 kHz respectively. From now on, we will call 8 kHz data as narrowband (NB) and 16 kHz data as wideband (WB), and band between 4-8 kHz as upperband (UB). The conventional way of combining datasets with different sampling rates is to downsample all of them to the lowest sampling rate – 16 kHz to 8 kHz in our case. This results in loss of information in the UB of the WB data. The

downsampled microphone speech along with the telephone speech is then used to train the ASV system. Since the sampling frequency of entire training data is 8 kHz, we call this system as NB system. This method works fairly well when the evaluation data set is also telephone speech. However, it is not optimal when the evaluation data is WB since information in the UB is lost. This information loss could hurt the potential performance of the ASV system. Another option is to train the system only on the available WB corpus avoiding the need for downsampling. We call this system as WB system, since the sampling frequency of training data is 16 kHz. But this approach would lead to poor results when the available WB data is scarce. The experiments in this section demonstrate this scenario.

The rest of the chaper is organized as follows: we first explain the telephone and microphone datasets used in our experiments, and the NB and WB baseline systems in Section 3.2. In Section 3.3, we explain the training procedure of mixed BW system and present results. In Section 3.4, we discuss the neural BWE systems explored in this work. In Section 3.5, we summarize our experiments.

3.2 Baseline Systems

We first explain the datasets used in this work, and then explain the training procedure of NB and WB baseline systems.

3.2.1 Datasets Description

A small portion of WB dataset that we used in this chapter was collected from Mixer6 and NIST SRE08 corpus. These data are comprised of microphone recordings of telephone calls. The same speaker was recorded on several microphones. Hence, there exists a lot of redundancy. There is also speaker overlap between the telephone corpus and the microphone data. Major portion of the WB corpus comes from *VoxCeleb* dataset [53] which contains speech from celebrity speakers. WB data consist of 30974 utterances collected from 1871 speakers.

The NB data we used for this work comprises SwitchBoard 2 Phases 1, 2 and 3, SwitchBoard Cellular and NIST 2004 - 2010 including Mixer 6. For NIST SRE08 and MX6 there exists some speaker overlap between the NB and WB datasets. The NB dataset consists of 86594 utterances collected from 7001 speakers. As can be observed, the WB dataset is limited in size (1871 speakers) compared to the NB dataset (7001 speakers).

Speakers In The Wild (SITW) [68] is used as the WB dataset for evaluating

our models. SITW consists of variable length utterances from 6-240 seconds. Speech from this corpus consists of audio from videos of native English speakers, with naturally occurring noises, reverberation and device variability. The sampling frequency of this dataset is 16 kHz. We tested our models on the *Core* and *Assist* conditions of SITW (details in [68]).

3.2.2 Narrowband and Wideband Systems

We experimented with two x-vector systems: a NB system, and a WB system. The NB system used 23 MFCC features based on 23 mel filter-bank. Features were short-time mean normalized with 3 second sliding window and silence frames were removed. The x-vector system was trained using Kaldi [37]. Full rank PLDA [69] was used for scoring. PLDA scores were normalized using adaptive symmetric norm (S-Norm) [70]. NB system represents the conventional way of training speaker recognition systems – WB data was downsampled to match the sampling frequency of telephone data. Training data comprised of NB telephone data, and downsampled microphone (details above in Section 3.2.1).

WB system used 23 MFCCs with 30 filters Mel filter-bank. WB system was trained only on the available WB speech (details above in Section 3.2.1). Thus, this system was trained on much less speakers than NB system (1871 compared to 7001 speakers).

	SITW *Core*			**SITW *Assist-Multi***		
	EER	**DCF(1E-2)**	**DCF(1E-3)**	**EER**	**DCF(1E-2)**	**DCF(1E-3)**
NB system	6.01	0.5111	0.7105	8.88	0.5569	0.7351
WB system	8.02	0.5538	0.7505	8.54	0.5049	0.6880

Table 3.1: Comparison of narrowband (NB) and wideband (WB) systems. NB system was trained on data acquired from original telephone speech and down-sampled microphone speech (7001 speakers) whereas WB system was trained only on original microphone speech (1871 speakers).

Among the NB and WB systems, we observed NB system performed better than WB system, as shown in Table 3.1. This is mainly because of the limited amount of WB data available to train the data hungry x-vector model compared to NB system. The NB system, though performed better than WB system, has the disadvantage that the information in the UB of the WB training and evaluation data is lost. No information loss occurred in the WB system but the training data was much smaller, since we have not used NB data.

To overcome these two disadvantages we proposed two approaches in this work. We first experimented with a mixed BW system [34] where the audio samples from NB telephone corpora were upsampled with a low-pass filter interpolator and the WB microphone training and evaluation data were used without any modification (details in Section 3.3). We then experiment with several neural network architectures that upsamples (maps) NB features to WB domain [35], details in Section 3.4.

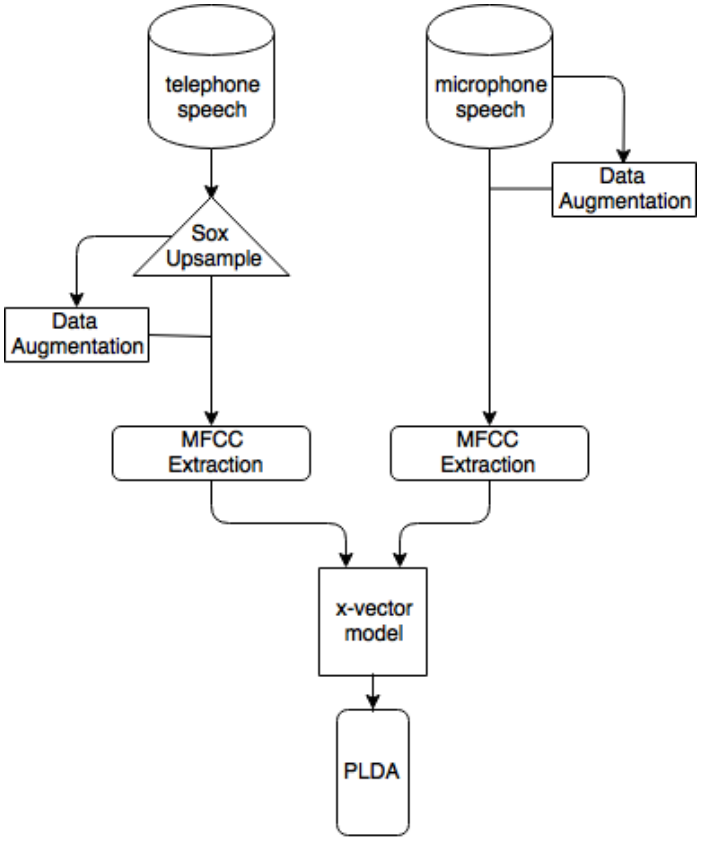


Figure 3.1: Training pipeline of mixed BW system [34]

3.3 Mixed Bandwidth System

The training procedure for mixed BW x-vector system is depicted in Figure 3.1. In this system, NB speech gets upsampled using a basic upsampler. The upsampled speech and original WB speech are used to train the x-vector network. Hence, this procedure falls under 'adaptation during training' strategy (discussed in Section 2.4.2.1). We first explain the upsampling procedure below, which will be followed by the training procedure.

3.3.1 Upsampling with low-pass filter interpolator

As shown in Figure 3.1, telephone speech gets upsampled using basic upsampler[1] in mixed BW system. Basic upsampling is a traditional method used in signal processing. In this technique, zeros are interpolated between each wav sample. A low pass filter eliminates the aliases created in the higher band, i.e., interpolates the unknown signal values. We used the implementation in SoX[2], which used a filter with 125 dB of attenuation for the rejected band. Note that this upsampling approach does not to fill in any additional frequency information in the upper half of the spectrum but preserves the information present in the lower half.

3.3.2 Mixed BW system trained without data augmentation

We first trained a mixed BW system without data augmentation. NB data was upsampled using the technique described above in Section 3.3.1. This upsampled data was combined with original WB data. Similar to the WB system (explained in Section 3.2.2), we extracted 23 dimensional MFCCs with 30 mel-

[1]We use the terms basic upsampler and linear upsampler interchageably in this work
[2]http://sox.sourceforge.net

	SITW *Core*			SITW *Assist-Multi*		
	EER	**DCF(1E-2)**	**DCF(1E-3)**	**EER**	**DCF(1E-2)**	**DCF(1E-3)**
Without data augmentation						
NB	6.01	0.5111	0.7105	8.88	0.5569	0.7351
WB	8.02	0.5538	0.7505	8.54	0.5049	0.6880
Mixed BW	**5.93**	**0.4711**	**0.6713**	**7.62**	**0.4890**	**0.6667**
With data augmentation						
NB	4.54	0.623	0.425	6.74	0.650	0.468
Mixed BW	**4.40**	**0.570**	**0.376**	**6.57**	**0.612**	**0.435**

Table 3.2: Results of mixed BW system on SITW with and without data augmentation. For description of NB and WB baseline systems, refer to Section 3.2.2. For description of mixed BW system, refer to Section 3.3

filterbanks on the training data. We then trained a mixed BW x-vector system on these features. Since, we used both NB and WB corpora for training, the mixed BW system was trained on the same amount of data as the NB system (explained in Section 3.2.2). Since the mixed BW system was trained at 16 kHz sampling frequency, no downsampling is required during evaluation.

The results of the mixed BW system trained without data augmentation on SITW are presented in the upper half of Table 3.2. As observed from the results, the mixed BW system performed better than the baseline systems on both test conditions – SITW *Core* and SITW *Assist-Multi*. We observed relative improvements of 7.82% and 12.2% (in terms of DCF) on SITW *Core* and *Assist-multi* conditions compared to NB baseline systems.

3.3.3 Mixed BW system trained with data augmentation

In this section, we trained a mixed BW system with data augmentation. Similar to the experiments in above section, we first upsampled the original NB data using the technique described in Section 3.3.1. This upsampled data was combined with original the WB data. We then upsampled the NB data used for augmentation. NB data for augmentation was obtained by adding noise to the original NB data. This additional data was augmented to the original training data. The combined data was used for training. We used data augmentation on the training data to increase the amount and diversity of the data. We extract 24 dimensional MFCCs with 30 mel-filterbanks on the original and augmented data. We then train a mixed BW x-vector system on these features. To compare the performance of mixed BW system with data augmentation, we also trained the NB system with data augmentation. The NB baseline system was trained on 23 dimensional MFCC features with 23 mel filter banks. Entire NB data available was used for training the system along with downsampled WB data. To test the NB baseline system, WB evaluation test set was downsampled to 8 kHz to match the sampling frequency of the training set. Similar to the mixed-BW system, the protocol we followed to augment the data is similar to Section 3.3 in the original x-vector work [14]. Protocol used to augment both systems

was mentioned in Section 2.3.

The results of mixed BW system on SITW trained with data augmentation are presented in the lower half of Table 3.2. The mixed BW system performed better for both evaluation conditions compared to the baseline. We observed relative improvements of 8.5% and 5.8% (in terms of DCF) on SITW *Core* and *Assist-multi* conditions compared to NB baseline systems. The improvement of mixed BW system can be attributed to the fact that the x-vector model, being a DNN, is able to optimize its neurons to respond to the lower half of the spectrum for NB data and to use the entire spectrum (all cepstral coefficients) for the WB data. The main advantage of mixed BW system is that the information in the upper half of WB training and evaluation data are retained as opposed to the NB system.

3.4 Neural bandwidth extension (BWE) Systems

Our experiments with mixed BW system in previous section demonstrated the advantage of preserving information present in the upper half of the WB spectrum during both training and evaluation. In mixed BW system, NB speech was upsampled using a basic upsampler during training. However, the basic upsampling approach do not estimate the spectrum in the upperband, which

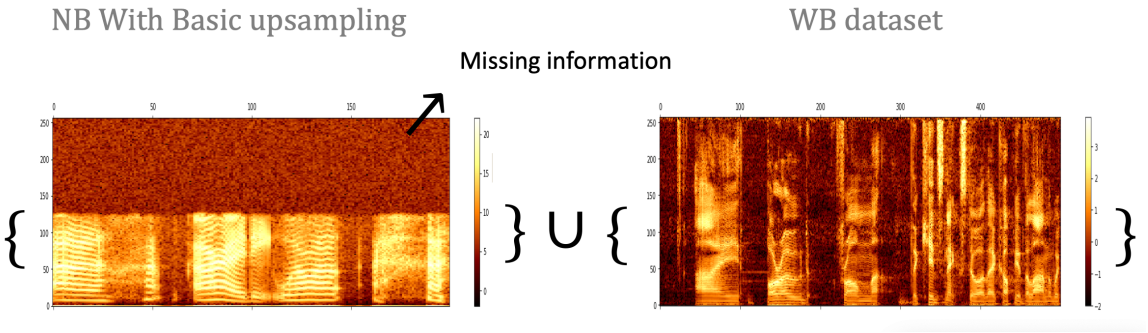


Figure 3.2: LPS feature of upsampled NB speech using basic upsampler (left) and original WB speech (right) used to train the mixed BW system. U indicates the combination of datasets.

is assumed to be null. The LPS features of the original WB speech and sox upsampled NB speech used in training the mixed BW system are shown in Figure 3.2. The lack of information in the upper half of the upsampled NB speech is clearly seen from the figure.

In this section, we experimented with neural BWE systems to upsample the NB speech. Instead of the linear upsampler used in mixed BW systems, we use a neural network to upsample the NB speech. The motivation behind using neural network is to predict information in the upper half (UB) of NB speech – hence, the name bandwidth extension (BWE). Instead of predicting just the UB from the NB and appending it to the NB features, we train the neural network to predict the entire spectrum of WB as suggested in [22]. The pipeline for training speaker recognition systems with neural BWE extension is as follows. We first train a DNN (architecture and training details in Sections 3.4.2 and 3.4.3 respectively) that acts as an upsampler network in the LPS feature do-

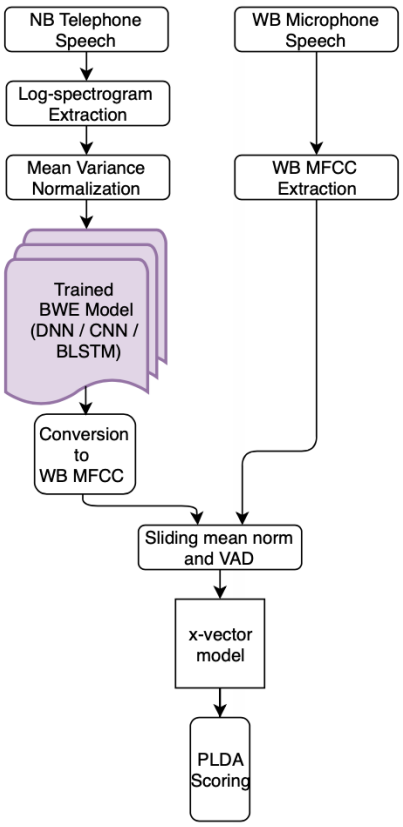


Figure 3.3: Training pipeline of x-vector network with neural BWE [34, 35]

main – network that maps NB LPS features to WB LPS domain. Once trained, the BWE network was used to upsample LPS features of telephone speech from training data. The upsampled WB LPS features are converted to MFCCs [22]. Finally, these MFCCs are combined with MFCCs extracted from the original WB data which forms the training data of the x-vector system. Pictorial representation of the training pipeline is outlined in Figure 3.3.

The system is similar to the mixed BW system but the upsampling process fills in the missing information in the UB of the upsampled telephone data, which is missing in case of linear upsampler. The experiments in this

section help us to investigate if filling in the missing information improves speaker recognition performance or not. Both the systems use the same training datasets and same MFCC feature configurations. Note that BWE networks were used only during the training (to upsample the NB training speech). Since the x-vector system was trained on WB speech, during evaluation the original WB evaluation corpus is used as it is.

We experimented with three different neural networks for BWE – (1) a fully connected neural network, termed as DNN-BWE, (2) a convolutional neural network [71], termed as CNN-BWE network and (3) a bi-directional long short term memory network, termed as BLSTM-BWE. We present the architectures of the BWE networks in Section 3.4.2. The training details of the networks are presented in Section 3.4.3 followed by results in Section 3.4.4.

3.4.1 BWE Training Overview

All BWE networks were trained on paired NB-WB data (for definition of paired data, refer to Section 2.4.3.1). Paired data was obtained from the WB training data via simulation – WB speech was downsampled to 8 kHz. The downsampled NB speech and the original WB speech formed the paired training data for all the BWE networks in this work. The LPS features extracted from the former were used as input to the network and the features of the later as the target output. The networks were trained to minimize MSE objective

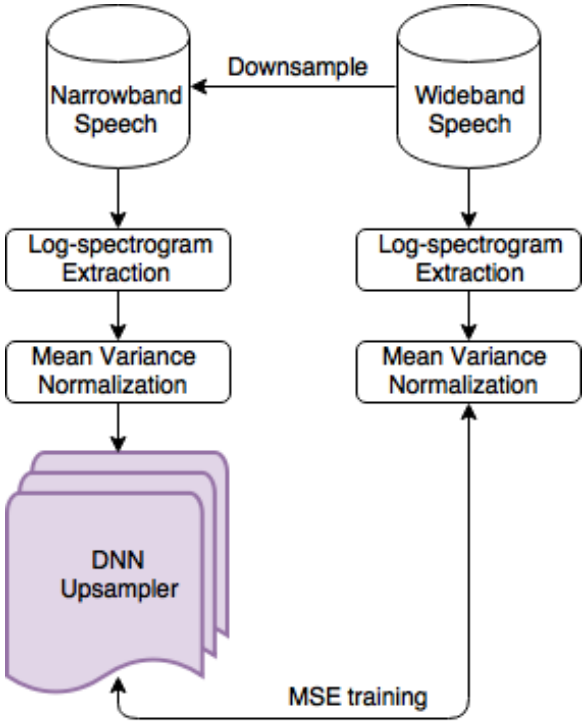


Figure 3.4: Neural BWE training pipeline

between predicted and original WB features. Figure 3.4 shows the pipeline to train the BWE networks.

3.4.2 BWE Networks

3.4.2.1 DNN-BWE Network

We experimented with a feed-forward fully connected DNN for BWE, similar to the network in [22]. The network had 3 hidden layers with 2048 neurons per layer each followed by ReLU nonlinearity. The last hidden layer is followed by an affine layer which projects the output of last hidden layer to WB LPS

features. A context of 5 past and 5 future frames was used at the input along with the current frame to predict the WB LPS of the current frame. Hence the input to the network is 1419 (11*129) dimensional and the output was of 257 dimensional.

3.4.2.2 CNN-BWE Network

The architecture of CNN-BWE network was similar to the full-convolutional deep residual network used by [72] for image style transfer and single image super resolution. The network consisted of two major building blocks: a downsampler (encoder) network and an upsampler (decoder) network, which makes it an encoder-decoder network[3]. The encoder network consisted of three 2D convolutional layers followed by several residual blocks [73]. The encoder maps the original features to a low-dimensional manifold while preserving the properties of speech. To do it, the network reduces the feature map dimensions by applying convolutions with stride = 2. For the details of kernel sizes, strides, number of filters and padding dimension in each layer of the encoder network and the residual network refer to Figures 3.5(a) and (b) respectively.

The decoder network consisted of two deconvolutional layers [74] followed by a final convolution layer. The network decodes the low-dimensional representation obtained from the encoder into the LPS feature of WB speech. The

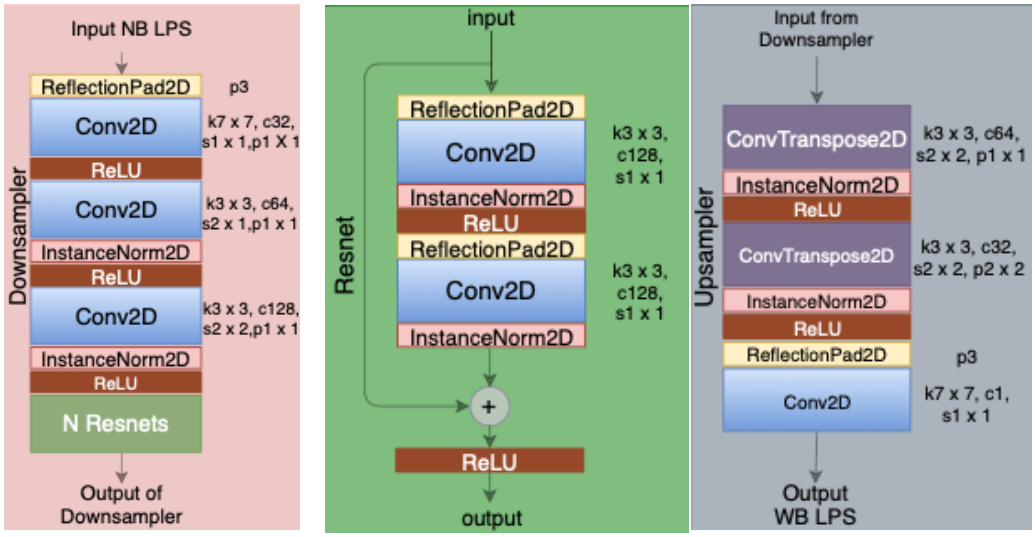
[3]The terms encoder-downsampler and decoder-upsampler are used interchangeably in CNN-BWE network description.

deconvolutional layers increase the feature map dimension by a factor of 2 by applying strides of $1/2$ while decreasing the number of filters by a factor of 2. For the details of kernel sizes, strides, number of filters and padding dimension in each layer of the upsampler refer to Figure 3.5(c).

The entire network consists of only convolutional and deconvolutional layers. No pooling or fully connected layers were included. Hence the network is termed as "deep residual fully-convolutional neural network". The fact that it is full convolutional allows the network to process variable length sequences without increasing the number of parameters. The exact number of convolutional layers (depth of the network) depends on the number of residual blocks used in the network. We experimented with number of residual blocks varying from 6 to 24 in this work (details in Section 3.4.3).

3.4.2.3 BLSTM-BWE Network

To compare both the feed forward architectures we described above, we also experimented with a recurrent architecture – a bidirectional long short term memory (BLSTM) network. The network has two 512 dimension hidden layers. The non linear activation used was ReLU. The output of BLSTM was fed to an affine layer, which made the network output dimension to match the required WB LPS dimension (257).



(a) Downsampler Network (b) Residual Network (c) Upsampler Network

Figure 3.5: CNN-BWE architecture details. (a) Downsampler (encoder) network architecture, (b) architecture of residual network used in downsampler, and (c) architecture of upsampler (decoder) network. Notation: k-kernel, s-strides, c-filters, p-padding, N-number of residual networks

3.4.3 BWE Networks Experimental Details

3.4.3.1 Training Data

All the three BWE extension networks were trained using the same paired NB-WB datasets. The WB training data (sampled from 1871 speakers) was downsampled to obtain paired training data. Training and cross validation utterances were selected randomly based on the number of speakers present in the WB dataset (90%-10% split between the speakers with no-overlap between the speakers). Once trained, the network was used to upsample original NB speech corpus (sampled from 7001 speakers) used to train the x-vector system. More detailed descriptions of the NB and WB training datasets can be found

in Section 3.2.1.

3.4.3.2 Feature configurations

As mentioned earlier, the BWE networks were trained to predict the entire WB. The input was 129 dimension NB LPS while the output was 257 dimension WB LPS, similar to [22]. NB and WB LPS were obtained with a window size of 25ms with an overlap of 15ms. *Povey* window in Kaldi Speech Recognition toolkit [37] was used. Energy based VAD was applied on the training data. Utterance level mean variance normalization was applied on each utterance. WB LPS are then converted into 23 dimensional MFCCs. Finally, they were combined with MFCCs of the original microphone speech – which formed the training data of x-vector network.

3.4.3.3 Training procedure

BWE systems were implemented using PyTorch [75]. MSE objective was minimized. All the networks were trained for a maximum of 50 epochs, where an epoch is completed when we have observed a random sample from each utterance in the training set. Early stopping was enforced when validation error does not decrease for 5 successive epochs. Learning rate was decreased by a factor of 2 when the validation error does not decrease for two successive epochs. The networks were initialized with values drawn from normal distri-

bution with mean 0 and standard deviation 0.02. Optimizer used for training depends on individual systems (details below). Dropout [76] was used as regularizer with 0.4 drop probability (except for CNN-BWE).

Input to the DNN-BWE network was arranged as three dimensional tensors of size $n \times F \times D$, where n, F and D stands for mini batch size, sequence length (number of frames) per mini-batch and feature dimension respectively. An utterance was selected at random for each mini batch, and 64 consecutive samples were drawn from that utterance. We add a context of 5 past and 5 future frames to each frame which makes input dimension $D = 11 \times 129 = 1419$. Mini batch size was set to 1. Given a mini batch of input NB features of size 1 x 64 x 1419 dimensions, the network was trained to predict the output WB features of size 1 x 64 x 257. Stochastic Gradient Descent (SGD) optimizer was used to train the network with an initial learning rate of 0.01 and a momentum of 0.9. Figure 3.6 shows the LPS output of the DNN-BWE network along with the ground truth for a randomly picked utterance from the development set (the network was not trained on this utterance). The figure demonstrates the ability of the DNN-BWE to fill missing information in the top half of the spectrum.

The training procedure of CNN-BWE network was as follows: since we used 2D convolutions in CNN-BWE network, input to the system was arranged as four dimensional tensors of size $n \times C \times F \times D$, where n, C, F, D stands for mini

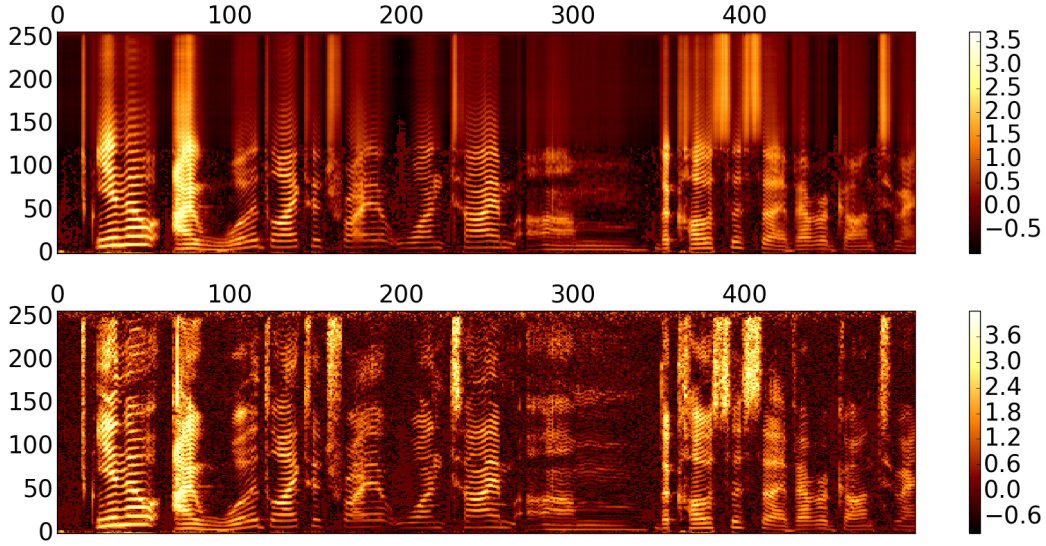


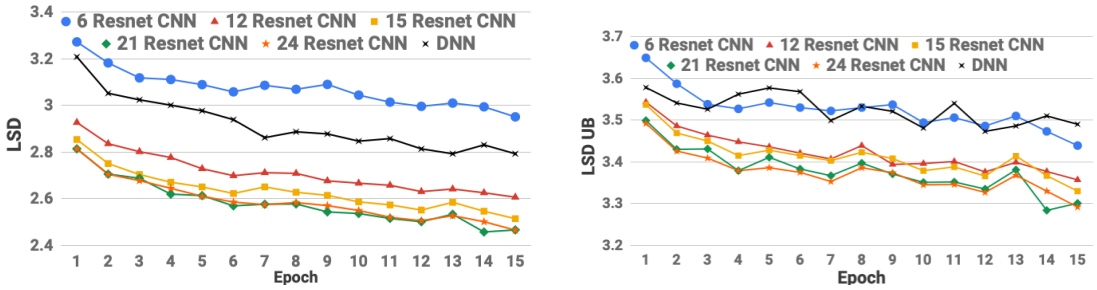
Figure 3.6: Log power spectrogram (LPS) feature comparison of estimated WB from DNN-BWE (upper) and original WB (lower)

batch size, channels size, sequence length and feature dimension respectively. Mini batch dimension n and number of input channels C were set to 1.

We experimented with the number of residual blocks in the encoder network. We observed that the number of residual blocks in the downsampler of CNN-BWE system plays an important role in improving the prediction quality of the network. The prediction quality was measured using the metric log-spectral distortion (LSD) given in Equation 3.1 below.

$$\mathrm{LSD}(X, \hat{X}) = \frac{1}{L}\sum_{l=1}^{L}\sqrt{\frac{1}{K}\sum_{k=1}^{K}(X(l,k) - \hat{X}(l,k))^2} \tag{3.1}$$

X and $\hat{X}$ in the equation stands for the reference and predicted LPS features.



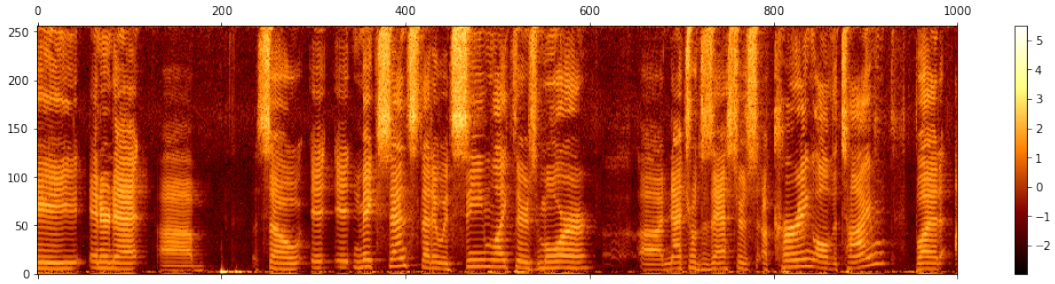
(a) LSD (b) LSD-UB

Figure 3.7: LSD and LSD-UB comparison of CNN-BWE and DNN-BWE Systems. We experimented with different number of residual blocks for CNN-BWE.

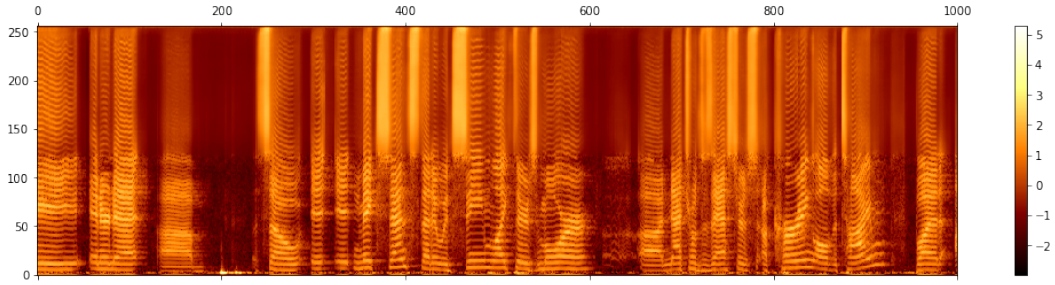
LSD measures the reconstruction quality of individual frequencies in LPS domain.

Figure 3.7(a) compares LSDs of several CNN-BWE systems – each differ in the number of residual blocks used in the encoder network. The figure also includes a comparison of CNN-BWE with DNN-BWE. Since, our main goal is to predict the missing information in the UB of the LPS, we also compute the LSD for the top half of the predicted spectrum, denoted as LSD-UB. Figure 3.7(b) presents comparison of LSD-UB. Both LSD and LSD-UB are measured on the development set.

As observed from the figure, for six residual blocks, the performance of DNN-BWE was better than the CNN-BWE. Increasing the number of residual blocks improved CNN-BWE. However, going beyond 21 blocks did not significantly improve the CNN-BWE system. Increasing the number of residual blocks increased the depth of the network which increased the receptive field



(a) Reference WB spectrogram

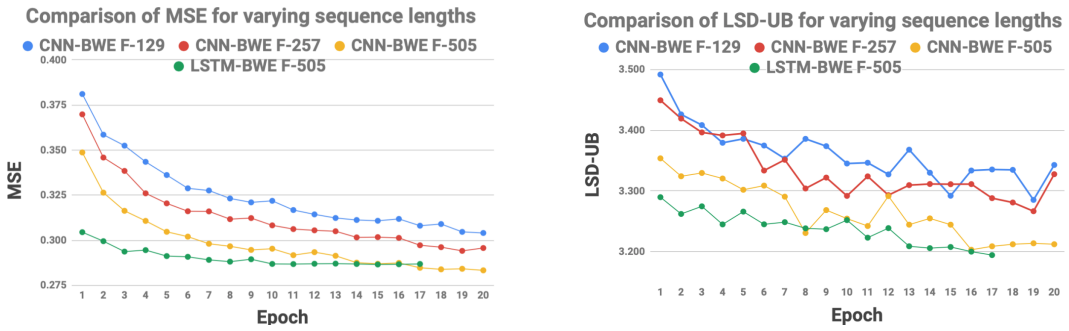


(b) Predicted WB spectrogram by CNN-BWE

Figure 3.8: Comparison of reference and predicted LPS by CNN-BWE

of the network, which explains the improvements. For the rest of the experiments, we used CNN-BWE systems with 21 residual blocks. In total, the network had 46 convolutional layers and two deconvolutional layers. In spite of the network depth, the number of parameters of CNN-BWE ($\sim 6M$ for network with 21 residual networks) are less than that of the DNN-BWE ($\sim 16M$). Figure 3.8 shows comparison of spectrograms of original WB utterance from validation set and predicted WB utterances from a CNN-BWE system.

The training procedure of BLSTM-BWE was similar to the DNN-BWE except that the sequence length F was set to 505. The motivation for experimenting with BLSTM was as follows: the limitation of DNN-BWE was that it was



(a) MSE (b) LSD-UB

Figure 3.9: Comparison of MSE and LSD-UB between CNN-BWE and BLSTM-BWE systems when trained on different sequence lengths.

trained on limited amount of context (5 past and 5 future frames). Any increase in the context would increase the number of parameters in the input layer. CNN-BWE overcame this disadvantage. Since, we used a full-convolutional network, we were able to train the network on longer sequences (129, 257 and 505), without increasing the number of parameters of the network. BLSTM networks are known for training on longer sequences. Hence, we trained a BLSTM network with sequence length $F = 505$, to compare the performance of CNN with a network trained on longer sequence. Figure 3.9 shows that MSE and LSD-UB decrease as we increase the sequence lengths. The BLSTM-BWE network converged faster (only trained for 17 epochs) compared to CNN-BWE in terms of number of epochs.

3.4.4 BWE Speaker recognition systems

The output spectrograms and LSD metric observed in previous section demonstrate the ability of all three networks in extending the bandwidth of NB speech. In this section, we assess the goodness of these networks in improving the performance of ASV systems. Performance was measured on the SITW evaluation set.

3.4.4.1 Baseline Systems

We experimented with three baseline x-vector systems: a NB baseline system, a WB baseline system and a mixed BW baseline system. The baseline systems are described in previous Section 3.1 and Section 3.3. No data augmentation was used in this experiments.

3.4.4.2 BWE Speaker recognition Systems

The training pipeline of x-vector network with BWE features is shown in Figure 3.3. The training procedure was similar to the mixed BW system except that the NB speech was upsampled using BWE. The x-vector network was trained on upsampled speech and original WB speech.

Table 3.3 summarizes the results of the baseline and BWE-speaker recognition systems. The mixed BW baseline system performed the best among the baselines. All the BWE systems improved over the baseline systems in terms

	SITW Core			SITW Assist-Multi		
	EER	**DCF(1E-2)**	**DCF(1E-3)**	**EER**	**DCF(1E-2)**	**DCF(1E-3)**
Baseline						
NB	6.01	0.5111	0.7105	8.88	0.5569	0.7351
WB	8.02	0.5538	0.7505	8.54	0.5049	0.6880
mixed BW	5.93	0.4711	0.6713	7.62	0.4890	0.6667
BWE-speaker ID results						
DNN-BWE	**5.55**	0.4705	0.6516	7.10	0.4812	0.6481
CNN-BWE (F-129)	5.74	0.4737	0.6630	**7.08**	0.4737	0.6671
CNN-BWE (F-257)	5.71	**0.4560**	**0.6480**	7.29	**0.4680**	**0.6475**
BLSTM-BWE (F-505)	5.74	0.4621	0.6523	7.35	0.4717	0.6510

Table 3.3: Results of speaker verification on SITW when trained on bandwidth extended features (F - sequence length)

of both EER and DCF. CNN-BWE system trained on sequence of length 257 performed the best in terms of DCF. However, the relative difference between the CNN-BWE and BLSTM-BWE system was not very significant. In terms of relative improvement of DCF at prior 0.01, the CNN-BWE improved by 10.78% and 15.96% for SITW *Core* and *Assist-multi* speaker conditions respectively over the NB baseline. Compared to the mixed BW system, the relative improvements were 3.21% and 4.13% for Core and Assist-multi-speaker conditions, which justifies our attempt to predict information in the UB for NB speech and use it for speaker recognition.

3.5 Summary

The experimental setting in this chapter is as follows: we considered two types of datasets for training – telephone speech sampled at narrowband (NB)

8 kHz frequency, and microphone speech sampled at wideband (WB) 16 kHz frequency. We assumed the evaluation dataset was WB. We further assumed that WB speech available for training was limited in number of speakers (and utterances) compared to NB speech. The conventional way of training systems under this scenario is to downsample WB speech to match the frequency of NB speech during training and evaluation. This procedure throws away information in the upperband (UB) of microphone speech. In this chapter, we investigated several procedures to combine both the datasets during training without downsampling the WB speech.

We first experimented with a basic upsampler that upsamples telephone speech to match the sampling frequency of microphone speech without predicting any information in the UB. The upsampled telephone speech was used to train the x-vector network. We termed this ASV system as mixed BW sytem [34]. This upsampling technique, though simple and computationally inexpensive, proved very effective, yielding impressive results (details in Section 3.3). The main advantage of mixed BW system was that, we were able to use original microphone speech during training and evaluation without any downsampling as done in conventional way of training. We, thus, were able to retain information in the UB of WB speech which helped improve the verification performance.

We then continued our investigation by experimenting with several neu-

ral network architectures to upsample NB speech before training x-vector networks. The motivation was to overcome the limitation of linear upsampler in predicting the information present in the upper band of telephone speech. We experimented with a feed forward fully connected DNN, a deep residual full-convolutional network (CNN) and a BLSTM network for BWE [35]. Individual x-vector based speaker recognition systems were trained on bandwidth extended features obtained from each of these three systems. All the BWE speaker recognition systems improved in performance compared to conventional training and the mixed BW system. The best performer in terms of DCF was the CNN-BWE system trained on sequence lengths of 257. In terms of DCF, the CNN-BWE system showed relative improvement of 10.78% and 15.96% in the SITW eval *Core* and *Assist-Multi* condition respectively w.r.t. the conventionally trained NB baseline; and improved by 3.21% and 4.13% w.r.t. to the mixed BW baseline. In terms of number of parameters, the CNN-BWE model with 21 resnets is the most light weight with $\sim 6M$ parameters compared to DNN-BWE with $\sim 16M$ parameters and BLSTM-BWE with $\sim 18M$ parameters (details in Section 3.4).

Chapter 4

Unsupervised Domain Adaptation using CycleGAN for Channel Mismatch

4.1 Introduction

In this chapter, we address a channel mismatch scenario where the training (*source* domain) and evaluation (*target* domain) data of the automatic speaker verification (ASV) system are sampled from telephone and microphone corpora respectively. In the previous chapter, we addressed the same scenario as a mismatch in sampling frequencies of speech from both the domains. Telephone speech and microphone speech were termed as narrowband (NB) and wide-

band (WB) speech with sampling frequencies 8 kHz and 16 kHz respectively. The LPS features of NB speech were upsampled to match the sampling frequency of WB data using a bandwidth extension (BWE) network. BWE network can be considered as a feature mapping function that maps NB features to WB domain. The network was trained using paired NB-WB data (procedure detailed in Section 3.4). The NB input data used in training was obtained by downsampling the WB speech. The network, thus trained using paired NB-WB data obtained from microphone speech, was used to upsample the features of telephone data.

The above approach gave encouraging results. However, it limited us from using real data from both domains to train the feature mapping function – e.g. to upsample the telephone speech features using BWE network, we trained the network on paired data extracted from microphone speech using simulation. To address this limitation, in this chapter, we experimented with an unsupervised domain adaptation (UDA) approach where a feature mapping function was trained on unpaired and unlabelled data from both the domains (for descriptions of unpaired data and UDA, refer to Section 2.4.3.2 and Section 2.4.1 respectively). Once trained, the network was used to map features from *target* to *source* domain during evaluation. Hence, this procedure would fall under 'adaptation during testing' scheme (discussed in Section 2.4.2.1). Instead of approaching the problem as a mismatch in sampling frequencies, we approach

the problem as a mismatch in channels used to acquire speech (details in Section 2.5.2).

The nature of training data dictated the loss functions we used to train feature mapping networks in this work. In previous chapter, since, BWE network was trained on paired data, we used MSE objective between predicted and ground truth features. Since, the training data is unpaired in the current setting, MSE objective cannot be used. To achieve the required mapping from *source* to *target* domain, we train the network using adversarial loss [28, 54] – the network is trained such that its output fools a binary classifier, which was trained to distinguish between original and mapped *target* domain features. In addition to adversarial loss, we also use cycle-consistency loss that reconstructs the original *source* domain features from the mapped features in *target* domain with the help of a second network. This second constraint ensures no useful information is lost while transferring the features from *source* to *target* domain. This framework is termed as CycleGAN in the literature [16]. We demonstrate the effectiveness of this approach for channel adaptation task [30, 31].

We also experimented with training the feature mapping function in a low-resource scenario – when limited amount of data is available from *target* domain. This setting is of interest to us because this opens up the possibility of maximally taking advantage of small development sets found in real data, and not use simulated sets (as done commonly in practice). We observed that

CycleGAN framework tends to overfit when trained with limited amount of data. We present a simple, but effective, regularization technique to overcome this disadvantage [31].

The outline of this chapter is as follows: we first describe the unadapted baseline system in Section 4.2. We describe the CycleGAN based adaptation framework in Section 4.3 and present results of our adaptation system. In Section 4.4, we present our low-resource experiments. We summarize our observations in Section 4.5.

4.2 Baseline System

The ASV system trained on *source* domain features and tested on original *target* domain features is referred as the baseline system. Speaker embeddings were extracted using an x-vector network with Extended TDNN (ETDNN) architecture [15]. The x-vector system was trained for 3 epochs using Kaldi [37]. We used the same setup as in SRE16 Kaldi recipe[1] but without any data augmentation. The x-vector network was trained on telephone corpus and tested on evaluation corpus from SITW, referred as SITW *eval*. Telephone data used for training consisted of recordings from datasets SRE 04-10, Mixer6 and Switchboard 1-Phase 1, 2, and 3. This gave us 90946 utterances from 6986 speakers. The system used 40-dimensional log Mel filter-bank (log mel-FB)

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2

with short-time centering (300 frames). Energy-based VAD was applied to remove the non-speech frames.

Once trained, x-vector network was used to extract speaker embeddings for the training and evaluation corpora. The x-vectors were centered, projected to 150 dimensions using linear discriminant analysis (LDA) and length normalized. Full-rank PLDA [69] was used to get the scores. Finally, scores were normalized using adaptive symmetric norm (S-Norm) [70]. In the baseline system, both the x-vector network and PLDA backend were trained on telephone speech and tested on microphone speech. Similar to Chapter 3, we evaluated our baseline system on SITW. During evaluation, the microphone speech was down-sampled to 8 kHz to match the sampling frequency of telephone speech. We present the results for baseline system on SITW in next section (in Table 4.4) after discussing the adaptation system.

4.3 Channel Adaptation Using CycleGAN

Our adaptation approach is as follows. We first trained an x-vector network [14] on training data sampled from telephone domain. The training procedure was very similar to the baseline system described in Section 4.2. During evaluation, we mapped the features of evaluation data sampled from microphone domain to telephone domain. We used the mapped features to

evaluate the ASV system trained on telephone domain. This was first accomplished by using the mapped features to extract speaker embeddings from the x-vector network, and use those embeddings for PLDA scoring. Hence, the overall speaker verification system can be considered as trained and evaluated on *source* domain. The feature mapping of evaluation data to *source* domain was accomplished by using the CycleGAN framework. Below we describe CycleGAN and its training procedure along with its objectives.

4.3.1 CycleGAN Description

As explained in Section 4.1, our adaptation procedure involved training two feature mapping functions using unpaired unlabelled data sampled from *source* and *target* domains. One mapping function maps features from *target* to *source* domain, whereas the other mapping function maps features from *source* to *target* domain. Both mapping functions are trained jointly. The training procedure was inspired from the *unpaired image-to-image translation* work done in [16], popularly known as cycle-consistent generative adversarial networks (CycleGAN). The mapping functions in CycleGAN are known as *generators*. The generators, represented as $G_{\mathrm{S}\to\mathrm{T}}$ and $G_{\mathrm{T}\to\mathrm{S}}$, map features from *source-to-target* and *target-to-source* respectively. Once the generators are trained, we use the generator $G_{\mathrm{T}\to\mathrm{S}}$ as the feature mapping function that maps evaluation features to source domain. Below we explain the training procedure of

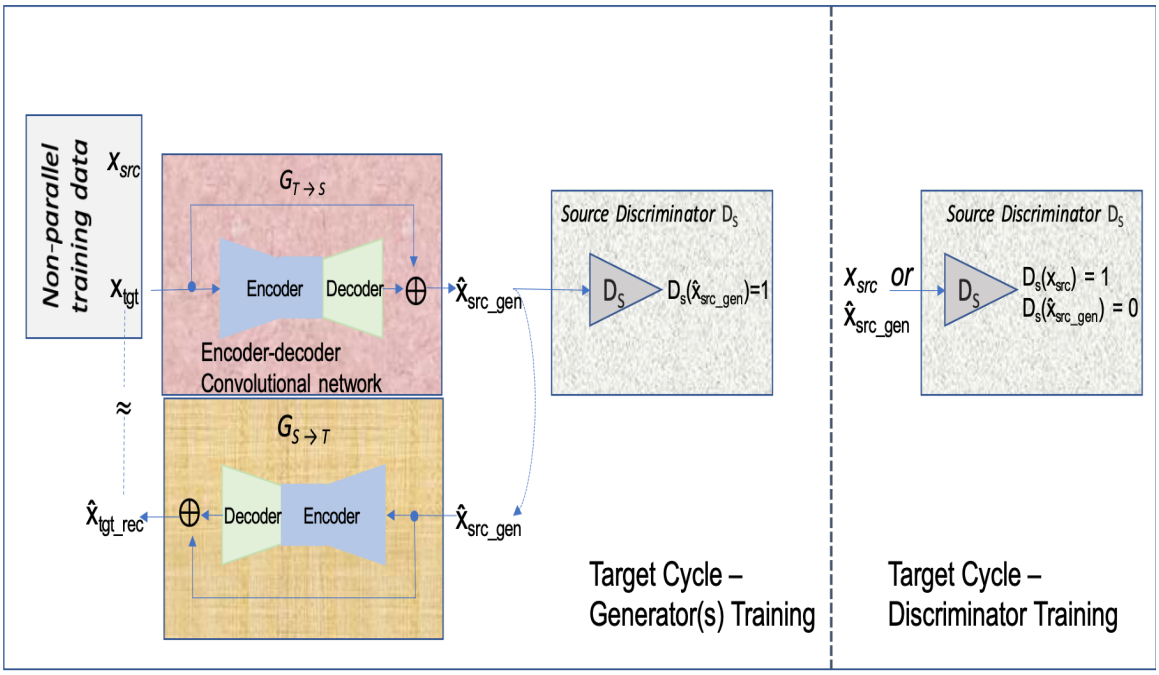
CycleGAN used in this work.

4.3.1.1 CycleGAN Training Procedure and Objectives

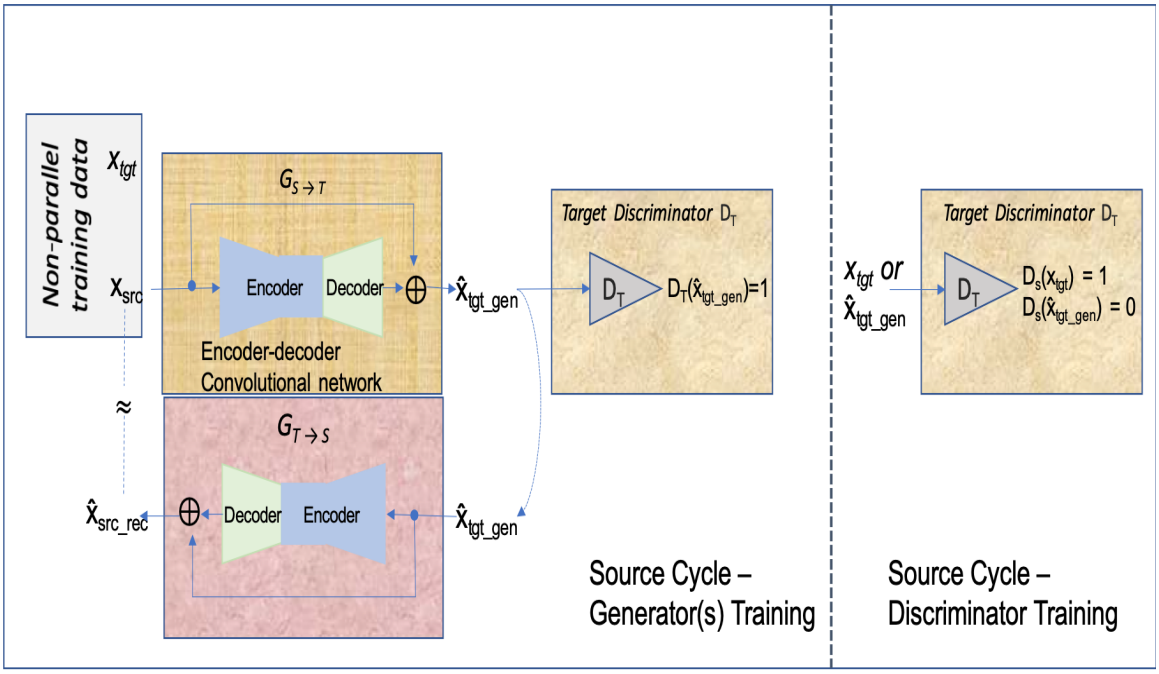
The unpaired corpora to train CycleGAN consisted of audio samples $\mathbf{A}_{\mathrm{S}} = \{\mathbf{a}_{\mathrm{S},i}\}_{i=1}^{N}$ and $\mathbf{A}_{\mathrm{T}} = \{\mathbf{a}_{\mathrm{T},i}\}_{i=1}^{M}$ drawn from two different domains: *source* S and *target* T with distributions $\mathbf{a}_{\mathrm{S},i} \sim q_{\mathrm{S}}(\mathbf{a})$ and $\mathbf{a}_{\mathrm{T},i} \sim q_{\mathrm{T}}(\mathbf{a})$ respectively. log mel-FB features were extracted from audio samples of *source* and *target* domain distributions denoted as $\mathbf{X}_{\mathrm{S}} = \{\mathbf{x}_{\mathrm{S},i}\}_{i=1}^{N}$ and $\mathbf{X}_{\mathrm{T}} = \{\mathbf{x}_{\mathrm{T},i}\}_{i=1}^{M}$. The *source* and *target* domain distributions in feature space are represented as $\mathbf{x}_{\mathrm{S},i} \sim p_{\mathrm{S}}(\mathbf{x})$ and $\mathbf{x}_{\mathrm{T},i} \sim p_{\mathrm{T}}(\mathbf{x})$ respectively. $\mathbf{X}_{\mathrm{S}}$ and $\mathbf{X}_{\mathrm{T}}$ are used as features from two different distributions to train the feature mapping functions[2].

The training procedure is as follows: two mini batches of features $\mathbf{x}_{\mathrm{S}}$ and $\mathbf{x}_{\mathrm{T}}$ are sampled from the distributions $\mathbf{X}_{\mathrm{S}}$ and $\mathbf{X}_{\mathrm{T}}$ corresponding to the *source* and *target* domain features respectively. The generator $G_{\mathrm{T}\to\mathrm{S}}$ maps $\mathbf{x}_{\mathrm{T}}$ to *source* domain, producing features $\hat{\mathbf{x}}_{\mathrm{S\_gen}}$. A discriminator D_{S} is trained to discriminate between original ($\mathbf{x}_{\mathrm{S}}$) and generated *source* domain features ($\hat{\mathbf{x}}_{\mathrm{S\_gen}}$), thus serving the task of a binary classifier. D_{S} outputs 1 for the original (*real*) source domain features and 0 for generated (*fake*) features as shown in Figure 4.1 and represented in Equation 4.1.

[2]Since, CycleGAN is also used in far-field adaptation work (discussed in Chapter 5), we explain the training procedure using generic terms *source* and *target*, instead of *microphone* and *telephone*



(a) Target cycle (*target - source - target*)



(b) Source cycle (*source - target - source*)

Figure 4.1: Overview of CycleGAN framework

$$D_{\mathrm{S}}(\mathbf{x}_{\mathrm{S}}) = 1$$

$$D_{\mathrm{S}}(\hat{\mathbf{x}}_{\mathrm{S\_gen}}) = D_{\mathrm{S}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x}_{\mathrm{T}})) = 0 \tag{4.1}$$

To achieve this, the discriminator D_{S} is trained to minimize a *least square loss* [77] as given in Equation 4.2.

$$L_{\mathrm{disc}}(G_{\mathrm{T}\to\mathrm{S}}, D_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}}, \mathbf{X}_{\mathrm{S}}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{T}}}[D_{\mathrm{S}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x}))^2] + \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{S}}}[(D_{\mathrm{S}}(\mathbf{x}) - 1)^2] \tag{4.2}$$

The generator $G_{\mathrm{T}\to\mathrm{S}}$ is then trained to output features $\hat{\mathbf{x}}_{\mathrm{S\_gen}}$ that appear to be drawn from the *source* domain distribution $\mathbf{X}_{\mathrm{S}}$. This is accomplished by flipping the label of the *fake* features given to the discriminator, *i.e.* assigning a label 1 to the *fake* features instead of 0. The procedure is detailed in [28, 54]. The adversarial objective of the generator is given in Equation 4.3.

$$L_{\mathrm{adv}}(G_{\mathrm{T}\to\mathrm{S}}, D_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{T}}}[(D_{\mathrm{S}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x})) - 1)^2] \tag{4.3}$$

Equivalently, the other generator-discriminator ($G_{\mathrm{S}\to\mathrm{T}}$, D_{T}) pair is trained

in a similar fashion to transfer features from *source* domain to *target* domain.

A single generator-discriminator pair trained with combination of *adversarial* and *least square* losses would suffice, in theory, to transfer features from one domain to the opposite domain. However, this leads to an ill-posed problem with *adversarial* loss putting a weak constraint on the generators. Thus, the generator could create many possible features which appear to be drawn from the true distribution but may fail to preserve the information present in the signal like the linguistic information, speaker and gender information. To restrict the space of possible mappings from the generator, CycleGAN enforce *cycle-consistency* constraint on the generators – reconstructing the original features, e.g. $\mathrm{x_T}$, from the generated features in the opposite domain, e.g., $\hat{\mathrm{x}}_{\mathrm{S\_gen}}$ by passing them through the second generator $G_{\mathrm{S}\to\mathrm{T}}$. The reconstructed features, represented as $\hat{\mathrm{x}}_{\mathrm{T\_rec}}$, should be identical $\mathrm{x_T}$. Mathematically, *cycle-consistency* on the *target* and *source* domain features are explained in Equation 4.4.

$$
\begin{aligned}
\hat{\mathbf{x}}_{\mathrm{T\_rec}} &= G_{\mathrm{S}\to\mathrm{T}}(\hat{\mathbf{x}}_{\mathrm{S\_gen}}) \\
&= G_{\mathrm{S}\to\mathrm{T}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x}_{\mathrm{T}})) \\
&\approx \mathbf{x}_{\mathrm{T}} \\
\text{Similarly} \quad \hat{\mathbf{x}}_{\mathrm{S\_rec}} &= G_{\mathrm{T}\to\mathrm{S}}(\hat{\mathbf{x}}_{\mathrm{T\_gen}}) \\
&= G_{\mathrm{T}\to\mathrm{S}}(G_{\mathrm{S}\to\mathrm{T}}(\mathbf{x}_{\mathrm{S}})) \\
&\approx \mathbf{x}_{\mathrm{S}}
\end{aligned} \tag{4.4}
$$

Cycle-consistency of *source* features is achieved by minimizing the objective in Equation 4.5 between $\mathbf{x}_{\mathrm{S}}$ and $\hat{\mathbf{x}}_{\mathrm{S\_rec}} = G_{\mathrm{T}\to\mathrm{S}}(\hat{\mathbf{x}}_{\mathrm{T\_gen}})$ where we used L_1 distance as the metric. We refer to this loss as *forward cycle-consistency* loss. Similarly, the loss computed between $\mathbf{x}_{\mathrm{T}}$ and $\hat{\mathbf{x}}_{\mathrm{T\_rec}}$ is referred to as *backward cycle-consistency* loss. The final *cycle-consistency* loss is the combination of both these objectives and is given in Equation 4.6.

$$
L_1(G_{\mathrm{S}\to\mathrm{T}}, G_{\mathrm{T}\to\mathrm{S}}, \mathbf{X}_{\mathrm{S}}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{S}}} ||G_{\mathrm{T}\to\mathrm{S}}(G_{\mathrm{S}\to\mathrm{T}}(\mathbf{x})) - \mathbf{x}||_1 \tag{4.5}
$$

$$L_{\text{cyc}}(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}, \mathbf{X}_{\text{S}}, \mathbf{X}_{\text{T}'}) = L_1(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}, \mathbf{X}_{\text{S}}) + L_1(G_{\text{T}\to\text{S}}, G_{\text{S}\to\text{T}}, \mathbf{X}_{\text{T}}) \tag{4.6}$$

Finally, both the generators of CycleGAN are trained using multi-task objective: by minimizing both the *adversarial* and *cycle-consistency* objectives as shown in Equation 4.7. λ_{cyc} and λ_{adv} in Equation 4.7 denote the weights assigned to *cycle-consistency* loss and *adversarial* loss respectively.

The training objectives of both the *source* and *target* discriminators are given in Equations 4.8 and 4.9 respectively.

Objective of generators

$$\begin{aligned} & L(G_{\text{T}\to\text{S}}, G_{\text{S}\to\text{T}}, D_{\text{T}}, D_{\text{S}}, \mathbf{X}_{\text{S}}, \mathbf{X}_{\text{T}}) \\ & = \lambda_{adv} L_{\text{adv}}(G_{\text{T}\to\text{S}}, D_{\text{S}}, \mathbf{X}_{\text{T}}) \\ & + \lambda_{\text{adv}} L_{\text{adv}}(G_{\text{S}\to\text{T}}, D_{\text{T}}, \mathbf{X}_{\text{S}}) \\ & + \lambda_{\text{cyc}} L_{\text{cyc}}(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}, \mathbf{X}_{\text{S}}, \mathbf{X}_{\text{T}}) \end{aligned} \tag{4.7}$$

Objective of *source* discriminator D_{S}

$$\begin{aligned}\theta^{\mathrm{D_S}*} &= \underset{\theta^{\mathrm{D_S}}}{\operatorname{argmin}} L_{\mathrm{GAN}}(G_{\mathrm{T}\to\mathrm{S}}, D_{\mathrm{S}}, \mathbf{X}_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}}) \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{S}}}[(D_{\mathrm{S}}(\mathbf{x})-1)^2] + \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{T}}}[D_{\mathrm{S}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x}))^2] \end{aligned} \tag{4.8}$$

Objective of *target* discriminator D_{T}

$$\begin{aligned}\theta^{\mathrm{D_T}*} &= \underset{\theta^{\mathrm{D_T}}}{\operatorname{argmin}} L_{\mathrm{GAN}}(G_{\mathrm{S}\to\mathrm{T}}, D_{\mathrm{T}}, \mathbf{X}_{\mathrm{T}}, \mathbf{X}_{\mathrm{S}}) \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{T}}}[(D_{\mathrm{T}}(\mathbf{x})-1)^2] + \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{S}}}[D_{\mathrm{T}}(G_{\mathrm{S}\to\mathrm{T}}(\mathbf{x}))^2] \end{aligned} \tag{4.9}$$

4.3.1.2 Identity Loss

As explained in Equation 4.7, the generators are trained using a combination of *adversarial* and *cycle-consistency* losses. Along with this two constraints, we place an additional constraint on the generator – to make an identity mapping when features from the opposite domain are provided as input to the generator. The identity mappings for both the generators are achieved by training them with *identity* loss [16, 78] given in Equation 4.10.

Identity loss

$$L_{\text{idt}}(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}) = \tag{4.10}$$
$$\mathbb{E}_{\mathbf{x}\sim p_{\text{T}}}[||G_{\text{S}\to\text{T}}(\mathbf{x}) - \mathbf{x}||_1]$$
$$+ \mathbb{E}_{\mathbf{x}\sim p_{\text{S}}}[||G_{\text{T}\to\text{S}}(\mathbf{x}) - \mathbf{x}||_1]$$

Identity loss was used in the original CycleGAN work as a regularizer during training. Besides acting as a regularizer, identity loss helps the generators learn to preserve linguistic information. Since, the objective is similar to that of an auto encoder – the generator is trained to reconstruct the input.

Combined objective for training generators with *identity*, *cycle-consistency* and *adversarial* losses is given in Equation 4.11 below. λ_{idt} in the equation stands for the weight assigned to identity loss.

Overall objective of generators trained with identity loss

$$L(G_{\text{T}\to\text{S}}, G_{\text{S}\to\text{T}}, D_{\text{T}}, D_{\text{S}}, \mathbf{X}_{\text{S}}, \mathbf{X}_{\text{T}'})$$
$$= \lambda_{adv} L_{\text{adv}}(G_{\text{T}\to\text{S}}, D_{\text{S}}, \mathbf{X}_{\text{T}'})$$
$$+ \lambda_{\text{adv}} L_{\text{adv}}(G_{\text{S}\to\text{T}}, D_{\text{T}}, \mathbf{X}_{\text{S}})$$
$$+ \lambda_{\text{cyc}} L_{\text{cyc}}(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}, \mathbf{X}_{\text{S}}, \mathbf{X}_{\text{T}'})$$
$$+ \lambda_{idt} L_{\text{idt}}(G_{\text{S}\to\text{T}}, G_{\text{T}\to\text{S}}) \tag{4.11}$$

4.3.1.3 Network Architectures used in CycleGAN

CycleGAN consists of two generators and two discriminators. In this section, we give details of generator and discriminator architectures. The architecture for generator[3] is given in Table 4.1. Similar to BWE network in previous chapter, the generator is a full-convolutional network [79] with an encoder-decoder architecture. The encoder has three convolutional layers [71, 80] followed by nine layers of residual network (ResNet) [73]. All the convolutional layers are followed by an *instance normalization* layer [81] and ReLU [82] activation except for the first convolutional layer which is followed by ReLU activation. Each ResNet is followed by ReLU activation.

The architecture of ResNet is given in Table 4.2. ResNet has two convolutional layers. The first convolutional layer in the ResNet is followed by an *instance normalization* layer and ReLU activation. The second layer is only followed by *instance normalization* layer, the output of which is added to the input of the ResNet *via* short-cut connection, which becomes the final output of the ResNet. The output of last ResNet, followed by the ReLU activation, becomes the input to the decoder network.

The decoder architecture has two transposed convolutional layers [74] each followed by an *instance normalization* layer and ReLU activation. The transposed convolutional layers are followed by a final convolutional layer. The gen-

[3]Both the generators have identical architectures. Hence, we explain the architecture of only one generator.

Layer	**Kernel size (*nf*, *k*, *s*)**	**Output**
Shortcut	-	h x w x 1
Encoder		
Convolutional, ReLU	[3,3,1]	h x w x 32
Convolutional, *instance normalization*, ReLU	[3,3,2]	h/2 x w/2 x 64
Convolutional, *instance normalization*, ReLU	[3,3,2]	h/4 x w/4 x 128
(ResNet, ReLU) x 9	-	h/4 x w/4 x 128
Decoder		
Deconvolutional, *instance normalization*, ReLU	[3,3,2]	h/2 x w/2 x 64
Deconvolutional, *instance normalization*, ReLU	[3,3,2]	h x w x 32
Convolutional	[3,3,1]	h x w x 1
Addition	-	h x w x 1

Table 4.1: Architecture of generators used in CycleGAN. *nf*, *k* and *s* represent the number of filters, kernel size and stride respectively. (h,w) represent the shape of the input to the generator. ResNet stands for residual network (details in Table 4.2)

Layer	**Kernel size (*nf*, *k*, *s*)**	**Output**
Shortcut	-	h/4 x w/4 x 128
Convolutional, *instance normalization*, ReLU	[3,3,1]	h/4 x w/4 x 128
Convolutional, *instance normalization*	[3,3,1]	h/4 x w/4 x 128
Addition	-	h/4 x w/4 x 128

Table 4.2: Architecture of residual network (ResNet) used in generators of CycleGAN. *nf*, *k* and *s* represent the number of filters, kernel size and stride respectively. (h,w) represent the shape of the input to the generator.

Layer	**Kernel size (*nf*, *k*, *s*)**	**Output**
Convolutional, LeReLU	[4,4,2]	h/2 x w/2 x 64
Convolutional, LeReLU	[4,4,2]	h/4 x w/4 x 128
Convolutional, LeReLU	[4,4,2]	h/8 x w/8 x 256
Convolutional, LeReLU	[4,4,1]	h/8 x w/8 x 512
Convolutional	[4,4,1]	h/8 x w/8 x 1

Table 4.3: Architecture of discriminators used in CycleGAN. *nf*, *k* and *s* represent the number of filters, kernel size and stride respectively. (h,w) represent the shape of the input to the generator.

erator has a short-cut connection – the output of the last convolutional layer of the decoder is added to the input to the generator (encoder) which becomes the final output of the generator. For details of number of filters *nf*, kernel size *k* and stride *s* used in each individual convolutional and transposed convolutional layers, refer to Table 4.1.

The architecture used for discriminator[4] is presented in Table 4.3. It has five convolutional layers, the first four of which are followed by leaky rectified linear unit (LeReLU) [83] non-linear activation. The slope of all LeReLU functions are set to 0.2. Since, we train the discriminator with a *least squares* loss there is no non linear activation after the last convolution layer in the discriminator. For details of number of filters *nf*, kernel size *k* and stride *s* used in each individual convolutional layer of the discriminator, refer to Table 4.3.

[4]Similar to the generators, both the discriminators have identical architectures. Hence, we explain the architecture of only one discrminator.

4.3.2 Adaptation System Training

In this section, we explain the training process of adaptation system. In Section 4.2, we discussed baseline system. It was trained on telephone domain data and tested on microphone domain data. For the adaptation system, the x-vector network and PLDA were same as in the baseline. Feature adaptation was done in the evaluation stage. First, the log mel-FB features of the evaluation corpus were mapped from microphone to telephone domain by forward passing through the $G_{\mathrm{T}\to\mathrm{S}}$ generator of CycleGAN network. Then, the mapped features were used to extract the x-vectors for the evaluation data and used for PLDA scoring. As explained above, CycleGAN is at the heart of adaptation system. We first explain the datasets used for training CycleGAN and then we give the training details.

4.3.2.1 Description of *Source* and *Target* Domain Datasets

Telephone domain data (*source* domain) used to train CycleGAN system was the same as the data used to train x-vector system. It consisted of recordings from datasets SRE04-10, Mixer6 and Switchboard 1-Phase 1, 2, and 3. This comprised of 90946 utterances from 6986 speakers. Microphone data used in CycleGAN training was sampled from two corpora: Development corpus of SITW (referred to as SITW *dev*) and *VoxCeleb*1 [53]. Together they consist of 24581 utterances. SITW *dev* contains 4439 utterances from 119 speakers.

SITW evaluation corpus (referred as SITW *eval*) was used to evaluate the system. No speaker overlap exists between SITW *eval* and *dev* corpora. Training of CycleGAN does not require speaker labels from both the domains. The microphone speech was down-sampled to 8 kHz to match the sampling frequency of telephone speech.

4.3.2.2 CycleGAN Training Procedure

Similar to x-vector system, CycleGAN system was trained on 40-dimensional log mel-FB features with short time centering. Energy VAD was applied on centered features to remove the non-speech frames. Two mini batches of features were sampled randomly from *source* and *target* domain during each training step. Since no parallel data exists between both the domains, the batches were drawn in a completely random fashion. The sizes of both mini batches were set to 32. For the number of contiguous frames sampled from each utterance (sequence length, denoted as F in Table 4.4), we experimented with sequence lengths 11 and 127. Since we used 2D CNNs, all the mini batches were arranged as four dimensional tensors of size (32, 1, F, 40). The model was trained for 50 epochs. Each epoch was set to be complete when all the telephone utterances have appeared once in that epoch. Adam Optimizer was used with momentum $\beta_1 = 0.5$ as suggested by [84]. The learning rates for the generators and discriminators were set to 0.0003 and 0.0001 respectively. The learning

rates were kept constant for the first 15 epochs and, then, linearly decreased until they reach the minimum learning rate (1e-6). The cycle loss weight was set to 2.5 and adversarial loss weight was set to 1.0. We used PyTorch [75, 85] for the CycleGAN implementation.

4.3.3 Adaptation Results

Table 4.4 presents results of both baseline system and two adaptation systems used in this work. Both the adaptation systems differ in the way the respective CycleGANs used in adaptation were trained.

As discussed in Section 4.3.1.1, CycleGAN was trained using a combination of *cycle-consistency* and *adversarial* losses. The first adaptation system used a CycleGAN trained with *identity loss* along with the above two losses mentioned(details in Section 4.3.1.2 and Equation 4.10). The system was termed as 'adaptation system using CycleGAN trained with identity loss'. The advantage of using identity loss for training CycleGAN was two fold – it acted as a regularizer and it helped the generators learn to preserve useful information while trying to achieve domain transfer. However, it increased the training time of CycleGAN since features from both domains were passed through each generator. For instance, source domain features of training data were passed through $G_{\mathrm{S}\to\mathrm{T}}$ to minimize the adversarial and cycle consistency losses. Additionally, the target domain features of training data were passed through

	SITW Core		SITW Assist-Multi	
	EER	**DCF**	**EER**	**DCF**
Baseline System	10.14	0.6842	12.72	0.6941
Adaption System using CycleGAN trained				
with *identity* loss (F-11)	9.74	0.6754	12.25	0.6816
with short-cut connection (F-11)	9.19	0.6649	11.51	0.6797
with short-cut connection (F-127)	**8.87**	**0.6548**	**10.78**	**0.6643**

Table 4.4: Comparison of CycleGAN domain adaptation on the SITW *eval* (F stands for sequence length used to train CycleGAN).

the same generator to minimize identity loss. This adaptation system yielded improvements over baseline system but slowed down the training process.

One other way to help the generators learn to preserve information was to add a short cut connection from input to output. The input is added to the output of last convolutional layer. Hence, the overall output of the generator is given by x + conv(x), where x is the the input to the generator. The short-cut connection was shown as a part of the generator in Table 4.1. For the CycleGAN trained with identity loss, the generator architecture is similar to Table 4.1 but without the short-cut connection. Since, the short-cut connection preserved the information, identity loss was not used in the training. The objective used in training was Equation 4.7. This adaptation system, was referred as 'adaptation system using CycleGAN trained with short-cut connection'. As observed from the table, CycleGAN trained with short-cut connection yielded better results than the one trained with *identity* loss. The former also was faster to train

compared to the later. Both the CycleGANs discussed so far were trained with 11 frames as input (represented as F=11 in the table). We then experimented with training the generator with a longer context (F=127) which yielded better results than the one trained with 11 frames. For the rest of this chapter all the CycleGANs were trained with a short-cut connection in the generator with the objective Equation 4.7 and with 127 number of frames as input.

4.3.4 Adaptation During Testing vs. Adaptation During Training

The experiments in previous section fall under the 'adaptation during testing' scheme, since the evaluation data was mapped to *source* domain data before extracting the x-vectors. In this section, we compare it with 'adaptation during training' - adapting the training corpora to *target* domain using $G_{\mathrm{S}\to\mathrm{T}}$. The x-vector network was trained on the mapped features, which are now in microphone domain. In this scenario, we use the original evaluation data (no mapping required) and forward pass them through the x-vector network trained on mapped features. The comparison of both the schemes is presented in Table 4.5. The adaptation system used in the former scenario is equivalent to train and test on telephone domain. Hence, termed as telephone system in Table 4.5. In the later scenario, adaptation system is equivalent to train and

	Domain		SITW Core		SITW Assist-Multi	
	train	**test**	**EER**	**DCF**	**EER**	**DCF**
Baseline	tel	mic	10.14	0.684	12.72	0.694
Telephone	tel	mic $\rightarrow$ tel	8.87	0.655	10.78	0.664
Microphone	tel $\rightarrow$ mic	mic	8.28	0.643	10.44	0.643

Table 4.5: Adaptation during testing vs. adaptation during training

test on microphone domain. Hence, it is termed as microphone system. The results suggest that 'adaptation during training' yielded better results compared to its counterpart.

4.4 Low-Resource Domain Adaptation

Our unsupervised adaptation approach using CycleGAN yielded encouraging results on SITW *eval* as shown in previous section. We used development portion of SITW and the much larger VoxCeleb1 dataset [53] as *target* domain data to train CycleGAN. In this section we experimented with a low-resource scenario where we have limited amount of data[5] from target domain to train CycleGAN. This is a more practical scenario, since it is not always reasonable to expect large amounts of data from *target* domain during the training stage. To motivate our experiments, we trained a CycleGAN on only SITW *dev* and compared it to the CycleGAN we presented in previous section – which was

[5]Limited amount of data, in this chapter, refers to having lower amount of data (in number of hours) and also collected from limited number of speakers.

	SITW Core-Core		**SITW Assist-Multi**	
	EER	**DCF**	**EER**	**DCF**
Baseline system S	10.14	0.6842	12.72	0.6941
Adaptation system	**8.87**	**0.6548**	**10.78**	**0.6643**
Adaptation system low-resource	9.51	0.6608	11.43	0.6683
Baseline system S & T	7.90	0.6226	10.14	0.6418

Table 4.6: Results of low-resource channel adaptation system

trained on SITW *dev* and VoxCeleb1. SITW *dev* has only 191 number of speakers. We refer to the adaptation system that uses CycleGAN trained on limited amount of data as 'adaptation system low-resource'. The experimental results are presented in Table 4.6. In addition to the baseline system that was only trained on *source* domain data, referred as 'baseline system S', we also compare our adaptation systems with a baseline system that is trained on both *source* and *target* domain. We refer to the later system as 'baseline system S & T'. Specifically, it is trained on all the *source* domain data, mentioned in the previous section. In addition, we also use SITW *dev*, which was sampled from the *target* domain, to train the ASV system. Since, we used the *target* domain data along with its speaker labels we consider the second baseline system as a supervised adapted system. Hence, the performance of this baseline system serves as a upper limit for the unsupervised adaptation system.

The results in Table 4.6 demonstrate that CycleGAN trained with more amount of data performs better than the CycleGAN trained with limited amount

of data. We attribute the drop in performance to over-fitting phenomenon of CycleGAN, when it was trained on limited amount of training data. However, it is not always practical to assume the availability of large amounts of *target* domain data during training. In the following section, we present a training strategy that acts as a regularizer and improves the performance of CycleGAN trained under low-resource scenario. The specific training strategy we are talking about is to add noise to the *target* domain data. Below, we explain the training procedure of CycleGAN with noise addition.

4.4.1 Training Procedure with Noise Addition

Similar to previous section, we assume the availability of training corpora to train CycleGAN. It consists of audio samples $\mathbf{A}_{\mathrm{S}} = \{\mathbf{a}_{\mathrm{S},i}\}_{i=1}^{N}$ and $\mathbf{A}_{\mathrm{T}} = \{\mathbf{a}_{\mathrm{T},i}\}_{i=1}^{M}$ drawn from two different domains: *source* S and *target* T with distributions $\mathbf{a}_{\mathrm{S},i} \sim q_{\mathrm{S}}(\mathbf{a})$ and $\mathbf{a}_{\mathrm{T},i} \sim q_{\mathrm{T}}(\mathbf{a})$ respectively. In this section, we assume $M << N$. Noise is then added to the *target* domain audio samples (procedure in Section 4.4.3) which results in a transformed *target* domain distribution T'. Audio samples with noise added to them are represented as $\mathbf{A}_{\mathrm{T}'} = \{\mathbf{a}_{\mathrm{T}',i}\}_{i=1}^{M}$. Audio samples from $\mathbf{A}_{\mathrm{S}}$ and $\mathbf{A}_{\mathrm{T}'}$ are considered as the *source* and *target* domain distributions to traing the CycleGAN. The rest of the training procedure is identical to the training procedure in previous section.

Similar to the training procedure in Section 4.3.1.1, speaker labels from

either domain are not needed to train the feature mapping function. Noise is added to the *target* domain audio only during training. During evaluation original audio samples of the evaluation data sampled from the *target* domain data are used to extract filter bank features which are then mapped to the *source* domain. Similar to the previous section, the *target* domain data used to train and evaluate the feature mapping system has no speaker overlap.

4.4.2 Modified CycleGAN Objectives

As explained in previous section, the target domain audio samples are sampled from the transformed domain (obtained from noise addition) $\mathrm{A}_{\mathrm{T}'}$ instead of the original domain A_{T}. Log mel-filter bank features are extracted from audio samples of *source* and transformed *target* domain distributions denoted as $\mathbf{X}_{\mathrm{S}} = \{\mathbf{x}_{\mathrm{S},i}\}_{i=1}^{N}$ and $\mathbf{X}_{\mathrm{T}'} = \{\mathbf{x}_{\mathrm{T}',i}\}_{i=1}^{M}$. The distributions in filter bank space are $\mathbf{x}_{\mathrm{S},i} \sim p_{\mathrm{S}}(\mathbf{x})$ and $\mathbf{x}_{\mathrm{T}',i} \sim p_{\mathrm{T}'}(\mathbf{x})$ respectively. $\mathbf{X}_{\mathrm{S}}$ and $\mathbf{X}_{\mathrm{T}'}$ are used as features from two different distributions to train the feature mapping functions.

The generator objective with noise addition to target domain is represented as in Equation 4.12. L_{adv} and L_{cyc} are defined in Equation 4.6 and Equation 4.3 respectively.

$$
\begin{aligned}
& L(G_{\mathrm{T}\to\mathrm{S}}, G_{\mathrm{S}\to\mathrm{T}}, D_{\mathrm{T}}, D_{\mathrm{S}}, \mathbf{X}_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}'}) \\
& = \lambda_{adv} L_{\mathrm{adv}}(G_{\mathrm{T}\to\mathrm{S}}, D_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}'}) \\
& + \lambda_{\mathrm{adv}} L_{\mathrm{adv}}(G_{\mathrm{S}\to\mathrm{T}}, D_{\mathrm{T}}, \mathbf{X}_{\mathrm{S}}) \\
& + \lambda_{\mathrm{cyc}} L_{\mathrm{cyc}}(G_{\mathrm{S}\to\mathrm{T}}, G_{\mathrm{T}\to\mathrm{S}}, \mathbf{X}_{\mathrm{S}}, \mathbf{X}_{\mathrm{T}'})
\end{aligned} \tag{4.12}
$$

4.4.3 Noise Addition Procedure

Noise addition procedure to the *target* domain audio samples is as follows. To add noise we used 930 ”noise” samples from MUSAN [47] corpus. Noise was added as foreground noise [48] at the interval of 1 second with the signal-to-noise ratio (SNR)s ranging from 0 to 15 dB. The ”music” and ”babble” portions of MUSAN corpus were not used in this work. Noise addition was done only on *target* domain data during the training of CycleGAN system. The original *target* domain data (without noise) was not used during training. While forward passing the SITW *eval* features through CycleGAN, noise was not added.

4.4.4 Results

In this section, we present results of low-resource adaptation system. The intuition behind adding noise is that the addition of noise acts as a regularizer

	SITW Core-Core		SITW Assist-Multi	
	EER	**DCF**	**EER**	**DCF**
Baseline system S	10.14	0.6842	12.72	0.6941
Adaptation system low-resource without noise	9.51	0.6608	11.43	0.6683
Adaptation system low-resource with noise	8.91	**0.6495**	**10.71**	**0.6608**
Adaptation system	**8.87**	0.6548	10.78	0.6643
Baseline system S & T	7.90	0.6226	10.14	0.6418

Table 4.7: Results for low-resource channel adaptation trained with noise

during training and prevents over-fitting. Results are in Table 4.7.

Adaptation system LT trained with noise had much better performance compared to *Baseline system S* and slightly better results compared to *Adaptation system*, which was trained with larger *target* domain data. In next chapter, we provide extensive empirical evidence that noise addition acts as a regularizer and prevents over-fitting.

4.5 Summary

In this chapter, we experimented with a channel mismatch scenario where the ASV system was trained on data acquired using *telephone* channel and evaluated on *microphone* data. To address this we presented an unsupervised channel adaptation technique *via* feature mapping. Our feature mapping network is a 'deep residual convolutional network' trained on *unpaired* data ac-

quired from both the domains using the CycleGAN framework. Our approach yielded 10.1% and 4.5% relative improvements on EER and *minDCF* on SITW *core* condition, when microphone features were mapped to telephone domain during evaluation. We also experimented with mapping telephone features to microphone domain, and use the mapped features for training. This approach yielded slightly better results compared to the previous approach [30] (details in Section 4.3).

We also experimented with training the feature mapping network in a low-resource scenario, where limited *target* domain data was available for training. We observed that the CNN tended to overfit in this scenario. To circumvent this, we presented a simple regularization technique, using which the CNN trained on limited data performed almost similar to the CNN trained on much larger data [31] (details in Section 4.4).

The main advantage of our approach is to make use of *unpaired* real data from both domains to learn a domain adaptation mechanism, thus, avoiding the need for using simulation. In the next chapter, we extend this approach to increase the robustness of SV systems to far-field testing scenarios.

Chapter 5

Far-field Feature Enhancement

5.1 Introduction

Speech signals get contaminated by various background noises, reverberation and other unwanted variabilities present during their acquisition. An ideal automatic speaker verification (ASV) system should be robust to any background noises and reverberation effects present. During the time this thesis was written, developing robust ASV systems has been a very active research area. Several challenges were organized such as NIST Speaker Recognition Evaluation (SRE) 2019, VOiCES from a Distance Challenge [86], and VoxCeleb Speaker Recognition Challenge 2019. One of the main research topics of 2019 sixth Frederick Jelinek memorial summer workshop was '*Speaker detection in adverse scenarios with a single microphone*' [87].

An ASV system trained solely on clean speech may not perform well on noisy test conditions. Similarly, a system trained on close talk speech may not perform well on far-field test conditions. One approach to improve the robustness of verification system to degraded speech[1] is to train it on noisy and reverberant speech. However, acquiring speech data with speaker labels (also referred as labelled speech) from such degraded conditions is not very practical, considering the expenses involved in data acquisition and manual data labelling. An alternate way to obtain labelled degraded speech for training is to use a simulation strategy [14, 48]. Noisy data can be created by artificially adding noise files to the original training data – considered clean and close talk speech in this chapter. Reverberant speech can be simulated by artificially convolving room impulse responses (RIRs) with close talk speech [48, 88]. The simulated degraded speech, along with the original clean training data, can be used to train the verification system. This method, known as data augmentation, has proven to be very effective in improving the performance of ASV systems, yielding SOTA results on various tasks [14, 15, 89]. Data augmentation for x-vector (and PLDA) training was originally proposed as an inexpensive approach to gain access to large amounts of speaker labelled data. Training x-vector network on large amounts of degraded speech reduces overfitting, thus improves the generalization ability of the network. It also enables training larger (deeper

[1] In this chapter, we use the term degraded speech to refer to reverberant and/or noisy speech

and wider) x-vector networks, which improves the expressive nature of the embedding networks.

In this chapter, we study training ASV system using data augmentation from a robustness perspective. Increasing the robustness of the system to far-field speech by data augmentation can be seen as a supervised domain adaptation (SDA) approach (for definition of SDA, refer to Section 2.4.1). The simulated data, comprising of far-field and noisy speech, is treated as adaptation data. Adaptation data along with the speaker labels is combined with the original clean training data, which is used for training the system. However, such simulation strategies do not take into account the amount and type of degradation the test utterances may have. In other words, if the test condition happens to be different from the training condition, the performance of the system suffers. A recent study on Speaker Diarization on children's speech [90] demonstrates various challenges that x-vector systems face in adverse scenarios.

In this work, we experimented with a 'single channel wide-band far-field feature enhancement'[2] approach to improve (enhance) the quality of speech features with the end goal of improving the performance of ASV systems. Our approach was to train a DNN to enhance the features of evaluation data, considered as degraded. The network was trained on acoustic features extracted from both degraded and clean data. Speaker labels were not required to train

[2]We used the term feature enhancement to refer to improving the quality of acoustic features. Speech enhancement, on the other hand, refers to improving intelligibility and/or overall perceptual quality of degraded speech signal. In this work we focus on feature enhancement.

network. Once trained, the DNN mapped the features of evaluation data to clean domain. The enhanced features of evaluation data were used to extract speaker embeddings, which were then used for scoring. We experimented with two enhancement networks in this work (details below in this section), both were trained to enhance acoustic features extracted from single channel far-field wide-band speech. Hence, we call our approach as 'single channel wide-band far-field feature enhancement'. Since, the training of the enhancement networks does not require access to unlabelled data, the enhancement approach can be treated as a special case of 'unsupervised domain adaptation' (UDA) approach.

We experimented with two types of ASV systems in this chapter: one trained solely on clean training data, and other trained using data augmentation method described above. The system trained only on clean data and tested on degraded data can be seen as operating under mismatched conditions. Hence, the system would suffer a loss in performance. The enhancement network, in this case, helps improve the performance by mapping the degraded evaluation features to clean domain. Hence, the ASV system trained on clean data and equipped with an enhancement network, can be seen as trained and evaluated on clean condition. This approach yielded encouraging results, as shown later in Section 5.3 and Section 5.4. The approach also falls under 'adaptation during testing' scenario (details in Section 2.4.2.1).

The motivation for using enhancement to improve the performance of ASV system remains clear in the above scenario where the system was trained using only data from clean domain. In the second scenario, where ASV system was trained using data augmentation, we investigated if enhancing the test features would further improve the performance. From a robustness perspective, training the system using data augmentation procedure described earlier can be considered as *multi-condition* training (since, training data comprised of clean, noisy and far-field speech). Enhancement, in this case, would be beneficial if the test condition happens to be different from any of the conditions seen during training. From domain adaptation perspective, as explained earlier, training the ASV system using data augmentation can be considered as a SDA approach to increase the robustness of the system to degraded testing conditions. Since, the training of enhancement network required unlabelled data, enhancement approach can be considered as an unsupervised domain adaptation approach. Hence, in this scenario our approach was motivated to investigate if an ASV system trained using *multi-condition* data (a SDA approach) would further benefit from enhancement (an UDA approach).

Training the enhancement networks required access to unlabelled data from both degraded and clean speech. We experimented with both unsupervised and supervised approaches[3] for training the enhancement networks in this work.

[3]The usage of the terms supervised and unsupervised approaches to train the enhancement networks should not be confused with supervised and unsupervised domain adaptation ap-

Unsupervised and supervised enhancement approaches stand for the usage of *unpaired* (non-parallel) and *paired* (parallel) degraded-clean speech to train the enhancement network. For details on description of *unpaired* and *paired* data, refer to Section 2.4.3. The enhancement networks trained with unsupervised and supervised approaches were termed as unsupervised enhancement network (UEN) and supervised enhancement network (SEN) respectively.

We first experimented with unsupervised enhancement approach (details in Section 5.3). Motivation behind taking an unsupervised approach for training the Unsupervised Enhancement Network (UEN) was to incorporate the knowledge of the target (adverse) domain in the enhancement training procedure with the help of some unlabeled training data (different from evaluation) from that domain, thus avoiding the need for simulation. Encouraged by the success of CycleGAN [16] for the unsupervised channel adaptation task in previous chapter, we experimented with CycleGAN framework for the unsupervised enhancement task (details in Section 5.3). UEN is the generator in CycleGAN that maps features from degraded domain to clean domain. As explained in previous chapter, CycleGAN trained using *unpaired* data optimizes a combination of cycle-consistency and adversarial losses [28]. Adversarial loss helps the network learn it's own loss function [29] to make the target domain

proaches. The supervised (or unsupervised) enhancement approach refers to usage of *paired* (or *unpaired*) data for training the enhancement network. The supervised (or unsupervised) domain adaptation approach refers to usage of *labelled* (or *unlabelled*) data to adapt the system to some target domain.

features match with the source domain features (clean in this case).

supervised enhancement network (SEN), on the other hand, was trained on *paired* reverb-clean data. The *paired* training data was obtained by simulation – reverberant speech was simulated from clean training speech, A conventional approach to train enhancement network with *paired* data is to force the output features of enhancement network to appear same as the clean features, usually achieved by minimizing distance metrics like L_1 or MSE. We provide training details of SEN and a comparison of both approaches in Section 5.4.

Along with SEN and UEN, we also experimented with a domain adaptation network (DAN) that maps the far-field features to some chosen domain different from clean. domain adaptation network (DAN) was mainly targeted at improving the performance of ASV system trained on multi-condition data. The motivation behind mapping far-field features to some other domain but clean using a DAN was as follows: if there existed a domain which improved the performance of ASV system when trained on, compared to performance of a system trained on clean domain, then, mapping the evaluation features to that domain would improve the performance. Similar to UEN, DAN was also trained using a CycleGAN on *unpaired* training data obtained from reverb domain and the chosen domain (details in Section 5.5).

SEN, UEN and DAN were all trained on log mel-FB features. Hence, the enhancement was termed as feature enhancement. Since, all these networks

were trained using unlabelled data, the overall adaptation procedure was unsupervised.

Our experimental approach was as follows: we first trained SEN, UEN and DAN with their respective training data and objectives. Once trained, we used them individually to enhance/adapt the features of the test data (enrollment and evaluation data) before extracting x-vectors. In the case where *multi-condition* data was available to train the system, we also used the networks described above to enhance/adapt the features of *multi-condition* data and train the x-vector on enhanced features. In the later case adaptation was done during both training and testing.

The rest of the chapter is organized as follows. In Section 5.2 we first describe both the baseline ASV systems used in this work. In Section 5.3, we describe the training details of UEN and provide experimental results of using UEN to improve the performance of an ASV system trained solely on clean data (without any data augmentation). In Section 5.4, we describe the training procedure of SEN and compare it with UEN. In Section 5.5, we describe the training procedure of DAN and compare it's performance with UEN. In Section 5.6, we discuss the efficacy of using all three networks in improving the performance of ASV system trained using *multi-condition* data. Finally in Section 5.7, we summarize the experimental observations and we highlight the main contributions of this chapter.

5.2 Baseline Systems

In this chapter, we mainly consider two baseline ASV systems: one trained solely on *clean* data (without any data augmentation) and other trained on *multi-condition* data (comprised of clean, noisy and far-field corpora). The former baseline was referred as 'baseline ASV *clean*' in this chapter, while the later was referred as 'baseline ASV *multi-condition*'. We refer to both the baselines as systems trained on *source* domain data (with or without data augmentation) and tested on *target* domain data *without any enhancement*. All the baseline systems were trained on wide-band (WB) data. The dimension of MFCC was 40. For the x-vector network in our baseline system, we experimented with an ETDNN architecture [15]. ETDNN improves upon TDNN [14] by interleaving dense layers in between the convolution layers. More details on the ETDNN network and the pipeline can be found in [15, 87].

5.2.1 Details of Training Data

The clean training data used for training 'baseline ASV *clean*' was obtained from VoxCeleb1 [53] and VoxCeleb2 [91]. The files from the same YouTube video of VoxCeleb1 and VoxCeleb2 were concatenated, denoted as VoxCelebCat, to obtain longer audio sequences. Since VoxCelebCat was collected in wild conditions and contained unwanted background noise, we filtered the files

based on their SNR, estimated by Waveform Amplitude Distribution Analysis (WADASNR) algorithm [31, 32, 92, 93]. We retained the files with SNR greater than 19 decibel (dB). The high SNR signals, thus obtained, termed as VoxCelebCat *clean*, consisted of 1665 hours of speech from 7104 speakers.

Augmentation data used for training the 'baseline ASV *multi-condition*' was obtained by simulation. Simulation strategy we followed in this work is similar to the work in [14]. We applied two types of simulation - additive noise based and simulation of far-field speech. We used noise files from MUSAN [47] corpus for additive noise based simulation and RIRs available at `http://www.openslr.org/26`, which consisted of both simulated and real RIRs, for far-field speech simulation. MUSAN corpora consists of noise files from three different subcategories: *music* files from several genres ($\approx$ 42 hours and 31 minutes), *speech* from twelve languages ($\approx$ 60 hours) and *noise* files consisting of wide assortment of technical and non-technical noises ($\approx$ 6 hours). The simulated RIRs used for far-field simulation were divided into three sets based on the ranges from which width and length of the room were sampled from: *small*, *medium* and *large* room sets. The width and length of *small*, *medium* and *large* rooms sets were uniformly sampled from ranges 1-10m, 10-30m and 30-50m respectively. In all the three sets, room height was sampled uniformly from 2-5m; and absorption coefficient was sampled uniformly from [0.2; 0.8]. In each set, 200 rooms were sampled and 100 RIRs were sampled in

each room based on speaker and receiver position. The distance between the speaker and the receiver was not greater than 5m.

The goal of our work in this chapter is to develop adaptation techniques to make ASV system robust to far-field and noisy test conditions. Hence, for the robustness study we tested our systems on both real speech acquired in wild (uncontrolled) conditions and simulated test conditions. An ideal adaptation system should generalize to unseen far-field and noisy conditions. To simulate such conditions, we made sure there was no overlap between the noise files and RIRs used for simulating the train and test conditions. To accomplish this, the *noise*, *speech* and *music* portions of MUSAN corpus were split into two sets: one for simulating the train speech and other for simulating the test conditions. We followed a 90-10 split strategy: 90% of noise files were used for simulating the training data and rest 10% were used for simulating the test corpora. In addition to noise files from MUSAN, we also used noise files acquired from Chime-3 [94] corpora for creating noisy test corpora, referred as *chime3bg*.

Simulated RIRs were first split into four different subsets based on their Reverberation Time (RT)60 – the time taken for the sound pressure level to decrease by 60 dB value. Sabine's formula was used for computing RT60 from RIRs and is given by

$$RT60 \approx 0.1611 sm^{-1} V/S\alpha \tag{5.1}$$

- RT60 in Equation 5.1 stands for time taken for the sound pressure level to decrease by 60 dB value, in *seconds*,
- V stands for the volume of the room, in *cubic meters*,
- S stands for the surface area of the room, in *square meters*, and
- α stands for the absorption coefficient value.

The RT60 ranges considered were 0.0-0.5, 0.5-1.0, 1.0-1.5 and 1.5-4.0[4]. Within each subset 90-10 strategy was used for splitting RIRs into train and test sets. Similar splitting strategy was applied to real RIRs (315 in number) also. Figure 5.1 summarizes the noise and RIR files used for simulation in this work.

'Baseline ASV *clean*' was trained on VoxCelebCat *clean* described above. To train 'baseline ASV *multi-condition*', VoxCelebCat *clean* was combined with a random subset of VoxCelebCat *noise*, VoxCelebCat *music*, VoxCelebCat *babble* and VoxCelebCat *reverb_noise*. VoxCelebCat *noise*, VoxCelebCat *music* and VoxCelebCat *babble*, together referred as VoxCelebCat *addtitive*, were obtained

[4]RIR to RT60 mapping is available in the file `https://github.com/jsalt2019-diadet/jsalt2019-diadet/blob/master/src/kaldi_augmentation/simrir2rt60.info`

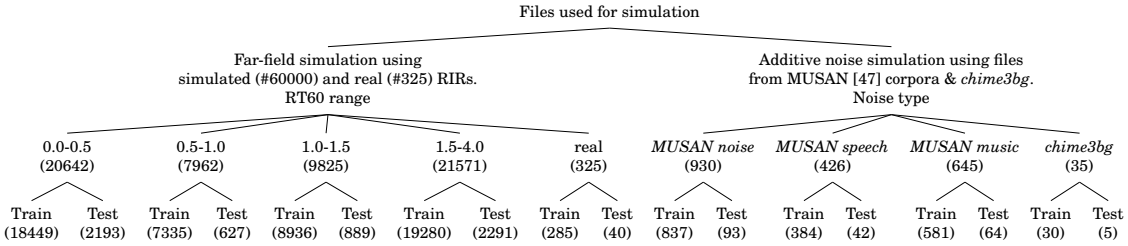


Figure 5.1: Summary of additive noise files and RIRs used in simulation of train and test data. Far-field simulation was done using simulated and real RIRs. Simulated RIRs were further split into four different sets based on their RT60 values. Additive noise simulation was done by using MUSAN corpora and noise files from Chime3 (denoted as *chime3bg*). The files used for train and test simulation were obtained by making a 90-10 split. Total number of files in each category are shown in parenthesis.

by adding assorted *noise*, *music* and *speech* files from MUSAN corpus to VoxCelebCat *clean*. The SNR of the noisy files obtained were randomly chosen from 15, 10, 5 and 0 dB. *noise* files from MUSAN were added as *foreground* noise [48], *music* and *speech* were added as background noise. The *multi-condition* data also consisted of far-field speech, termed as VoxCelebCat *reverb_noise*, which was obtained by first convolving the simulated RIRs (described above) with VoxCelebCat *clean* and then adding the *noise* files from MUSAN corpus as foreground noise. Simulated RIRs whose RT60 were in range 0.0-1.0 were used in simulation. The files used for simulation were sampled only from the training portion, as described in Figure. 5.1. Finally, the random subset was chosen to be twice the size of VoxCelebCat *clean* (in terms of number of utterances). Table 5.1 summarizes the datasets used for training the baseline systems.

Train				
Real	**Simulated**			
Clean	Additive noise			Far-field
VoxCelebCat *clean*	VoxCelebCat *noise*	VoxCelebCat *babble*	VoxCelebCat *music*	VoxCelebCat *reverb_noise*

Table 5.1: Summary of the datasets used in training the baseline ASV systems. We experimented with two baseline systems in this work: 'baseline ASV *clean*' and 'baseline ASV *multi-condition*'. 'baseline ASV *clean*' was trained only on VoxCelebCat *clean*. 'baseline ASV *multi-condition*' was trained by combining VoxCelebCat *clean* with a random subset of simulated datasets shown in the Table. For more details on individual datasets refer to Section 5.2.1

5.2.2 Datasets Used For Testing

The main goal of the enhancement work in this chapter is to make ASV system robust to far-field and noisy test conditions. To facilitate the study, we first obtained far-field and noisy test conditions *via* simulation. Once we demonstrate the effectiveness of our enhancement approach on simulated test conditions, we also test our approach on real speech acquired in wild (uncontrolled) conditions. Below we explain datasets used for testing.

5.2.2.1 Real Datasets Used For Testing

For the real testing conditions, we used three different corpora [87] collected in different scenarios:

- **Meeting** (AMI Meeting Corpus (AMI) [95]): with a setting of 3 different meeting rooms with 4 individual headset Microphones, 8 Multiple Distant Microphones forming a microphone array; 180 speakers x 3.5 sessions per speaker (sps). Since

we are exploring enhancement with single microphone, we focused only on the mix Headset.

- **Indoor controlled** (Stanford Research Institute (SRI) data [96]): with a setting of 23 different microphones placed throughout 4 different rooms; controlled backgrounds, 30 speakers x 2 sessions and 40 h, live speech along with background noises (TV, radio).

- **Wild** (***BabyTrain***): with an uncontrolled setting, 450 recurrent speakers, up to 40 sps (longitudinal), 225hrs; suitable for diarization and detection.

The enrollments for speaker verification were generated by accumulating non-overlapping speech (5, 15 and 30s duration) of every target speaker along one or multiple utterances. For the test, we cut the audio into 60 second chunks. We did a cartesian product between the enrollments and the test segments to generate all possible trials. Then, based on certain criteria, some trials were filtered out. For example, same session and same microphones were not allowed to produce a target-trial pair. For more details on the testing corpora refer to [87].

5.2.2.2 Simulated Datasets Used For Testing

We treated SITW [68, 97] as clean test set in this chapter. We obtained all simulated test conditions from SITW *eval core-core* conditions, simply referred

as SITW. Similar to the training datasets (discussed in Section 5.2.1), we experimented with two types of simulations: far-field and additive noise. Simulated reverberant test set was obtained from SITW, labelled as SITW *reverb*, by convolving speech files from SITW with simulated and real RIRs (discussed in Section 5.2.1 and Figure. 5.1). We split the simulated RIRs into four sets based on their RT60 values: 0.0-0.5, 0.5-1.0, 1.0-1.5, 1.5-4.0. By using these four sets of RIRs we obtained four different copies of simulated reverberant speech. We also used real RIRs to obtain a copy of reverberant speech. All the five copies of reverberant speech, obtained *via* simulation, is referred as SITW *reverb* in this chapter. We test each system individually on these five different copies and we take an average of all the results.

Though our enhancement networks were trained to perform dereverberation task, the networks should be able to enhance noisy test signals as well, to make the overall setup work in wild conditions, where background noise also exist in speech recordings. To make sure our enhancement setup also enhances noisy signals, we prepared simulated noisy test conditions by adding noise to SITW. We added *music*, *speech* and *noise* files from MUSAN corpus to SITW. The resultant noisy test conditions were termed as SITW *music*, SITW *babble* and SITW *noise* respectively. We also added noise files from *chime3bg* to SITW to obtain a noisy test set termed as SITW *chime3bg*. The SNRs used for testing were different from training. For testing we used SNRs 17, 12, 7, 2 and -5 dB

whereas for training we randomly sampled SNR from 15, 10, 5 and 0 dB. We finally average the results of all individual SNRs within each noisy condition. Thus, our overall setup ensured we test on unseen conditions (*music*, *babble* and *chime3bg*) and SNR levels.

We ensured noise files and RIRs used for training and testing simulations were disjoint, as shown in Figure 5.1. Figure 5.2 shows a summary of all test corpora used for evaluation in this work.

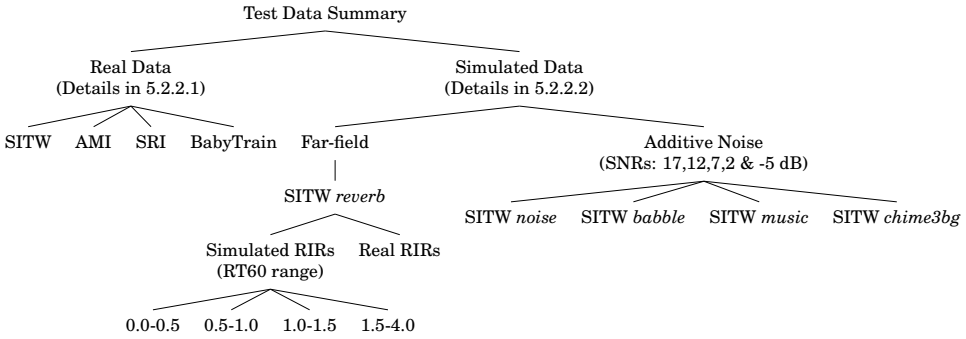


Figure 5.2: Summary of test datasets used in far-field adaptation study

5.2.3 Results of Baseline Systems

In this section, we present results of the two baseline systems – *clean* and *multi-condition*. Our experimental approach was as follows: we first used the 'baseline ASV *clean*' to tune and demonstrate the effectiveness of enhancement networks – UEN and SEN – in improving the performance of speaker verification. The testing was done on both simulated and real conditions. Once tuned, we tested the effectiveness of the enhancement networks to improve the perfor-

Test Condition	RIR Type, RT60 range	EER	minDCF
SITW	-,-	5.02	0.327
SITW *reverb*	Simulated, 0.0-0.5	5.84	0.405
	Simulated, 0.5-1.0	6.95	0.504
	Simulated, 1.0-1.5	6.34	0.459
	Simulated, 1.5-4.0	6.03	0.415
	Real, -	6.53	0.450
SITW *reverb*	Average	6.34	0.447

Table 5.2: Results of 'baseline ASV *clean*' tested on SITW and SITW *reverb*. For details on SITW *reverb*, refer to Section 5.2.2.2.

mance of ASV system trained on *multi-condition* data by testing on real data acquired from wild conditions.

We first present results of 'baseline ASV *clean*', which was trained on VoxCelebCat *clean* (details in Section 5.2.1) and tested on simulated conditions (details in Section 5.2.2.2). Table 5.2 presents results on SITW and SITW *reverb*. SITW *reverb* consisted of several test conditions, which were classified based on the RIR type used (simulated or real) and the RT60 value of the RIR (details in Section 5.2.2.2). The final result was presented by taking an average of all the test conditions. It can be observed from the results that reverberation deteriorates verification performance compared to clean testing conditions: SITW and SITW *reverb* yielded 0.327 and 0.447 respectively (in terms of *minDCF*). Hence, the usage of enhancement of far-field speech to improve the verification performance was justified.

Speech acquired in wild conditions can also have different types of background noises. To make the ASV system perform well in wild conditions,

we should make it robust to noisy speech as well. In order to study this, we tested 'baseline ASV *clean*' on four noisy conditions – SITW *noise*, SITW *babble*, SITW *music* and SITW *chime3bg* – obtained *via* simulation (details in Section 5.2.2.2). Results are presented in Table 5.3. We tested each noise condition at different SNR levels, different from SNRs used in training. Finally, we average the results across all SNRs within each noise condition. We represented the original SITW condition (considered as *clean*) as having infinite (∞) SNR. Some observations from the table:

- Compared to clean condition (∞ SNR), the verification performance deteriorated slightly at SNR condition (17 dB). For instance, SITW *music* at 17 dB obtained 0.345 *minDCF* compared to 0.327 of SITW. This observation remains consistent across all four test conditions.

- When the SNR dropped, the performance dropped significantly. At -5 dB (very low SNR condition), the performance deteriorated quite significantly (0.864 on SITW *music* compared to 0.327 on SITW).

- Among all the conditions, *babble* condition was more challenging – obtained very poor results compared to the other three conditions.

As explained earlier, we used 'baseline ASV *clean*' and simulated test conditions to tune the enhancement networks and demonstrate the effectiveness of enhancement in improving the speaker verification performance. Once tuned,

SNR	**SITW *music***		**SITW *babble***		**SITW *noise***		**SITW *chime3bg***	
(dB)	EER	*minDCF*	EER	*minDCF*	EER	*minDCF*	EER	*minDCF*
Baseline ASV *clean*								
∞	5.02	0.327	5.02	0.327	5.02	0.327	5.02	0.327
17	5.14	0.345	5.36	0.349	5.14	0.351	5.10	0.341
12	5.55	0.368	5.96	0.386	5.54	0.381	5.44	0.380
7	6.36	0.424	7.6	0.497	6.35	0.440	6.48	0.473
2	8.65	0.563	12.94	0.764	8.00	0.544	9.81	0.693
-5	16.13	0.864	28.48	0.999	12.98	0.754	21.46	0.993
Avg.	8.37	0.513	12.07	0.599	7.60	0.494	9.66	0.576

Table 5.3: Results of 'baseline ASV clean' tested on four different noisy conditions: SITW *music*, SITW *babble*, SITW *noise* and SITW *chime3bg*. For comparison, we presented the results of original SITW (represented with ∞ SNR). For details on the testing conditions, refer to Section 5.2.2.2.

we used those enhancement networks to improve the verification performance when tested on wild conditions. In the second scenario, we considered training ASV system on both *clean* data and *multi-condition* data, which is a more practical scenario. We present baseline results on three different real datasets – AMI, SRI and BabyTrain (details in Section 5.2.2.1) – in Table 5.4. As can be observed from the table, all the three datasets posed significant challenge to the verification performance (yielded very high EER and DCF). The SDA approach to train ASV system on *multi-condition* data improved the results compared to the system trained on *clean*.

	AMI		SRI		BabyTrain	
	EER	*minDCF*	EER	*minDCF*	EER	*minDCF*
Baseline ASV system						
clean	26.51	0.940	21.11	0.767	28.13	0.970
multi-condition	18.79	0.688	14.55	0.583	11.72	0.553

Table 5.4: Results of baseline ASV systems on real test conditions. For details on train and real test conditions, refer to Section 5.2.1 and Section 5.2.2.1 respectively.

5.3 Feature Enhancement with Unpaired Data

The main aim of this chapter was to improve the robustness of ASV system when tested on degraded speech (reverberant speech with background noise). Our approach to achieve this was to enhance the features of the evaluation data and use those enhanced features to extract x-vectors for scoring. We experimented with two enhancement networks: unsupervised enhancement network (UEN) [32] and supervised enhancement network (SEN), trained using *unpaired* (non-parallel) and *paired* (parallel) data, respectively. In this section, we present the training details of UEN and its use in improving the performance of ASV system trained on *clean* condition. Our experimental approach was as follows: we first demonstrated the usage of UEN in enhancing the features of degraded speech obtained via simulation. We used those enhanced features to improve the performance of ASV system trained on *clean* condition. We compared the result with 'baseline ASV *clean*' (discussed in Section 5.2).

We then used this UEN to improve the performance of system trained on *multi-condition* data (discussed in Section 5.6.1).

UEN was trained using the CycleGAN framework [16]. Unpaired training data was gathered from clean (*source*) and far-field (*target*) domains. CycleGAN framework consist of two generators: one maps features from *source* to *target* and the other maps features from *target* to *source*. The former generator is termed as UEN. Details on CycleGAN framework (loss functions and architectures used in training) are explained in Section 4.3.1.

The main advantage of training an enhancement network on unpaired data is that real data from both domains can be used for training, thus, avoiding the need for simulation. To validate this, we experimented with both simulated and real training data, and compared their results. We used VoxCelebCat *clean* (details in Section 5.2.1) as *source* domain for all the experiments in this chapter. For training the UEN with simulated data, we used VoxCelebCat *reverb_noise* (details in Section 5.2.1) as *target* domain data. We also experimented with VoxCelebCat *reverb*, which was simulated similar to VoxCelebCat *reverb_noise* except that *noise* from MUSAN was not added after reveberant speech was obtained. For experiments with using real data as target domain, we considered the domains AMI, SRI and *BabyTrain*. We used training data (different from evaluation data) from those individual domains to train the UEN (for SRI, since training data does not exist, we used training data from

Chime5 for training the UEN). For more details on training datasets, refer to Section 5.2.1.

The UEN trained on simulated data was evaluated on both simulated and real datasets. The UENs trained on real data from individual domains were only evaluated on those domains (for example, UEN trained on AMI was evaluated only on AMI). For more details on real and simulated test corpora used in chapter, refer to Section 5.2.2. Table 5.5 summarizes the training and test datasets used for individual UEN networks.

Train Datasets		**Test Datasets**					
Source	***Target***	**Simulated**		**Real**			
Clean	*Far-field*	SITW *noisy*	SITW *reverb*	SITW	AMI	SRI	BabyTrain
VoxCelebCat clean	VoxCelebCat *reverb*(sim)	✓	✓	✓	✓	✓	✓
	VoxCelebCat *reverb_noise*(sim)	✓	✓	✓	✓	✓	✓
	AMI train(real)	✗	✗	✗	✓	✗	✗
	Chime5 train(real)	✗	✗	✗	✗	✓	✗
	BabyTrain train(real)	✗	✗	✗	✗	✗	✓

Table 5.5: Overview of datasets used for training and testing unsupervised enhancement networks (UENs). We experimented with several *unpaired* training datasets. The *target* domain data for training the UEN was obtained either *via* simulation (denoted as sim in the table) or by sampling from *target* domain (denoted as real in the table). The *source* (clean) domain data for all the UENs remains the same. For description of training and testing datasets, refer to Section 5.2.1 and Section 5.2.2 respectively.

The training procedure was as follows: CycleGAN framework was trained on 40-dimensional log mel-FB features. Short-time mean centering and energy based VAD was applied on the features. Two batches of features were sampled from clean and degraded speech during each training step. Since, the training process was unsupervised both the mini batches were drawn in a completely

random fashion with no correspondence between the two batches. Batch size and sequence length were set to 32 and 127 respectively. The model was trained for 100 epochs. Each epoch was set to be complete when one random sample from each of the utterances of clean training corpus has appeared once in that epoch. Adam Optimizer was used with momentum $\beta_1 = 0.5$. The learning rates for the generators and discriminators were set to 0.0003 and 0.0001 respectively. The learning rates were kept constant for the first 15 epochs and, then, linearly decreased until they reach the minimum learning rate (1e-6). The cycle and adversarial loss weights were set to 2.5 and 1.0 respectively. More details on the objectives used for training CycleGAN can be found in Section 4.3.1 and in our work on domain adaptation [30–32].

5.3.1 Unsupervised Feature Enhancement (UEN) with Simulated Data

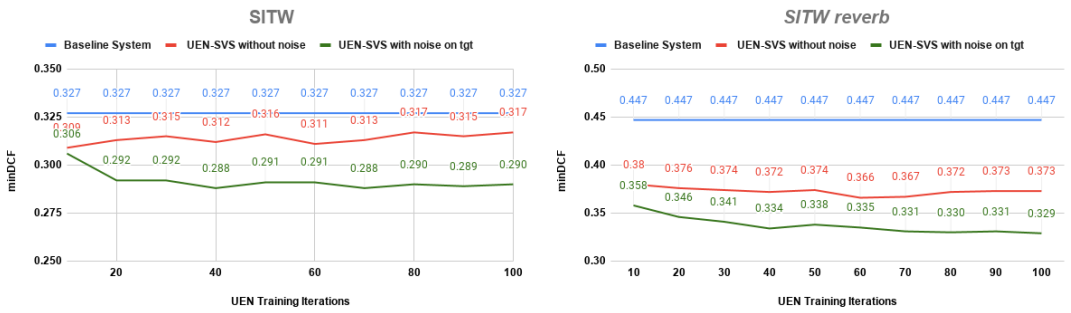
As explained earlier, our far-field enhancement procedure involved training a CycleGAN with *unpaired* data and then use UEN – the generator that maps features from far-field (*target*) domain to clean (*source*) domain – to enhance the features of evaluation data. In this section, we experimented with using simulated far-field data (obtained from clean data) for training the UEN. We experimented with two UENs, each differ from the *target* domain data used for

training. We train an UEN with VoxCelebCat *reverb* as *target* domain data and a second UEN with VoxCelebCat *reverb_noise* as target domain. The *source* domain data, VoxCelebCat *clean*, was the same for both the networks. The former was termed as UEN trained without noise whereas the later was termed as UEN trained with noise. The ASV system equipped with UEN trained with noise was termed as 'UEN ASV with noise'. Similarly, the ASV system whose test data was enhanced with UEN trained without noise was termed as 'UEN ASV w/o noise'.

Figure 5.3 shows comparison of speaker verification performance of 'UEN ASV with noise' and 'UEN ASV w/o noise' when evaluated on SITW and SITW *reverb*. We compared these systems with 'baseline ASV clean' - ASV system trained on VoxCelebCat *clean* and evaluated on SITW and SITW *reverb* (without enhancement). Figure 5.3a presents comparison of *minDCF* of both the systems on SITW as a function of number of iterations used for training the UEN. Figure 5.3b compares the results of both the UEN ASV systems on SITW *reverb*. The plot presents results averaged across all the SITW *reverb* test conditions created using the simulated and real RIRs. For details on SITW *reverb* refer to Section 5.2.2.2. For baseline results on SITW and SITW *reverb*, refer to Section 5.2.3.

Two main observations can be made from the plot:

1. Both the UEN ASV systems improve over 'baseline ASV *clean*' on both



(a) SITW **(b)** SITW *reverb*

Figure 5.3: Results of unsupervised enhancement on SITW and SITW *reverb*. The baseline system was trained on clean and evaluated without any enhancement. Enhancement network trained with VoxCelebCat *reverb* tends to overfit. Adding *noise* to *target* domain data helped prevent over-fitting.

test conditions. The enhancement approach, though intended to boost the performance of SITW *reverb*, also improved the performance of original SITW which was considered clean test corpora in our work [32].

2. The UEN trained without noise starts to overfit after iteration 10 (Figure 5.3a) but the UEN trained with noise continue to improve performance up to iteration 50 (upon which the performance flattens). *Adding noise to the target domain data, thus, acts like a regularizer [31, 32].*

The performance of 'UEN ASV *noise*' and 'baseline ASV *clean*' at each individual RT60 ranges are presented in Table 5.6. The advantage of our enhancement approach is clearly evident from the results (yielding 26% relative improvement in *minDCF* on average).

The results discussed so far in this section demonstrated excellent dereverberation capabilities of UEN on simulated far-field test conditions. However, in

	Baseline ASV *clean*		**UEN ASV *noise***	
	EER	DCF	EER	DCF
Sim. RIRs RT60 range				
0.0-0.5	5.84	0.405	5.47	0.320
0.5-1.0	6.95	0.504	5.80	0.356
1.0-1.5	6.34	0.459	5.33	0.330
1.5-4.0	6.03	0.415	5.21	0.311
Real RIRs	6.53	0.450	5.66	0.335
Avg	6.34	0.447	**5.49**	**0.330**

Table 5.6: Results of unsupervised enhancement using UEN on SITW *reverb*. The baseline system was trained on VoxCelebCat *clean* and evaluated on SITW *reverb*. Similar to baseline, enhancement system was trained on *clean* condition but SITW *reverb* was enhanced using UEN during evaluation. UEN in this table was trained on VoxCelebCat *reverb_noise* and VoxCelebCat *clean* as *target* and *source* domains respectively.

real world, speech often gets corrupted by several background noises. An ideal, enhancement system should be able to deal with both far-field speech (dereverberation task) and additive noise (denosing task). It should also generalize to unseen conditions seen during the training of enhancement network. To test the generalization ability of UEN, we evaluated the performance of both the UEN ASV systems on SITW *noisy*. As discussed in Section 5.2.2.2, SITW *noisy* consists of four different additive noise test conditions - SITW *noise*, SITW *music*, SITW *babble* and SITW *chime3bg*. Out of the four different testing conditions, only MUSAN *noise* was added to the training data of UEN. The remaining three conditions (MUSAN *speech*, MUSAN *music* and *chime3bg*) were not used during the training of UEN. Hence, these three conditions are consid-

SNR	**SITW *music***		**SITW *babble***		**SITW *noise***		**SITW *chime3bg***	
(dB)	EER	DCF	EER	DCF	EER	DCF	EER	DCF
baseline ASV *clean*								
17	5.14	0.345	5.36	0.349	5.14	0.351	5.1	0.341
12	5.55	0.368	5.96	0.386	5.54	0.381	5.44	0.380
7	6.36	0.424	7.6	0.497	6.35	0.440	6.48	0.473
2	8.65	0.563	12.94	0.764	8.00	0.544	9.81	0.693
-5	16.13	0.864	28.48	0.999	12.98	0.754	21.46	0.993
Avg.	8.37	0.513	12.07	0.599	7.60	0.494	9.66	0.576
UEN ASV *noise*								
17	4.89	0.290	5.04	0.296	4.85	0.292	4.90	0.286
12	5.03	0.301	5.66	0.324	5.02	0.306	5.00	0.298
7	5.54	0.329	7.32	0.429	5.35	0.329	5.44	0.330
2	6.71	0.396	12.22	0.656	6.23	0.385	6.84	0.413
-5	11.60	0.623	27.74	0.994	9.07	0.508	12.90	0.645
Avg.	**6.75**	**0.388**	**11.60**	**0.540**	**6.10**	**0.364**	**7.02**	**0.394**

Table 5.7: Unsupervised enhancement results on simulated additive noise conditions. For details on test conditions, refer to Section 5.2.2.2

ered as unseen conditions used to test the generalization ability of UEN. The results at each individual SNR are presented in Table 5.7. 'UEN ASV *noise*' yielded consistent improvements on all four noise conditions at all SNRs. More pronounced improvements were observed at 0dB and -5dB SNRs. Figure 5.4 presents the comparison of both the UEN ASV systems as a function of number of iterations used for training UEN. The *minDCF* reported in the figure was obtained by averaging the *minDCF*s across the individual SNR conditions. Once again the advantage of training UEN with *noise* (avoiding over fitting) can be observed from the plots.

The results and the figures showed that the UEN we devised exhibited good

(a) SITW *noise* **(b)** SITW *chime3bg*

(c) SITW *music* **(d)** SITW *babble*

Figure 5.4: Results on SITW *noisy*

dereverberation and denoising capabilities, and also good generalization ability to unseen noise conditions (SITW *music*, SITW *speech* and SITW *chime3bg*). Since, UEN trained with *noise* outperformed the UEN trained without *noise* on all testing conditions, we refer to 'UEN ASV *noise*' as 'UEN ASV' for the rest of the chapter and we report results only on UENs trained with noise.

5.3.1.1 Comparison with Weighted Prediction Error

In this section, we compare the performance of UEN in improving the performance of ASV with Weighted Prediction Error (WPE) [98, 99], a commonly used approach for speech dereverberation. We termed the ASV, whose evaluation data was enhanced with WPE as 'WPE ASV'. Table 5.8 presents the comparison results for 'UEN ASV with *noise*' and 'WPE ASV'. The training data for both the systems remains the same as 'baseline ASV'. Enhancement was only applied during evaluation. We obtained 21% and 22% relative improvements on *minDCF* of SITW *reverb* over 'baseline ASV' and 'WPE ASV'.

	SITW		**SITW reverb**	
	EER	minDCF	EER	minDCF
Baseline	**5.23**	0.340	6.78	0.460
WPE	5.69	0.370	6.48	0.466
UEN	5.68	**0.323**	**6.09**	**0.363**

Table 5.8: Comparison between UEN and WPE

5.3.2 Enhancement in Low-resource Setting

UEN used in unsupervised enhancement experiments discussed in Section 5.3.1 was trained on simulated far-field data obtained from VoxCelebCat *clean*. The simulated far-field data used was comparable in size to that of the clean data ($\tilde{1}$600 hours of speech from 7104 speakers). In this section, we tested the ability of UEN's performance when trained on limited amount of data. In this work, limited target domain data referred to data obtained from limited number of speakers. We trained an UEN on SITW *dev reverb_noise* as far-field data. Similar to VoxCelebCat *reverb_noise*, SITW *dev reverb_noise* was obtained by convolving RIRs to SITW *dev* and then MUSAN *noise* was added on top of it. For the low resource experiments in this section, we used real RIRs instead of simulated RIRs (details in Section 5.2.1). SITW *dev reverb_noise* and VoxCelebCat *reverb_noise* has 191 and 7104 number of speakers respectively. Since, the former data set has much less number of speakers compared to latter, we considered this a low resource scenario. We used SITW *eval* and VOiCES corpus [86, 100] for evaluation. Results are presented in Table 5.9. The low resource system was denoted as 'UEN ASV low resource' in the Table. From the results, it can be observed that both the UEN ASV systems improved the results compared to baseline ASV *clean* and WPE ASV on both the evaluation datasets: SITW and VOiCES *eval*. More importantly, the low resource system performed very similar to UEN ASV.

	SITW *reverb*		VOiCES *eval*	
	EER	**DCF**	**EER**	**DCF**
Baseline ASV	7.24	0.560	11.12	0.787
WPE ASV	6.52	0.520	9.36	0.687
UEN ASV	6.34	**0.492**	**8.95**	0.678
UEN ASV low resource	**6.12**	0.497	9.08	**0.673**

Table 5.9: Unsupervised enhancement results [31] trained in a low-resource setting. 'UEN ASV low resource' and 'UEN ASV' were trained using SITW *dev reverb_noise* and VoxCelebCat *reverb_noise* as target domain data respectively. The source domain data, VoxCelebCat *clean*, remains the same for both the networks.

5.3.3 Unsupervised Feature Enhancement with Real Data

UEN used in enhancement experiments discussed so far in Section 5.3.1 was trained on simulated far-field data obtained from VoxCelebCat *clean*. The UEN, thus trained on simulated data, was used to enhance several simulated test conditions. In this section, we used the UEN trained using simulated data to enhance evaluation data obtained from domains such as AMI, SRI and BabyTrain. In addition to training UEN on simulated data, we also experimented with training the UEN on real data obtained from target domains. Experiments in Section 5.3.2 demonstrated the ability of UEN trained on limited amount of data to improve the verification results compared to baseline system. However, the low resource system in Section 5.3.2 was also trained on simulated data (SITW *dev reverb_noise*). In this section, we experimented

with training the UEN on real degraded data obtained from target domain – the domain on which the system would be tested on. The term unsupervised in UEN refers to use of *unpaired* data to train the enhancement network. The ability to train the enhancement network on *unpaired* data motivated us to experiment with real data to train the UEN, thus avoiding the need for simulated data. We considered real data as degraded data obtained from real domains. The real data obtained from target domain used for training UEN was different from evaluation data. We termed ASV system enhanced with UEN trained on real data as 'UEN ASV real' whereas the system trained on simulated data (VoxCelebCat *reverb_noise*) was still termed as 'UEN ASV'.

We experimented with two domains in this section: AMI and SRI. We trained two UENs in this section: one for AMI and other for SRI. The target domain data for training the UEN for enhancing AMI evaluation data was sampled from training set of AMI [95]. However, the UEN for SRI was trained on Chime5 [94] as target domain data for lack of availability of training set for SRI corpus. AMI was recorded in a setting of 3 different meeting rooms, 180 speakers $\times$ 3.5 sessions per speaker. Out of these 180 speakers, 135 speakers were used for training the UEN and 45 for testing. Chime5 corpus was recorded in an indoor uncontrolled setting of kitchen, dining, living room with 80 speakers. Similar to simulated setup, we added *noise* from MUSAN to the recordings of AMI and Chime5. Addition of noise to reverberant speech fol-

lowed from our experiments in Section 5.3.1 where we showed that noise addition improves the performance of UEN by avoiding over fitting. The source domain data (VoxCelebCat *clean*) used for training remained the same for the real and simulated networks. The real target domain has much less speakers (135 from AMI) compared to simulated setup (7104). Enhancement experiments performed using real data can be seen as a special case of low resource experiments since the training data available from the target domain was limited in number of speakers.

Experimental results are presented in Table 5.10. As shown in results, both the enhancement systems –'UEN ASV' and 'UEN ASV *real*'– improved in performance compared to the baseline for both the test domains. For AMI, 'UEN ASV *real*' performed better than 'UEN ASV' even though UEN *real* was trained on smaller amount of target domain data (obtained from 135 speakers) compared to the UEN trained on simulated data (7104 speakers). For SRI, unlike AMI, 'UEN ASV' performed better than 'UEN ASV *real*'. The difference in domains between SRI (testset) and Chime5 (training set) used might have resulted in slightly poor performance of the real system compared to its simulated counterpart. From these experiments we observed that when training and evaluation conditions matched closely (like in AMI) use of real data over simulated data offered advantage, which justifies our approach for training enhancement network on *unpaired* data.

	AMI		SRI	
	EER	minDCF	EER	minDCF
Baseline ASV	26.51	0.940	21.11	0.767
UEN ASV	20.22	0.766	**18.63**	**0.714**
UEN ASV *real*	**19.66**	**0.726**	19.92	0.732

Table 5.10: Comparison of UEN trained on real data vs UEN trained on simulated data on AMI and SRI

5.4 Feature Enhancement Using Paired Data

UEN [31,32], discussed in Section 5.3, was trained on *unpaired* reverb-clean data to minimize a multi-task objective – a combination of cycle-consistency and adversarial losses. UEN transforms features from reverberant to clean domain. As discussed in Section 5.3.1, UEN has shown good dereverberation and denoising capabilities by improving performance on simulated reverberant, simulated noisy, and real datasets collected in wild/uncontrolled environments. UEN also obtained better verification performance compared to the widely used WPE based speech dereverberation approach [98, 99] (details in Section 5.3.1.1). Additionally, since UEN does not require *paired* data for training, it can be trained on real data from reverberant (target) and clean (source) domains[5], thus avoiding the need for using simulated data (details in Section 5.3.3).

[5]We used the terms clean/source and reverberant/target interchangeably in this paper

In this section, we compare the performance of UEN with an enhancement network trained using *paired* reverberant-clean corpora – termed as supervised enhancement network (SEN). One of the standard approaches for training enhancement networks with *paired* data is to learn a mapping function from acoustic features of degraded speech to clean speech using a DNN [23]. This feature mapping approach solves a *non-linear regression* problem by minimizing distance metrics like L_1 or L_2 between the output and reference clean features. This is a supervised approach, since the DNN is trained on *paired* clean-degraded speech usually obtained by simulation. The L_2 objective trains a regression network that outputs the mean of all plausible outputs, which is known to produce smooth and/or distorted features. [2, 29]. This issue is well-noted in enhancement community [101] and also observed in image super resolution task in computer vision [29]. In this work, we explored the usage of adversarial loss [28] to overcome the distortions introduced by the feature mapping approach.

5.4.1 Supervised Enhancement Network (SEN)

SEN was trained using *paired* reverberant-clean corpora to minimize a combination of feature mapping objective and adversarial loss. We chose L_1 metric for the mapping objective, and is given in Equation 5.2.

$$L_1(\mathrm{SEN}, \mathbf{X}_{\mathrm{reverb}}, \mathbf{X}_{\mathrm{clean}}) = \mathbb{E}_{(\mathbf{x}_{\mathrm{reverb}}, \mathbf{x}_{\mathrm{clean}}) \sim (p_{\mathrm{reverb}}, p_{\mathrm{clean}})} ||\mathrm{SEN}(\mathbf{x}_{\mathrm{reverb}}) - \mathbf{x}_{\mathrm{clean}}||_1 \quad (5.2)$$

L_1 objective usually distorts the output by making it smooth [2,29,101]. The additional adversarial loss [28] avoids this. This was accomplished by training a discriminator – a binary classifier that discriminates between the enhanced and original clean features. SEN was then trained to trick the discriminator in believing that the output features were sampled from the original clean feature distribution instead of the enhanced feature distribution. At the end of the training, the enhanced and original clean features become indistinguishable by the discriminator, making the enhanced features more realistic, thus avoiding distortion. We used *least-squares* objective [77] to train the discriminator as given in Equation 5.3,

$$\begin{aligned} L_{\mathrm{Disc}}(\mathrm{SEN}, D_{\mathrm{clean}}, \mathbf{X}_{\mathrm{reverb}}, \mathbf{X}_{\mathrm{clean}}) = \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{clean}}}[(D_{\mathrm{clean}}(\mathbf{x}) - 1)^2] \\ + \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{reverb}}}[(D_{\mathrm{clean}}(\mathrm{SEN}(\mathbf{x})))^2] \end{aligned} \quad (5.3)$$

The adversarial objective for the SEN is

$$L_{\text{adv}}(\text{SEN}, D_{\text{clean}}, \mathbf{X}_{\text{reverb}}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{reverb}}}[(D_{\text{clean}}(\text{SEN}(\mathbf{x})) - 1)^2] \tag{5.4}$$

The final multi-task objective for training the SEN is given by

$$\begin{aligned} L(\text{SEN}, D_{\text{clean}}) = \lambda_1 L_1(\text{SEN}, \mathbf{X}_{\text{reverb}}, \mathbf{X}_{\text{clean}}) \\ + \lambda_{\text{adv}} L_{\text{adv}}(\text{SEN}, D_{\text{clean}}, \mathbf{X}_{\text{reverb}}) \ , \end{aligned} \tag{5.5}$$

where λ_1 and λ_{adv} represent the weights assigned to individual objectives.

5.4.2 Training Details of Supervised Enhancement Network

The training of SEN required *paired* reverb-clean data which was obtained by simulation. Similar to UEN, we used VoxCelebCat *reverb_noise* and VoxCelebCat *clean* as reverb and clean data respectively. As explained in Section 5.3, we obtained the former data set by convolving the later with simulated RIRs and then added noise files from MUSAN *noise* corpus. During training, mini batches of *paired* reverb-clean data were drawn from training corpora which was then used to minimize the objectives given in Equation 5.5. We compared the performance of SEN with that of UEN. To make an ideal

comparison between both the networks, similar to UEN, the learning rates used for SEN and discriminator were set to 0.003 and 0.001 respectively. The learning rates were kept constant for the first 15 epochs and, then, linearly decreased until they reach the minimum learning rate (1e-6). Mini batch size in SEN training was set to 32, similar to UEN. SEN was trained for 50 epochs. Each epoch was set to be complete when one random sample from each of the utterances of clean training corpus has appeared once in that epoch. Adam Optimizer was used with momentum $\beta_1 = 0.5$. The architectures for SEN and discriminator used were exactly the same as that of UEN and discriminator used in CycleGAN. Once the SEN was trained, we used it to enhance the features of the evaluation data which were then used to compute the x-vectors used for scoring the ASV system.

5.4.3 Paired vs Unpaired Enhancement Approaches

The SEN network was trained using a multi-task objective – a weighted sum of L_1 and adversarial loss (details in Section 5.4.1). To demonstrate the effectiveness of adversarial loss for training the SEN, we first performed an ablation study on the losses by training two SENs on individual losses. Results on simulated test set SITW *reverb* are presented in Table 5.11.

Baseline system was trained solely on clean and tested without enhancement. SEN trained solely with adversarial loss in Equation 5.4, learns its own

ASV	**Loss Weights**		**SITW**		**SITW *reverb***	
system	λ_{FM}	λ_{adv}	EER	minDCF	EER	minDCF
Baseline	-	-	5.02	0.327	6.34	0.448
UEN	-	-	4.77	0.297	5.63	0.345
SEN(L_1)	1.0	0.0	6.39	0.462	8.87	0.626
SEN(L_{adv})	0.0	1.0	8.28	0.442	9.91	0.535
SEN(both)	1.0	1.0	4.19	0.275	5.06	0.317
SEN(both)	1.0	0.1	**4.01**	**0.260**	**4.63**	**0.299**
SEN(both)	1.0	0.01	6.28	0.462	8.72	0.612

Table 5.11: Comparison of SEN vs UEN on ASV evaluated on SITW and SITW *reverb*. x-vector network was trained on VoxCelebCat *clean* without data augmentation and enhancement was applied during evaluation. UEN and SEN stand for unsupervised (*unpaired*) and supervised (*paired*) enhancement networks respectively.

loss function to generate clean features that matches closely with the reference clean features. The ASV system, whose evaluation data was enhanced using SEN trained with L_1 loss only, did not improve over baseline system. Similarly, the ASV system whose evaluation data was enhanced using SEN trained with adversarial loss alone also did not improve over the baseline. The above results suggest that either of the loss functions alone are not enough to achieve enhancement task. Moreover, SEN with adversarial loss yielded better *minDCF* (0.535) compared to SEN with L_1 (0.626) on SITW *reverb*, which justified the usage of adversarial loss. We then trained a SEN using a combination of both the losses giving them equal weights (1.0). As shown in Table 5.11, use of this SEN to enhance SITW *reverb* improved the performance compared to both baseline and UEN ASV system. Moreover, to demonstrate the effectiveness of adversarial loss in eliminating the distortion in output features, we used

the following analysis. Since, SITW *reverb* was obtained *via* simulation from SITW, a distortion free enhancement of SITW *reverb* should be able to retrieve SITW. In other words, enhanced SITW *reverb* and original SITW, when tested on same baseline should give identical performance. When the above happens, it is safe to conclude that enhancement process is distortion free. The results are consistent with the statement above. SITW *reverb* enhanced using SEN (trained with both objectives with equal loss weights) yielded 5.06 and 0.317 in terms of EER and *minDCF* respectively. SITW yielded similar performance on the baseline system (5.02 and 0.327 in terms of EER and *minDCF* respectively). The results suggest that training enhancement network on adversarial loss along with conventional feature mapping objective (like L_1 loss) would be able to eliminate any distortions present in enhancement.

We further experimented with tuning the adversarial loss weight λ_{adv}. We experimented with 1.0, 0.1 and 0.01. Setting adversarial loss weight to 0.01 made it insignificant compared to L_1 loss (SEN with λ_{adv} set to 0.01 had slight improvements over SEN trained with L_1 alone). We obtained better results with the adversarial loss weight of 0.1, as shown in Table 5.11. SEN trained with λ_{adv} set to 0.1 yielded 20.5% and 33.5% percent relative improvements in terms of *minDCF* on SITW and SITW *reverb* compared to the baseline system. For comparison, UEN ASV yielded 9.1% and 23% relative improvements. Similar to UEN, SEN did not deteriorate performance of SITW (considered *clean*)

but improved it. For the rest of this work, use of the term SEN refers to SEN trained with 0.1 and 1.0 for λ_{adv} and λ_1 respectively.

5.5 Comparison of Feature Enhancement and Feature Adaptation

Both UEN and SEN, described above, transformed reverberant features to clean domain. Hence, we call the transformation *reverberant feature enhancement* or *feature dereverberation*. As shown in Section 5.3.1, the enhancement networks trained for doing feature dereverberation also generalizes to unseen *noisy* conditions. Thus, we use the term degraded features to denote both reverberant and/or *noisy* conditions. The ASV system experimented so far in this chapter was trained on VocCelebCat *clean*, considered as *clean* condition, and tested on degraded conditions. Both the enhancement networks, UEN and SEN, were used to improve the performance of this system by enhancing the features of the evaluation data. The enhanced system in this case can be considered as trained and tested on *clean* condition whereas the baseline system can be considered as trained on *clean* and tested on degraded conditions. In this setting, both the enhancement networks – UEN and SEN – yielded impressive results across a wide range of simulated and real testing conditions, as demonstrated in Section 5.3 and Section 5.4.

	SITW		**SITW *reverb***	
	EER	**DCF**	**EER**	**DCF**
Baseline ASV system				
trained on *clean*	5.02	0.327	6.29	0.446
trained on *noise*	**4.07**	**0.273**	**4.98**	**0.347**

Table 5.12: Comparison of baseline ASV systems trained on two different conditions: VoxCelebCat *clean* and VoxCelebCat *noise*. No enhancement/adaptation was done during evaluation

In this section, we experimented with mapping the degraded features to some chosen domain other than *clean* via a domain adaptation network (DAN). The choice of domain to which the degraded features would be mapped was based on the following criteria: we train two ASV systems, one trained on *clean* (as earlier) and the other trained on some domain we chose (details below) other than clean. We observed that ASV trained on the chosen domain yielded better results on unseen conditions compared to ASV trained on *clean*. Table 5.12 presents results of two different baseline ASV systems – one trained on VoxCelebCat *clean* and other trained on VoxCelebCat *noise* (chosen domain). VoxCelebCat *noise* was obtained by artificially adding assorted noise files from MUSAN *noise* corpora to VoxCelebCat *clean*. The noise addition was done as *foreground* noise [48] at randomly chosen SNR levels from 15,10,5 and 0 dB. For the experiments in Table 5.12, both the systems were tested on original evaluation corpora (enhancement was not applied during evaluation).

Table 5.12 suggests that training x-vector system on *noise* yielded better results on both SITW and SITW *reverb*. The former test set can be considered as

an unseen condition for both the baselines (far-field data is not seen in training for both the systems). Training on VoxCelebCat *noise* improved performance on SITW *reverb* (an unseen condition) compared to baseline trained on VoxCelebCat *clean*. Moreover, it also improved the performance on SITW, which was considered clean testing condition. We observed that x-vector network of ASV trained on *noise* has higher validation accuracy and a smaller difference between train and validation accuracy (avoiding over fitting) compared to the x-vector trained on *clean*. Training the ASV on *noise*, thus, acted as a regularizer in training and yielded better performance on both *clean* and *reverb* (unseen) testing conditions. Since, training on *noise* yielded better performance than training on *clean* we experimented with mapping the far-field evaluation corpora to *noise* domain *via* a domain adaptation network (DAN).

5.5.1 Domain Adaptation Network (DAN)

As explained above, DAN maps the degraded features from reverberant to *noise* domain. This mapping was attained by training a CycleGAN, similar to the one trained for UEN, except that the source domain was VoxCelebCat *noise* instead of VoxCelebCat *clean*. The target domain data used, VoxCelebCat *reverb_noise*, was same for both the networks. During evaluation, the reverberant features were transferred to the source domain using the corresponding generator in the CycleGAN, termed as DAN. Except for the difference in train-

ing data used, the procedure for training the DAN was identical to UEN.

5.5.2 Results: UEN vs DAN

Table 5.13 presents comparison between UEN and DAN on two baseline systems: one trained on *clean* and other trained on *noise* condition. UEN ASV and DAN ASV refer to the systems where the evaluation features were mapped using UEN and DAN respectively. For the ASV system trained on *clean* condition, UEN yielded better results compared to DAN. On the other hand, for the ASV system trained on *noise* condition, DAN yielded better results compared to UEN. Results suggested that mapping features of the evaluation data to the domain on which the x-vector was trained yielded better results. The experiments demonstrated the ability of CycleGAN to map features to an arbitrary domain (*noise* in this case). It can also be observed that enhancement deteriorated performance of ASV system that was trained on *noise*. Hence, DAN becomes a strong candidate to improve the performance of ASV trained on *multi-condition* data (discussed in next section).

	ASV trained on *clean*				**ASV trained on *noise***			
	SITW		**SITW *reverb***		**SITW**		**SITW *reverb***	
	EER	**DCF**	**EER**	**DCF**	**EER**	**DCF**	**EER**	**DCF**
Baseline ASV	5.02	0.327	6.29	0.446	**4.07**	0.273	4.98	0.347
UEN ASV	**4.74**	**0.290**	**5.45**	**0.329**	4.39	0.277	5.12	0.315
DAN ASV	4.76	0.308	5.73	0.376	4.39	**0.264**	**4.81**	**0.309**

Table 5.13: Comparison of UEN vs DAN. We present results on two different ASV systems: one trained on *clean* and other trained on *noise* condition.

5.6 Enhancement Experiments on System Trained with Data Augmentation

Encouraged by the improvements obtained by enhancement networks on an ASV system trained solely on *clean* condition, we experimented with a more practical scenario - ASV trained on *multi-condition* data. As explained in Section 5.1, *multi-condition* training data comprised of *noisy* and *far-field* conditions along with the original *clean* condition. The *noisy* and *far-field* conditions were obtained from *clean via* simulation. Using the *multi-condition* data for training, a method popularly known as data augmentation, yielded SOTA results on various test scenarios [14,15]. As explained in Section 5.1, training the ASV on multi-condition data can also be seen as a supervised domain adaptation strategy to increase the robustness of the system to degraded conditions.

With the following experiments in this section we investigated whether our unsupervised domain approach, we discussed in previous sections, would complement the supervised *multi-condition* approach.

5.6.1 Enhancement for x-vector vs. Enhancement for PLDA

In this experiment, we first experimented with UEN to boost the performance of *multi-condition* system. Results on two domains – AMI and SRI – are presented in Table 5.14. The first row presents results of *multi-condition* system. Second row presents the results when only test (*eval* and enrollment) data was enhanced using UEN. On both AMI and SRI, enhancing only test data deteriorated performance. We then trained the PLDA on xvectors extracted using enhanced *multi-condition* training data instead of original *multi-condition* data (results in row 3). Similar to the baseline, x-vector network, in this case, was still based on *multi-condition* data. On SRI, this improved the results slightly compared to the baseline system (better EER and similar *minDCF*). On AMI, this training strategy still deteriorated the performance compared to baseline. We then trained a homogeneous system – where x-vector and PLDA were trained on enhanced *multi-condition* data instead of original *multi-condition* data. This system gave better performance in terms

Details of data used			**AMI**		**SRI**	
x-vector	PLDA	test	EER	minDCF	EER	minDCF
aug	*aug*	*orig*	**18.79**	**0.688**	14.55	0.583
aug	*aug*	*enh*	18.96	0.711	15.43	0.644
aug	*enh*	*enh*	19.41	0.690	**14.26**	0.583
enh	*enh*	*enh*	18.81	0.690	14.78	**0.559**

Table 5.14: UEN ASV results on AMI and SRI with x-vector and PLDA augmentation. *aug*, *enh* and *orig* in the table stands for augmented data, enhanced data and original data with no enhancement respectively

of minDCF (4.2% relative) on SRI compared to baseline while giving almost similar performance as the baseline on AMI. The results indicated that ASV system trained using data augmentation (*multi-condition* data) can take advantage of enhancement. To benefit from enhancement the system needed to be trained in an homogeneous style – both x-vector and PLDA were trained on enhanced *multi-condition* data and evaluation data was also enhanced [32].

The homogeneous system discussed above yielded slightly better performance on SRI compared to the baseline. However, we observed a larger difference in training and validation accuracies of x-vector networks trained on enhanced *multi-condition* data and original *multi-condition* data. This indicated that the homogeneous system, though gave a slight improvement in performance, exhibited a tendency to over-fit to the training data. In the following experiments in Section 5.6.2, we explored several training strategies to over come the over-fitting scenario to improve the performance further.

The analysis in Table 5.14 was done using UEN. In the following experiments in Section 5.6.2, we also tested the efficacy of SEN and DAN, along

with UEN, in improving the verification performance when *multi-condition* data was used for training the ASV system.

5.6.2 Comparison of All Adaptation Approaches

Along with the homogeneous training strategy we discussed in previous section, we experimented with two different training schemes – both differ in data used for training x-vector and PLDA. In the homogeneous scheme, the entire *multi-condition* training data was enhanced [32]. We termed the ASV system trained in homogeneous fashion as ASV *enh*. In the second scheme, only the far-field component (VoxCelebCat *reverb_noise*) of *multi-condition* training data was enhanced. VoxCelebCat *clean* and VoxCelebCat *additive* were kept unchanged. The motivation for enhancing only the far-field training data was two fold: (1) all the enhancement networks in this work were trained to do dereverberation task. Hence, we used them to enhance only far-field portion of training data, and (2) leaving the additive noise training data unchanged would let us train the x-vector on noisy data, which could help prevent the over-fitting scenario observed in homogeneous system. We termed the system trained using this training strategy as ASV *ff-enh*. In the third scheme, we trained the x-vector on both original *multi-condition* training data and also its enhanced version. We termed this system as ASV *aug & enh*. Since the x-vector network in this case had double the training data compared to the first two scenarios,

it was trained only for 1.5 epochs compared to 3 epochs in other cases, thus making the systems comparable. In all schemes, the PLDA was trained on x-vectors extracted from enhanced features only (making the PLDAs comparable too). In all three cases, the evaluation data was enhanced. All the three training schemes were repeated for UEN, DAN and SEN separately. This resulted in total 9 different systems for comparison – three different training schemes repeated for all three enhancement networks. Results on three different real testing conditions – AMI, SRI and BabyTrain – are presented in Table 5.15.

In Section 5.5, we observed that when experimented with two conditions – *clean* and *noise*, matched condition training yielded better performance. In other words, when ASV system was trained on *clean* condition, enhancement yielded better results compared to DAN. On the other hand, when ASV system was trained on *noise* condition, domain adaptation (using DAN) yielded better results compared to enhancement. In this section, we trained the ASV system on *multi-condition* data, as opposed to training on a single condition in the previous sections. Hence, we devised the above experiments to determine whether mapping the features to *clean* domain (enhancement) or to *noise* domain (adaptation) yields better performance in a *multi-condition* training scenario. The goal to experiment with all the 9 systems and to test on several domains was not to obtain best performance on individual domains, but to determine a training strategy that consistently improves performance across all

Enh. Type	**ASV**	**AMI**		**SRI**		**BabyTrain**	
		EER	minDCF	EER	minDCF	EER	minDCF
Baseline	*multi-condition*	18.79	0.688	14.55	0.583	11.72	0.551
UEN	ASV *enh*	18.84(↓)	0.688(↑)	15.29(↓)	0.580(↑)	9.88(↑)	0.391(↑)
	ASV *ff-enh*	19.48(↓)	0.689(↓)	14.69(↓)	0.585(↓)	11.45(↑)	0.474(↑)
	ASV *aug & enh*	18.98(↓)	0.682(↑)	14.02(↑)	0.559(↑)	13.14(↓)	0.441(↑)
SEN	ASV *enh*	19.97(↓)	0.697(↓)	15.16(↓)	0.563(↑)	9.61(↑)	0.428(↑)
	ASV *ff-enh*	18.89(↓)	0.692(↓)	14.50(↑)	0.523(↑)	8.70(↑)	0.402(↑)
	ASV *aug & enh*	20.03(↓)	0.706(↓)	13.13(↑)	0.533(↑)	10.09(↑)	0.405(↑)
DAN	ASV *enh*	18.78(↑)	0.713(↓)	15.18(↓)	0.584(↓)	12.32(↓)	0.488(↑)
	ASV *ff-enh*	19.38(↓)	0.704(↓)	14.40(↑)	0.578(↑)	11.07(↑)	0.523(↑)
	ASV *aug & enh*	**18.54**(↑)	**0.673**(↑)	**14.10**(↑)	**0.550**(↑)	**8.71**(↑)	**0.377**(↑)

Table 5.15: Results on ASV system trained on *multi-condition* data (↑ and ↓ indicate that the enhancement system's performance *improved* or *deteriorated* respectively compared to the *multi-condition* baseline. Results in bold indicate system has improved performance across all test conditions. For description of terms ASV *enh*, ASV *ff-enh* and ASV *aug & enh* refer to Section 5.6.2)

test conditions compared to the baseline system.

The results from Table 5.15 suggests that 'DAN ASV *aug & enh*' yielded improvements on all three domains – AMI, SRI and BabyTrain – on both the evaluation metrics (EER and *minDCF*). This system yielded relative improvements on *minDCF* of 2.2%, 6% and 31.6% on AMI, SRI and BabyTrain respectively compared to *multi-condition* baseline. On SRI and BabyTrain, both 'SEN ASV *ff-enh*' and 'SEN ASV *aug & enh*' yielded improvements compared to baseline but deteriorate the performance on AMI. However, 'DAN ASV *aug & enh*' yielded improvements on all three domains. Hence, we conclude that adapting the features to *noise* domain and training the ASV on both *multi-condition* data and enhanced data yields good improvements over the baseline in *multi-condition* scenario.

5.7 Summary

In this chapter, we experimented with single channel far-field feature adaptation for improving the performance of speaker verification systems when tested on uncontrolled 'wild' conditions. To enhance the quality of features during evaluation, we experimented with two enhancement networks – unsupervised enhancement network (UEN) and supervised enhancement network (SEN). The former was trained on *unpaired* data from both training and evaluation domains, while the later was trained on *paired* data, usually obtained by simulation. In the initial phase of our experiments, we experimented with an ASV system trained solely on clean speech. Since, in this stage enhancement is only applied during evaluation, we used this system to tune the enhancement networks and test their efficacy. *Unpaired* enhancement approach can be used to train on real data from *target* domain, thus, the need for simulation can be avoided. We experimented with *paired* enhancement approach to benchmark the *unpaired* approach. We presented a simple regularization technique to prevent the enhancement networks from over-fitting. We presented results of our approach on several simulated and real degraded conditions [32] (details in Section 5.3 and Section 5.4).

We also observed empirically that conventional non-linear regression based enhancement approach using loss functions like L_1 would not improve SV per-

formance. We attribute this performance drop to the distortions in output feature space. We found that adversarial loss circumvents this distortions by aligning the enhanced features close to real clean features (details in Section 5.4).

We also experimented with transferring the evaluation features to an arbitrary domain (instead of clean). To achieve this, we trained a DNN to map features from degraded domain to an assorted *noise* domain – network termed as domain adaptation network (DAN). We observed that mapping evaluation features to the domain on which the network was trained improves verification performance (details in Section 5.5).

In the last stage of our experiments, we tested our feature mapping approach on improving the performance of ASV system trained on multi-condition data, the approach used in SOTA SV systems. In this setting, we found that enhancing the evaluation data alone would not suffice to improve the performance. Training PLDA on embeddings extracted from enhanced features improved the performance. Further gains are achieved by training the entire pipeline on both enhanced and multi-condition features. In the multi-condition scenario, enhancement during training and evaluation gave considerable improvements. However, adapting features to *noise* domain (using DAN) gave consistent improvements on all the domains we experimented with (details in Section 5.6).

Chapter 6

Summary and Future Work

In this chapter, we summarize the observations from our experiments and discuss some potential approaches for extending our work.

6.1 Summary

6.1.1 Robustness to Mismatch in Sampling Frequency

In Chapter 3, the experimental setting was as follows: we considered two types of datasets for training – telephone speech sampled at narrowband (NB) 8 kHz frequency, and microphone speech sampled at wideband (WB) 16 kHz frequency. We assumed the evaluation dataset was WB. We further assumed

that WB speech available for training was limited in number of speakers (and utterances) compared to NB speech. The conventional way of training systems under this scenario is to downsample WB speech to match the frequency of NB speech during training and evaluation. This procedure throws away information in the upperband (UB) of microphone speech. In this chapter, we investigated several procedures to combine both the datasets during training without downsampling the WB speech.

We first experimented with a linear upsampler that upsamples telephone speech to match the sampling frequency of microphone speech without predicting any information in the UB. The upsampled telephone speech was used to train the x-vector network. We termed this ASV system as mixed BW sytem [34]. This upsampling technique, though simple and computationally inexpensive, proved very effective, yielding impressive results (details in Section 3.3). The main advantage of mixed BW system was that, we were able to use original microphone speech during training and evaluation without any downsampling as done in conventional way of training. We, thus, were able to retain information in the UB of WB speech which helped improve the verification performance.

We then continued our investigation by experimenting with several neural network architectures to upsample NB speech before training x-vector networks. The motivation was to overcome the limitation of basic upsampler in

predicting the information present in the upper band of telephone speech. We experimented with a feed forward fully connected DNN, a deep residual full-convolutional network (CNN) and a BLSTM network for BWE [35]. Individual x-vector based speaker recognition systems were trained on bandwidth extended features obtained from each of these three systems. All the BWE speaker recognition systems improved in performance compared to conventional training and the mixed BW system. The best performer in terms of DCF was the CNN-BWE system trained on sequence lengths of 257. In terms of DCF, the CNN-BWE system showed relative improvement of 10.78% and 15.96% in the SITW eval *Core* and *Assist-Multi* condition respectively w.r.t. the conventionally trained NB baseline; and improved by 3.21% and 4.13% w.r.t. to the mixed BW baseline. In terms of number of parameters, the CNN-BWE model with 21 resnets is the most light weight with $\sim 6M$ parameters compared to DNN-BWE with $\sim 16M$ parameters and BLSTM-BWE with $\sim 18M$ parameters (details in Section 3.4).

6.1.1.1 Contributions

Our main contributions with this approach are two fold: 1) we designed an experimental framework – termed mixed BW approach – to train a x-vector embedding network on original WB speech and upsampled telephone speech. This approach performed better than the conventional way of training ASV

systems on both telephone and microphone data – by downsampling the WB speech. 2) We also proposed a CNN with an encoder-decoder architecture to extend the band-width of telephone speech. The CNN architecture performed better in improving SV performance compared to a previously proposed fully connected DNN [22].

6.1.2 Robustness to Channel Mismatch Scenario

In Chapter 4, we experimented with a channel mismatch scenario where the ASV system was trained on data acquired using *telephone* channel and evaluated on *microphone* data. To address this, we presented a feature mapping based unsupervised channel adaptation technique, where we map the features of *microphone* domain to *telephone* domain. Our feature mapping network is a 'deep residual convolutional network' trained on *unpaired* data acquired from both the domains using the CycleGAN framework. Our approach yielded 10.1% and 4.5% relative improvements on EER and *minDCF* on SITW *core* condition, when microphone features were mapped to telephone domain during evaluation. We also experimented with mapping telephone features to microphone domain, and use the mapped features for training. This approach yielded slightly better results compared to the previous approach [30] (details in Section 4.3).

We also experimented with training the feature mapping network in a low-resource scenario, where limited *target* domain data was available for training.

We observed that the CNN tended to overfit in this scenario. To circumvent this, we presented a simple regularization technique, using which the CNN trained on limited data performed almost similar to the CNN trained on much larger data [31] (details in Section 4.4).

The main advantage of our approach is to make use of *unpaired* real data from both domains to learn a domain adaptation mechanism, thus, avoiding the need for using simulation. In the next chapter, we extend this approach to increase the robustness of SV systems to far-field testing scenarios.

6.1.2.1 Contributions

We proposed an experimental framework to train a CNN on unpaired data acquired from two different domains – telephone and microphone. Our training procedure is inspired by the CycleGAN framework proposed for unpaired image-to-image translation. Our work was the first to use this framework for microphone-telephone channel adaptation of ASV systems. Our work also proposed a regularization scheme to avoid over fitting of the CycleGAN framework when trained on limited amount of data.

6.1.3 Robustness to Far-field and Noisy Data

In Chapter 5, we experimented with single channel far-field feature adaptation for improving the performance of speaker verification systems when

tested on uncontrolled 'wild' conditions. To enhance the quality of features during evaluation, we experimented with two enhancement networks – unsupervised enhancement network (UEN) and supervised enhancement network (SEN). The former was trained on *unpaired* data from both training and evaluation domains, while the later was trained on *paired* data, usually obtained by simulation. In the initial phase of our experiments, we experimented with an ASV system trained solely on clean speech. In this stage enhancement is only applied during evaluation. We used this system to tune the enhancement networks and test their efficacy. *Unpaired* enhancement approach can be used to train on real data from both *source* and *target* domains, thus, the need for simulation can be avoided. On the other hand, *paired* enhancement approach is trained on simulated data, using a combination of MSE and adversarial losses (details in Section 5.4). We presented a simple regularization technique to prevent the enhancement networks from over-fitting. We presented results of our approach on several simulated and real degraded conditions [32] (details in Section 5.3 and Section 5.4).

We also experimented with transferring the evaluation features to an arbitrary domain (instead of clean). To achieve this, we trained a DNN to map features from degraded domain to an assorted *noise* domain – network termed as domain adaptation network (DAN). We observed that mapping evaluation features to the domain on which the network was trained improves verification

performance (details in Section 5.5).

In the last stage of our experiments, we tested our feature mapping approach on improving the performance of ASV system trained on multi-condition data, the approach used in SOTA SV systems. In this setting, we found that enhancing the evaluation data alone would not suffice to improve the performance. Training PLDA on embeddings extracted from enhanced features improved the performance. Further gains are achieved by training the entire pipeline on both enhanced and multi-condition features. In the multi-condition scenario, enhancement during training and evaluation gave considerable improvements. However, adapting features to *noise* domain (using DAN) gave consistent improvements on all the domains we experimented with (details in Section 5.6).

6.1.3.1 Contributions

Our work was the first to show the effectiveness of the CycleGAN framework for enhancement of far-field speech, and its application to SV. Unlike previous approaches [25–27], which showed the benefits of enhancement network on simulated datasets, our work showed improvements on several simulated (noisy and far-field) datasets and real datasets acquired from wild/uncontrolled conditions.

6.2 Future Work

As discussed above, our feature mapping approach demonstrated strong results on various domain mismatch scenarios. In this section, we discuss some potential extensions for our approach.

6.2.1 Multi-Domain Adaptation

Feature mapping networks used in this work facilitate mapping from one domain to a different domain. In order to adapt an ASV system to multiple domains, our proposed framework requires us to train a feature mapping DNN separately for each of this individual domains. This approach not only increases the computational complexity of training, it also limits us from using the shared knowledge between domains. An ideal setup would be to train a single feature mapping DNN to map features between multiple domains. To facilitate this, the DNN needs to be conditioned on some auxiliary information of the domain to which features would be mapped.

For instance, one single DNN can be trained to adapt features from AMI, BabyTrain and SRI domains (details in Section 5.3.3) to clean domain. The network can be conditioned on one-hot vector for each individual domains. One other instance would be speech denoising scenario, where babble, music and

clean[1] can be treated as individual noise types. In this case, the auxiliary information could be noise type and SNR level. The noise type information can be encoded in the form of one-hot vector. Conditional *GAN* [58, 102] facilitate such conditioning. Hence, become a strong candidate for exploration.

6.2.2 Domain Specific Data Augmentation

As is evident from multiple discussions in this thesis, data augmentation using simulation helped make x-vector embeddings more robust, and thus, achieve SOTA results on various test sets (domains). In this work, we trained a feature mapping DNN for an enhancement task – DNN maps features from degraded (*target*) domain to clean (*source*) domain and use them for evaluation. We trained the DNN on *unpaired* data from both domains using adversarial and cycle-consistency loss. The training procedure with cycle-consistency involved using an auxiliary network that maps enhanced features back to the domain of interest. This bi-directional feature mapping mechanism opens up opportunities for one more application – domain specific data augmentation.

In this scenario, during training, we would map the original training features from *source* domain to *target* domain (the domain on which model would be evaluated), and use those features along with the simulated training data during training. Using the domain specific features in training, along with the

[1]Clean signals can be considered as an additional noise type with infinite SNR

simulated features, would increase the robustness of embeddings to the *target* domain of interest. The procedure can be extended to multi-domain adaptation using the techniques discussed in Section 6.2.1.

6.2.3 Domain Adaptation for ASR

The feature mapping DNNs in this work are trained using unlabeled data from both domains using a combination of adversarial and cycle-consistency loss functions. Since the loss functions employed are task-independent (speaker verification in this work) and adaptation is done in the acoustic feature space, the feature mapping approach can be used for other speech applications. One interesting application of our work would be to test the efficacy of this approach in making ASR systems robust to several domains.

Bibliography

[1] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[3] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Twelfth annual conference of the international speech communication association*, 2011.

[4] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Twelfth annual conference of the international speech communication association*, 2011.

[5] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop*

on Automatic Speech Recognition and Understanding. IEEE, 2013, pp. 55–59.

[6] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[7] A. Fedorova, O. Glembek, T. Kinnunen, and P. Matějka, "Exploring ann back-ends for i-vector based speaker age estimation," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[9] A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, 2012.

[10] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

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

[12] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[14] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors : Robust DNN Embeddings for Speaker Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*. Alberta, Canada: IEEE, apr 2018, pp. 5329–5333.

[15] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, L. P. García-Perera, F. Richardson, R. Dehak *et al.*, "State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations," *Computer Speech & Language*, vol. 60, p. 101026, 2020.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint*, 2017.

[17] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[18] D. Serdyuk, K. Audhkhasi, P. Brakel, B. Ramabhadran, S. Thomas, and Y. Bengio, "Invariant representations for noisy speech recognition," *arXiv preprint arXiv:1612.01928*, 2016.

[19] D. Liang, Z. Huang, and Z. C. Lipton, "Learning noise-invariant representations for robust speech recognition," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 56–63.

[20] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," *arXiv preprint arXiv:1608.06019*, 2016.

[21] Z. Meng, Z. Chen, V. Mazalov, J. Li, and Y. Gong, "Unsupervised adaptation with domain separation networks for robust speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 214–221.

[22] K. Li, Z. Huang, Y. Xu, and C.-H. Lee, "DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[23] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[24] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 7398–7402.

[25] Z. Meng, J. Li, Y. Gong *et al.*, "Cycle-consistent speech enhancement," *arXiv preprint arXiv:1809.02253*, 2018.

[26] Y. Shi, Q. Huang, and T. Hain, "Robust speaker recognition using speech enhancement and attention model," *arXiv preprint arXiv:2001.05031*, 2020.

[27] S. Shon, H. Tang, and J. Glass, "Voiceid loss: Speech enhancement for speaker verification," *arXiv preprint arXiv:1904.03601*, 2019.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[29] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[30] P. S. Nidadavolu, J. Villalba, and N. Dehak, "Cycle-gans for domain adaptation of acoustic features for speaker recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6206–6210.

[31] P. S. Nidadavolu, S. Kataria, J. Villalba, and N. Dehak, "Low-Resource Domain Adaptation for Speaker Recognition Using Cycle-GANs," in *Accepted at IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019*. Sentosa, Singapore: IEEE, 2019.

[32] P. S. Nidadavolu, S. Kataria, J. Villalba, P. Garcia-Perera, and N. Dehak, "Unsupervised feature enhancement for speaker verification," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7599–7603.

[33] P. S. Nidadavolu, S. Kataria, P. García-Perera, J. Villalba, and N. Dehak, "Single channel far field feature enhancement for speaker verification in the wild," *arXiv preprint arXiv:2005.08331*, 2020.

[34] P. S. Nidadavolu, C.-I. Lai, J. Villalba, and N. Dehak, "Investigation on bandwidth extension for speaker recognition." in *INTERSPEECH*, 2018, pp. 1111–1115.

[35] P. S. Nidadavolu, V. Iglesias, J. Villalba, and N. Dehak, "Investigation on neural bandwidth extension of telephone speech for improved speaker recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6111–6115.

[36] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[37] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, ePFL-CONF-192584.

[38] A.-r. Mohamed, "Deep neural network acoustic models for asr." Ph.D. dissertation, University of Toronto, 2014.

[39] J. Li, D. Yu, J.-T. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in cd-dnn-hmm," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 131–136.

[40] J. Droppo and A. Acero, "Environmental robustness," in *springer handbook of speech processing*. Springer, 2008, pp. 653–680.

[41] B. S. Atal, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," *the Journal of the Acoustical Society of America*, vol. 55, no. 6, pp. 1304–1312, 1974.

[42] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[43] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[44] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.

[45] N. Brümmer and J. Du Preez, "Application-independent evaluation of

speaker detection," *Computer Speech & Language*, vol. 20, no. 2-3, pp. 230–275, 2006.

[46] D. A. Van Leeuwen and N. Brümmer, "An introduction to application-independent evaluation of speaker recognition systems," in *Speaker classification I*. Springer, 2007, pp. 330–353.

[47] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[48] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

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

[50] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[51] C. Avendano, H. Hermansky, and E. A. Wan, "Beyond nyquist: Towards the recovery of broad-bandwidth speech from narrow-bandwidth speech,"

in *Fourth European Conference on Speech Communication and Technology*, 1995.

[52] X. Zhuang, A. Ghoshal, A.-V. Rosti, M. Paulik, and D. Liu, "Improving DNN Bluetooth Narrowband Acoustic Models by Cross-bandwidth and Cross-lingual Initialization," *Proc. Interspeech 2017*, pp. 2148–2152, 2017.

[53] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[54] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[55] D. Michelsanti and Z.-H. Tan, "Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification," *arXiv preprint arXiv:1709.01703*, 2017.

[56] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. PMLR, 2018, pp. 1989–1998.

[57] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial

networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3722–3731.

[58] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[59] M. Mimura, S. Sakai, and T. Kawahara, "Cross-domain speech recognition using nonparallel corpora with cycle-consistent adversarial networks," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 134–140.

[60] T. Kaneko and H. Kameoka, "Parallel-data-free voice conversion using cycle-consistent adversarial networks," *arXiv preprint arXiv:1711.11293*, 2017.

[61] F. Fang, J. Yamagishi, I. Echizen, and J. Lorenzo-Trueba, "High-quality nonparallel voice conversion based on cycle-consistent adversarial network," *arXiv preprint arXiv:1804.00425*, 2018.

[62] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "Stargan-vc: Non-parallel many-to-many voice conversion with star generative adversarial networks," *arXiv preprint arXiv:1806.02169*, 2018.

[63] E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher, "A multi-discriminator cyclegan for unsupervised non-parallel speech domain adaptation," *arXiv preprint arXiv:1804.00522*, 2018.

[64] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[65] F. G. Germain, Q. Chen, and V. Koltun, "Speech denoising with deep feature losses," *arXiv preprint arXiv:1806.10522*, 2018.

[66] S. Kataria, P. S. Nidadavolu, J. Villalba, N. Chen, P. Garcia-Perera, and N. Dehak, "Feature enhancement with deep feature losses for speaker verification," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7584–7588.

[67] S. Kataria, P. S. Nidadavolu, J. Villalba, and N. Dehak, "Analysis of deep feature loss based enhancement for speaker verification," *arXiv preprint arXiv:2002.00139*, 2020.

[68] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database." in *Interspeech*, 2016, pp. 818–822.

[69] N. Brümmer and E. De Villiers, "The speaker partitioning problem." in *Odyssey*, 2010, p. 34.

[70] N. Brümmer and A. Strasheim, "Agnitio's speaker recognition system for evalita 2009," in *The 11th Conference of the Italian Association for Artificial Intelligence*. Citeseer, 2009.

[71] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[72] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[74] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[75] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," 2017.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[77] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2813–2821.

[78] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[79] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[80] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[81] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[82] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural net-

works for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8609–8613.

[83] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[84] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[85] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[86] M. K. Nandwana, J. Van Hout, M. McLaren, C. Richey, A. Lawson, and M. A. Barrios, "The voices from a distance challenge 2019 evaluation plan," *arXiv preprint arXiv:1902.10828*, 2019.

[87] P. García, J. Villalba, H. Bredin, J. Du, D. Castan, A. Cristia, L. Bullock, L. Guo, K. Okabe, P. S. Nidadavolu *et al.*, "Speaker detection in the wild: Lessons learned from jsalt 2019," *arXiv preprint arXiv:1912.00938*, 2019.

[88] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating

small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[89] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin *et al.*, "State-of-the-art speaker recognition for telephone and video speech: the jhu-mit submission for nist sre18," in *Interspeech*, 2019, p. accepted for publication.

[90] J. Xie, L. P. García-Perera *et al.*, "Multi-plda diarization on children's speech," *Proc. Interspeech 2019*, pp. 376–380, 2019.

[91] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[92] C. Kim and R. M. Stern, "Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[93] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[94] J. Barker *et al.*, "The third chime speech separation and recognition challenge: Dataset, task and baselines," in *2015 IEEE Workshop on Auto-*

matic Speech Recognition and Understanding (ASRU). IEEE, 2015, pp. 504–511.

[95] I. McCowan, J. Carletta *et al.*, "The ami meeting corpus," in *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, vol. 88, 2005, p. 100.

[96] D. C. et al., "Ldc2019e60, distant microphone conversational speech in noisy environments," Private communication in support of the 2019 JHU/CLSP Summer Workshop, 2019.

[97] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The 2016 speakers in the wild speaker recognition evaluation." in *Interspeech*, 2016, pp. 823–827.

[98] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1717–1731, 2010.

[99] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind mimo impulse response shortening," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 10, pp. 2707–2720, 2012.

[100] C. Richey, M. A. Barrios, Z. Armstrong, C. Bartels, H. Franco, M. Graciarena, A. Lawson, M. K. Nandwana, A. Stauffer, J. van Hout, P. Gamble, J. Hetherly, C. Stephenson, and K. Ni, "Voices obscured in complex environmental settings (voices) corpus," in *Proc. Interspeech 2018*, 2018, pp. 1566–1570.

[101] P. Wang, K. Tan, and D. Wang, "Bridging the gap between monaural speech enhancement and recognition with distortion-independent acoustic modeling," *arXiv preprint arXiv:1903.04567*, 2019.

[102] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

Vita

Phani Sankar Nidadavolu is a Ph.D. student in the Center for Language and Speech Processing (CLSP) at Johns Hopkins University. His primary research area is developing domain adaptation techniques for improving the robustness of speaker recognition systems. He is advised by Najim Dehak and Jesús Villalba. He has also worked with Hynek Hermansky on developing a multi-stream approach for improving the robustness of automatic speech recognition (ASR) systems to noisy data. He collaborated with Daniel Povey on developing a data augmentation recipe for training ASR systems in Kaldi. During his Ph.D., he worked on several speech applications like speaker diarization, age estimation, and gender identification. Before joining CLSP, he was as a research associate in Speech and Image Processing (SIP) lab at Indian Institute of Technology (IIT) Hyderabad. Currently, he is working as an 'Applied Scientist' in acoustic modelling team at Amazon.com, Inc.