DEEP SPEAKER EMBEDDING FOR ROBUST SPEAKER VERIFICATION

YOUZHI TU

PhD

The Hong Kong Polytechnic University

2022

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Deep Speaker Embedding for Robust Speaker Verification

Youzhi TU

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

October 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this report is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

_____(Signed)

_____Youzhi TU_____(Name of student)

Abstract

Speaker verification (SV) aims to determine whether the speaker identity of a test utterance matches that of a target speaker. In SV, the identity of a variable-length utterance is typically represented by a fixed-dimensional vector. This vector or its modeling process is referred to as speaker embedding. Although state-of-the-art deep speaker embedding has achieved outstanding performance, deploying SV systems to adverse acoustic environments still faces a number of challenges. First, today's SV systems rely on the condition that the training and test data share the same distribution. Once this condition is violated, domain mismatch will occur. The problem will be exacerbated when the speaker embeddings violate the Gaussianity constraint. Second, because the temporal feature maps produced by the last frame-level layer are highly non-stationary, it is not desirable to use their global statistics as speaker embeddings. Third, current speaker embedding networks do not have any mechanisms to let the frame-level information flow directly into the embeddings layer, causing information loss in the pooling layer.

This thesis develops three strategies to address the above challenges. First, to jointly address domain mismatch and the Gaussianity requirement of probabilistic linear discriminant analysis (PLDA) models, the author proposes a variational domain adversarial learning framework with two specialized networks: variational domain adversarial neural network (VDANN) and information-maximized VDANN (InfoVDANN). Both networks leverage domain adversarial training to produce speaker discriminative and domain-invariant embeddings and apply variational autoencoders (VAEs) to perform Gaussian regularization. The InfoVDANN, in particular, avoids posterior collapse in VDANNs by preserving the mutual information (MI) between the domain-invariant embeddings and the speaker embeddings. Second, to mitigate the effect of non-stationarity in the temporal feature maps, the author proposes short-time spectral pooling (STSP) and attentive STSP to transform the temporal feature maps

into the spectral domain through short-time Fourier transform (STFT). The zero- and low-frequency components are retained to preserve speaker information. A segment-level attention mechanism is designed to produce spectral representations with fewer variations, which results in better robustness to the non-stationary effect in the feature maps. Third, to allow information in the frame-level layers to flow directly to the speaker embedding layer, MI-enhanced training based on a semi-supervised deep InfoMax (DIM) framework is proposed. Because the dimensionality of the frame-level features is much larger than that of the speaker embeddings, the author proposes to squeeze the frame-level features via global pooling before MI estimation. The proposed method, called squeeze-DIM, effectively balances the dimension between the frame-level features and the speaker embeddings.

We evaluate the proposed methods on VoxCeleb1, VOiCES 2019, SRE16, and SRE18-CMN2. Results show that the VDANN and InfoVDANN outperform the DANN baseline, indicating the effectiveness of Gaussian regularization and MI maximization. We also observed that attentive STSP achieved the largest performance gains, suggesting that applying segment-level attention and leveraging low spectral components of temporal feature maps can produce discriminative speaker embeddings. Finally, we demonstrate that the squeeze-DIM outperforms the DIM regularization, suggesting that the squeeze operation facilitates MI maximization.

AUTHOR'S PUBLICATIONS

Journal Papers

1. **Y. Z. Tu** and M. W. Mak, "Aggregating Frame-Level Information in the Spectral Domain With Self-Attention for Speaker Embedding", *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 30, pp. 944–957, 2022.

2. **Y. Z. Tu**, M. W. Mak, and J. T. Chien, "Variational Domain Adversarial Learning with Mutual Information Maximization for Speaker Verification", *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 28, pp. 2013–2024, 2020.

Conference Papers

1. **Y. Z. Tu** and M. W. Mak, "Mutual Information Enhanced Training for Speaker Embedding", in *Proc. Annual Conference of the International Speech Communication Association*, 2021, pp. 91–95.

2. **Y. Z. Tu** and M. W. Mak, "Short-Time Spectral Aggregation for Speaker Embedding", in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 6708–6712.

3. **Y. Z. Tu**, M. W. Mak, and J. T. Chien, "Information Maximized Variational Domain Adversarial Learning for Speaker Verification", in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6449–6453.

4. **Y. Z. Tu**, M. W. Mak, and J. T. Chien, "Variational Domain Adversarial Learning for Speaker Verification", in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 4315–4319.

ACKNOWLEDGMENTS

I would like to express deep gratitude to my supervisor Dr. Man-Wai Mak. I appreciate his expertise in many areas and his academic rigor. I am deeply impressed by his patience and encouragement, which greatly help me to build confidence in the research. Without his help, I cannot achieve such progress at this stage. I would also like to thank Prof. Jen-Tzung Chien, who is our coauthor, for his constructive comments and suggestions to improve our papers.

I would like to thank all the teachers who have taught me and shared their research or life experiences. I would like to express special thanks to the Department of Electronic and Information Engineering, a creative platform where I am studying, and many thanks to the staff in the General Office for their various support.

I would like to express sincere gratitude to our group members Eddy Tan, Sean Xu, Weiwei Lin, Longxin Li, Lu Yi, Lingjun Zhao, and Xiaoquan KE, for their encouragement and assistance in both academic and daily life.

I am indebted to my family for their unconditional support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AAE	adversarial autoencoder
AEVB	autoencoding variational Bayes
AM-Softmax	additive margin softmax
BN	batch normalization
CNN	convolutional neural network
DA	domain adaptation
DANN	domain adversarial neural network
DAT	domain adversarial training
DCF	detection cost function
DCT	discrete cosine transform
DET	detection error tradeoff
DFT	discrete Fourier transform
DIM	deep InfoMax
DNN	deep neural network

EER	equal error rate
ELBO	evidence lower bound
FA	factor analysis
FAR	false acceptance rate
FRR	false rejection rate
G-PLDA	Gaussian PLDA
GAN	generative adversarial network
GMM	Gaussian mixture model
HT-PLDA	heavy-tailed PLDA
InfoVAE	information-maximized VAE
InfoVDANN	information-maximized VDANN
JS	Jensen–Shannon
KL	Kullback–Leibler
LDA	linear discriminant analysis
LSTM	long short-term memory
MAP	maximum *a posteriori*

MFCC	Mel-frequency cepstral coefficient
MI	mutual information
MLP	multi-layer perceptron
MMD	maximum mean discrepancy
NAP	nuisance attribute projection
PLDA	probabilistic linear discriminant analysis
ROC	receiver operating characteristic
STFT	short-time Fourier transform
STSP	short-time spectral pooling
SV	speaker verification
TDNN	time delay neural network
UBM	universal background model
VAD	voice activity detection
VAE	variational autoencoder
VB	variational Bayes
VDANN	variational domain adversarial neural network
WCCN	within-class covariance normalization

Chapter 1

INTRODUCTION

In this chapter, we introduce the topic speaker recognition and discuss the latest development of deep speaker embedding. Specifically, several challenges in state-of-the-art speaker embedding will be highlighted. These challenges motivate us to develop the strategies in later chapters as solutions. We then conclude the contributions of this thesis and outline the thesis' organization.

1.1 Speaker Recognition

Identity authentication plays a crucial role in almost all aspects in our daily life, e.g., granting access to personal devices or private premises, personalizing and customizing remote services, performing financial transactions, etc. Compared with the traditional password- or token-based mechanisms, biometric authentication provides a more convenient, efficient, and effective way to recognize an individual's identity. Such authentication exploits a person's physical or behavioral characteristics such as fingerprint, face, voice, iris, gait, etc. to build the identity profiles [6]. Because speech is a natural means of human-to-human interaction and the voices of individuals are different, voice biometrics is one of the most user-friendly biometric authentication methods. The individuality of human voices is primarily due to the variability of the vocal apparatus across speakers. Besides, voice biometrics has attracted widespread interest from academic and industrial communities due to its non-intrusive way of identity determination and the ease of accessing to speech data.

Briefly speaking, speaker recognition refers to the technology that uses a speaker's voice to recognize his/her identity. It is also called voice biometrics. Such technology has been applied to many practical scenarios such as access control, service customization, financial transactions, criminal surveillance, national security and so on.

In general, speaker recognition can be divided into two major categories: speaker identification and speaker verification (SV). The former aims to determine the identity of a test speaker from a known set of speakers, whereas the latter is to detect whether a test speaker's identity matches that of a target voice. Because SV is a one-to-one mapping evaluation, the verification result is a yes/no decision. This is in contrast to speaker identification, which is a one-to-many mapping task, i.e., determining who in a given speaker set produces the query utterance. On the other hand, according to the dependence on the text content in the input speech, speaker recognition can be text dependent or text independent. Because the lexicon in text-dependent speaker recognition systems is constrained to limited words or phrases, the degree of phonetic variability is lower than that of text-independent systems. As a result, the performance of text-dependent speaker recognition is usually better than that of the text-independent counterpart, although the duration of the input speech is typically shorter in text-dependent systems. In other words, text-independent speaker recognition is more difficult. This thesis will focus on *text-independent speaker verification*.

1.2 Development of Speaker Modeling

Modern SV systems are complex and are comprised of many components including voice activity detection (VAD), feature extraction, speaker modeling, and backend modeling (see Chapter 2 for details). During the past years, each component has evolved rapidly, especially with the advance of deep learning. Among these components, speaker modeling has experienced dramatic development and has contributed to the greatest performance gains. This section overviews the latest development in

speaker modeling.

1.2.1 I-vector

One classic speaker modeling method is the i-vector approach proposed in 2009 [7, 8]. The i-vector method uses factor analysis (FA) to model the speaker and non-speaker variabilities. Through the FA model, variable-length utterances are represented by the posterior means of the low-dimensional speaker factors and channel factors. Under the i-vector framework, the posterior means of speaker factors are used as speaker embeddings, which represent speaker-specific traits. Because of the low dimensionality of the embeddings, various channel compensation methods can be applied to suppress the channel variability in the i-vectors. Such methods include linear discriminant analysis (LDA), nuisance attribute projection (NAP) [9], within-class covariance normalization (WCCN) [10], and probabilistic linear discriminant analysis (PLDA) [11, 12]. Specifically, the i-vector/PLDA framework can produce excellent performance under a variety of practical conditions, making it the dominant approach for speaker recognition since 2010. Even today, i-vectors are frequently used.

Although the i-vector approach has achieved significant improvement in performance, it still has some drawbacks. First, the i-vector approach can be seen as an unsupervised dimension reduction method operated on the Gaussian mixture model (GMM) supervectors. Unsupervision means that the method cannot exploit the speaker labels in the training set. Therefore, there is no guarantee that speaker information is explicitly aggregated into the i-vectors. This is a huge disadvantage because large-scale labeled datasets have become widely available. Second, an i-vector is a maximum *a posteriori* (MAP) point estimate of the latent variables in an FA model. Therefore, i-vectors do not carry any information regarding their reliability and uncertainty. This is particularly problematic for short-utterance SV because the reliability of i-vectors drops when the utterance duration decreases. For example, in VoxCeleb1 [13] and VOiCES 2019 [14] where the average duration is about 8 seconds

and 16 seconds, respectively, the performance of i-vectors is not good. Third, the performance of i-vectors can quickly reach a plateau when the amount of training data increases. For example, in the i-vector system reported by [3], incorporating data augmentation to i-vector training can only obtain a limited performance gain. This observation suggests that the i-vector approach cannot fully exploit the speaker information in large training sets.

To overcome the above limitations, we will explore a more advanced speaker modeling method in this thesis. With the success of deep learning in various areas [15–18], deep speaker embedding has gradually become a mainstream approach.

1.2.2 Deep Speaker Embedding

Over the years, using deep neural networks (DNNs) to extract speaker embeddings has advanced rapidly. The advancements appear not only in network architectures but also in optimization strategies and learning metrics. In particular, end-to-end speaker embedding networks offer superior performance and have become a norm in learning speaker representation. Nevertheless, there is no consistent definition of "end-to-end". Some researchers view end-to-end from the perspective of SV protocols and derive the end-to-end losses directly from SV decisions [19–21]. Others consider end-to-end as a speaker identification problem and derive a softmax-like classification loss [3, 22–25]. In this section, we will use both definitions and give a brief overview of deep speaker embedding.

Early attempts on deep speaker embedding dated back to the d-vector approach proposed in 2014 [22]. A d-vector system uses a multi-layer perceptron (MLP) with a softmax output layer to learn frame-level features, and the d-vector is the average of the frame-level outputs at the last hidden layer. To make the d-vectors better fit the SV task, a tuple-based end-to-end model was proposed in [19], which maps an

enrollment–test pair to a decision probability. To highlight the difficult pairs[1] during training and to remove the sample selection requirement in [19], a generalized end-to-end loss was proposed in [20]. Together with a long short-term memory (LSTM), the method in [20] maps the frame-level features to an utterance-level embedding. Similar studies applying end-to-end losses can also be found in [21, 26, 27]. Although [20,21,26,27] are specialized in SV tasks, the LSTM/MLP-based embeddings can also be used for other applications.

Another line of speaker embedding extraction is based on end-to-end speaker identification networks. The embedding networks in this category have a similar structure: a convolutional neural network (CNN)/LSTM-based frame-level subnetwork, a pooling layer, and a fully-connected utterance-level network. Compared with the speaker embeddings trained by end-to-end losses, this kind of speaker embedding can generalize better to unseen data and are more robust to noise, reverberation, and domain mismatch [28–31]. A classical speaker embedding following this line is the x-vector [3], which uses time delay neural networks (TDNNs) to extract frame-level features and applies statistics pooling to summarize these features in the form of fixed-length vectors. Since the emergence of the x-vector, it has been a baseline for speaker embedding.

With the development of more advanced DNNs, ResNets [32], DenseNets [33], and Res2Nets [34] have been widely used to extract the frame-level information in speaker embedding networks [23–25]. For example, lightweight ResNets were adapted from ResNet-34 and ResNet-50 in [23] and [35]. In both [24] and [36], DenseNets were used in the frame-level subnetwork. Also, the authors of [25] applied a Res2Net to process frame-level information. A key advantage of using CNNs or more advanced architectures in the frame-level subnetwork is that the complex temporal relations across frames can be better captured compared with the traditional MLPs. This is the

[1]The speaker-specific trait of the enrollment utterance is similar to that of the test utterance.

reason that the CNN-based speaker embedding has become the mainstream speaker modeling method at present. Unless otherwise stated, the deep speaker embeddings in this thesis refer to the utterance-level representations extracted from CNN-based end-to-end speaker identification networks.

Besides the development in frame-level subnetworks, the pooling layer has advanced simultaneously for more efficient utterance-level aggregation. For instance, the authors of [27] projected the channel-wise mean vectors of the frame-level features onto a speaker embedding space. In the x-vector extractor, both the means and the standard deviations of the frame-level features are computed through a statistics pooling layer [3]. By simultaneously pooling over the features from different frame-level layers, multi-level pooling was proposed in [37]. Besides, the authors of [38] proposed the learnable dictionary encoding where the encoded vectors act like the means of a GMM. In [23], a NetVLAD layer [39] was applied for utterance-level aggregation. Instead of performing attention across the frames, channel- and context-dependent statistics pooling [25] extends attention along the channel dimension to highlight the contribution of individual channels.

Another popular category of frame-level aggregation is the attentive pooling. For instance, an attention mechanism was introduced to weight the temporal frames so that the attended frames have greater contribution to the speaker embedding vector [40]. To increase the representation capacity of the aggregated embeddings, multi-head attentive pooling was proposed to attend the convolutional features from multiple perspectives [41]. The authors of [42] further extended this multi-head idea and diversified the attention heads by allowing different resolutions in different heads. Different from [41] where each head attends the frame-level features across all channels, the authors of [43] applied each head to a subset of the channels. By integrating the attention mechanism and GMM clustering, the authors of [24] proposed a mixture of attentive pooling from a probabilistic perspective.

The above pooling methods operate in the temporal domain. Yet we can perform

aggregation in the spectral domain. Inspired by the work in [44], the author proposed short-time spectral pooling (STSP) in [45] to preserve speaker information. To emphasize the discriminative segments, the author further introduced attentive STSP in [46] as an extension of STSP. Results show that the attentive STSP can further enhance the aggregation of frame-level information.

1.3 Motivations of the Thesis

Although CNN-based speaker embedding has achieved state-of-the-art performance, there are still several challenges to be overcome. These challenges motivate us to develop advanced strategies to make SV systems more practical.

1. *Domain mismatch:* To develop practical SV systems, we assume that the training data (source-domain data) share the same distribution with the test data (target-domain data). In practice, however, the distributions of the training and test data can differ due to the discrepancy in the conditions from which the data were collected, e.g., different channels, languages, noises, etc. As a result, the assumption can hardly be met and domain mismatch occurs, which poses a great challenge to SV. Therefore, it is necessary to adapt the trained models based on some target-domain data. This strategy is known as domain adaptation (DA). On the other hand, due to the high cost of data labeling, usually only a small amount of labeled data or even no labeled data from the target domain are available. This difficulty motivates us to seek advanced DA methods to alleviate the domain mismatch problem.

2. *Gaussianity requirement of PLDA models:* PLDA models have been playing a key role in backend scoring since the era of i-vectors. Although deep speaker embeddings are amenable to the simple cosine scoring and rely less on the PLDA model, the PLDA scorer still outperforms the cosine scorer (especially in EER)

for median-sized embedding networks like the x-vector extractor [38, 47, 48]. However, PLDA models require the x-vectors to follow a Gaussian distribution. Once this assumption is violated, the SV performance will be degraded severely. Deep speaker embeddings, on the other hand, are not guaranteed to be Gaussian and thus may not meet the Gaussianity requirement of PLDA backends. This introduces a challenge in applying PLDA models. Traditional solutions to this problem include using heavy-tailed PLDA [1, 49] or applying the x-vector length normalization [50]. However, the former is more computationally expensive than the Gaussian PLDA and the latter is not really a Gaussianization procedure but a sub-optimal compromise. Therefore, we need to develop Gaussian-regularized optimization for x-vectors.

3. *Utterance-level aggregation:* Modern speaker embedding networks typically apply a pooling layer to aggregate the frame-level information into utterance-level embeddings. One baseline aggregation strategy is to use channel-wise means and standard deviations of the last frame-level feature maps as the summarization of the whole utterance [3]. However, due to the sharp dimensionality reduction in the pooling operation, some speaker information will inevitably be lost in the aggregation process. This motivate us to preserve as much speaker information as possible during the pooling operation. On the other hand, most pooling methods are performed in the temporal domain. Due to the high non-stationarity in the final feature maps, it is not beneficial to exploit the temporal statistics during aggregation. Thus, aggregation methods that aim to preserve information and to exploit the stationarity in the feature maps should be developed.

4. *Frame-level information enhanced learning:* As introduced in Section 1.2.2, the frame-level subnetwork of a speaker embedding network contributes the largest performance gains. It is therefore crucial to enhance the flow of the frame-level

information through the network so that more speaker information can reach the speaker embedding layer. Mainstream embedding networks use CNNs, ResNets, or DenseNets to facilitate the frame-level information flow. Nevertheless, this information flow is constrained within the frame-level subnetwork only and the frame-level information has to be aggregated through an additional pooling operation before utterance-level processing. This not only reduces the efficiency of information flow but also leads to information loss. Because the goal of speaker embedding networks is to produce utterance-level representations, it makes sense to let the frame-level information flow directly into the utterance-level subnetwork, bypassing the pooling layer. This motivate us to propose a frame-level information preservation framework for speaker embedding.

1.4 Contributions of the Thesis

This thesis contributes to the field of SV by addressing the challenges introduced in Section 1.3. The main contributions are as follows.

1. *Variational domain adversarial learning:* To jointly address domain mismatch and the Gaussianity constraint, the author incorporates a variational autoencoder (VAE) [2] into the conventional domain adversarial neural network (DANN) [51] and proposes a novel variational domain adversarial neural network (VDANN) [31,52,53]. The DANN part aims to alleviate the domain mismatch between the training and test data through adversarial training [54], whereas the VAE is to constrain the learned embeddings to be Gaussian so that the Gaussian PLDA backend can be directly applied. This strategy is called variational domain adversarial learning in this thesis.

   However, a potential limitation of the VDANN is that posterior collapse [55–58] may occur while training the VAE. Because posterior collapse occurs when the produced latent vectors (embeddings) are independent of the inputs, posterior

collapse can lead to non-informative speaker embeddings. This outcome is undesirable because our objective is to learn meaningful embeddings. To address this limitation, the author adopts the idea of InfoVAE [59] and incorporates a mutual information (MI) term into the objective function so that the dependence of the speaker embeddings on the inputs can be enhanced. The resulting architecture is called information-maximized VDANN (InfoVDANN) [31, 53].

2. *Utterance-level aggregation in the spectral domain:* To preserve as much speaker information as possible when aggregating the frame-level representations, the author proposes short-time spectral pooling (STSP) [45] and performs aggregation in the spectral domain. From a Fourier perspective, the conventional statistics pooling [3] only exploits the DC (zero frequency) components of the last frame-level feature maps. STSP improves statistics pooling by retaining more frequency components besides the DC ones to preserve richer information. The author proves that STSP is a generalized statistics pooling method.

   To compute the spectral embeddings in STSP, we simply average the spectrogram along the temporal axis for each utterance. However, this brute average ignores the importance of individual windowed segments in the spectrogram. To emphasize on the discriminative segments, the author extends STSP by using a self-attention mechanism in the spectral domain and proposes a novel attentive STSP method [46]. Due to the segment-level attention mechanism, attentive STSP can produce spectral embeddings with less variation than attentive pooling, making it more robust against the non-stationarity in the feature maps.

3. *Mutual information enhanced training:* To directly feed the frame-level information into the speaker embeddings, the author proposes to maximize the MI between the frame-level features and the utterance-level embeddings. The author adopts the Deep InfoMax (DIM) framework [60] to perform MI estimation

through semi-supervised learning. However, a straightforward implementation of DIM may pose a dimensionality imbalance problem, leading to unreliable MI estimation and performance degradation. To overcome this problem, the author proposes to squeeze the frame-level features before MI estimation through some global pooling methods. The resulting structure is called squeeze-DIM [61] regularizer, which facilitates the MI maximization.

1.5 Thesis Organization

The remainder of the thesis is organized as follows:

Chapter 2 presents a literature survey on SV. Specifically, the key components of a typical SV system, such as the feature extraction module, the speaker embedding network, and the backend, are introduced.

In Chapter 3, the author reviews two modern deep learning models: generative adversarial networks (GANs) and VAEs. These models will be used in later chapters.

In Chapter 4, the author introduces the variational domain adversarial learning framework and details the principle of VDANN and InfoVDANN, which are used to jointly address domain mismatch and Gaussianity requirement of the PLDA models.

Chapter 5 is to address the utterance-level aggregation challenge. The author proposes to aggregate the frame-level information in the spectral domain. In particular, STSP and attentive STSP are introduced to preserve richer speaker information in the aggregated embeddings.

In Chapter 6, the author performs MI enhanced training so that the information in the frame-level layers can be directly fed into the speaker embeddings. The author will introduce the DIM framework for MI maximization and detail the proposed squeeze-DIM regularization.

Finally, the author gives conclusions and suggests possible future work in Chapter 7.

Chapter 2

SPEAKER VERIFICATION

This chapter presents a literature survey on speaker verification, including the processing pipeline, the key components of speaker verification systems, and the performance evaluation metrics.

2.1 System Overview

Speaker verification (SV) is to determine whether the identity of a claimed utterance matches a target identity. As shown in Figure 2.1, a typical SV task consists of a training phase and a verification phase. In the training phase, a speaker embedding network is learned from a large amount of training data. For the SV system relying on the probabilistic linear discriminant analysis (PLDA) [11, 12] backend, a PLDA model is also trained in this phase. The verification phase actually contains an enrollment stage and a test stage. In the enrollment stage, one or more utterances from an enrolled speaker are provided for speaker modeling. During the test stage, the claimant's embedding is extracted from the test utterance and verified against the enrolled speaker embedding by comparing the score of the enrollment–test trial with a threshold. If the score is larger than the threshold, we accept the hypothesis that the test utterance has the same identity as that of the enrollment utterance; otherwise, the enrollment and test utterances come from different speakers.

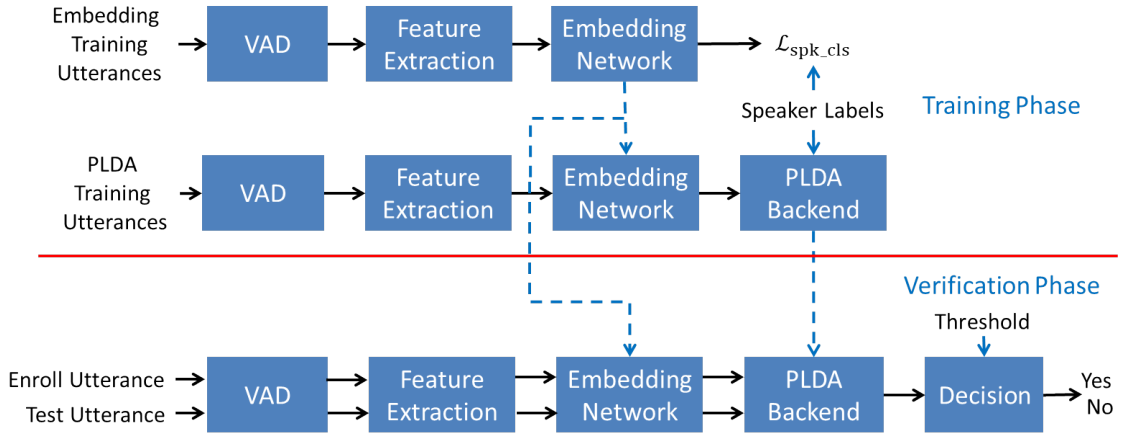


Figure 2.1: Overview of a typical speaker verification (SV) system. The schematics above the horizontal red line denote the training phase, which contains a speaker embedding network training stage and a PLDA backend training stage. The verification phase is illustrated below the red line, where an enrollment stage and a test stage are involved. In the verification phase, an enrollment–test embedding pair is scored by the backend and the score is compared with a threshold to decide whether the identity of the test utterance is the same as that of the enrolled speaker.

2.2 Voice Activity Detection

Voice activity detection (VAD) is to determine the speech and non-speech segments in a speech signal. Although it is a simple binary classification task, it is important for a variety of speech applications, e.g., keyword spotting [62], SV [63, 64], etc. VAD is beneficial to SV in mainly two aspects. First, it filters out the non-speech activities such as noises, which reduces the effect of noises and increases the contribution of the speech activities. Second, VAD allows the subsequent pipelines to process the speech segments only and reduces the computational cost in the verification stage. This is important for real-time applications and personalized VAD [65].

A naive implementation of VAD is the energy-based VAD, which is the default speech/non-speech detection in the Kaldi's SRE16 recipe[1] and Kaldi's VoxCeleb

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

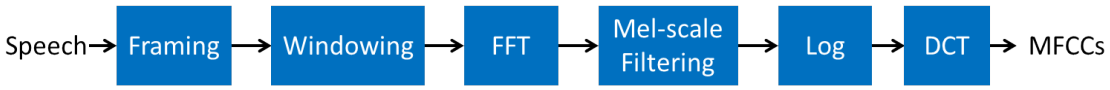


Figure 2.2: Extraction pipeline of MFCCs.

recipe.[2] More advanced VAD involves statistical models, GMMs or application-specific processing [63]. Recently, DNN-based VADs are becoming popular, e.g., [66] and [67] are based on the CNN-BiLSTM architecture.

Note that not all SV systems have a VAD module. For example, in [68], the VAD is replaced by a temporal gating operation and this module can be flexibly placed after any frame-level layer.

2.3 Feature Extraction

Before the emergence of DNN-based speaker embedding, acoustic features play a crucial role in traditional speaker modeling such as the GMM universal background model (UBM) and i-vector. One of the popular acoustic features in SV are Mel-frequency cepstral coefficients (MFCCs) [69], which are widely used in GMM-based speaker models. Figure 2.2 shows the processing pipeline of MFCCs. Because speech signals are non-stationary, we rely on short-term statistics to represent the characteristics of a speech signal. This is done by first segmenting the speech into short and overlapped frames. A typical configuration is to use a duration of 25ms with a 10ms sliding step. A Hamming or Hanning window is usually applied on each frame to suppress the spectral leakage incurred by Fourier transform. After that, a nonlinear Mel-scale filter-bank analysis is performed on the spectrogram so that the produced filter-bank coefficients (spectrum energy in each Mel-scale frequency band) can better approximate the response of the human auditory system. After applying a logarithm

[2] https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb/v2.

operation, we obtain the filter-bank energy coefficients. Finally, discrete cosine transform (DCT) is applied to the filter-bank features and the leading coefficients are retained as MFCCs.

Although MFCCs are mainstream acoustic features during the pre-DNN era, recent studies on deep speaker embedding have shown that MFCCs are not advantageous over the simple filter-bank features [18, 28, 29, 48]. The acoustic features can even be the simpler spectrograms when 2-D CNNs are used in the frame-level layers of the embedding network [13, 23, 27]. In short, with a decreasing dependence on the acoustic features, their choices are becoming more and more diverse for deep speaker embedding.

Note that this spectral feature extraction module may not be necessary if raw waveforms are directly used for training an embedding network [68, 70–73]. However, in this thesis, we focus on the deep speaker embedding that uses the dominant spectrum-based features such as filter-bank energy coefficients and MFCCs.

2.4 Speaker Embedding Networks

Compared with the traditional i-vectors, DNN-based speaker embeddings offer several advantages. Firstly, given a large amount of speech data, DNNs are able to fit more complex functions than GMMs. Thus, this data-driven approach can fully exploit the speaker-related information in large-scale training data. Moreover, attributed to the large modeling capacity of DNNs, deep speaker embeddings can better capture the variability in short utterances. Secondly, with an increasing amount of labeled data, deep speaker embedding can sufficiently aggregate the speaker-discriminative information from training data through supervised learning. Thirdly, DNNs are less dependent on the assumption of the input acoustic features. For example, in the i-vector method, the GMMs normally use a diagonal covariance matrix for each mixture component, and this requires that the elements of an observed acoustic feature vector

Table 2.1: Architecture of the speaker embedding network used in this thesis. BN represents batch normalization [4]. T denotes the total number of frames in an utterance and N_{spk} is the number of training speakers.

Layer	Layer Type	Layer Context	Total Context	Output Dimension
1	TDNN-BN-ReLU	$[t-2, t+2]$	5	512
2	TDNN-BN-ReLU	$\{t-2, t, t+2\}$	9	512
3	TDNN-BN-ReLU	$\{t-3, t, t+3\}$	15	512
4	Dense-BN-ReLU	$\{t\}$	15	512
5	Dense-BN-ReLU	$\{t\}$	15	1,500
6	Pooling	$[0,\ T)$	T	3,000
7	Dense-BN	$[0,\ T)$	T	256
8	AM-Softmax	$[0,\ T)$	T	N_{spk}

should be decorrelated. This is the reason why MFCCs [69] are widely used in i-vector modeling. However, DNNs are more competent in exploring the correlation among the feature components. Therefore, even simple filter-bank energy features can outperform MFCCs when large DNNs are used [18, 48]. In this section, we mainly introduce the baseline speaker embedding network (a modified x-vector extractor) that will be used in later chapters.

2.4.1 Network Architecture

As illustrated in Table 2.1, the configuration of the speaker embedding network used in this thesis is almost identical to that of [3]. One difference is that the additive margin softmax (AM-Softmax) [74] is used in the output layer of the former system. The motivation is that the AM-Softmax loss takes both inter-speaker variations and within-speaker variations into account, thereby being more suitable for the verification task. Another difference is that the former uses a dense layer of 256 nodes for utterance-level processing, because this configuration produces better performance under the AM-Softmax loss. During frame-level processing, each TDNN layer aggregates several contextual frames from the previous layer, resulting in a temporal

context of 15 acoustic frames at Layer 3. For each utterance, its speaker embedding vector is the affine output at Layer 7.

We can also use more advanced convolutional layers to replace the TDNN layers in the frame-level subnetwork for better performance. An example is shown in Chapter 6, where the middle TDNN layers are replaced by three Res2Net blocks so that the information flows within each block can be diversified.

2.4.2 Pooling Strategy

We introduce three baseline pooling methods for utterance-level aggregation: statistics pooling, multi-head attentive pooling, and channel- and context-dependent statistics pooling.

Statistics Pooling

The default pooling method for x-vector is statistics pooling. Denote $\mathbf{H} = \{\mathbf{h}_t\}_{t=0}^{T-1} \in \mathbb{R}^{C\times T}$ as a feature map at the last convolutional layer, where C is the number of channels in the feature map $\mathbf{H}$ and T is the number of frames. $\mathbf{H}$ comprises a sequence of frame-level vectors fed to the pooling layer. The aggregated representation $\mathbf{z}$ is expressed as

$$\mathbf{z} = [\boldsymbol{\mu}^\top, \boldsymbol{\sigma}^\top]^\top, \tag{2.1}$$

where

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{h}_t, \tag{2.2}$$

and

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{T}\operatorname{diag}\left(\sum_{t=0}^{T-1}\mathbf{h}_t\mathbf{h}_t^\top - \boldsymbol{\mu}\boldsymbol{\mu}^\top\right)}. \tag{2.3}$$

In (2.3), diag$(\cdot)$ means constructing a vector using the diagonal elements of a square matrix and the square root is operated element-wise. In short, the aggregated representation $\mathbf{z}$ is the concatenation of channel-wise means and standard deviations of

the feature map.

Multi-head Attentive Pooling

In [41], an attention mechanism with multiple heads was introduced to attend the frame-level features from various perspectives. Let us consider an H-head attention network with a *tanh* hidden layer of D nodes and a linear output layer. The attention weight matrix $\mathbf{A} = (a_{t,h}) \in \mathbb{R}^{T \times H}$ can be computed as

$$\mathbf{A} = \text{Softmax}\left(\tanh\left(\mathbf{H}^\top \mathbf{W}_1\right)\mathbf{W}_2\right), \tag{2.4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{C \times D}$ and $\mathbf{W}_2 \in \mathbb{R}^{D \times H}$ are trainable weight matrices and the softmax function is operated column-wise. For the h-th head ($h \in \{1, \ldots, H\}$), the attended mean and standard deviation vectors are computed as follows:

$$\boldsymbol{\mu}_h = \sum_{t=0}^{T-1} a_{t,h}\mathbf{h}_t, \tag{2.5}$$

and

$$\boldsymbol{\sigma}_h = \sqrt{\text{diag}\left(\sum_{t=0}^{T-1} a_{t,h}\mathbf{h}_t\mathbf{h}_t^\top - \boldsymbol{\mu}_h\boldsymbol{\mu}_h^\top\right)}. \tag{2.6}$$

Finally, we have the aggregated vector as follows:

$$\mathbf{z} = \left[\boldsymbol{\mu}_1^\top, \boldsymbol{\sigma}_1^\top, \ldots, \boldsymbol{\mu}_H^\top, \boldsymbol{\sigma}_H^\top\right]^\top. \tag{2.7}$$

A major difference between statistics pooling and attentive pooling is that the latter scales the feature map by an attention weight vector $\{a_{t,h}\}_{t=0}^{T-1}$ for each head h during the pooling process (see (2.5) and (2.6)). The purpose of the attention weight vector is to emphasize discriminative frames for information aggregation. Because multi-head attentive pooling has H independent attention weight vectors in $\mathbf{A}$, its capacity for

information preservation is larger than that of single-head attentive pooling.

Channel- and Context-Dependent Statistics Pooling

Multi-head attentive pooling applies attention weights independently on channel-wise feature sequences, assuming that each channel has equal importance to the discriminative power of speaker embeddings. In [25], channel- and context-dependent statistics pooling (CCDSP) was proposed to account for the contribution of individual channels. To enable the attention network to take the utterance's global properties (such as noise or recording conditions) into consideration, we concatenate $\mathbf{h}_t$ in (2.2) with the global non-weighted mean $\boldsymbol{\mu}$ (see (2.2)) and standard deviation $\boldsymbol{\sigma}$ (see (2.3)) along the channel axis, i.e., $\tilde{\mathbf{h}}_t = [\mathbf{h}_t^\top, \boldsymbol{\mu}^\top, \boldsymbol{\sigma}^\top]^\top$. Then, we compute the attention scores as

$$e_{t,c} = \boldsymbol{v}_c^\top f\left(\boldsymbol{W}_3 \tilde{\mathbf{h}}_t + \boldsymbol{b}\right) + k_c, \quad c = 1, \ldots, C, \tag{2.8}$$

where $\boldsymbol{W}_3 \in \mathbb{R}^{D' \times 3C}$ and $\boldsymbol{b} \in \mathbb{R}^{D'}$ are the channel independent weight matrix and bias vector to be learned, respectively, and $f(\cdot)$ is a non-linear activation function. $\boldsymbol{v}_c \in \mathbb{R}^{D'}$ and $k_c \in \mathbb{R}$ are the learned channel-dependent weight vector and bias of the c-th channel, respectively. $e_{t,c}$ is then normalized along the frame dimension via a softmax function to compute the channel-dependent attention weights

$$a_{t,c}^{\mathrm{CD}} = \frac{\exp\left(e_{t,c}\right)}{\sum_{\tau=0}^{T-1} \exp\left(e_{\tau,c}\right)}. \tag{2.9}$$

Finally, the weighted mean vector $\boldsymbol{\mu}^{\mathrm{CD}} = \left\{\mu_c^{\mathrm{CD}}\right\}_{c=1}^{C}$ and the standard deviation vector $\boldsymbol{\sigma}^{\mathrm{CD}} = \left\{\sigma_c^{\mathrm{CD}}\right\}_{c=1}^{C}$ are concatenated to form the aggregated embedding, where

$$\mu_c^{\mathrm{CD}} = \sum_{t=0}^{T-1} a_{t,c}^{\mathrm{CD}}\, h_{t,c}, \tag{2.10}$$

and

$$\sigma_c^{\mathrm{CD}} = \sqrt{\sum_{t=0}^{T-1} a_{t,c}^{\mathrm{CD}}\, h_{t,c}^2 - (\mu_c^{\mathrm{CD}})^2}. \tag{2.11}$$

2.4.3 Additive Margin Softmax Loss

X-vector extractor uses a softmax activation function at the output layer. Because the softmax loss aims to separate different classes, it is not good at reducing the within-class variations. To reduce within-speaker variations and obtain more discriminative embeddings, we may use the additive margin softmax loss [74]:

$$\begin{aligned}\mathcal{L} &= -\frac{1}{N_{\mathrm{trn}}}\sum_{i=1}^{N_{\mathrm{trn}}} \log \frac{e^{s\left(\cos\theta_{y_i}-m\right)}}{e^{s\left(\cos\theta_{y_i}-m\right)} + \sum_{j=1,j\neq y_i}^{N_{\mathrm{spk}}} e^{s\cos\theta_j}} \\ &= -\frac{1}{N_{\mathrm{trn}}}\sum_{i=1}^{N_{\mathrm{trn}}} \log \frac{e^{s\left(\tilde{\mathbf{w}}_{y_i}^\top \mathbf{f}^i-m\right)}}{e^{s\left(\tilde{\mathbf{w}}_{y_i}^\top \mathbf{f}^i-m\right)} + \sum_{j=1,j\neq y_i}^{N_{\mathrm{spk}}} e^{s\,\tilde{\mathbf{w}}_j^\top \mathbf{f}^i}},\end{aligned} \tag{2.12}$$

where m and s denote the cosine margin and the scaling factor, respectively and N_{trn} is the number of training samples in a mini-batch. $\mathbf{f}^i$ is the affine output of the Dense-BN layer (Layer 7) in Table 2.1 for the i-th training sample, and y_i is the corresponding speaker label. $\tilde{\mathbf{w}}_j$, which corresponds to the j-th output node, is the j-th column of the weight matrix $\widetilde{\mathbf{W}}$, i.e., $\widetilde{\mathbf{W}} = \{\tilde{\mathbf{w}}_j\}_{j=1}^{N_{\mathrm{spk}}}$. Note that both $\tilde{\mathbf{w}}_j$ and $\mathbf{f}^i$ are normalized to unit length.

2.5 Backends

In SV systems, a backend is used to compute the scores for decision-making given a pair of enrollment and test speaker embeddings. Probabilistic linear discriminant analysis (PLDA) [11,12] is one of the commonly used backend. PLDA is a probabilistic generative modeling method that can be seen as supervised factor analysis. Compared with the cosine scorer, the PLDA backend is able to compensate for the channel variabilities in the speaker embeddings. According to the assumption on the input

distributions, PLDA backends can be divided into Gaussian PLDA and heavy-tailed PLDA.

2.5.1 Gaussian Probabilistic Linear Discriminant Analysis

Gaussian PLDA (G-PLDA) has been widely used since the i-vector era. There are three types of G-PLDA: the standard PLDA [12], the simplified PLDA [75, 76], and the two-covariance model [11]. Given a training set $\mathcal{X} = \{\mathbf{x}_{ij}; i \in [1, N], j \in [1, N_i]\}$ of N speakers and each with N_i embeddings, the standard PLDA formulation can be expressed as

$$\mathbf{x}_{ij} = \mathbf{m} + \mathbf{V}\mathbf{z}_i + \mathbf{U}\mathbf{y}_{ij} + \boldsymbol{\epsilon}_{ij}, \tag{2.13}$$

where $\mathbf{m} \in \mathbb{R}^D$ is the global mean vector of the training embeddings, $\mathbf{V} \in \mathbb{R}^{D\times M}$ contains the basis spanning a low-dimensional speaker subspace, and $\mathbf{U} \in \mathbb{R}^{D\times P}$ is a low-rank matrix whose columns span the channel subspace. The latent variable $\mathbf{z}_i$'s are speaker factors with a standard Gaussian distribution, i.e., $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, whereas $\mathbf{y}_{ij}$'s are channel factors with $\mathbf{y}_{ij} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. $\boldsymbol{\epsilon}_{ij}$ denotes the Gaussian residue with zero mean and a diagonal covariance matrix $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. If we use a full covariance matrix for $\boldsymbol{\epsilon}_{ij}$, we can merge the channel term ($\mathbf{U}\mathbf{y}_{ij}$) into the residue and obtain the simplified G-PLDA model:

$$\mathbf{x}_{ij} = \mathbf{m} + \mathbf{V}\mathbf{z}_i + \boldsymbol{\epsilon}_{ij}. \tag{2.14}$$

The parameters $\boldsymbol{\omega} = \{\mathbf{m}, \mathbf{V}, \boldsymbol{\Sigma}\}$ can be optimized by an EM algorithm [77].

During the verification stage, given a pair of target and test speaker embeddings $(\mathbf{x}_s, \mathbf{x}_t)$, the likelihood ratio score is calculated as [75]

$$S(\mathbf{x}_s, \mathbf{x}_t) = \frac{P(\mathbf{x}_s, \mathbf{x}_t|\text{ same })}{P(\mathbf{x}_s, \mathbf{x}_t|\text{ different })} = \frac{\int p(\mathbf{x}_s, \mathbf{x}_t, \mathbf{z}|\boldsymbol{\omega})\, \mathrm{d}\mathbf{z}}{\int p(\mathbf{x}_s, \mathbf{z}_s|\boldsymbol{\omega})\, \mathrm{d}\mathbf{z}_s \int p(\mathbf{x}_t, \mathbf{z}_t|\boldsymbol{\omega})\, \mathrm{d}\mathbf{z}_t}, \tag{2.15}$$

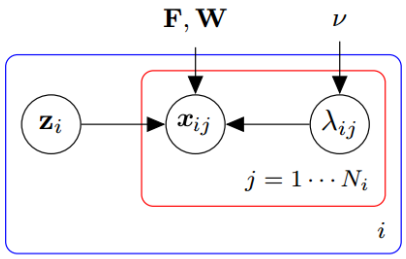


Figure 2.3: Graphical illustration of simplified heavy-tailed PLDA [1].

where $\boldsymbol{\omega}$ denotes the parameters of the G-PLDA model.

2.5.2 Heavy-Tailed Probabilistic Linear Discriminant Analysis

G-PLDA assumes that the observed speaker embeddings follow a Gaussian distribution. This is, however, unlikely the case in practice because speaker embeddings often exhibit large deviations from the mean vector [78] if no additional Gaussianization procedures are performed. To accommodate the large deviation from the mean, heavy-tailed priors such as Student's t distributions were used to describe the latent variables, which results in heavy-tailed PLDA (HT-PLDA) [78]. In [1], an efficient implementation was introduced to reduce the computational cost of HT-PLDA.

The graphical representation of the efficient HT-PLDA is illustrated in Figure 2.3. For speaker i, $\mathbf{x}_{ij} \in \mathbb{R}^D$ and $\mathbf{z}_i \in \mathbb{R}^M$ are the observed speaker embedding and normally distributed speaker factors, respectively. λ_{ij} denotes the precision scaling factor which follows a Gamma distribution $\mathcal{G}(\alpha, \beta)$ parametrized by $\alpha = \beta = \nu/2$. The parameter ν is known as the degrees of freedom. The HT-PLDA model is formulated as [1]

$$p\left(\mathbf{x}_{ij}|\mathbf{z}_i, \lambda_{ij}\right) = \mathcal{N}\left(\mathbf{x}_{ij}|\mathbf{F}\mathbf{z}_i, \left(\lambda_{ij}\mathbf{W}\right)^{-1}\right), \tag{2.16}$$

where $\mathbf{F} \in \mathbb{R}^{D\times M}$ is the speaker factor loading matrix and $\mathbf{W} \in \mathbb{R}^{D\times D}$ is the precision

matrix of the residues. The model parameters are $\{\nu, \mathbf{F}, \mathbf{W}\}$. However, there is no closed-form solution for the M-step of the EM algorithm for optimizing the model parameters, nor is there any closed-form expression for exact HT-PLDA scoring. Either $\mathbf{z}_i$ or λ_{ij} can be integrated out in closed form, but not both. However, Gaussian likelihood approximation [79] can be applied to address this problem.

2.6 Evaluation Metrics

Two types of errors are used to measure the performance of an SV system: false rejection rate (FRR) and false acceptance rate (FAR). The FRR represents the chance of falsely rejecting the true speakers, and it is also known as the miss rate:

$$P_{\text{Miss|Target}} = N_{\text{Miss}}/N_{\text{Target}}, \tag{2.17}$$

where N_{Miss} is the number of false rejections given a decision threshold and N_{Target} is the total number of true-speaker trials. Whereas the FAR is the chance of falsely accepting the imposters, also known as the false alarm rate:

$$P_{\text{FalseAlarm|Nontarget}} = N_{\text{FalseAlarm}}/N_{\text{Nontarget}}, \tag{2.18}$$

where $N_{\text{FalseAlarm}}$ is the number of false acceptances and $N_{\text{Nontarget}}$ is the total number of imposter trials.

A well-performed SV system should achieve a low FAR and also a low FRR. However, because FRRs and FARs have opposite trends when the decision threshold changes, it is impossible to decrease both error rates simultaneously. The equal error rate (EER) [80], the detection cost function (DCF) [81], and the detection error tradeoff (DET) curves [82] are commonly used as evaluation metrics. These metrics are all derived from FRR and FAR.

- *Equal Error Rate:* The EER [80] refers to the operating point at which the FAR

is equal to the FRR. EER indicates the system performance independent of the threshold. The lower the ERR, the better the performance.

- *Detection Cost Function:* The DCF [81] is defined as a weighted sum of the FRR and FAR at a specific decision threshold θ:

$$C_{\text{Det}}(\theta) = C_{\text{Miss}} \times P_{\text{Target}} \times P_{\text{Miss|Target}}(\theta) + C_{\text{FalseAlarm}} \times (1 - P_{\text{Target}}) \times P_{\text{FalseAlarm|Nontarget}}(\theta), \tag{2.19}$$

where C_{Miss} and $C_{\text{FalseAlarm}}$ are the costs of miss detection and false acceptance, respectively, and P_{Target} is the prior probability of target speakers. C_{Det} is often normalized by C_{Default} for better interpretation of the performance, with C_{Default} defined as the best cost that could be obtained without the input utterances:

$$C_{\text{Norm}}(\theta) = C_{\text{Det}}(\theta)/C_{\text{Default}}(\theta), \tag{2.20}$$

where $C_{\text{Default}}(\theta) = \min\{C_{\text{Miss}} \times P_{\text{Target}}, C_{\text{FalseAlarm}} \times (1 - P_{\text{Target}})\}$. In practice, the minimum DCF (minDCF)—obtained by sweeping the thresholds—and the actual DCF (actDCF)—determined by an application-specific threshold—are mainly used as performance metrics.

- *Detection Error Tradeoff:* A DET curve [77,82] is similar to a receiver operating characteristic (ROC) curve except that both axes in the DET curve follow a non-linear scale. When the distributions of the true-speaker trials and the imposter trials are normal, the DET curve will be a straight line. This facilitates the comparison of similar systems: the closer the curve is to the origin, the better the performance. An example of the DET curve is shown in Figure 2.4.

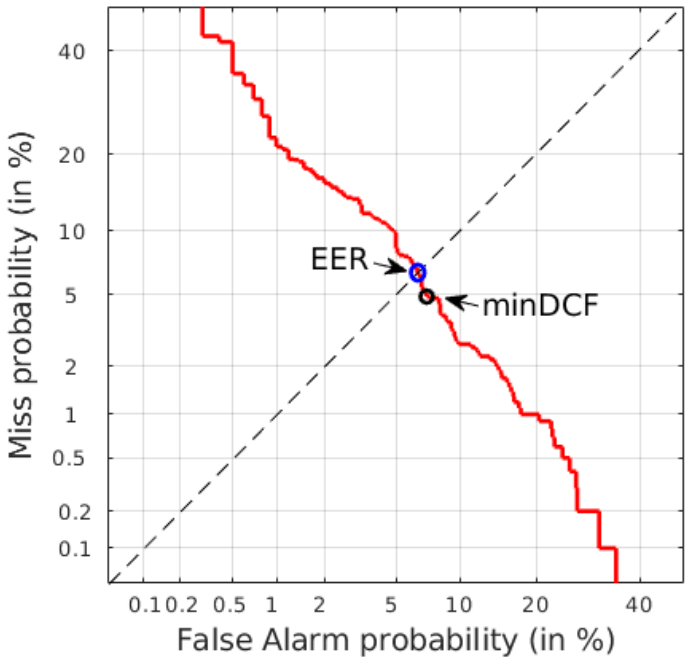


Figure 2.4: Illustration of the DET curve (in red color). The blue circle, which locates at the intersection of the first quadrant angle bisector and the DET curve, refers to the operating point where EER is achieved. The black circle denotes the operating point with minDCF.

Chapter 3

MODERN DEEP LEARNING MODELS

In this chapter, we introduce generative adversarial networks (GANs) and variational autoencoders (VAEs). These networks will be used in later chapters.

3.1 Generative Adversarial Networks

The generative adversarial network (GAN) is a well-known framework for generative modeling through an adversary. As shown in Figure 3.1, the standard GAN consists of a generative model which aims to learn the distribution of the real data, and a discriminative model that is to differentiate the real samples from the generated fake samples. The training of GANs is a minimax optimization of the objective function. Specifically, the discriminative model is trained by maximizing the GAN objective so that it can correctly distinguish between the real data and the generated data, whereas the generative model is optimized by minimizing the objective function to fool the discriminative model.

Given a training set $\mathcal{X}$ with true data distribution $p_{\mathcal{D}}(\mathbf{x})$ ($\mathbf{x} \in \mathcal{X}$), from a generative modeling perspective, the aim of the generative model is to learn a distribution $p_g(\mathbf{x})$ as close to $p_{\mathcal{D}}(\mathbf{x})$ as possible. Denote the generative model by a differentiable function $G(\cdot;\theta_g)$ parameterized by a DNN with parameters θ_g; $G(\mathbf{z};\theta_g)$ aims to map noise samples $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ to $\mathbf{x}$ in the original data space. Also, we denote the discriminative model by $D(\cdot;\theta_d)$, which is to classify the samples from $p_{\mathcal{D}}(\mathbf{x})$ and $p_g(\mathbf{x})$ into correct and fake categories, respectively. The minimax optimization of GANs can be

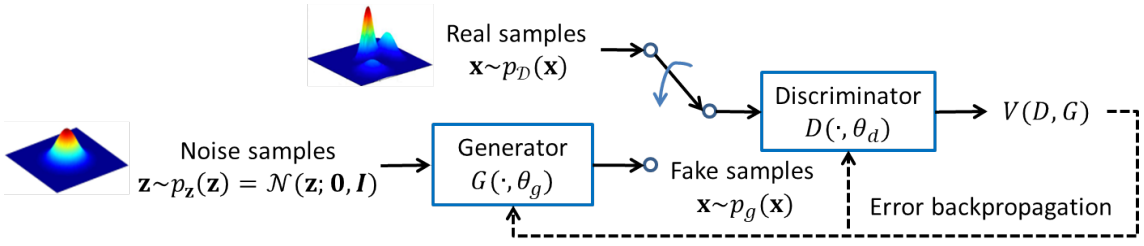


Figure 3.1: Schematic of the standard GAN. The black solid arrows illustrate the forward signal flows, the black dotted arrows denote the error backpropagation flows, and the blue arrow indicates the switch of signal flow from the real data to the generated data.

expressed as

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))], \quad (3.1)$$

where $V(D,G)$ is called value function (or GAN objective function) [54]. In fact, $V(D,G)$ is the negative of the binary cross-entropy loss function with $D(\mathbf{x})$ corresponding to the correct label 1 and $D(G(\mathbf{z}))$ corresponding to label 0. In practice, we optimize D (with G fixed) for k ($k \geq 1$) steps before updating G once (with D fixed) and iterate this process until reaching an equilibrium condition where D cannot discriminate between the real samples and the fake samples.

To analyze the equilibrium condition, instead of using an iterative training procedure, we first optimize D to completion with a fixed G. The optimal D is obtained as follows [54]:

$$D_G^*(\mathbf{x}) = \frac{p_{\mathcal{D}}(\mathbf{x})}{p_{\mathcal{D}}(\mathbf{x}) + p_g(\mathbf{x})}. \quad (3.2)$$

Under the optimal D_G^*, the inner term of (3.1) becomes

$$
\begin{aligned}
V(G,D) &= \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log D_G^*(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D_G^*(G(\mathbf{z}))\right)\right] \\
&= \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log D_G^*(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\log\left(1 - D_G^*(\mathbf{x})\right)\right] \\
&= \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log \frac{p_{\mathcal{D}}(\mathbf{x})}{p_{\mathcal{D}}(\mathbf{x}) + p_g(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\log \frac{p_g(\mathbf{x})}{p_{\mathcal{D}}(\mathbf{x}) + p_g(\mathbf{x})}\right] \\
&= \mathrm{KL}\left(p_{\mathcal{D}}(\mathbf{x}) \middle\| \frac{p_{\mathcal{D}}(\mathbf{x}) + p_g(\mathbf{x})}{2}\right) + \mathrm{KL}\left(p_g(\mathbf{x}) \middle\| \frac{p_{\mathcal{D}}(\mathbf{x}) + p_g(\mathbf{x})}{2}\right) - \log 4 \\
&= 2\mathrm{JS}(p_{\mathcal{D}}(\mathbf{x}) \| p_g(\mathbf{x})) - \log 4, \qquad (3.3)
\end{aligned}
$$

where $\mathrm{KL}(\cdot\|\cdot)$ and $\mathrm{JS}(\cdot\|\cdot) \geq 0$ are the Kullback–Leibler (KL) divergence and Jensen–Shannon (JS) divergence between two data distributions, respectively. From (3.3), we observe that when $p_{\mathcal{D}}(\mathbf{x}) = p_g(\mathbf{x})$, $V(G,D)$ achieves the minimum value of $-\log 4$. Therefore, when the training process reaches the equilibrium, the discriminator is not able to distinguish the real data from the generated data, i.e., $D_G^*(\mathbf{x}) = 1/2$.

Note that (3.3) indicates that optimizing D to completion actually defines the JS divergence between the real data distribution and the fake data distribution. Thus, adversarial learning through a minimax game is effectively minimizing the JS divergence between two distributions. When the equilibrium condition is attained, the JS divergence term becomes zero and these two distributions are equal. Because JS divergence is a distance measure between two distributions, adversarial learning can be used to minimize their distance. This is useful in domain adaptation where the objective is to minimize the source data distribution and the target data distribution. Following this strategy, domain adversarial training [51] was proposed and has been widely used for alleviating domain mismatch [31, 52, 83, 84]. In Chapter 4, the author adopts the same idea and applies an adversarial autoencoder (AAE) [85] in the latent space to minimize the distance between the aggregated posterior and the prior probability of the latent variables outputted from the encoder of the AAE–VDANN.

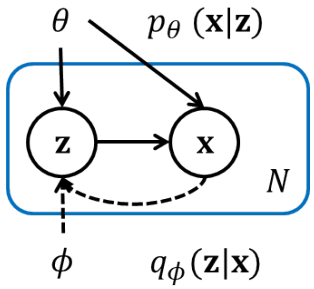


Figure 3.2: Graphical illustration of the probabilistic model involved in the autoencoding variational Bayes (AEVB) approach [2]. The solid arrows denote the generative modeling process with parameters θ, i.e., $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, and the dashed arrows represent the approximate inference process with a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ parameterized by ϕ, which is an approximate to the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. ϕ and θ are jointly learned by AEVB.

3.2 Variational Autoencoders

Variational autoencoders (VAEs) are another category of generative models that use the variational Bayes (VB) method to perform density estimation. The autoencoding VB (AEVB) approach provides an efficient VB implementation to approximate the intractable posterior [2].

Suppose we have a training set $\mathcal{X}$ with true data distribution $p_\mathcal{D}(\mathbf{x})$ ($\mathbf{x} \in \mathcal{X}$) and its underlying generation is determined by a latent variable set $\mathcal{Z}$. Figure 3.2 presents a graphical illustration of the latent variable model used in the AEVB framework, where $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational posterior to approximate the intractable true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The objective of AEVB is to jointly learn the variational parameters ϕ and the generative model parameters θ.

VAE is a specific implementation of the AEVB approach where DNNs implement the posterior inference model and the generative model. As shown in Figure 3.3, a VAE consists of an encoder (recognition model) and a decoder (generative model), whose parameters are denoted as ϕ and θ, respectively. Note that both the encoder and the decoder are probabilistic models because given a sample $\mathbf{x}$ (or $\mathbf{z}$), the output

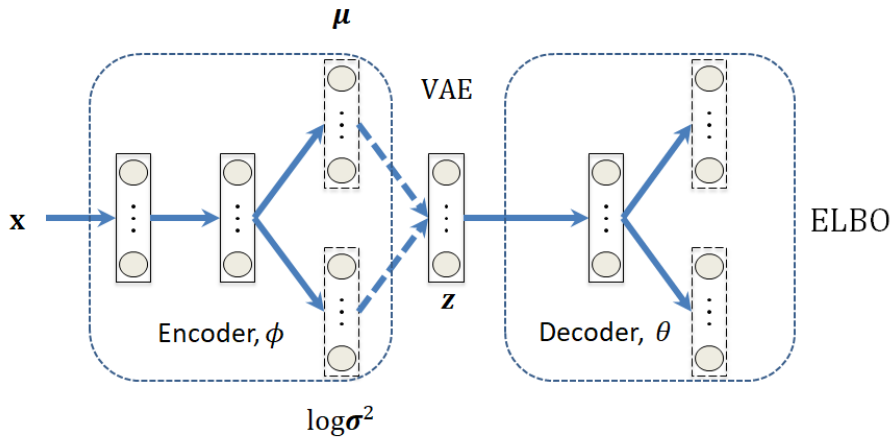


Figure 3.3: Schematic of a VAE. The solid and dashed arrows represent network connections and stochastic sampling, respectively.

of the encoder (or decoder) is a distribution (we use a Gaussian distribution in Figure 3.3). A VAE can be optimized by maximizing the evidence lower bound (ELBO) of log-likelihood [2, 59]:

$$\max_{\phi,\theta} \text{ELBO}(\phi,\theta) = \max_{\phi,\theta} \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} \left[-\text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]\right], \quad (3.4)$$

where $p_\theta(\mathbf{z})$ is the prior of $\mathbf{z}$ which is generally a standard Gaussian $\mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$.

Because the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is effectively a probabilistic model (with a sampling operation), the gradient of the ELBO w.r.t. ϕ cannot be directly computed. A naive solution is to use the Monte Carlo gradient estimator. However, due to the high variance of this gradient estimator, it cannot be applied in practice. To estimate the gradient w.r.t. ϕ, we can use the reparameterization trick [2].

Given a latent variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, we reparameterize $\mathbf{z}$ using a differentiable transformation function $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ with an auxiliary random variable $\boldsymbol{\epsilon}$ as follows:

$$\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}). \quad (3.5)$$

Following this reparameterization, for some function $f(\mathbf{z})$, we can estimate its expectation w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$ as

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f\left(g_{\boldsymbol{\phi}}\left(\boldsymbol{\epsilon}, \mathbf{x}\right)\right)\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f\left(g_\phi\left(\boldsymbol{\epsilon}_l, \mathbf{x}\right)\right), \tag{3.6}$$

where $\boldsymbol{\epsilon}_l \sim p(\boldsymbol{\epsilon})$ and L is number of latent samples corresponding to a single $\mathbf{x}$. Note that (3.6) can provide a gradient estimate of $f(\mathbf{z})$ w.r.t. $q_\phi\left(\mathbf{z} \mid \mathbf{x}\right)$ with much lower variance than that using a simple Monte Carlo gradient estimator. Finally, the ELBO can be estimated as

$$\text{ELBO}(\phi, \theta) \simeq \frac{1}{N}\sum_{i=1}^{N}\left[-\text{ KL}\left(q_\phi(\mathbf{z}|\mathbf{x}_i)\|p_\theta(\mathbf{z})\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(\mathbf{x}_i|\mathbf{z}_{il}\right)\right], \tag{3.7}$$

where $\mathbf{z}_{il} = g_\phi\left(\boldsymbol{\epsilon}_l, \mathbf{x}_i\right)$ and N is the number of training samples.

For the VAE used in this thesis, as shown in Figure 3.3, we use a Gaussian distribution for $q_\phi(\mathbf{z}|\mathbf{x})$ with mean vector $\boldsymbol{\mu}$ and a diagonal covariance matrix $\text{diag}\left(\boldsymbol{\sigma}^2\right)$, i.e., $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}, \text{diag}\left(\boldsymbol{\sigma}^2\right)\right)$.[1] According to (3.5), we obtain the l-th latent sample as $\mathbf{z}_l = g_\phi\left(\boldsymbol{\epsilon}_l, \mathbf{x}\right) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_l$, where $\boldsymbol{\epsilon}_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ is the Hadamard product. For the Gaussian variational posterior case, we can obtain an analytical expression for the KL divergence term in (3.4) and the final estimate of ELBO becomes

$$\text{ELBO}(\phi, \theta) \simeq \frac{1}{NL}\sum_{i=1}^{N}\sum_{l=1}^{L}\log p_\theta\left(\mathbf{x}_i|\mathbf{z}_{il}\right) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{J}\left[1 + \log\sigma_{ij}^2 - \mu_{ij}^2 - \sigma_{ij}^2\right], \tag{3.8}$$

where J is the dimension of the latent variable $\mathbf{z}$. As suggested in [2], we may use $L = 1$ in (3.8) during the sampling operation.

We will use (3.8) in Section 4.3 for variational domain adversarial learning. Nevertheless, we should also note that there are other decompositions or modifications of the ELBO for various interpretations of the VAE. For example, the ELBO has been

[1] Both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ depend on $\mathbf{x}$ and ϕ.

widely adapted for disentangled factor learning [86–89]. In [59], the ELBO in (3.4) was reformulated to provide intuition for improving the training of VAEs:

$$\begin{aligned}\text{ELBO}(\phi,\theta) &= \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}\left[-\text{ KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]\right] \\ &\propto -\text{ KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right) - \mathbb{E}_{q_\phi(\mathbf{z})}\left[\text{KL}\left(q_\phi(\mathbf{x}|\mathbf{z})\|p_\theta(\mathbf{x}|\mathbf{z})\right)\right], \end{aligned} \tag{3.9}$$

where $q_\phi(\mathbf{z})$ is the aggregated posterior [85,90]: $q_\phi(\mathbf{z}) = \int_{\mathbf{x}} p_{\mathcal{D}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{x}$. Note that $q_\phi(\mathbf{z})$ requires an aggregation over the entire training set $\mathcal{X}$, it cannot be computed exactly. In practice, we can approximate $q_\phi(\mathbf{z})$ by a Monte Carlo estimate [77,91]. (3.9) is useful in the interpretation of information-maximized VAE (InfoVAE) introduced in Section 4.4.1.

Besides being used as a generative modeling method, the VAE can be used as an inference model in which the latent variable $\mathbf{z}$'s are extracted as the output representations. Also, we note that the KL divergence term in (3.4) can be used as a regularizer on $\mathbf{z}$'s so that the latent representations can retain some desirable properties. For example, in Chapter 4, the author uses a VAE to perform Gaussian regularization on the learned speaker embeddings by minimizing the KL divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and a Gaussian distribution $p_\theta(\mathbf{z})$ to make the embeddings more Gaussian. This property is beneficial when a Gaussian PLDA backend is used for scoring.

Chapter 4

VARIATIONAL DOMAIN ADVERSARIAL LEARNING FOR SPEAKER VERIFICATION

4.1 Introduction

One challenge to SV is domain mismatch. Domain mismatch refers to the problem where the distribution of the (source-domain) training data differs from that of the (target-domain) test data due to different languages, channels, noises, etc. Domain mismatch can cause severe performance degradation, and domain adaptation (DA) is usually adopted to alleviate this problem. In this chapter, we focus on the scenario where only a small amount of unlabeled target-domain data are available besides large-scale source-domain data. This situation requires unsupervised DA.

There are many attempts to perform unsupervised DA in SV and these methods can be roughly divided into two categories: model-level DA and embedding-level DA. The first category directly adapts the covariance matrices of PLDA models to make the PLDA parameters better match the target distribution, e.g., CORAL+ [92] and Kaldi's PLDA adaptation.[1] The second category aims to learn a domain-invariant space so that the target-domain data match the source-domain data in the transformed space. Traditional methods in this category include inter-dataset variability compensation [93], dataset-invariant covariance normalization [94], correlation alignment (CORAL) [95], and feature-Distribution Adaptor [96]. More advanced methods are based on DNNs, e.g., autoencoder-based DA [97], maximum mean discrepancy (MMD) [98] based DA [30, 99, 100], and domain adversarial training (DAT) [51] based

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

DA [83, 84, 101, 102]. In particular, DAT has shown promising performance in the Domain Adaptation Challenge 2013 [84].

Another challenge to SV is that the speaker embeddings are required to follow a Gaussian distribution if the standard PLDA backend is used for scoring. However, there is no guarantee that the embeddings are Gaussian in practice, which leads to poor performance when Gaussian PLDA (G-PLDA) backends are applied. Heavy-tailed PLDA (HT-PLDA) [1, 49] can be used to address this problem but it is very computationally expensive. Length normalization [50] is another feasible strategy but it is only a sub-optimal compromise. Yet we may directly regularize the speaker embeddings to be Gaussian while training the embedding network. Such works include Gaussian-constrained training [103] and variational autoencoders (VAEs) based regularization [104, 105].

In this chapter, the author proposes a variational domain adversarial learning framework to jointly address the domain mismatch challenge and the Gaussianity requirement of the G-PLDA model. Specifically, a variational adversarial neural network (VDANN) and an information-maximized VDANN (InfoVDANN) are proposed in Section 4.3 and Section 4.4, respectively.

4.2 Domain Adversarial Neural Network

Domain adversarial neural network (DANNs) [51] aim to learn a domain-invariant latent space through adversarial training for unsupervised DA. DANNs have been widely applied to alleviate domain mismatch. In [84], DAT is used to generate speaker discriminative and domain-invariant representations in the Domain Adaptation Challenge 2013. Also, an end-to-end DANN was implemented in [83] to produce embeddings that are invariant to languages.

As shown in Figure 4.1, a standard DANN consists of three subnetworks: an encoder E, a label predictor C, and a domain discriminator D. Their parameters are

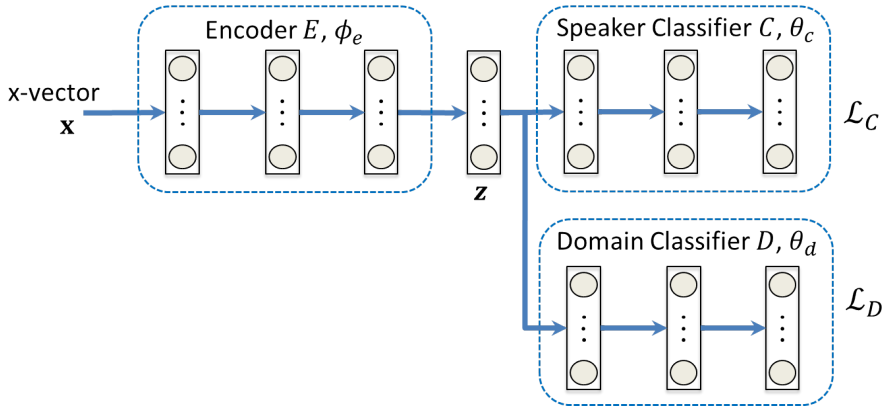


Figure 4.1: Schematic of a DANN. After training, the transformed features are extracted from the $\mathbf{z}$ nodes.

denoted by ϕ_e, θ_c, and θ_d, respectively. Given a source domain set $\mathcal{X}^S = \{\mathbf{x}_1^S, \ldots, \mathbf{x}_{N_S}^S\}$ and a target domain set $\mathcal{X}^T = \{\mathbf{x}_1^T, \ldots, \mathbf{x}_{N_T}^T\}$, where N_S and N_T are the number of samples in $\mathcal{X}^S$ and $\mathcal{X}^T$, respectively. Denote $\mathbf{y} = \{y_i\}$ corresponding to $\mathcal{X}^S$ as the one-hot speaker labels and $\mathbf{d} = \{d_i\}$ corresponding to $\{\mathcal{X}^S, \mathcal{X}^T\}$ as the domain labels, respectively. Define the loss function of DANN as

$$\begin{aligned}\mathcal{L}(\theta_c, \theta_d, \phi_e) &= \mathcal{L}_C(\theta_c, \phi_e) - \alpha \mathcal{L}_D(\theta_d, \phi_e) \\ &= \sum_{\mathbf{x}_i \in \mathcal{X}^S} \mathcal{L}_C\left(C(E(\mathbf{x}_i)), y_i\right) - \alpha \sum_{\mathbf{x}_i \in \{\mathcal{X}^S, \mathcal{X}^T\}} \mathcal{L}_D\left(D(E(\mathbf{x}_i), d_i)\right), \end{aligned} \tag{4.1}$$

where $\mathcal{L}_C(\cdot)$ and $\mathcal{L}_D(\cdot)$ are the loss functions for C and D, respectively. α weights the domain discrimination loss during training. The minimax optimization in DANN is denoted as follows

$$\min_{\theta_c, \phi_e} \max_{\theta_d} \mathcal{L}(\theta_c, \theta_d, \phi_e). \tag{4.2}$$

After successful training, the features encoded by the extractor are not only task discriminative but also domain-invariant. This feature extractor is used to calculate embeddings for later tasks.

4.3 Variational Domain Adversarial Neural Network

Although prior findings suggest DAT's superiority to conventional DA [83, 84], there is no guarantee that the learned features follow a Gaussian distribution, which is essential for the G-PLDA backend. To alleviate this limitation, the author incorporates a VAE into DAT so that the learned features are not only speaker discriminative and domain-variant but also Gaussian distributed. The resulting network is referred to as variational DANN (VDANN).

As explained in Section 3.2, one desirable property of VAE is that the first term on the right-hand side of (3.4) can be considered as a regularizer that constrains the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the desired prior $p_\theta(\mathbf{z})$. Therefore, if we constrain $p_\theta(\mathbf{z})$ to be a multivariate Gaussian distribution, the encoder is encouraged to produce Gaussian latent vectors, which is amenable to PLDA modeling.

Suppose we have a training set $\mathcal{X} = \{\mathcal{X}^r\}_{r=1}^R$ comprising samples from R domains, where $\mathcal{X}^r = \{\mathbf{x}_1^r, \ldots, \mathbf{x}_{N_r}^r\}$ contains N_r samples from the r-th domain. Also, we denote $\mathbf{y}$ and $\mathbf{d}$ as the one-hot speaker and domain labels, respectively. We define the Gaussian VAE loss as the negative of the ELBO in (3.8):

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) \simeq -\sum_{r=1}^{R}\sum_{i=1}^{N_r}\left\{\frac{1}{2}\sum_{j=1}^{J}\left[1 + \log\left(\sigma_{ij}^r\right)^2 - \left(\mu_{ij}^r\right)^2 - \left(\sigma_{ij}^r\right)^2\right] + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(\mathbf{x}_i^r|\mathbf{z}_{il}\right)\right\}, \tag{4.3}$$

where J is the dimension of $\mathbf{z}$ and L denotes the number of latent samples. In practice, we set $L = 1$.

As shown in Figure 4.2, the proposed VDANN consists of a speaker predictor C, a domain classifier D and a VAE. The latter contains an encoder E and a decoder G. The network parameters are denoted as θ_c, θ_d, ϕ_e and θ_g, respectively. Through adversarial training, the VDANN learns a domain-invariant space across multiple

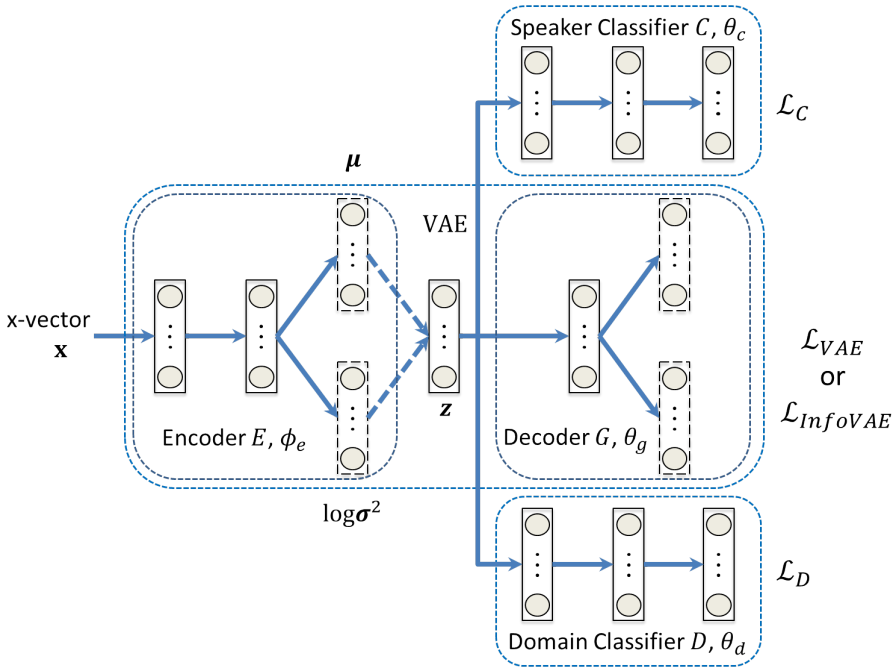


Figure 4.2: Schematic of a VDANN (and an InfoVDANN). The solid and dashed arrows represent network connections and stochastic sampling, respectively. For VDANN, $\mathcal{L}_{\text{VAE}}(\phi_e, \theta_g)$ is used, whereas $\mathcal{L}_{\text{InfoVAE}}(\phi_e, \theta_g)$ is used for InfoVDANN.

domains. Specifically, applying adversarial training on E while minimizing the speaker classification loss with respect to ϕ_e will make E produce a domain-invariant but speaker discriminative representation through the nodes denoted by $\mathbf{z}$ in Figure 4.2.

To train this network, we define the loss of VDANN as:

$$\mathcal{L}_{\text{VDANN}}(\theta_c, \theta_d, \phi_e, \theta_g) = \mathcal{L}_C(\theta_c, \phi_e) - \alpha\mathcal{L}_D(\theta_d, \phi_e) + \beta\mathcal{L}_{\text{VAE}}(\phi_e, \theta_g), \tag{4.4}$$

where

$$\mathcal{L}_C(\theta_c, \phi_e) = \frac{1}{N}\sum_{r=1}^{R}\sum_{i=1}^{N_r}\left\{-\sum_{k=1}^{K} y_{ik}^r \log C\left(E\left(\mathbf{x}_i^r\right)\right)_k\right\}, \tag{4.5}$$

$$\mathcal{L}_D(\theta_d, \phi_e) = \frac{1}{N}\sum_{r=1}^{R}\sum_{i=1}^{N_r}\left\{-d_i^r \log D\left(E\left(\mathbf{x}_i^r\right)\right)_r\right\}, \tag{4.6}$$

and $\mathcal{L}_{\text{VAE}}$ takes similar form as in (4.3) except that the parameters of the encoder and decoder change to ϕ_e and θ_g, respectively. The subscript k in the categorical cross-entropy loss of the speaker classifier C in (4.5) indexes the speakers and represents the k-th output of the classifier. The hyperparameters α and β control the contribution of individual losses that shape the features produced by E.

During training, for each mini-batch, we first optimize D by minimizing the domain classification loss. Parameters of D are then fixed while training the remaining parts of the VDANN. To incorporate speaker information into E, speaker prediction loss is minimized; simultaneously we maximize the domain classification loss so that we can learn a domain-invariant space for E. Moreover, the VAE loss is minimized to regularize the learned features to be Gaussian. To summarize, we optimize the VDANN as follows:

$$\min_{\theta_c,\phi_e,\theta_g} \max_{\theta_d} \mathcal{L}_{\text{VDANN}}(\theta_c, \theta_d, \phi_e, \theta_g). \tag{4.7}$$

(4.7) can be divided into the following minimax procedure:

$$\hat{\theta}_d = \operatorname*{argmax}_{\theta_d} \mathcal{L}_{\text{VDANN}}(\hat{\theta}_c, \theta_d, \hat{\phi}_e, \hat{\theta}_g), \tag{4.8}$$

$$\left(\hat{\theta}_c, \hat{\phi}_e, \hat{\theta}_g\right) = \operatorname*{argmin}_{\theta_c,\phi_e,\theta_g} \mathcal{L}_{\text{VDANN}}(\theta_c, \hat{\theta}_d, \phi_e, \theta_g), \tag{4.9}$$

where symbols with a hat (e.g., $\hat{\theta}_c$) on the right-hand side of (4.8) and (4.9) mean that they are fixed when optimizing the target parameters. After training, we may extract the transformed features from the $\mathbf{z}$ nodes of the encoder E. Since the variational approximate posterior is regularized to follow a Gaussian distribution, features produced from the encoder will also likely to be Gaussian.

SRE16 development set was used as the cross-validation set to determine the hyperparameters α and β in (4.4) and we used simple grid search to tune these hyperparameters. Note that the DANN in Section 4.2 is a special case of VDANN, i.e., if we set $R = 2$ and $\beta = 0$ in (4.4), the VDANN loss reduces to the DANN loss

in (4.1).

4.4 Information-Maximized Variational Domain Adversarial Neural Network

A potential problem of VDANN is that training the VAE by maximizing the ELBO of log-likelihood could cause failure in learning informative latent representations [55, 59, 90, 106]. In particular, if the decoder is flexible enough, a VAE can produce noninformative latent vectors independent of the inputs. This problem is referred to as posterior collapse [55, 57, 58], which is undesirable for learning meaningful representations.

Several methods have been proposed to address the posterior collapse problem, e.g., applying the KL cost annealing [107], using a variational mixture of posteriors [108], reducing the amortization gap [57], skipping the connections in the decoder [58], aggressively training the encoder [55], applying a self-attention mechanism [109], etc. In [59], the InfoVAE, a variant of VAE with the information-maximized latent representation, was proposed to address posterior collapse. The idea is to explicitly enhance the dependence of the latent vectors on the inputs by maximizing the mutual information (MI) between the latent variables and the inputs.

In this section, the author adopts the idea of InfoVAE and extends the VDANN for unsupervised DA. With the InfoVAE, the learned features can sufficiently reflect the meaningful information from the inputs, while simultaneously retain the benefit of VDANN to produce Gaussian distributed features. The author refers to the resulting information-maximized VDANN as InfoVDANN.

4.4.1 Information-Maximized VAE

Maximizing the ELBO directly can lead to some problems. First, according to (3.9), if the dimension of $\mathbf{x}$ is much higher than that of $\mathbf{z}$, maximization of the ELBO will

emphasize the second term, i.e., data reconstruction. This bias in emphasis can easily cause overfitting. Second, as mentioned earlier, if the decoder is flexible enough, VAE training will ignore the information between the latent features and the inputs, leading to noninformative representation.

In [59], a new objective function was proposed based on (3.9) to address the problems in VAEs. The objective includes 1) adding a scalar to increase the contribution of $\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ and to counteract the dimension imbalance between $\mathcal{X}$ and $\mathcal{Z}$ and 2) incorporating an MI term that explicitly retains high mutual information between $\mathbf{x}$ and $\mathbf{z}$. The resulting model is called InfoVAE whose objective is expressed as follows:

$$\begin{aligned}
\mathrm{ELBO}_{\mathrm{InfoVAE}} &= -\lambda \mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right) + \eta I_q(\mathbf{x};\mathbf{z}) - \mathbb{E}_{q_\phi(\mathbf{z})}\left[\mathrm{KL}\left(q_\phi(\mathbf{x}|\mathbf{z})\|p_\theta(\mathbf{x}|\mathbf{z})\right)\right] \\
&= -\lambda \mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right) \\
&\quad + \eta\left\{\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\left[\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)\right] - \mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)\right\} \\
&\quad + \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\left[-\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]\right] \\
&\quad + \mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right) - \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\left[\log p_\mathcal{D}(\mathbf{x})\right] \\
&\propto \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - (1-\eta)\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\left[\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)\right] \\
&\quad - (\lambda - 1 + \eta)\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right), \qquad (4.10)
\end{aligned}$$

where $I_q(\mathbf{x};\mathbf{z})$ is the MI between $\mathbf{x}$ and $\mathbf{z}$ under $q_\phi(\mathbf{x},\mathbf{z})$. The hyperparameter λ compensates for the dimension imbalance between $\mathbf{x}$ and $\mathbf{z}$, so that the variational inference and data reconstruction can be balanced. η signifies the importance of maintaining high mutual information between the original and latent vectors. We do not explicitly use $I_q(\mathbf{x};\mathbf{z})$ in the final expression because the MI term is difficult to compute directly, especially in high-dimensional spaces. Note that we can further generalize the $\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ in (4.10) to broader divergence families for efficient optimization, e.g., we may use MMD [98] as a divergence measure or introduce a

discriminator and apply adversarial training to distinguish samples drawn from $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$, similar to the adversarial autoencoder (AAE) [85].

4.4.2 Information-Maximized VDANN

To improve the training of a VDANN, the author proposes an InfoVDANN by incorporating an InfoVAE into the DANN [51] to learn features that can sufficiently characterize the latent information from the inputs while simultaneously leverage the benefit of the VDANN.

The InfoVDANN has a similar structure as the VDANN, as shown in Figure 4.2. The only difference between the InfoVDANN and VDANN is the use of the variational loss function: the InfoVAE loss function is used in InfoVDANNs, whereas the VDANN applies a standard VAE objective.

We follow the same mathematical settings as in VDANN and define the InfoVAE loss function as the negative of (4.10)

$$\begin{aligned}\mathcal{L}_{\text{InfoVAE}}\left(\phi_e, \theta_g\right) = -\frac{1}{N}\sum_{r=1}^{R}\sum_{i=1}^{N_r}\Bigg\{ & \frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(\mathbf{x}_i^r|\mathbf{z}_{il}^r\right) \\ & - \frac{1-\eta}{2}\sum_{j=1}^{J}\left[\left(\mu_{ij}^r\right)^2 + \left(\sigma_{ij}^r\right)^2 - 1 - \log\left(\sigma_{ij}^r\right)^2\right] \\ & - (\lambda - 1 + \eta)\frac{1}{L}\sum_{l=1}^{L}\left[\log q_\phi(\mathbf{z}_{il}^r) - \log p(\mathbf{z}_{il}^r)\right]\Bigg\}, \end{aligned} \tag{4.11}$$

where J is the dimension of $\mathbf{z}$ and L denotes the number of sampled latent variables for a given $\mathbf{x}$. The hyperparameters λ and η are consistent with those in (4.10). The first term on the right-hand side of (4.11) is the data reconstruction error, whereas the second term is the analytical expression of $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$. The third term is the Monte Carlo estimate of $\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ in (4.10), i.e.,

$$\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right) = \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z}) - \log p_\theta(\mathbf{z})\right]. \tag{4.12}$$

To train the InfoVDANN, we define the loss of InfoVDANN as

$$\mathcal{L}_{\text{InfoVDANN}}(\theta_c, \theta_d, \phi_e, \theta_g) = \mathcal{L}_C(\theta_c, \phi_e) - \alpha\mathcal{L}_D(\theta_d, \phi_e) + \beta\mathcal{L}_{\text{InfoVAE}}(\phi_e, \theta_g), \tag{4.13}$$

where $\mathcal{L}_C(\theta_c, \phi_e)$ and $\mathcal{L}_D(\theta_d, \phi_e)$ have been defined in (4.5) and (4.6), respectively. The training procedure of InfoVDANN is similar to that of VDANN in (4.7) except that $\mathcal{L}_{\text{InfoVDANN}}(\theta_c, \theta_d, \phi_e, \theta_g)$ instead of $\mathcal{L}_{\text{VDANN}}(\theta_c, \theta_d, \phi_e, \theta_g)$ is applied.

4.4.3 MMD–VDANN and AAE–VDANN

To compute the term $\text{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ in $\mathcal{L}_{\text{InfoVAE}}$, we first need to obtain samples $\mathbf{z}$'s from $q_\phi(\mathbf{z})$. This can be easily addressed by ancestral sampling [55, 58, 106], that is, we first uniformly sample $\mathbf{x}$'s from the training data, and then draw samples $\mathbf{z}$'s from $q_\phi(\mathbf{z}|\mathbf{x})$. Take a mini-batch of B training samples as an example, this can be denoted as follows:

$$\mathbf{z} \sim q_\phi\left(\mathbf{z}|\mathbf{x}_b\right), \quad b \sim \text{ Uniform } (1, \ldots, N). \tag{4.14}$$

After getting samples from $q_\phi(\mathbf{z}|\mathbf{x}_b)$, the aggregate posterior can be calculated through the Monte Carlo estimate:

$$q_\phi(\mathbf{z}_s) \approx \frac{1}{B}\sum_{b=1}^{B} q_\phi\left(\mathbf{z}_s|\mathbf{x}_b\right), \quad s = 1, \ldots, B. \tag{4.15}$$

If we use these B $\mathbf{z}_s$'s to estimate $\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right]$, i.e., the first term of $\text{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ in (4.12), we have

$$\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right] \approx \frac{1}{B}\sum_{s=1}^{B}\left[\log\frac{1}{B}\sum_{b=1}^{B} q_\phi\left(\mathbf{z}_s|\mathbf{x}_b\right)\right]. \tag{4.16}$$

Since the estimate in (4.16) is biased and can only give a lower bound on the true expectation [55], the estimate of $\mathrm{KL}\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ will also be biased. This means that we need to set the mini-batch size B to a large value during training to make the estimate of the KL divergence reliable, which would lead to heavy computation.

Although there are other methods to estimate $q_\phi(\mathbf{z})$ and $\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right]$ [88, 89], these methods are restricted to using KL divergence as the "distance" between $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$. Inspired by the work in [59, 87], we generalize the KL divergence to other probability distance metrics. Setting $L = 1$, the resulting objective of the InfoVAE is then reformulated as

$$\begin{aligned}\hat{\mathcal{L}}_{\text{InfoVAE}}\left(\theta_g, \phi_e\right) = -\frac{1}{N}\sum_{r=1}^{R}\sum_{i=1}^{N_r}\Bigg\{&\log p_\theta\left(\mathbf{x}_i^r|\mathbf{z}_i^r\right)\\ &-\frac{1-\eta}{2}\sum_{j=1}^{J}\left[\left(\mu_{ij}^r\right)^2+\left(\sigma_{ij}^r\right)^2-1-\log\left(\sigma_{ij}^r\right)^2\right]\Bigg\}\\ &+(\lambda-1+\eta)\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right),\end{aligned} \tag{4.17}$$

where $\mathrm{D}_g(\cdot\|\cdot)$ denotes a generalized distance metric.

If we apply MMD [98] as the distance metric in (4.17), the resulting InfoVDANN is called MMD–VDANN. MMD characterizes the distance between two distributions as the Euclidean distance in the Hilbert space, which can be efficiently computed by the kernel trick. Given a suitable kernel, MMD can match up to infinite moments of their distributions. An unbiased empirical estimate of MMD between datasets $\mathcal{X}$ and $\mathcal{Y}$ is given by

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{X},\mathcal{Y}) = &\frac{1}{N(N-1)}\sum_{n=1}^{N}\sum_{n'\neq n}^{N}k\left(\mathbf{x}_n,\mathbf{x}_{n'}\right)+\frac{1}{N'(N'-1)}\sum_{n=1}^{N'}\sum_{n'\neq n}^{N'}k\left(\mathbf{y}_n,\mathbf{y}_{n'}\right)\\ &-\frac{2}{NN'}\sum_{n=1}^{N}\sum_{n'=1}^{N'}k\left(\mathbf{x}_n,\mathbf{y}_{n'}\right),\end{aligned} \tag{4.18}$$

where N and $N^{'}$ denote the number of samples in $\mathcal{X}$ and $\mathcal{Y}$, respectively, and $k\left(\cdot,\cdot\right)$ represents a kernel.

Alternatively, we may use adversarial learning to minimize the distance between two distributions in the latent space as in AAEs [85]. This can be fulfilled by introducing a discriminator to distinguish the samples drawn from $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$. The author calls the InfoVDANN that implements the minimization of $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ by adversarial learning as AAE–VDANN.

The optimization of MMD–VDANN and AAE–VDANN is the same as in (4.7), except that $\mathcal{L}_{\mathrm{VAE}}\left(\phi_e,\theta_g\right)$ is replaced by $\hat{\mathcal{L}}_{\mathrm{InfoVAE}}\left(\theta_g,\phi_e\right)$ (see (4.17)) for the $\mathcal{L}_{\mathrm{InfoVDANN}}$. Take the MMD–VDANN as an example, we use MMD between $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$ as the $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ term in (4.17). It is straightforward to implement this by replacing $\mathbf{x}$'s and $\mathbf{y}$'s in (4.18) with the samples drawn from $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$, respectively. Also, we set both N and $N^{'}$ to be the mini-batch size B during training. The latent samples $\mathbf{z}$'s from $q_\phi(\mathbf{z})$ can be drawn according to (4.14).

MMD–VDANN and AAE–VDANN are proposed as two specialized variants of the InfoVDANN to leverage MI. The difference between MMD–VDANN and AAE–VDANN is that MMD–VDANN minimizes the MMD between the aggregated posterior $q_\phi(\mathbf{z})$ and the prior $p_\theta(\mathbf{z})$ as a proxy to minimize the generalized divergence $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$, whereas AAE–VDANN applies adversarial training to minimize this divergence. The author proposes these two variants to demonstrate that the performance of InfoVDANN is *not* sensitive to how the generalized divergence is minimized. In other words, InfoVDANN would *not* be biased towards a specific divergence between $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$. We will verify this through the experiments in Section 4.6.

4.4.4 Relation to Previous Works

One common characteristic among [92–96] is that they performed DA *without* using DNNs. For DNN-based DA methods, [97] and [30, 99, 100] applied Euclidean distance and MMD to measure the discrepancy between different distributions, respectively.

Different from these distance metrics, [84] and [83] used adversarial training to learn domain-invariant representations. But because there is no constraint on the latent features learned by DANN, the adversarial training may lead to non-Gaussian latent vectors, which would break the assumption of the Gaussian PLDA backend. VDANN overcomes this limitation by regularizing the learned embeddings using a VAE in DAT so that they were Gaussian distributed. Thus, VDANN differs from DANN in this *variational* regularization.

However, a potential limitation of the VDANN is that posterior collapse may occur while training the VAE, leading non-informative speaker representations. The proposed InfoVDANN follows the framework of *variational* DAT in that it performs domain adaptation and Gaussianity regularization simultaneously. However, a major difference with VDANN is that it addresses the posterior collapse problem by explicitly incorporating an MI term in the loss function. Maximizing this term enables the InfoVDANN to preserve more speaker information into the learned embeddings, which is the contribution of this method.

In Section 4.3, we see that DANN is a special case of the VDANN. By removing the sampling operation and the decoder in the VAE, VDANN becomes the DANN. In fact, both the VDANN and DANN are special cases of InfoVDANN: InfoVDANN becomes the VDANN if the MI term is removed from the objective function, and it becomes the DANN if we further remove the sampling operation and the decoder in the VAE.

4.5 Experimental Setup

The performance of various DA methods was evaluated on SRE16 [110] and SRE18-CMN2 [111]. All experiments were based on x-vectors [3] using the x-vector extractor available from the Kaldi repository.[2] Unless otherwise stated, the InfoVDANN men-

[2]http://kaldi-asr.org/models/m3

tioned in the latter sections represents both MMD–VDANN and AAE–VDANN.

4.5.1 Training of InfoVDANN, VDANN and DANN

We used data from four domains as shown in Table 4.1 to train the InfoVDANN, VDANN, and DANN. Each dataset corresponds to a domain. To briefly summarize, SRE04–10 mainly consist of conversational telephone speech in English. VoxCeleb1 was a wideband corpus extracted from the YouTube videos spanning a wide range of ethnicities with real-world noises. SwitchBoard-2 was an English corpus of two-sided telephone conversations collected in the 90's. Similar to the VoxCeleb1 dataset, SITW was a collection from the open source media with unconstrained acoustic conditions. In contrast, the SRE16 evaluation set is composed of telephone conversations spoken in Tagalog and Cantonese, while SRE18-CMN2 contains mainly conversational telephone speech in Tunisian Arabic. Although there is overlap in the collection conditions amongst these datasets, basically, they differed from each other in channels, languages, noises, etc. Therefore, there are mismatches between the training data and the test data and mismatches within the training data.

The statistics of the four training sets are shown in Table 4.1. Note that each training set is a subset of the original set. For example, the minimum number of x-vectors per speaker is 30 for both SRE04–10 and VoxCeleb1. SwitchBoard-2 was selected from Phases I–III to ensure that there are at least 20 x-vectors for each speaker, whereas each speaker in SITW has at least 15 x-vectors.

Table 4.1: Statistics of Training Sets

Dataset	**No. of speakers**	**No. of utterances**
SRE04–10	1,796	53,880
VoxCeleb1	1,181	35,430
SwitchBoard-2	268	6,812
SITW	198	3,572

As shown in Figure 4.2, there are four sub-networks in the VDANN and InfoVDANN. The encoder has two hidden layers and each layer has 1,024 nodes. We used ReLU as the activation function in each layer, followed by batch normalization (BN) and dropout. The dimension of the latent space was set to 400. There is only one hidden layer with 2,048 nodes in the decoder. The output layers of both the encoder and decoder are linear. For the speaker classifier, we used a 1024-1024 hidden-layer structure with Leaky ReLU activation functions, and BN and dropout layers were appended after each layer. The output layer has 3,443 nodes with a softmax function, which correspond to 3,443 speakers. The configuration of the domain classifier is similar to that of the speaker classifier except that the number of nodes in the two hidden layers are 128 and 32, respectively. There are four output nodes which correspond to the four domains in Table 4.1. The dropout rate was set to 0.2 for all dropout layers in the network. For the AAE–VDANN, we included an additional latent-variable discriminator to Figure 4.2 to differentiate the samples drawn from $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$. This discriminator has two hidden layers with 128 and 16 nodes, respectively, followed by ReLU activation and BN in each layer. The number of trainable parameters of each network is summarized in Table 4.2.

Table 4.2: Number of trainable parameters of the embedding transformation networks

Network	**No. of parameters**
DANN	7.02 M
VDANN	9.89 M
MMD–VDANN	9.89 M
AAE–VDANN	9.94 M

We used the Adam optimizer to train the InfoVDANN, VDANN and DANN with a learning rate of 1.0×10^{-3}. The mini-batch size was set to 128. MMD was computed using a mixture of seven radial basis functions (RBFs) with width being set to 0.1, 0.2, 0.4, 1.0, 4.0, 16.0, 256.0, respectively for the MMD–VDANN. For the DANN, we

set $\alpha = 0.1$ and $\beta = 0$ in (4.13), whereas for the VDANN, we set $\alpha = 0.1$ and $\beta = 0.1$ with $\eta = 0$ and $\lambda = 1.0$ in (4.17). For the InfoVDANN, we set $\alpha = 0.1$, $\beta = 1.0$, $\eta = 0.2$, and $\lambda = 1.0$. These hyperparameters were determined by simple grid search on the SRE16 development set.

4.5.2 PLDA Training and Scoring

We used the Gaussian PLDA (G-PLDA) backend and the heavy-tailed PLDA (HT-PLDA) [1] for scoring. For SRE16, the baseline G-PLDA and HT-PLDA models were both trained on the augmented SRE04–10 x-vectors. For SRE18, Mixer6 and its augmentation were also added to the training sets. The augmentation step followed the Kaldi's SRE16 recipe. Before G-PLDA training, the x-vectors were centered and projected to a 150 dimensional space by an LDA transformation matrix, followed by whitening and length normalization. The LDA projection matrix was trained on the same dataset as for training the PLDA models. The dimension of the LDA projection was selected according to the EER on the development set. Evidences of why the projection dimension was set to 150 can be found in Appendix C. As for the HT-PLDA training, we used the same setup as that in [1]. For example, the degree of freedom in the heavy-tailed distribution and the dimension of the speaker subspace were set to 2 and 150, respectively. Also, the LDA projection and length normalization were excluded from the preprocessing.

For all the G-PLDA backends, we also applied Kaldi's PLDA adaptation as an *extra* adaptation step. Specifically, SRE16 unlabeled data were used to adapt the PLDA model for SRE16, whereas we used SRE18 unlabeled data for PLDA adaptation for SRE18. While for the HT-PLDA systems, since Kaldi's PLDA adaptation was not compatible with the HT-PLDA model, we used the unsupervised domain adaptation [112] via parameter interpolation as in [1]. The interpolation factor was set to 0.9 for the out-of-domain part.

For the evaluations of InfoVDANN, VDANN, and DANN, we applied the same

processing as the baseline using the transformed x-vectors rather than the raw x-vectors.

4.6 Results and Discussions

4.6.1 SRE Performance

We followed the Kaldi's SRE16 recipe for SRE16/18 evaluations for the G-PLDA backends. For the baseline, the x-vectors were centered, LDA-transformed, whitened, and length-normalized before PLDA scoring. The same preprocessing was applied to the transformed x-vectors for the InfoVDANN, VDANN and DANN. As for the preprocessing of x-vectors for HT-PLDA models, only centering and whitening were applied.

Table 4.3 shows the performance of different systems on SRE16 and SRE18-CMN2. The first part (rows 1–5) and the second part (rows 6–10) are the results applying HT-PLDA and G-PLDA scoring, respectively. For evaluations based on HT-PLDA, without PLDA adaptation, the HT-PLDA backends only outperform the G-PLDA counterparts in minDCF under SRE16-All and SRE16-Tagalog. For SRE18-CMN2 evaluations, the HT-PLDA backends even cannot compete with the G-PLDA models. Besides, we see that the unsupervised DA failed on SRE16-All and SRE16-Tagalog, and it only worked on SRE16-Cantonese and SRE18-CMN2 with a slight performance gain over the unadapted version. In general, there is no consistent observation that InfoVDANN with HT-PLDA backends outperforms the other systems. It seems that although the HT-PLDA model is theoretically capable of addressing the non-Gaussian embeddings, it failed to achieve consistent gains compared with those of the G-PLDA backend (rows 6–10). Maybe the (transformed) x-vectors do not present a strong heavy-tailed characteristic. Or there may be a need for LDA to find more discriminative directions in the speaker subspace before performing HT-PLDA scoring.

The performance based on the G-PLDA backend is shown in the second part (rows

6–10) in Table 4.3. Without Kaldi's PLDA adaptation under SRE16-All, we can observe that although VDANN can reduce domain mismatch in terms of both EER and minDCF, both MMD–VDANN and AAE–VDANN consistently outperform the VDANN. We also performed Kaldi's PLDA adaptation as an *extra* domain adaptation. From columns 4–5 in the second part, we see that Kaldi's PLDA adaptation is still helpful in further reducing domain mismatch. From the improvement due to the PLDA adaptation, we may conclude that adversarial domain adaptation and PLDA adaptation are complementary. The fact that the performance of InfoVDANN is better than that of DANN verifies that imposing the variational regularization on the transformed x-vectors is effective for domain adaptation. The performance of the Cantonese and Tagalog partitions is consistent with that of SRE16-All. These findings suggest both types of InfoVDANNs (MMD–VDANN and AAE–VDANN) benefit DA in extracting additional speaker discriminative information from the training data compared with the VDANN, and that incorporating an MI term in the loss function is effective for reducing domain mismatch.

From the last four columns in the second part of Table 4.3, we obtain similar conclusions for SRE18-CMN2 as in SRE16: maintaining high MI between the latent features and the inputs can feed more speaker information into the learned embeddings, which enhances speaker recognition performance.

As shown in Appendix B, the P-values of the McNemar's tests [5] between both InfoVDANNs and the others are mostly zeros for SRE16-All and SRE18-CMN2. This means that the improvement of both InfoVDANNs over VDANN, DANN and the baseline is statistically significant.

We also present the performance of some state-of-the-art end-to-end systems in the third part (rows 11–14) of Table 4.3. Generally, results show that end-to-end systems outperform the embedding–backend cascaded systems. This is reasonable because the end-to-end systems have greater capacity to learn the domain invariance.

Table 4.3: Performance on SRE16 and SRE18-CMN2. The first part (rows 1–5) and the second part (rows 6–10) show the performance based on the heavy-tailed PLDA (HT-PLDA) and the Gaussian PLDA (G-PLDA) backends, respectively. The third part (rows 11–14) presents the performance of some state-of-the-art end-to-end domain adaptation systems. The DCF refers to the average minDCF at the operating points 0.01 and 0.005.

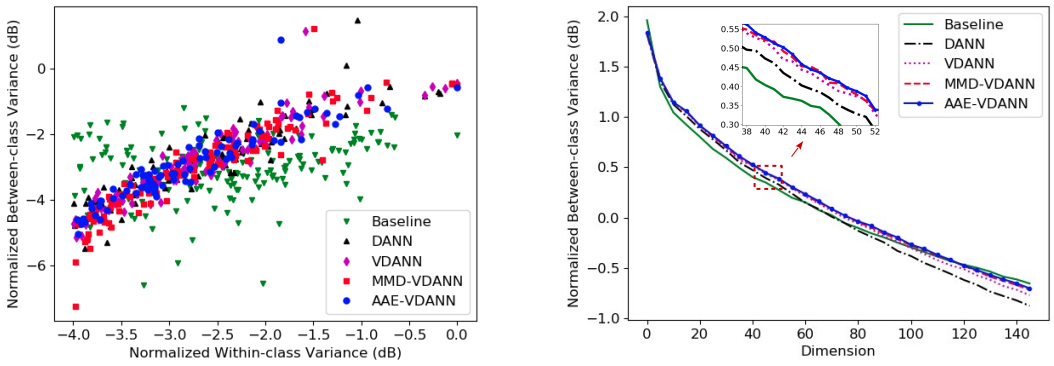
	SRE16-All				SRE16-Cantonese				SRE16-Tagalog				SRE18-CMN2			
	w/o adp		w/ adp		w/o adp		w/ adp		w/o adp		w/ adp		w/o adp		w/ adp	
	EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF
Baseline (HT)	11.48	0.830	12.43	0.871	7.03	0.648	6.43	0.605	**16.36**	0.926	16.84	0.929	**10.42**	0.674	10.21	0.668
DANN (HT)	12.03	0.844	12.23	0.850	6.81	0.576	6.37	0.579	16.97	0.940	17.06	0.946	11.06	0.683	10.72	0.665
VDANN (HT)	11.57	**0.776**	11.68	**0.800**	6.48	0.568	6.15	0.555	16.51	**0.888**	16.63	**0.897**	10.81	0.674	10.41	0.658
MMD–VDANN (HT)	**11.47**	0.797	11.65	0.816	6.24	0.544	6.06	**0.533**	16.40	0.910	16.63	0.905	10.69	**0.655**	10.19	**0.636**
AAE–VDANN (HT)	11.50	0.795	**11.62**	0.820	**6.20**	**0.543**	**6.04**	0.539	16.42	0.907	**16.55**	0.917	10.59	0.657	**10.15**	0.643
Baseline (G)	11.30	0.890	8.27	0.604	7.16	0.579	4.69	0.427	15.60	0.972	11.93	0.751	11.21	0.676	9.60	0.575
DANN (G)	11.64	0.894	8.22	0.604	6.85	0.570	4.33	0.439	16.54	0.973	12.24	0.742	10.77	0.674	9.29	0.574
VDANN (G)	10.92	0.852	8.13	0.587	6.62	0.540	4.54	0.413	15.33	0.963	11.76	0.731	10.29	0.664	9.18	0.572
MMD–VDANN (G)	10.67	0.835	**7.91**	**0.581**	6.41	0.538	4.07	0.402	**15.03**	0.958	**11.59**	**0.723**	9.92	**0.653**	8.95	0.570
AAE–VDANN (G)	**10.61**	**0.832**	**7.91**	0.582	**6.35**	**0.535**	**4.06**	**0.400**	15.08	**0.955**	**11.59**	0.726	**9.91**	0.654	**8.85**	**0.559**
WGAN [83]	13.25	0.899	9.15	0.677	7.39	0.561	–	–	19.12	0.968	–	–	10.35	0.658	9.21	0.602
WGAN+lan+sup [83]	9.59	0.746	**8.00**	0.651	**5.59**	**0.497**	–	–	**13.70**	**0.880**	–	–	8.88	0.619	8.25	0.576
LSGAN [113]	11.74	–	–	–	7.90	–	–	–	15.63	–	–	–	–	–	–	–
Multi-level [30]	**9.03**	**0.585**	8.29	**0.546**	–	–	–	–	–	–	–	–	**8.33**	**0.520**	**8.09**	**0.521**

4.6.2 Speaker Discriminative Features

By explicitly incorporating an MI term in the objective function, InfoVDANN is able to feed *extra* information into the speaker embeddings, making them more discriminative than those of the VDANN and DANN. To investigate the discriminativeness of the InfoVDANN, we plot the between-class variances against the within-class variances using the SRE04–10 dataset (with augmentation). In Figure 4.3(a), the x-axis denotes the logarithm of normalized within-class variances: $\log[(V_w - \min(V_w))/(\max(V_w) - \min(V_w))]$, where V_w is the within-class variance. Therefore, different systems can be compared in the same scale. The scale of the y-axis is $\log[(V_b - \min(V_w))/(\max(V_w) - \min(V_w))]$, where V_b denotes the between-class variance. Each datapoint in Figure 4.3(a) corresponds to a dimension of the embedding vector. We see that InfoVDANN generally has larger between-class variances than the baseline for each within-class variance. This means that the x-vectors transformed by the InfoVDANN are more discriminative compared with the baseline because the speaker clusters become more separated after the transformation. However, it is not clear if InfoVDANN outperforms the VDANN and DANN because their scatters overlap with each other in the plot.

Since LDA was applied before PLDA scoring, we further evaluate the discriminativeness of these systems after LDA projection. The sorted between-class variances after normalization were plotted in Figure 4.3(b). Note that the within-class variances are all normalized to 1. According to Figure 4.3(b), we see that for the first 150 dimensions, InfoVDANN consistently has larger between-class variances than the other systems. This suggests that the embeddings transformed by InfoVDANN are more discriminative after LDA projection, which verifies the advantage of InfoVDANN over the VDANN and DANN.

We also compare the t-SNE [114] plots of the raw x-vectors randomly selected from 20 speakers with those of their transformed ones. The selected x-vectors do not appear



(a) (b)

Figure 4.3: Illustration for (a) between-class variances versus within-class variances before LDA and (b) between-class variances after LDA.

in the training sets. Specifically, for each domain we randomly chose 5 speakers, with each speaker providing at least 20 utterances. The results are illustrated in Figure 4.4. In the figure, each color corresponds to a domain, and each marker represents a speaker whose identity is indicated in the legend.

As shown in Figure 4.4, except for the utterances in SITW, the x-vectors transformed by the InfoVDANN, VDANN and DANN are more speaker discriminative across all the four domains. To be specific, the clusters from 3 speakers annotated by the orange oval (i.e., sw_5069, sw_5143 and sw_5397) in Figure 4.4(a) are close to each other and we cannot differentiate them easily. However, these x-vectors are more speaker discriminative after DAT as observed in Figs. 4.4(b)–(e). Similarly, the x-vectors from SRE04–10 indicated by the black arrows (speaker identities 8918, 8962 and 8977) become more distinct after InfoVDANN transformation (Figure 4.4(e)). For Voxceleb1, the embeddings within the red clouds (speaker IDs id11247 and id11248) show the same tendency: DAT makes the speaker representation more discriminative. Because these speakers are not in the training set, the results suggest that the InfoVDANN can generalize to unknown speakers. This is the reason why the InfoVDANN

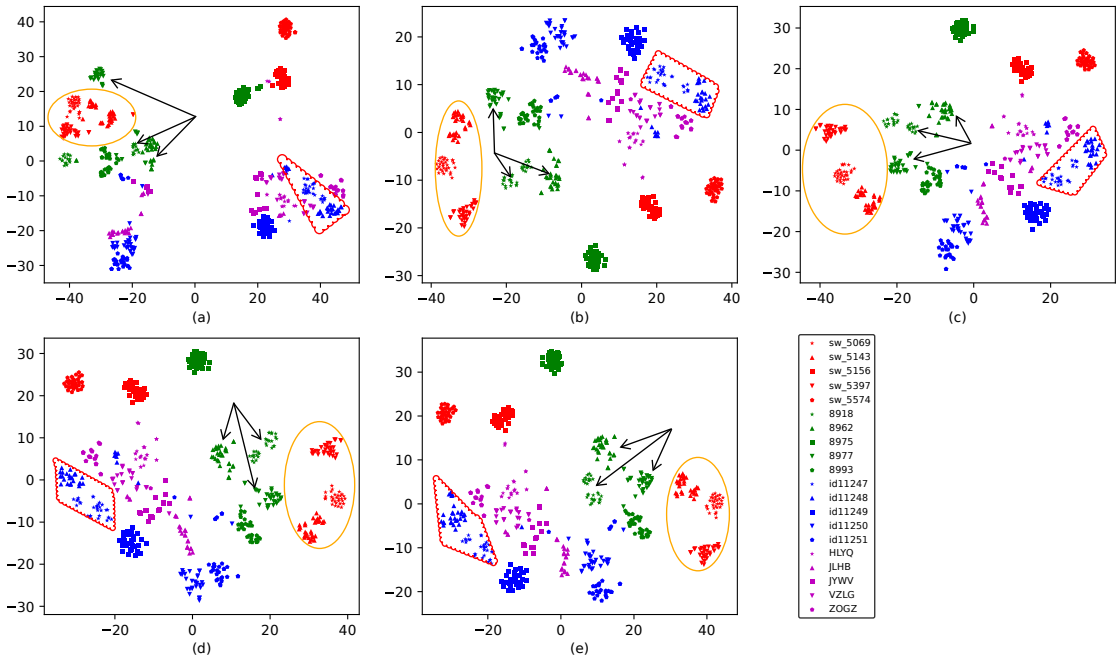


Figure 4.4: Comparison of t-SNE plots of (a) raw x-vectors, (b) DANN-transformed x-vectors, (c) VDANN-transformed x-vectors, (d) MMD–VDANN transformed x-vectors, and (e) AAE–VDANN transformed x-vectors. Each color denotes a domain, e.g., red, green, blue and magenta indicate SwitchBoard-2, SRE04–10, VoxCeleb1 and SITW, respectively, and each marker represents a speaker. Speaker identities are illustrated in the legend.

can improve the performance of x-vectors in SRE16 and SRE18-CMN2.

4.6.3 Effect on Gaussian Regularization

The InfoVDANN and the VDANN apply variational regularization on the learned embeddings so that the transformed x-vectors will follow a Gaussian distribution. To investigate the effectiveness of this Gaussian regularization, we present the normal Q–Q plots [115] of two randomly selected dimensions of the raw x-vectors and the x-vectors transformed by InfoVDANN, VDANN, and DANN. These x-vectors were selected from the CMN2 part of the SRE18 evaluation set. Evidently, as shown in Figure 4.5, the distributions of all the x-vectors transformed by the MMD–VDANN,

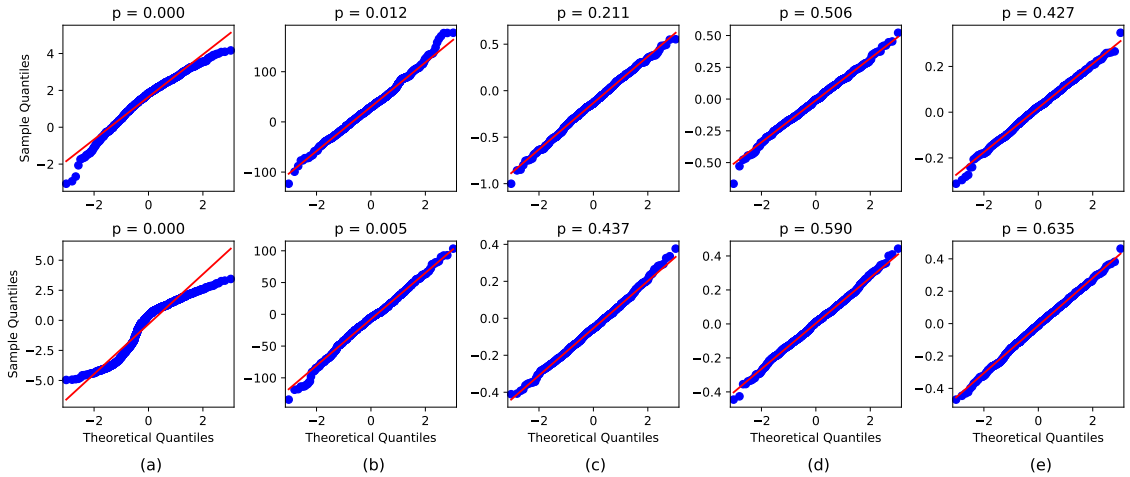


Figure 4.5: Quantile-quantile (Q–Q) plots of the 151-st (Row 1), and 301-st (Row 2) components of (a) raw x-vectors, (b) DANN-transformed x-vectors, (c) VDANN-transformed x-vectors, (d) MMD–VDANN transformed x-vectors, and (e) AAE–VDANN transformed x-vectors. The vertical and horizontal axes correspond to the samples under test and the samples drawn from a standard normal distribution, respectively. The red line represents the case of perfectly Gaussian. The p-values above the graphs were obtained from Shapiro-Wilk tests, with $p > 0.05$ meaning failing to reject the null hypothesis that the test samples come from a Gaussian distribution (i.e., the larger the p, the more Gaussian the distribution).

AAE–VDANN, and VDANN are closer to a Gaussian distribution than the DANN and the baseline systems, whereas the InfoVDANN-transformed ones seem to be more Gaussian than those transformed by the VDANN. This suggests that the InfoVAE loss can make the latent vectors $\mathbf{z}$'s to follow a Gaussian distribution.

The histograms of the p-values obtained from Shapiro-Wilk tests [116, 117] are illustrated in Figure 4.6. We can see that generally the MMD–VDANN, AAE–VDANN, and VDANN have larger p-values than the DANN and the baseline. This suggest that the distributions of InfoVDANN- and VDANN-transformed x-vectors are closer to the standard Gaussian than the DANN and baseline systems. This is reasonable because we perform variational regularization in these three systems to make their distributions more Gaussian.

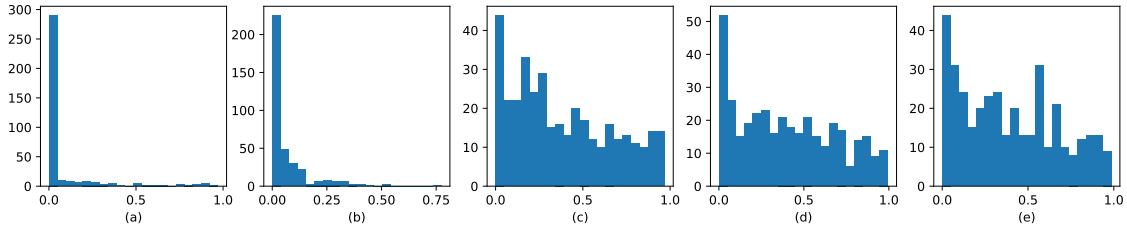


Figure 4.6: Histograms of the p-values of each dimension of (a) raw x-vectors, (b) DANN-transformed x-vectors, (c) VDANN-transformed x-vectors, (d) MMD–VDANN transformed x-vectors, and (e) AAE–VDANN transformed x-vectors. The p-values were obtained from Shapiro-Wilk tests, with $p > 0.05$ meaning failing to reject the null hypothesis that the test samples come from a Gaussian distribution.

Table 4.4: Estimates of the mutual information term ($I_q(\mathbf{x};\mathbf{z})$ in (4.10)) under SRE16 Evaluation set and SRE18-CMN2 Evaluation set

	SRE16-eval				SRE18-eval-CMN2			
	Enrollment		Test		Enrollment		Test	
	mean	var	mean	var	mean	var	mean	var
VDANN	4.466	1.092	5.078	1.115	3.922	1.045	4.567	1.077
MMD–VDANN	4.811	1.052	5.770	1.150	5.357	1.228	5.028	1.327
AAE–VDANN	5.114	1.047	6.263	1.151	5.038	1.163	5.031	1.248

4.6.4 Comparison of Mutual Information

The InfoVDANN explicitly incorporates an MI term in the objective function during DAT to additionally preserve meaningful information between the learned features and the input set. It makes sense to infer latent representations that are more speaker discriminative by maximizing this MI term together with the optimization of other sub-networks in the InfoVDANN. It has been verified in Table 4.3 that both the InfoVDANNs consistently outperform the VDANN based on the G-PLDA backend. To further evaluate the effectiveness of this MI maximization, we report the MI estimates on both the SRE16 Evaluation set and SRE18-CMN2 Evaluation set in Table 4.4.

The MI between the latent variable $\mathbf{z}$ and the input $\mathbf{x}$ is estimated by (4) in

Appendix A. We randomly selected 1,024 x-vectors from each set (e.g., SRE16-eval Enrollment and Test, SRE18-eval-CMN2 Enrollment and Test), and used these 1,024 samples as one single batch for each estimation. Each of the mean and variance was based on 200 simulations.

We can see from Table 4.4 that both MMD–VDANN and AAE–VDANN have higher MI between the learned features and the inputs than the VDANN, which contributes to the performance gain in SRE16 and SRE18-CMN2. According to (5) in Appendix A the MI estimates are bounded by 6.9314 (i.e., log(1024)) for these samples. Although there is some gap between the estimates and the upper bound, the performance gains on SREs are still statistically significant as reported in Section 4.6.1.

4.6.5 Impact of Hyperparameters λ and η

In (4.13), we use four hyperparameters (α, β, λ and η) in the InfoVDANN's objective function. VDANN is a special case of InfoVDANN in that the InfoVDANN loss degenerates into the VDANN objective when $\eta = 0$, and $\lambda = 1$ in (4.17). The discrepancy between the two loss functions is the choice of $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ and the contribution of $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$ and $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ to the total loss, which are controlled by λ and η, respectively. To examine the superiority of InfoVDANN over VDANN, we present the impact of λ and η on SRE performance in Figure 4.7 by setting $\alpha = 0.1$, and $\beta = 1.0$ in (4.13).

From (4.10), we note that λ is to balance the variational inference and data reconstruction during the optimization of the VAE sub-network in Figure 4.2. In our experimental setup, because the difference between the dimension of the latent vector (400) and that of the input embedding (512) is not very large, we started with $\lambda = 1.0$ to evaluate the influence of η on SRE16-eval and SRE18-CMN2. As shown in Figure 4.7(a) and Figure 4.7(b), both MMD–VDANN and AAE–VDANN achieve the best performance at $\eta = 0.2$ for SRE16 with and without the *extra* Kaldi's PLDA

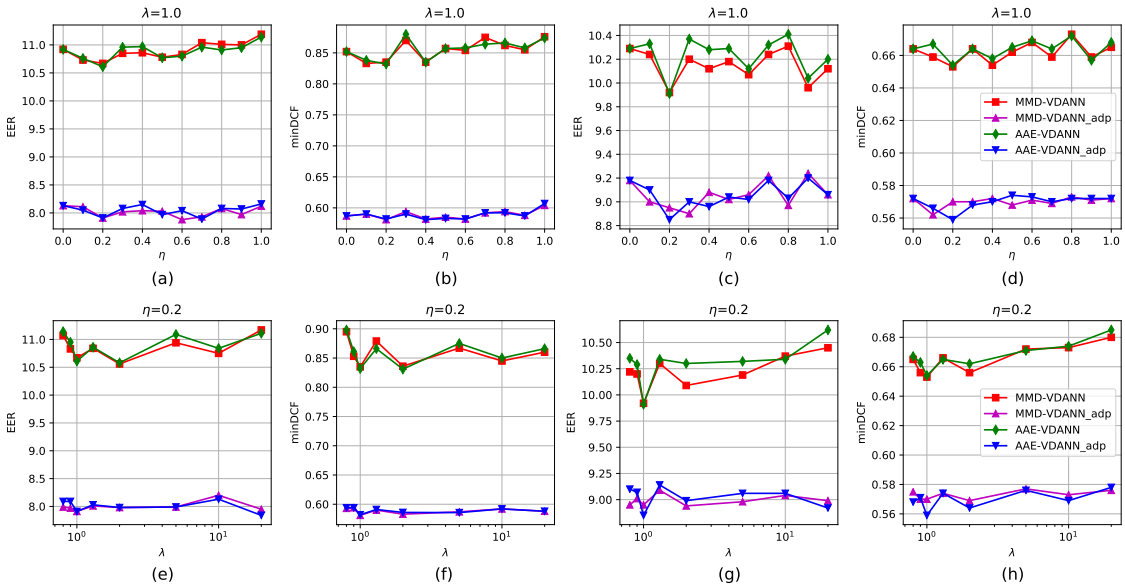


Figure 4.7: Impact of hyperparameters λ and η on the performance of SRE16 and SRE18-CMN2. The first row shows the impact of η on the performance of SRE16 [(a) and (b)] and SRE18-CMN2 [(c) and (d)] with λ fixed to 1.0. The second row illustrates the impact of λ on SRE16 [(e) and (f)] and SRE18-CMN2 [(g) and (h)] with $\eta = 0.2$. The instances with a suffix "_adp" in the legend denote the case using Kaldi's PLDA adaptation.

adaptation. In this regard, we fixed η to 0.2 when inspecting how λ impacts SRE16 performance. The result is illustrated in Figure 4.7(e) and Figure 4.7(f), from which we can see that within a wide range of $\lambda \in [0.8, 10]$, the SRE16 performance without PLDA adaptation does not change too much, while it achieves a slightly better performance at $\lambda = 1.0$. With Kaldi's PLDA adaptation, we can obtain similar performance at $\lambda = 1.0$ and $\lambda = 2.0$. Overall, with $\eta = 0.2$, both MMD–VDANN and AAE–VDANN obtained a consistently good performance at $\lambda = 1.0$ for SRE16 with and without PLDA adaptation. From Figs. 4.7(a), 4.7(b), 4.7(e) and 4.7(f), we conclude that $\eta = 0.2$ and $\lambda = 1.0$ are suitable choices for compromising the contribution of $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$ and $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$ to produce meaningful speaker embeddings.

We obtained a similar conclusion for SRE18-CMN2, as shown in Figs. 4.7(c), 4.7(d), 4.7(g) and 4.7(h). The performance over the choice of both hyperparameters with PLDA adaptation is more stable than that without it. For fixed λ, increasing η from 0 to an appropriate value (e.g., 0.2) can improve the performance of InfoVDANN. However, a larger value for η can have a detrimental effect on the performance. Because η represents the contribution of the MI term in (4.10), maintaining a high MI between the latent features and the inputs during training does not always produce better features. Note that when $\eta = 1.0$ and $\lambda = 1.0$, (4.10) becomes the standard AAE loss function [85]. This in turn verifies that $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$ is still very important as a regularization term on the learned embeddings. On the other hand, with $\eta = 0.2$, as shown in Figs. 4.7(g) and 4.7(h), the performance is insensitive to a wide range of λ.

From the above analysis, we thus used $\alpha = 0.1$, $\beta = 1.0$, $\eta = 0.2$, and $\lambda = 1.0$ for experimental comparisons in the previous subsections. From Figs. 4.7(a)–(h), we also observe that there is no big difference between the choice of $\mathrm{D}_g\left(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})\right)$, e.g., MMD and adversarial learning, since MMD–VDANN has a comparable performance with the AAE–VDANN for nearly all of the experiments. This also empirically verifies the robustness of InfoVDANN using different ways to minimize the generalized divergence. But due to the additional latent-variable discriminator (distinguishing the samples drawn from $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$) in AAE–VDANN, AAE–VDANN possesses more parameters compared with MMD–VDANN. Moreover, the minimax optimization of this latent-variable discriminator is not as efficient as the minimization of the MMD. As such, in practice, training AAE–VDANN takes slightly longer than training MMD–VDANN[3]. Thus, from the perspective of implementation, MMD–VDANN may be a better choice.

[3]It takes around 15% more time to train an AAE–VDANN than an MMD–VDANN.

Chapter 5

UTTERANCE-LEVEL AGGREGATION IN THE SPECTRAL DOMAIN

5.1 Introduction

A modern speaker embedding network is usually comprised of three components: a frame-level subnetwork, a pooling layer, and an utterance-level subnetwork. Because information will inevitably be lost in the pooling operation, it is crucial to preserve as much information as possible when aggregating the frame-level representations. Conventional pooling strategies such as statistics pooling [3] and self-attention based pooling [24,41,42] are operated in the temporal domain. However, due to the high non-stationarity in the temporal feature maps produced from the last frame-level layer, it is undesirable to use the global statistics (e.g., means and standard deviations, etc.) of the temporal feature maps as aggregated embeddings. This motivates us to explore more stationary spectral representations and perform aggregation in the spectral domain.

In [44], spectral pooling was proposed to replace max pooling for better information preservation in computer vision. This method involves three steps: 1) transforming the convolutional features from the spatial domain to the spectral domain by discrete Fourier transform (DFT), 2) cropping and retaining the low spectral components, and 3) performing inverse DFT on the cropped features to transform them back to the spatial domain. Because most energy of the spectral representations locates in the low-frequency region, spectral pooling is able to preserve most of the feature information by retaining the low spectral components.

However, because DFT can only be applied to deterministic or wide-sense stationary signals, it is not suitable for non-stationary speech signals [118]. In this chapter, the author replaces DFT by short-time Fourier transform (STFT) to account for the non-stationarity in the convolutional feature maps. Specifically, two spectral pooling strategies, i.e., short-time spectral pooling (STSP) [45] and attentive STSP [46], are proposed to preserve rich information and mitigate the non-stationarity in the frame-level feature maps during aggregation. The principles of STSP and attentive STSP are detailed in Section 5.2 and 5.3, respectively.

5.2 Short-time Spectral Pooling

Figure 5.1 shows the process of STSP. Because the pooling layer sits between the frame-level subnetwork and the utterance-level subnetwork, spectral analysis is performed on the output feature maps of the last convolutional layer, not on the MFCCs or filter-bank features. Given the c-th channel feature $\mathbf{x}_c = \{x_c(t)\}_{t=0}^{T-1}$ of a convolutional feature map $\{\mathbf{x}_c\}_{c=1}^{C} \in \mathbb{R}^{C\times T}$ with C channels, its short-time Fourier transform (STFT) [119] is expressed as follows:

$$X_c(n,k) = \sum_{t=0}^{T-1} x_c(t) w(t-nS) e^{-\frac{j2\pi}{L}kt}, \tag{5.1}$$

where $w(\cdot)$ is a window function of length L, S denotes the sliding step of the window, n indexes the temporal segments (sliding windows), and $k = 0, \ldots, L-1$ indexes the frequency components. Note that in this chapter, *we always make sure that the STFT length (the length of Fourier transform during STFT) is equal to the window length* L. (5.1) suggests that by sliding the window, we may apply multiple STFTs on a 1-D sequence to produce a 2-D spectral feature map with a temporal index n and a frequency index k for each channel.

To compute the spectral representation for each channel, we average the windowed

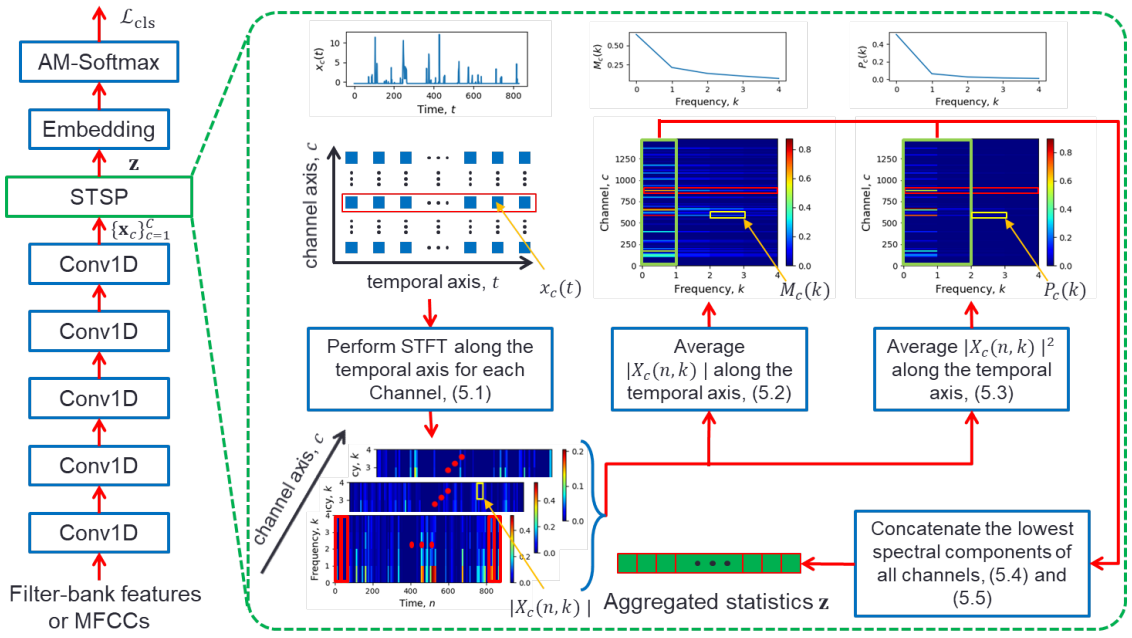


Figure 5.1: Schematic of short-time spectral pooling (STSP). The left part depicts the signal flow within the embedding network, whereas the right part details the pipeline of STSP. In the green dashed box, the left-most graph in the second row illustrates a temporal feature map extracted from the last convolutional layer. The bottom-left spectrograms were produced by STFT with length $L = 8$, and the vertical red boxes on top of the spectrogram denote spectral arrays to be averaged along the time axis by an attention weight matrix. The actual values of $\boldsymbol{M}_c$ and $\boldsymbol{P}_c$ after applying (5.2) and (5.3) are shown in the middle and the right-most maps in the second row, respectively. The top three plots correspond to the row vectors with elements $x_c(t)$, $M_c(k)$, and $P_c(k)$ in the red boxes, respectively. All the spectral features in the green boxes in the second row are concatenated to form the final utterance-level statistics (see (5.4) and (5.5) for details)

segments within the spectrogram $|X_c(n,k)|$ along the temporal axis and obtain the first-order spectral representation as follows:

$$M_c(k) = \frac{1}{N}\sum_{n=0}^{N-1} |X_c(n,k)|\,, \quad k = 0, \ldots, L-1 \tag{5.2}$$

where $N = \text{floor}((T-L)/S) + 1$ is the number of windowed temporal segments.

Similarly, by averaging the square of the spectrogram, we obtain the second-order spectral statistics as

$$P_c(k) = \frac{1}{N} \sum_{n=0}^{N-1} \left| X_c(n,k) \right|^2, \quad k = 0, \ldots, L-1. \tag{5.3}$$

During aggregation, we concatenate $M_c(0)$ and the square roots of the lowest R components of $P_c(k)$ to form the utterance-level representation of channel c:

$$\mathbf{z}_c = \left(M_c(0), \sqrt{P_c(0)}, \ldots, \sqrt{P_c(R-1)} \right). \tag{5.4}$$

The final utterance-level feature is produced by concatenating the spectral statistics of all channels:

$$\mathbf{z} = \left(\mathbf{z}_1, \ldots, \mathbf{z}_c, \ldots, \mathbf{z}_C \right). \tag{5.5}$$

Note that this STSP process ((5.1)–(5.5)) is applied to the speaker embedding network during both the training and inference phases.

One benefit of using the averaged representations $M_c(k)$ and $P_c(k)$ is that they have a low-pass characteristic. This characteristic facilitates the aggregation by keeping the *lowest* spectral components only, because these components contain most of the energy of the original features.[1]

In fact, STSP has a close relationship with statistics pooling [3]. For example, if we set $k = 0$ and use a rectangular window in (5.2) without any overlap between successive segments ($S = L$), the DC component $M_c(0)$ approximates the mean of $\mathbf{x}_c$. On the other hand, setting $k = 0$ in (5.3) resembles the power. In the extreme case where $S = L = 1$, we have $P_c(0) = \frac{1}{N} \sum_{n=0}^{N-1} x_c(n)^2$. This means that under this condition, using means and standard deviations for statistics pooling is an analogy to

[1] According to Parseval's theorem [120], the energy of a signal in the temporal domain is equal to that in the spectral domain.

using the DC components ($M_c(0)$ and $P_c(0)$ in (5.2) and (5.3)) for STSP. Thus, we may consider STSP as a generalized statistics pooling. Because we have *higher-frequency* components ($P_c(k)$'s for $k > 0$) for pooling, we can preserve more information than statistics pooling.

5.3 Attentive Short-time Spectral Pooling

5.3.1 Motivation of Attentive STSP

One limitation of STSP is that the brute average of the spectrograms along the temporal axis ignores the importance of individual windowed segments. In other words, all segments in a specific spectrogram were treated with equal importance when computing the spectral representations. In practice, however, this is not reasonable because phonetic information is rarely distributed uniformly across an utterance. As a result, different segments of an utterance have different speaker discriminative power.

To address the above limitation of STSP, the author proposes applying self-attention [121] on the windowed segments for each spectrogram while computing the spectral representations. As a result, the discriminative segments can be emphasized during aggregation. The author calls the proposed method *attentive STSP* in this chapter. Unlike the conventional attention mechanisms for speaker embedding that perform attention on temporal frames [24, 40–42], attentive STSP performs attention on individual windowed segments. Nevertheless, the rationale behind attentive STSP is the same as that of the conventional attentive pooling methods.

The intuition that exploiting the local stationarity in the convolutional feature maps is beneficial to utterance-level aggregation can be interpreted from a perspective of stochastic process. Specifically, if we consider a feature sequence at the final convolutional layer as a realization of a stationary stochastic process, its global statistics (e.g., mean, standard deviation, etc.) will not change with time. However, once the stationarity assumption is violated, which is common for the final convolutional

feature maps, these global statistics will become unreliable for summarizing the process. This suggests that the performance of statistics pooling and its attentive variants would suffer more severely on long utterances because of the non-stationarity in the feature sequence. Therefore, the conventional pooling methods that operate in the temporal domain can be sensitive to the duration variations in the evaluation sets. On the other hand, attentive STSP is more robust to duration variations, attributed to its ability to handle the local stationarity in the feature maps.

In fact, it has been observed in portfolio optimization that exploiting only the mean and variance of a non-stationary sequence is not sufficient. In [122], the authors pointed out that although the mean-variance optimization (MVO) has long been an optimal strategy for investment, it presents poor out-of-sample performance due to the non-stationarity in the financial time series. To overcome the inherent time-varying property of the price series, a complex spectral portfolio method was proposed to model the cyclostationarity of the time series.

5.3.2 Principle of Attentive STSP

The procedure of attentive STSP is shown in Figure 5.2. The difference between attentive STSP and STSP in Section 5.2 is the way the spectral representations are calculated. In attentive STSP, rather than brutely average the windowed segments within the spectrogram $|X_c(n,k)|$, we apply a weighted average of these segments by an H-head attention weight matrix and obtain a spectral sequence for each head as follows:

$$M_c^h(k) = \sum_{n=0}^{N-1} \alpha_n^h \left| X_c(n,k) \right|, \quad k = 0, \ldots, L-1 \tag{5.6}$$

where $N = \text{floor}((T-L)/S) + 1$ is the number of windowed temporal segments, $h \in [1, H]$ indexes the attention heads, and $\boldsymbol{\alpha}^h = \{\alpha_n^h\}_{n=0}^{N-1}$ denotes the attention weight vector corresponding to head h. Similarly, if we attend to the square of the

spectrogram, we obtain the second-order spectral statistics:

$$P_c^h(k) = \sum_{n=0}^{N-1} \alpha_n^h \left| X_c(n,k) \right|^2, \quad k = 0, \ldots, L-1. \tag{5.7}$$

The attention mechanism is shown in Figure 5.2(b). Note that the attention process is operated on the windowed segments within each spectrogram. We first average the spectrogram for each channel along the frequency axis to make sure that all the spectral components within a specific segment share the same attention weights. The resulting feature map is denoted as $\mathbf{G} = \{G_c(n)\}_{c=1}^{C} \in \mathbb{R}^{C \times N}$, where

$$G_c(n) = \frac{1}{L} \sum_{k=0}^{L-1} \left| X_c(n,k) \right|, \quad n = 0, \ldots, N-1. \tag{5.8}$$

Similar to (2.4), the attention weight matrix $\mathbf{A}^{\text{STSP}} = \{\boldsymbol{\alpha}^h\}_{h=1}^{H}$ is computed as

$$\mathbf{A}^{\text{STSP}} = \text{Softmax}\left(\tanh\left(\mathbf{G}^\top \mathbf{W}_1^{\text{STSP}}\right) \mathbf{W}_2^{\text{STSP}}\right), \tag{5.9}$$

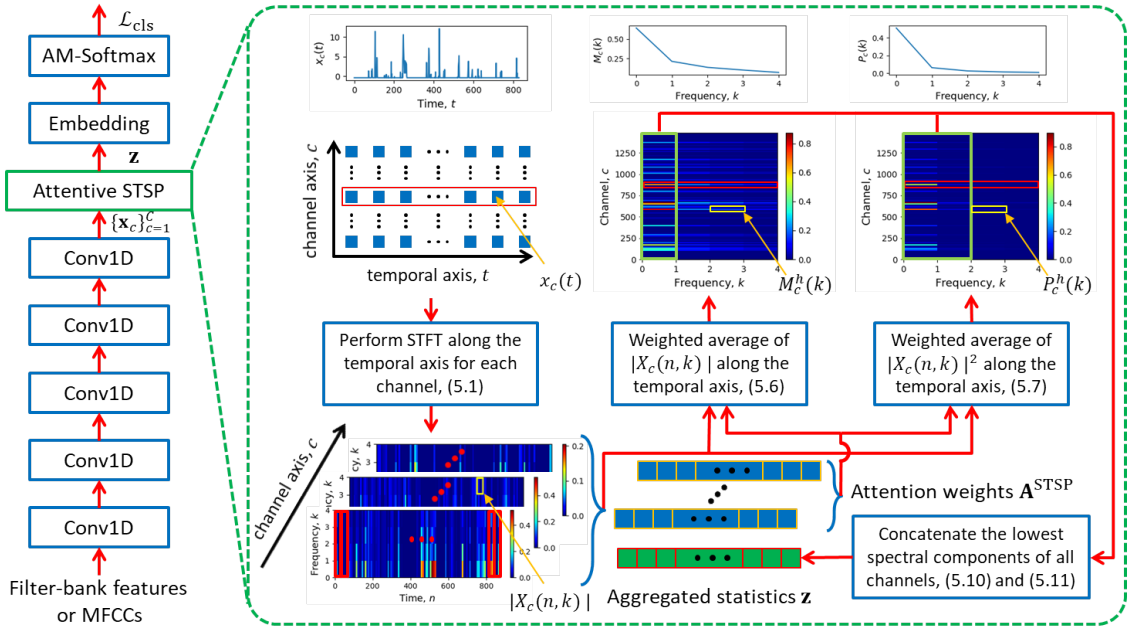
where $\mathbf{W}_1^{\text{STSP}} \in \mathbb{R}^{C \times D}$ and $\mathbf{W}_2^{\text{STSP}} \in \mathbb{R}^{D \times H}$ are trainable weight matrices.

To aggregate the frame-level information, we concatenate $M_c^h(0)$ and the square roots of the lowest R components of $P_c^h(k)$ to form the utterance-level representation of channel c for head h:

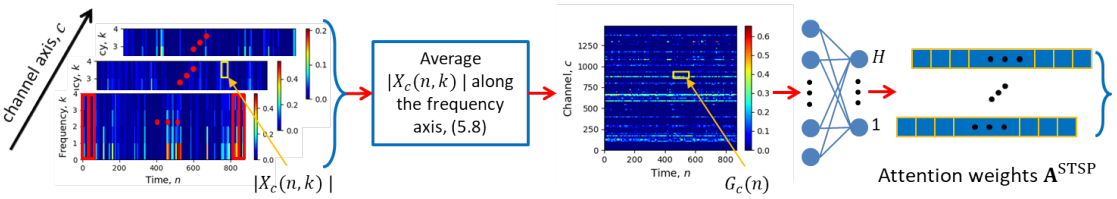
$$\mathbf{z}_c^h = \left(M_c^h(0), \sqrt{P_c^h(0)}, \ldots, \sqrt{P_c^h(R-1)} \right). \tag{5.10}$$

The final utterance-level feature is produced by concatenating the spectral statistics of all channels and all heads:

$$\mathbf{z} = \left(\mathbf{z}_1^1, \ldots, \mathbf{z}_c^h, \ldots, \mathbf{z}_C^H \right). \tag{5.11}$$



(a)



(b)

Figure 5.2: (a) Schematic of attentive STSP. The left part depicts the signal flow within the embedding network, whereas the right part details the pipeline of attentive STSP. In the green dashed box, the left-most graph in the second row illustrates a temporal feature map extracted from the last convolutional layer. The bottom-left spectrograms were produced by STFT with length $L = 8$, and the vertical red boxes on top of the spectrogram denote spectral arrays to be averaged along the time axis by an attention weight matrix. The actual values of $\boldsymbol{M}_c^h$ and $\boldsymbol{P}_c^h$ (only $h = 1$ is considered here) after applying (5.6) and (5.7) are shown in the middle and the right-most maps in the second row, respectively. The top three plots correspond to the row vectors with elements $x_c(t)$, $M_c^1(k)$, and $P_c^1(k)$ in the red boxes, respectively. All the spectral features in the green boxes in the second row are concatenated to form the final utterance-level statistics (see (5.10) and (5.11) for details). (b) Schematic of the attention mechanism used in attentive STSP. The middle feature map denotes the actual value of $\mathbf{G}$ and the node graph illustrates an H-head attention network. The attention weight matrix $\mathbf{A}^{\text{STSP}}$ is computed as in (5.9).

5.3.3 Rationale and Validity of Attentive STSP

As mentioned in Section 5.1, to facilitate the aggregation process, STSP requires that the energy of the features should concentrate in the low-frequency region [45]. To demonstrate that attentive STSP also satisfies this requirement, we provide empirical evidences by plotting the statistics of the spectral representations computed from (5.6) and (5.7). The procedure for computing the spectral representations is as follows:

1. Randomly select 20 utterances from each of the 1,211 speakers in the VoxCeleb1 development set.

2. Extract 40-dimensional filter-bank features from the selected utterances and perform mean normalization with a sliding window of 3 seconds.

3. Train an embedding system with a single-head attentive STSP layer ($H = 1$) using 5,984 speakers from the VoxCeleb2 development set.

4. Extract the feature maps from the last convolutional layer of the embedding network.

5. Compute the spectral representations $M_c^1(k)$ and $P_c^1(k)$ according to (5.6) and (5.7), respectively.

We followed the Kaldi's recipe[2] to prepare the acoustic features (the second step) to make sure that the data preparation is the same as that of the embedding training data. The training procedure of the embedding network is detailed in Section 5.4.1.

Figure 5.3 shows the statistics of $M_c^1(k)$ and $P_c^1(k)$ over 24,220 utterances. We observe that both the $M_c^1(k)$ and $P_c^1(k)$ of a particular channel that was randomly selected have most of their energy concentrated in the low-frequency region. This

[2]https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb/v2.

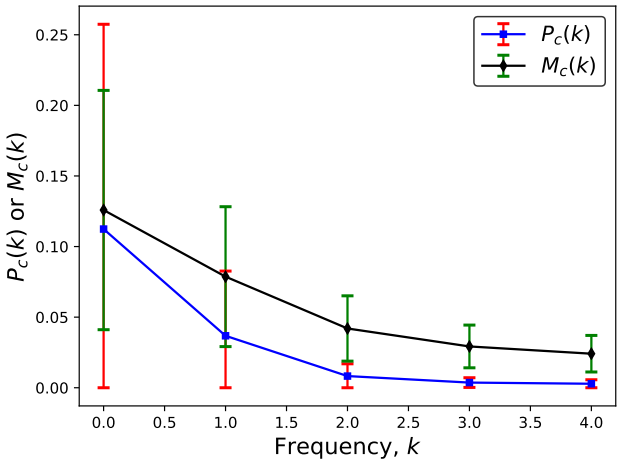


Figure 5.3: Statistics of $M_c^h(k)$ in (5.6) and $P_c^h(k)$ in (5.7) of a randomly selected channel c with respect to the frequency components k's under $H = 1$. Black diamonds and blue squares denote the means of $M_c^1(k)$ and $P_c^1(k)$ over 24,220 utterances in the VoxCeleb1 development set, respectively. The error bars represent one standard deviation. The STFT length L for computing $X_c(n, k)$ in (5.1) was set to 8. Only the left half of $M_c^1(k)$ and $P_c^1(k)$ $(0 \leq k \leq 4)$ are plotted due to the symmetry of spectrograms along the frequency axis.

validates the feasibility of using attentive STSP for utterance-level aggregation. Attributed to the desirable statistics of $M_c^h(k)$ and $P_c^h(k)$ in the spectral domain, attentive STSP uses the lowest spectral components for aggregation.

The property that most of the energy of the convolutional features concentrates in the low-frequency part in the spectral domain also reflects that the frame-level network is a low-pass filtering system. In [123], Rahaman *et al.* interpreted the generalization of DNNs [124, 125] from a Fourier perspective and revealed a learning bias of DNNs towards low-frequency functions (spectral bias). Although there is no exact clue that the low-pass characteristic of the speaker embedding network is completely attributed to the spectral bias of CNNs, we believe that this bias at least contributes partially to the low-pass property of the frame-level networks. On the other hand, in both temporal pooling [27] and statistics pooling [3], global averaging is used to extract

the mean vector of the whole temporal features. In fact, global averaging can be seen as mean filtering with a global kernel [126], which is a low-pass filtering operation. Therefore, the pooling methods in [27] and [3] have already *implicitly* exploited the low-pass characteristic of the CNNs, although they only use the DC components of the spectral representations. Similar to the vanilla STSP, the proposed attentive STSP *explicitly* explores the low-pass filtering effect and improves these pooling strategies by accounting for more spectral components besides the DC ones. Thus, attentive STSP preserves more speaker information than the conventional statistics pooling during aggregation.

Note that for the k-th spectral component ($k > 1$), because $M_c^h(k)$ and $P_c^h(k)$ are both related to the k-th frequency, the information in $M_c^h(k)$ and $P_c^h(k)$ will be correlated. From Fig. 5.3, we observe that $P_c^1(k)$ decays faster than $M_c^1(k)$ and are more energy-concentrated towards the zero frequency. We hypothesize that using $P_c^h(k)$ can be more effective than using $M_c^h(k)$ alone (see Section 5.5.5 for details). Taking this observation into account and to avoid using repeated information, we only use $\sqrt{P_c^h(k)}$ ($k > 1$) in (5.4) for aggregation.

5.3.4 Relation to Previous Works

Attentive STSP is a generalized STSP in that if we apply equal attention weights produced from a single-head attention network on the windowed segments in (5.2) and (5.3), i.e., $\alpha_n^1 = 1/N$ for $n \in [1, N]$, attentive STSP reduces to the vanilla STSP in [45].

Similar to STSP, attentive STSP also generalizes statistics pooling. Under the condition where single-head attention is implemented and equal attention weights are applied, if we set $k = 0$ and use a rectangular window without any overlap between successive segments (i.e., $S = L$) in (5.2), the DC component $M_c^1(0) = \frac{1}{N}\sum_{t=0}^{NL-1} x_c(t)$

approximates the mean of $\mathbf{x}_c$ multiplied by a scaling factor L.[3] On the other hand, setting $k = 0$ in (5.3) resembles computing the power of $\mathbf{x}_c$. In the extreme case where $S = L = 1$, we have $P_c^1(0) = \frac{1}{T}\sum_{t=0}^{T-1}(x_c(t))^2$. This means that under these conditions, using the means and standard deviations in statistics pooling is an analogy to using the DC components ($k = 0$ in (5.2) and (5.3)) in attentive STSP. Therefore, attentive STSP can be seen as a generalized statistics pooling method.

Attentive STSP has a close relationship with multi-head attentive pooling [41] because they both apply an attention mechanism during aggregation. However, there are two major differences between these two methods. Firstly, as shown in (5.9), attentive STSP performs attention on a series of *windowed segments* in $\mathbf{G}$, whereas multi-head attentive pooling implements an attention network on a sequence of *frames* as in (2.4). Because the spectral components in each segment is (locally) stationary, segment-level attention can provide attenive STSP with better robustness against the non-stationarity in the feature maps than frame-level attention. Secondly, attentive STSP further preserves the speaker information by retaining the *informative* spectral components only. Note that not all the components in the spectral domain are beneficial for aggregation. Specifically, incorporating high-frequency components can cause detrimental effect to the speaker embeddings because these components are very noisy. In contrast, because multi-head attentive pooling takes all the temporal frames into account, it always includes all the spectral information during aggregation (due to the equivalence of information between the temporal domain and the spectral domain). Therefore, attentive STSP is advantageous to multi-head attentive pooling in information distillation.

Note that the windowed segment attention in attentive STSP is different from the sliding-window attention in [127] and [128], although both attention mechanisms

[3]In fact, the DC component should be $M_c^1(0) = \frac{1}{N}|\sum_{t=0}^{NL-1} x_c(t)|$ under $S = L$ according to (5.2). However, in this paper, because each convolutional layer is followed by an ReLU layer in the speaker embedding network (see Table 2.1), all the elements of the input feature $\mathbf{x}_c$ will be non-negative. Thus, we have $M_c^1(0) = \frac{1}{N}|\sum_{t=0}^{NL-1} x_c(t)| = \frac{1}{N}\sum_{t=0}^{NL-1} x_c(t) > 0$.

involve the term "window." In particular, the segment-level attention in this paper is operated on the windowed segments to account for the local stationarity of the temporal feature maps. The attention mechanism aims to learn the *global* relationships across all of the windowed segments in an utterance. In contrast, the sliding-window attention takes a series of tokens (equivalent to frames in speaker verification) as input and only models the *local* relationships of the tokens within each sliding window. The objective is to reduce computation relative to the full attention [121]. Therefore, these two methods differ completely in their inputs, operating mechanisms, and objectives.

Interestingly, attentive STSP is also related to the modulation spectrum of speech [129, 130] because the spectral representations in attentive STSP and modulation spectrum are both produced from spectrograms. However, due to the differences in the input, the way to produce the spectrograms, and the strategy to compute the spectral representations, attentive STSP differs substantially from modulation spectrum. First, attentive STSP is operated on the output feature maps at the last convolutional layer of a speaker embedding network, whereas modulation spectrum takes speech signals as input. Second, attentive STSP applies STFT to perform time-frequency transformation, whereas filter-bank analysis is typically adopted for computing the spectrograms in modulation spectrum. Third, to compute modulation spectra, handcrafted bandpass filtering is often applied to the spectrograms, e.g., a linear filter is applied to the log-transformed spectrograms in RASTA processing [131]. In contrast, we compute $M_c^h(k)$'s and $P_c^h(k)$'s through a weighted average of the spectrogram and its square.

5.4 Experimental Setup

Five pooling methods are compared in this chapter, i.e., statistics pooling [3], multi-head attentive pooling [41], channel- and context-dependent statistics pooling (CCDSP) [25], STSP [45], and the proposed attentive STSP. We evaluated the performance of

these pooling methods on the VoxCeleb1 test set (clean) [13], the VOiCES 2019 evaluation set [132], the SRE16 evaluation set [110], and the SRE18-CMN2 evaluation set [111].

5.4.1 Training of Speaker Embedding Extractor

For the evaluation on VoxCeleb1, only the VoxCeleb2 development subset (approximate 2 million utterances from 5,984 speakers) was used for training. Whereas both VoxCeleb1 development and VoxCeleb2 development data were used as the training set for VOiCES 2019, which amounts to about 2.1 million utterances from 7,185 speakers. We followed the Kaldi's VoxCeleb recipe to prepare the training data, i.e., using 40-dimensional filter bank features, performing energy-based voice activity detection, implementing augmentation (by adding reverberation, noise, music and babble to the original speech files), applying cepstral mean normalization with a window of 3 seconds, and filtering out utterances with a duration less than 4 seconds.[4] Totally, we had approximately twice the number of clean utterances for training the embedding network.

For both SRE16 and SRE18-CMN2 evaluations, we followed the Kaldi's SRE16 recipe to prepare the training data.[5] Instead of using the 40-dimensional filter bank features, 23-dimensional MFCCs were used for training. The training set consists of SRE04–10, Mixer 6, Switchboard Cellular, and Switchboard 2 (all phases). Totally, we had 238,618 utterances from 5,402 speakers in the training set.

We used the architecture in Table 2.1 to implement the statistics pooling baseline. For systems that use multi-head attentive pooling, we used an attention network with 500 *tanh* hidden nodes and H linear output nodes, where H is the number of attention heads (see (2.7)). We used $D' = 256$ in CCDSP (see 2.8) and the non-linearity is *tanh*. For STSP, we used a rectangular window function with length $L = 8$ and $S = L$. The

[4]https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb/v2.

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

attention network in attentive STSP follows the structure of multi-head attentive pooling and uses various window functions with length L ranging from 4 to 16. The step size S of each windowed segment varies from $L/4$ to L.

The additive margin softmax loss [74] was used for training. The additive margin m and the scaling factor s in (2.12) were set to 0.25 and 30, respectively. The mini-batch size was set to 128 for all evaluation tasks. There are around 2,337 mini-batches in one epoch for VoxCeleb1 and VOiCES 2019, and 4,220 mini-batches for SRE16 and SRE18-CMN2. Each mini-batch was created by randomly selecting speech chunks of 2–4s from the training data. We used a stochastic gradient descent (SGD) optimizer with a momentum of 0.9. The initial learning rate was 0.02 and it was linearly increased to 0.05 at Epoch 20. After that, it was decayed by half at Epochs 50, 80 and 95, respectively. Totally, the networks were trained for 100 epochs. Once training was completed, the speaker embedding was extracted from the affine output at Layer 7 in Table 2.1.

5.4.2 PLDA Training

We used a Gaussian PLDA backend [11] for all evaluations. For VoxCeleb1, the PLDA model was trained on the speaker embeddings extracted from the clean utterances in the training set for the embedding network. For VOiCES 2019, we trained the backend on the concatenated speech with the same video session and used utterances augmented with reverberation and noise. The PLDA training data for both SRE16 and SRE18-CMN2 were the embedding network's training set excluding the Switchboard part. Before PLDA training, the speaker embeddings were projected onto a 200-dimensional space by LDA for VoxCeleb1 and 150-dimensional space for VOiCES 2019, SRE16, and SRE18-CMN2, followed by whitening and length normalization. The LDA projection matrix was trained on the same dataset as for training the PLDA models. For VOiCES 2019, SRE16 and SRE18-CMN2, we also applied adaptive score normalization [133]. The cohort for VOiCES 2019 was selected from

the longest two utterances of each speaker in the PLDA training data; whereas for SRE16 and SRE18-CMN2, the cohort was the respective unlabeled development set.

5.5 Results and Discussions

5.5.1 Performance on Various Evaluations

The performance was evaluated in terms of EER and minDCF with $P_{\text{target}} = 0.01$. Table 5.1 shows the performance of different systems on VoxCeleb1 (clean), VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval. We can observe that all the pooling methods outperform the statistics pooling baseline. For attentive pooling, STSP and attentive STSP, we have the following analyses.

Multi-head attentive pooling

For all evaluation tasks, attentive pooling (Rows 2–5) achieves the best performance when the number of heads H was set to 2. When H further increases to 4, attentive pooling exhibits performance degradation, especially on SRE16 and SRE18-CMN2. Among many possibilities, the performance degradation can be caused by an increased number of non-stationary attention weight vectors produced by the attention network.

Take VoxCeleb1 for example, as shown in the first row of Fig. 5.4(a), the feature sequence ($\{h_{c,t}\}_{t=0}^{T-1}$ in (2.5)) presents a high non-stationarity along the temporal axis. To fit the drastic variations of the sequence, the attention network is trained to produce attention weights of large variations, as evident by the second row of Fig. 5.4(a). However, due to the substantial variations within the weight vectors, it is difficult for the attention network to generalize well on unseen utterances. Therefore, the non-stationarity of the attention weights could remarkably affect the performance of attentive pooling. On the other hand, a larger H does not necessarily indicate a larger degree of diversity in the attended feature sequences. For example, as shown in the third row of Fig. 5.4(a), the attended frames by Head 1 largely overlap those by Head

0. On the contrary, increasing the number of attention heads may introduce a larger degree of non-stationarity in the attention weights, causing poorer generalization to unseen data.

For SRE16-eval and SRE18-CMN2-eval, because the utterances in the evaluation sets are much longer than those in VoxCeleb1,[6] the degree of non-stationarity in the attention weights will be larger than that of the VoxCeleb1 test set. Thus, the performance drop on SRE is severer than that on VoxCeleb1 and VOiCES 2019.

Channel- and Context-Dependent Statistics Pooling

We evaluated the performance of CCDSP with and without the global context vector ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, see Section 2.4.2). As observed in Rows 6–7, including the global statistics does not make a remarkable difference in performance. Although CCDSP achieves comparable performance with attentive pooling and outperforms statistics pooling, it cannot compete with STSP and attentive STSP on VOiCES 2019 and SREs.

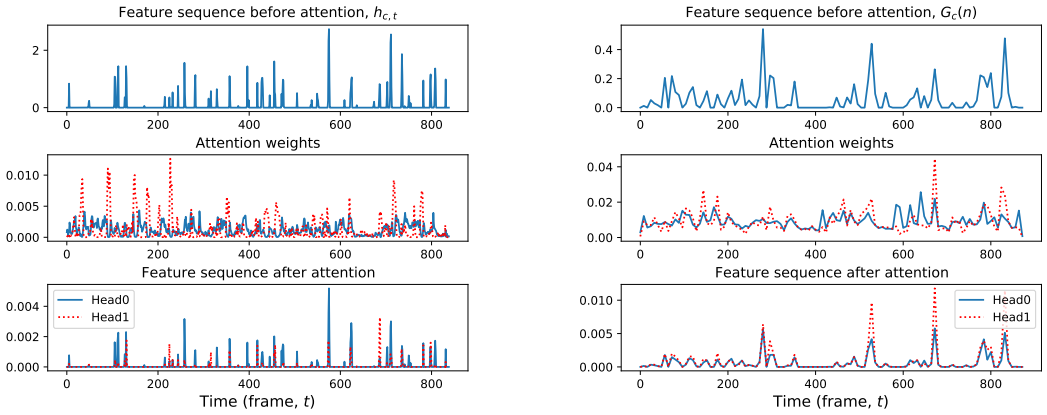
STSP

From Rows 8–10 of Table 5.1, we see that STSP achieves a consistent improvement in performance when the retained number of low-frequency components R in $P_c^h(k)$'s is increased from 1 to 3. However, further including the 4-th component will slightly degrade the performance, as can be seen in Row 11. This may be because there are more noises in the higher-frequency components. As demonstrated in [45], the magnitude of the spectral components $P_c(k)$'s in STSP approaches 0 when k becomes large. This suggests that we can hardly learn useful information from these vanishing components. Instead, the high frequency components can bring unwanted noise to the network during learning.

[6]The enrollment utterances of SRE16-eval and SRE18-CMN2-eval are approximate 60 seconds and the test utterances are 10–60s [110, 111], whereas the average duration of VoxCeleb1 is 8.2 seconds [13].

Table 5.1: Performance on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval. CCDSP refers to channel- and context-dependent statistics pooling (see Section 2.4.2). Rectangular window functions were used for STSP and attentive STSP under $L = S = 8$. H denotes the number of heads in multi-head attentive pooling (see (2.7)) and attentive STSP (see (5.5)), and R is the number of the lowest second-order spectral components in STSP and attentive STSP (see (5.3).

Row	Pooling method	No. of paras of emb. sys.	VoxCeleb1		VOiCES19-eval		SRE16-eval		SRE18-CMN2-eval	
			EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
1	Statistics pooling	3.48 M	2.08	0.225	5.78	0.498	8.07	0.499	7.63	0.475
2	Multi-head attentive pooling (H=1)	4.23 M	1.97	0.224	5.69	0.472	8.11	0.480	7.32	0.467
3	Multi-head attentive pooling (H=2)	5.00 M	1.89	**0.206**	**5.41**	**0.454**	**7.61**	**0.472**	**6.89**	**0.454**
4	Multi-head attentive pooling (H=3)	5.77 M	**1.86**	0.219	5.53	0.461	8.02	0.475	6.90	0.457
5	Multi-head attentive pooling (H=4)	6.54 M	2.04	0.228	5.61	0.467	8.39	0.484	7.41	0.469
6	CCDSP w/o context	4.26 M	1.92	**0.209**	5.51	0.450	7.78	**0.475**	7.11	0.458
7	CCDSP w/ context	5.02 M	**1.88**	0.211	**5.46**	**0.441**	**7.69**	0.479	**7.02**	**0.457**
8	Vanilla STSP (R=1)	3.48 M	2.12	0.229	5.66	0.467	7.11	0.474	6.79	0.465
9	Vanilla STSP (R=2)	3.86 M	1.92	**0.210**	5.44	0.448	7.07	0.468	6.68	0.457
10	Vanilla STSP (R=3)	4.25 M	**1.87**	0.213	**5.30**	**0.443**	**6.77**	**0.460**	**6.65**	**0.443**
11	Vanilla STSP (R=4)	4.63 M	1.89	0.226	5.52	0.461	6.90	0.473	6.75	0.451
12	Attentive STSP (R=1, H=1)	4.23 M	1.89	0.211	5.40	0.453	6.65	0.469	6.32	0.453
13	Attentive STSP (R=2, H=1)	4.61 M	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	6.22	**0.434**
14	Attentive STSP (R=3, H=1)	5.00 M	1.81	0.195	5.13	0.415	6.40	0.450	**6.20**	0.437
15	Attentive STSP (R=4, H=1)	5.38 M	1.83	0.220	5.28	0.455	6.47	0.451	6.24	0.445
16	Attentive STSP (R=2, H=2)	5.77 M	1.89	0.207	5.34	0.444	6.50	0.458	6.39	0.441
17	Attentive STSP (R=2, H=3)	6.93 M	1.96	0.238	5.63	0.461	6.76	0.467	6.57	0.448



(a) (b)

Figure 5.4: Illustration of the mechanism in (a) multi-head attentive pooling and (b) attentive STSP under H=2. We used a rectangular window function for attentive STSP under $L = S = 8$. Both the embedding networks were trained on the Voxceleb2 development data (see Section 5.4.1). For multi-head attentive pooling, the feature sequence ($\{h_{c,t}\}_{t=0}^{T-1}$ in (2.5)) in the first row corresponds to an utterance randomly selected from the VoxCeleb1 development set. For attentive STSP, the feature sequence is a random row vector in $\mathbf{G}$ of (5.8). Note that the unit in the horizontal axis is the frame index t in (2.5) and (5.1).

Comparing Rows 8–11 and Rows 2–5, we observe that STSP achieves similar performance as that of attentive pooling on VoxCeleb1 and VOiCES19-eval, but STSP remarkably outperforms attentive pooling on both the SRE evaluations. In fact, although both attentive pooling and STSP aim at preserving speaker information during aggregation, they fulfill the task from different perspectives. Specifically, attentive pooling emphasizes on discriminative *temporal frames* to enhance the information in the aggregated embeddings, whereas STSP emphasizes discriminative *spectral components*. Due to the duality of Fourier transform, the information in the temporal domain is equal to that in the spectral domain. This suggests that there should be little difference between attentive pooling and STSP, as can be verified by the comparison on VoxCeleb1. However, for long utterances, because of the high non-stationarity in the convolutional features (as analyzed in Section 5.5.1(multi-head attentive pool-

ing)), it can be difficult to extract discriminative information in the temporal domain. In contrast, the spectral components in STSP are smoother, making STSP more robust against the non-stationary feature maps, especially for long utterances as in the SRE evaluations. Therefore, STSP can achieve larger performance gain over attentive pooling on both SRE tasks.

Attentive STSP

As shown in Rows 12–17 of Table 5.1, the best performance of attentive STSP is achieved under $R = 2$ and $H = 1$. Compared with attentive pooling (Rows 2–5), a major advantage of attentive STSP is that because the attention mechanism is operated on the windowed segments instead of the frames, the produced attention weight vectors are much smoother than those by attentive pooling. This can be seen by comparing the second rows of Fig. 5.4(a) and Fig. 5.4(b). In fact, it is natural to obtain smoother attention weights by segment-level attention because the coarse-grained attention mechanism has taken the local stationarity in the feature sequences into account. After all, exploiting the local stationarity in the frame-level features is a fundamental difference between the STSP-based pooling and the spectral pooling in [44]. On the other hand, as explained in Section 5.5.1 (STSP), performing aggregation on long utterances in the spectral domain is superior to that in the temporal domain. Because of the segment-level attention and the spectral aggregation, attentive STSP obtains substantially better performance than attentive pooling, especially on the SRE16 and SRE18-CMN2.

Compared with STSP, the extra attention mechanism in attentive STSP can emphasize discriminative segments for information aggregation, leading to more discriminative speaker embeddings. This is why attentive STSP outperforms STSP on all the evaluation tasks, which verifies the motivation of this paper. An interesting observation is that different from STSP which achieves the best performance under $R = 3$, attentive STSP performs the best when $R = 2$. This indicates that the spectral en-

Table 5.2: Performance on the subsets of SRE16. Stats, MHAP, and Att_STSP refer to statistics pooling, multi-head attentive pooling, and attentive STSP, respectively.

	Female		Male		Cantonese		Tagalog	
Pooling method	EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
Stats	8.60	0.513	7.63	0.445	3.53	0.315	12.48	0.693
MHAP (H=2)	8.14	0.484	7.06	0.437	3.07	0.276	11.68	0.668
STSP (R=3)	6.82	0.472	6.20	0.438	2.82	0.270	9.58	0.647
Att_STSP (R=2)	**6.49**	**0.471**	**5.99**	**0.430**	**2.53**	**0.263**	**9.52**	**0.627**

ergy of attentive STSP are more concentrated at the low-frequency region than the STSP, which further facilitates the aggregation process.

We also investigated the effect of the number of heads H on the performance of attentive pooling. Comparing Row 13 and Rows 16–17 in Table 5.1, we see that increasing H does not offer any performance improvement on all evaluations. As illustrated in the third row of Fig. 5.4(b), the sequences after a two-head attention operation are almost the same. This suggests that more attention heads do not necessarily create richer diversity of the attended features. Instead, a larger H can introduce noises to the pooling operation because of the non-stationarity in the attention weights, as in attentive pooling in Section 5.5.1 (Multi-head attentive pooling). If not stated otherwise, in the rest of the paper, we only used a single-head attention network for attentive STSP, i.e., $H = 1$.

Furthermore, the performance improvement of STSP and attentive STSP on the subsets of the SRE16 evaluation data is also consistent. Specifically, we split the SRE16 evaluation data according to the gender and language information, and the results are shown in Table 5.2. We observe that both STSP and attentive STSP outperform statistics pooling and multi-head attentive pooling, and attentive STSP achieves the best performance consistently on all subsets.

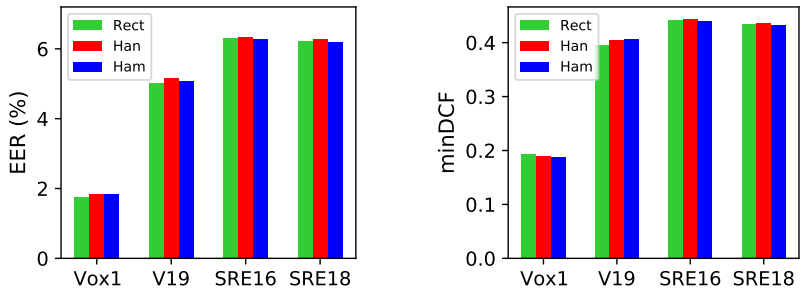


Figure 5.5: (a) EER and (b) minDCF of attentive STSP on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval with respective to various window functions under $L = S = 8$, $R = 2$, and $H = 1$.

5.5.2 Impact of Window Functions

In (5.1), a window function is applied to each temporal segment before performing DFT. To investigate the effect of the window function on performance, we implemented attentive STSP with the rectangular window, the Hanning window, and the Hamming window [134]. The performance is compared under $L = S = 8$ and $R = 2$.

As shown in Fig. 5.5(a) and Fig. 5.5(b), there is no significant difference in the performance of the three windows. We have also tried other configurations by varying R and L, but the results are almost the same. These suggest that attentive STSP is not sensitive to the window function.

5.5.3 Impact of STFT Length

In Section 5.3.2, we used STFT to exploit the local stationarity of temporal features for aggregation. Although each frame at the output of the last convolutional layer (Layer 5 in Table 2.1) contains the information of 15 speech frames, we cannot guarantee that the CNN's outputs are locally stationary. Because it is difficult to quantify the degree of local stationarity in the convolutional feature maps, we varied the STFT length L to investigate its influence on the performance of STSP. In the

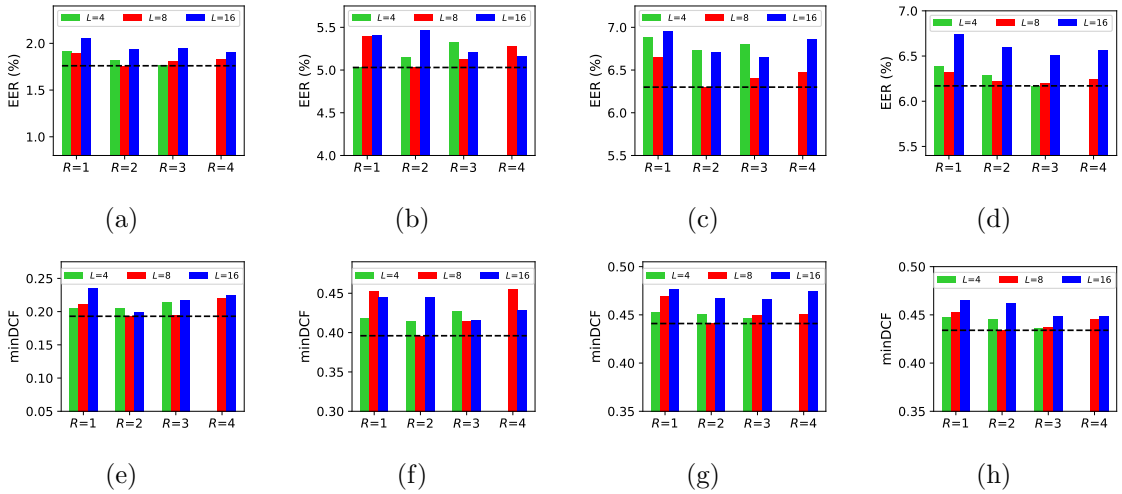
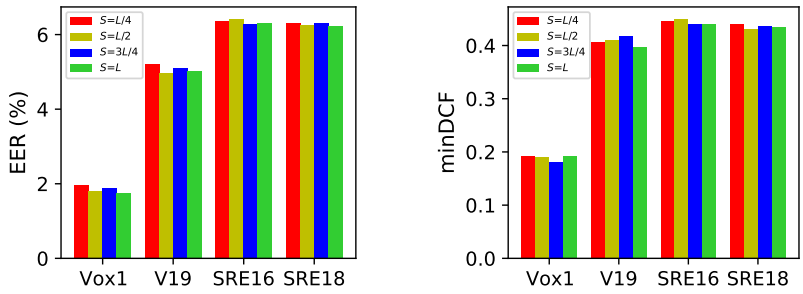


Figure 5.6: Performance of attentive STSP on (a) and (e) VoxCeleb1, (b) and (f) VOiCES19-eval, (c) and (g) SRE16-eval, and (d) and (h) SRE18-CMN2-eval with various STFT lengths under the setting $H = 1$ and $L = S$, where L and S are the STFT length and the step size of the sliding window, respectively. The black dashed line indicates the best result in the individual subfigure.

following experiments, the step size S of the window function is equal to L.

As shown in Figs. 5.6(a)–5.6(h), attentive STSP consistently achieves the best performance when $L = 8$ on all evaluation tasks. When L further increases to 16, the performance degrades in most cases, especially on SRE16-eval and SRE18-CMN2-eval. We hypothesize that the performance degradation is caused by the violation of the local stationarity required by STFT. When the STFT length approaches 16, the local stationarity of STFT may not hold and thus it would be difficult to obtain effective local information. Another disadvantage of using $L = 16$ is that because there are more spectral components in the frequency domain than those under $L = 8$, we need a larger R to include sufficient speaker information in the aggregated embeddings. This is not favorable for aggregation. Therefore we did not account for the case where L is larger than 16.

Interestingly, the best results under $L = 4$ are comparable with those under $L = 8$



(a) (b)

Figure 5.7: (a) EER and (b) minDCF of attentive STSP on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval with respective to the step sizes of the sliding window under $L = 8$, $R = 2$, and $H = 1$.

for VoxCeleb1, VOiCES19-eval, and SRE18-CMN2-eval. However, on SRE16-eval, the setting of $L = 8$ remarkably outperforms the case of $L = 4$. Although the local stationarity is largely satisfied under $L = 4$, there are insufficient components to hold speaker information in the spectral domain. Note that when $L = 4$, we can only have 3 spectral components in $P_c^h(k)$ because of the symmetry in STFT spectrograms.

From the above analysis, the configuration of $L = 8$ makes a compromise between the local stationarity and the spectral resolution. This is the reason we used $L = 8$ in Section 5.5.1 and Section 5.5.2.

5.5.4 Impact of Step Size

The step size S of windowed segments determines the degree of overlapping between successive segments and the number of segments in a temporal feature sequence. Because these factors can affect the results of STFT, which in turn affects the performance of STSP. To investigate the impact of step size on performance, we fixed the STFT length to 8 and varied S under $R = 2$.

As shown in Figure 5.7(a) and Figure 5.7(b), the step size does not have a substantial impact on the performance of attentive STSP across all the evaluations. This

means that attentive STSP is not sensitive to the step size of the sliding window. However, given fixed L, because a larger S results in a smaller number of windowed segments for a fixed-length feature sequence, the subsequent computational load of the spectral representations will be reduced. Therefore, it is favorable to use $S = L$ in attentive STSP to reduce the computational cost. This is the reason why we used $S = L$ in the former sections.

5.5.5 Effect of $M_c^h(k)$ and $P_c^h(k)$

$M_c^h(k)$ in (5.2) and $P_c^h(k)$ in (5.3) denote the weighted average of the magnitude and energy of the spectrogram along the temporal axis, respectively. A noteworthy observation is that, as shown in Fig. 5.3, $P_c^h(k)$'s contain more energy in the low frequency components than $M_c^h(k)$'s do.[7] This phenomenon can also be observed from the rightmost two plots in the middle row of Fig. 5.2(a), where the number of salient components of $P_c^h(k)$ is smaller than that of $M_c^h(k)$ for all channels. Based on the observation from both figures, we may ask a question: *Is $P_c^h(k)$ more effective than $M_c^h(k)$ for attentive STSP due to its more energy-concentrated property?*

To answer the above question, we modified the procedure of attentive STSP in Section 5.3.2 slightly by either excluding $M_c^h(0)$ or only including $M_c^h(0)$ in (5.4) and (5.5). The results of the modification are shown in Rows 5–9 of Table 5.3. Comparing Rows 1–4 and Rows 6–9, we observe that the attentive STSP without $M_c^h(0)$ obtains comparable results with the standard attentive STSP consistently across all the evaluations under various R's. This observation suggests that once $P_c^h(k)$'s are used in the aggregation process, $M_c^h(0)$ does not offer any effective performance gain. This argument can be further demonstrated by the comparison between Row 5 and Row 6. For example, when $M_c^h(0)$'s are used alone as the aggregated statistics, the performance of attentive STSP degrades substantially, as can be seen in Row 5. In

[7]The magnitude of $P_c^h(k)$'s presents a faster attenuation to zero than $M_c^h(k)$'s.

contrast, using $P_c^h(0)$'s alone (Row 6) remarkably outperforms that using $M_c^h(0)$'s alone (Row 5) across all the evaluations. Therefore, using $P_c^h(0)$ alone is much more effective than using $M_c^h(0)$ only for aggregation.

However, as verified in Section 5.3.4, attentive STSP is a generalized statistics pooling method in that using the DC components of the spectral representations is an analogy to using the means and standard deviations in statistics pooling. Therefore, to make attentive STSP complete and compatible with the historical statistics pooling method, we still keep $M_c^h(0)$ in attentive STSP.

5.5.6 Effect of Test Utterance Duration

From Table 5.1, we observe that compared with the baseline, the performance improvement of attentive STSP on SREs is much larger than that on VoxCeleb1 and VOiCES 2019. This observation suggests that attentive STSP can be more effective on long utterances (SREs) than on short ones (Voxeleb1 and VOiCES 2019). To investigate whether the superior performance gain of attentive STSP is related to the utterance duration or the dataset, we compared the performance of various pooling methods by truncating the test utterances.

In the experiments, we kept the duration of the enrollment utterances unchanged on all datasets. For VoxCeleb1 and VOiCES 2019, we randomly truncated the test utterances into 2s and 5s; whereas the test utterances are truncated to 5s and 20s for SRE16 and SRE18-CMN2. Note that if the duration of the original utterances is less than the target duration, we used the full-length utterances. The results are shown in Table 5.4. For each setting, the performance is an average of 5 runs.

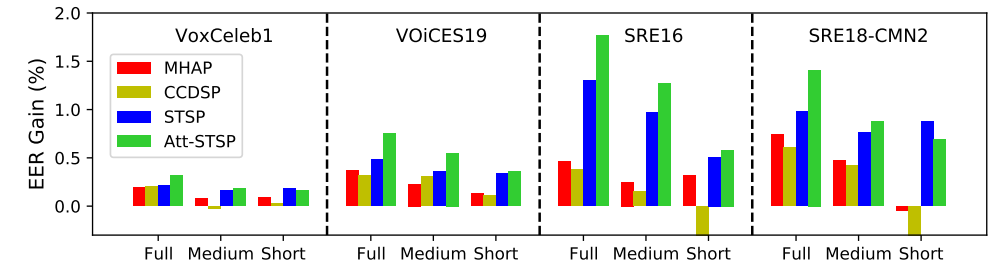
Table 5.3: Comparison of attentive STSP (Rows 1–4) and its modified version (Rows 5–9) with respective to the performance on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval. R is the number of the lowest second-order spectral components ($P_c^h(k)$) in (5.3). The number of attention heads was set to $H = 1$.

Row	Pooling method	No. of paras of emb. sys.	VoxCeleb1		VOiCES19-eval		SRE16-eval		SRE18-CMN2-eval	
			EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
1	Attentive STSP (R=1)	4.23 M	1.89	0.211	5.40	0.453	6.65	0.469	6.32	0.453
2	Attentive STSP (R=2)	4.61 M	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	6.22	**0.434**
3	Attentive STSP (R=3)	5.00 M	1.81	0.195	5.13	0.415	6.40	0.450	**6.20**	0.437
4	Attentive STSP (R=4)	5.38 M	1.83	0.220	5.28	0.455	6.47	0.451	6.24	0.445
5	Attentive STSP w/ only $M_c(0)$ (R=0)	3.85 M	2.46	0.274	6.00	0.478	8.47	0.523	8.17	0.528
6	Attentive STSP w/o $M_c(0)$ (R=1)	3.85 M	1.87	0.201	5.38	0.462	6.54	0.472	6.40	0.458
7	Attentive STSP w/o $M_c(0)$ (R=2)	4.23 M	**1.84**	**0.176**	**5.27**	0.432	**6.35**	**0.455**	**6.16**	0.441
8	Attentive STSP w/o $M_c(0)$ (R=3)	4.61 M	1.88	0.187	5.33	**0.417**	6.42	0.458	6.23	**0.438**
9	Attentive STSP w/o $M_c(0)$ (R=4)	5.00 M	1.87	0.194	5.37	0.434	6.51	0.461	6.27	0.442

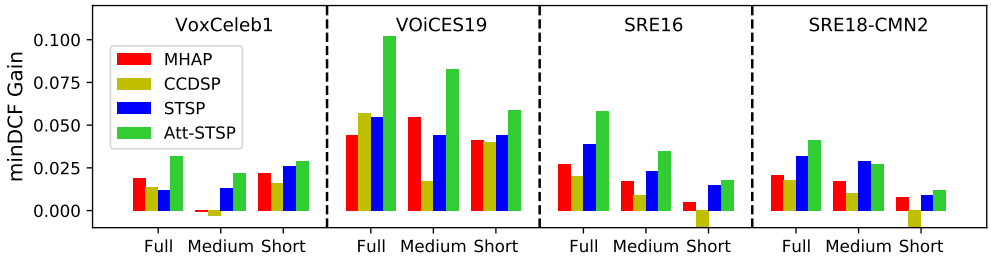
From Table 5.4, we observe that the performance of all pooling methods degrades severely when the test utterances were truncated. Generally, attentive STSP outperforms the other pooling strategies consistently across different test durations. The performance improvement of attentive pooling, CCDSP, STSP, and attentive STSP with respective to statistics pooling is shown in Fig. 5.8. We observe that the performance gain of attentive STSP becomes larger when the test utterances are longer. Besides, the performance improvement of attentive STSP consistently exceeds that of the other pooling methods. For example, in SRE16, when the duration of test utterances increases from short (5s) to medium (20s), the EER improvement ($\mathrm{EER}_{\mathrm{Stats}} - \mathrm{EER}_{\mathrm{Att-STSP}}$) increases from 0.58% to 1.27%. This observation suggests that attentive STSP favors long utterances and that attentive STSP is more resilient to the duration variations in the test utterances.

Table 5.4: Performance of various pooling methods on truncated test utterances. Stats and MHAP refer to statistics pooling and multi-head attentive pooling (H=2), respectively. *Full* means that we used the original duration of the test utterances without truncation. The *medium* duration denotes 5s for VoxCeleb1 and VOiCES19, and 20s for SRE16 and SRE18-CMN2. Similarly, the *short* duration represents 2s for VoxCeleb1 and VOiCES19, and 5s for both SREs.

Row	Test Utt. Dur.	Pooling method	VoxCeleb1		VOiCES19-eval		SRE16-eval		SRE18-CMN2-eval	
			EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
1	Full	Stats	2.08	0.225	5.78	0.498	8.07	0.499	7.63	0.475
2		MHAP	1.89	0.206	5.41	0.454	7.61	0.472	6.89	0.454
3		CCDSP	1.88	0.211	5.46	0.441	7.69	0.479	7.02	0.457
4		STSP	1.87	0.213	5.30	0.443	6.77	0.460	6.65	0.443
5		Att-STSP	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	**6.22**	**0.434**
6	Medium	Stats	2.35	0.264	8.59	0.697	9.52	0.580	9.21	0.571
7		MHAP	2.27	0.265	8.36	0.642	9.27	0.563	8.74	0.554
8		CCDSP	2.38	0.267	8.28	0.680	9.37	0.571	8.79	0.561
9		STSP	2.19	0.251	8.23	0.653	8.55	0.557	8.45	**0.542**
10		Att-STSP	**2.17**	**0.242**	**8.04**	**0.614**	**8.25**	**0.545**	**8.33**	0.544
11	Short	Stats	5.81	0.564	13.80	0.902	16.33	0.760	14.48	0.777
12		MHAP	5.72	0.542	13.67	0.861	16.01	0.755	14.53	0.769
13		CCDSP	5.78	0.548	13.69	0.862	20.52	0.828	15.70	0.834
14		STSP	**5.63**	0.538	13.46	0.858	15.82	0.745	**13.60**	0.768
15		Att-STSP	5.65	**0.535**	**13.44**	**0.843**	**15.75**	**0.742**	13.79	**0.765**



(a)



(b)

Figure 5.8: Improvement in (a) EER and (b) minDCF with respective to statistics pooling. *MHAP*: multi-head attentive pooling; *CCDSP*: channel- and context-dependent statistics pooling; *STSP*: short-time spectral pooling (proposed); *Att-STSP*: attentive short-time spectral pooling (proposed).

Chapter 6

MUTUAL INFORMATION ENHANCED TRAINING FOR SPEAKER EMBEDDING

6.1 Introduction

Studies have indicated that features at the lower layers of a DNN are generally more class-agnostic, while those at the upper layers are more class-specific [135, 136]. Accordingly, the prediction uncertainty decreases when signals flow from the lower layers to the upper layers, suggesting that training is a process of information loss. Therefore, some speaker information will inevitably diminish in the upper layers when training a speaker embedding network. As such, an intuitive way to enhance speaker information in the embeddings is to explicitly maximize the mutual information (MI) between the frame-level features and the segment-level embeddings, so that the embeddings can learn extra speaker information from the more general low-level features.

In fact, exploiting MI to learn meaningful speaker embeddings is not new. In [31], InfoVDANN was introduced to maximize the MI between the transformed embeddings and the input embeddings so that the transformed embeddings are more speaker discriminative. However, this method is operated at the segment level, which forbids it from leveraging useful information in the frame-level layers. Attributed to the deep InfoMax (DIM) [60] framework for representation learning, estimating the MI between the frame-level features and the segment-level embeddings has become feasible via MI neural estimators (MINEs) [137].

In this chapter, we aim to produce informative speaker embeddings through the DIM framework. However, a straightforward implementation of DIM may pose a

dimensionality imbalance problem. Because the dimensionality of the (flattened) frame-level features is generally much larger than that of the speaker embeddings, the learned MI estimators may be biased towards the frame-level features. In this case, MI cannot be accurately approximated, and directly applying MI maximization can fail to learn useful information. Therefore, it would be amenable to perform dimensionality reduction before MI estimation and maximization. Inspired by the squeeze operation in the SE networks [136], the author proposes to performing global pooling on the frame-level features for each channel before maximizing the MI between the frame-level features and the speaker embeddings. The global pooling essentially reduces the dimensionality of the frame-level features, which avoids imbalanced dimensionality in the MI estimation. The author calls the resulting method *squeeze-DIM*, which uses the MI estimation between the squeezed frame-level features and the speaker embeddings as a proxy to that between the original pairs. Different from DIM which aims for unsupervised learning, the author uses squeeze-DIM as a regularizer with the main task being speaker classification.

6.2 Deep InfoMax

MI is a useful tool in representation learning. The performance of representations through unsupervised learning can even be comparable with that through supervised learning on specific tasks [60]. The MI between two random variables X and Y is defined as the KL divergence between their joint distribution and the product of their marginals:

$$I(X;Y) = D_{\mathrm{KL}}(P(X,Y)\|P(X)P(Y)). \tag{6.1}$$

However, MI is difficult to estimate, especially when X and Y locate in a continuous, high-dimensional space. To scale with the dimension, various variational bounds are introduced in combination with neural networks when estimating MI. For the scenario where the objective is to learn encoded representations $Y = E_\phi(X)$ from the input

X, we maximize the MI between the input and output of the encoder [60, 138]:

$$(\hat{\phi}, \hat{\theta}) = \underset{\phi,\theta}{\operatorname{argmax}} I_\theta \left(X; E_\phi(X)\right), \tag{6.2}$$

where ϕ and θ parameterize the encoder E_ϕ and the MI estimator I_θ, respectively.

There are several popular variational lower bounds on MI [137,139,140]. The basic idea behind these bounds is that if we can train a discriminator (MI estimator) that is able to accurately differentiate the samples drawn from the joint distribution and those from the product of the marginals, we obtain a good estimate of the true MI.

One well-known variational estimator is called InfoNCE [139, 141]:

$$\begin{aligned} I(X;Y) &\geq \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{f(\mathbf{x}_i,\mathbf{y}_i)}}{\frac{1}{B}\sum_{j=1}^{B} e^{f(\mathbf{x}_j,\mathbf{y}_i)}}\right] \\ &\triangleq I_{\text{InfoNCE}}(X;Y), \end{aligned} \tag{6.3}$$

where the expectation is over B independent samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{B}$ drawn from the joint distribution $P(\mathbf{x}, \mathbf{y})$ and B is the mini-batch size. $f(\cdot, \cdot)$ denotes the *critic*, which takes a pair of samples and outputs a scaler score. Common critics can be a bilinear function $f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{W} \mathbf{y}$ where $\mathbf{W}$ is a weight matrix to be learned, a separable function $f(\mathbf{x}, \mathbf{y}) = g_{\theta_1}(\mathbf{x})^\top g_{\theta_2}(\mathbf{y})$ where $g_{\theta_1}(\cdot)$ and $g_{\theta_2}(\cdot)$ are functions characterized by networks with parameters θ_1 and θ_2, respectively, and a concatenated function $f(\mathbf{x}, \mathbf{y}) = h_\theta([\mathbf{x}, \mathbf{y}])$ where $h_\theta(\cdot)$ denotes a network parameterized by θ [139].

Because $I_{\text{InfoNCE}}(X;Y)$ is a multi-sample lower bound, it has low variance. However, $I_{\text{InfoNCE}}(X;Y)$ is biased and is upper bounded by $\log B$, which means that this bound will be loose when the true MI $I(X;Y) > \log B$.

Another MI estimator is based on the variational f-divergence estimation special-

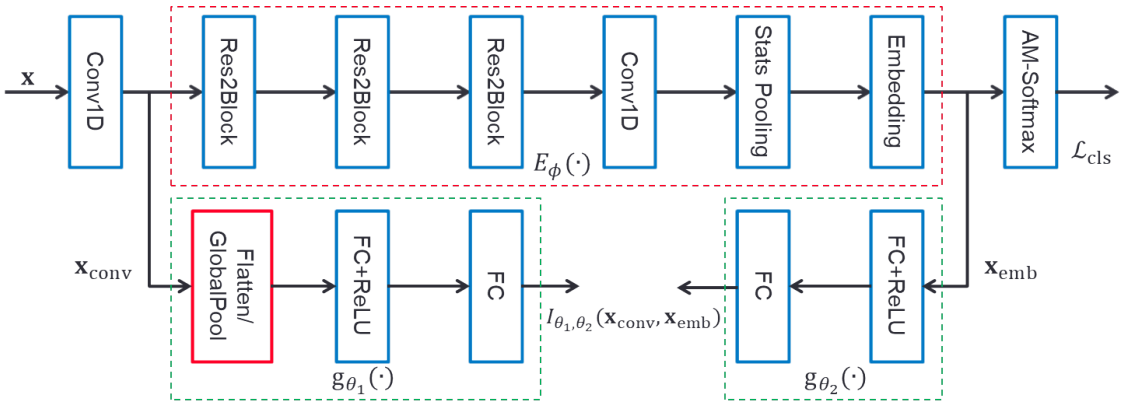


Figure 6.1: Schematic of MI-enhanced training. The whole network comprises two sub-networks: an upper speaker classifier C_ω and a lower MI estimator I_{est}. The MI estimator is instantiated by a separable critic as in (6.6) and (6.10) for DIM regularization and squeeze-DIM regularization, respectively. For DIM regularization, a flatten layer is used as the first layer of g_{θ_1}, while a global pooling layer is applied for squeeze-DIM regularization. FC denotes the fully-connected layer.

ized to KL divergence (f-GAN KL) [142]:

$$\begin{aligned} I(X;Y) &\geq \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[f(\mathbf{x},\mathbf{y})] - \mathbb{E}_{p(\mathbf{x})p(\mathbf{y})}\left[e^{f(\mathbf{x},\mathbf{y})-1}\right] \\ &\triangleq I_{\text{NWJ}}(X;Y). \end{aligned} \tag{6.4}$$

$I_{\text{NWJ}}(X;Y)$ is unbiased but presents high variance [139].

There are also other MI estimation such as the non-linearly interpolated lower bound [139] and the smoothed mutual information lower-bound [140]. These estimators have a better bias-variance trade-off than $I_{\text{InfoNCE}}(X;Y)$ and $I_{\text{NWJ}}(X;Y)$.

6.3 DIM Regularized Speaker Embedding

We adopt the DIM framework as a regularization on the embeddings so that more low-level information can be incorporated in the embeddings during training. As shown in Figure 6.1, there are two branches in the MI-enhanced training. The upper

branch represents a standard speaker classification task, while the lower is a DIM regularizer.

Let $\mathbf{x}$, $\mathbf{x}_{\text{conv}}$, $\mathbf{x}_{\text{emb}}$ be the input acoustic feature vectors, the immediate convolutional feature maps, and the speaker embeddings, respectively. Without loss of generality, we use a separable function as the *critic* in the MI estimator, although other *critic*s can also be applied. To preserve extra information in $\mathbf{x}_{\text{emb}}$, we maximize the MI between $\mathbf{x}_{\text{conv}}$ and $\mathbf{x}_{\text{emb}}$ as in (6.2):

$$(\hat{\phi}, \hat{\theta}_1, \hat{\theta}_2) = \underset{\phi,\theta_1,\theta_2}{\operatorname{argmax}} I_{\theta_1,\theta_2}(X_{\text{conv}}; X_{\text{emb}}), \tag{6.5}$$

where ϕ parameterizes the encoding network (within the red dashed box in Figure 6.1) between $\mathbf{x}_{\text{conv}}$ and $\mathbf{x}_{\text{emb}}$, i.e., $\mathbf{x}_{\text{emb}} = E_\phi(\mathbf{x}_{\text{conv}})$. θ_1 and θ_2 constitute the MI estimator with a separable *critic* as follows:

$$f(\mathbf{x}_{\text{conv}}, \mathbf{x}_{\text{emb}}) = g_{\theta_1}(\text{Flatten}(\mathbf{x}_{\text{conv}}))^\top g_{\theta_2}(\mathbf{x}_{\text{emb}}). \tag{6.6}$$

The MI estimator I_{θ_1,θ_2} can be I_{InfoNCE} and I_{NWJ} in (6.3) and (6.4), respectively.

Denote the classification loss in the upper branch of Figure 6.1 as

$$\mathcal{L}_{\text{cls}}(\omega) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log C_\omega(\mathbf{x}_{ik}), \tag{6.7}$$

where y_{ik} is an element of the one-hot speaker labels, and $C_\omega(\cdot)$ represents the whole speaker classifier parameterized by ω. N and K denote the number of training samples and the number of speakers, respectively. Note that the parameter of the encoder ϕ is a subset of ω. If we define the total loss of the network as

$$\mathcal{L}(\omega, \theta_1, \theta_2) = \mathcal{L}_{\text{cls}}(\omega) - \alpha I_{\theta_1,\theta_2}(X_{\text{conv}}; X_{\text{emb}}), \tag{6.8}$$

where α is a hyperparameter weighting the contribution of MI regularization, then

MI-enhanced training can be expressed as follows:

$$(\hat{\omega}, \hat{\theta}_1, \hat{\theta}_2) = \underset{\omega,\theta_1,\theta_2}{\operatorname{argmin}} \mathcal{L}(\omega, \theta_1, \theta_2). \tag{6.9}$$

6.4 Squeeze-DIM Regularized Speaker Embedding

One problem of the DIM regularized speaker embedding is that $I_{\theta_1,\theta_2}(X_{\text{conv}}; X_{\text{emb}})$ can be unreliable when the dimensionality of the flattened $\mathbf{x}_{\text{conv}}$ is much larger than that of $\mathbf{x}_{\text{emb}}$, because the *critic* $f(\mathbf{x}_{\text{conv}}, \mathbf{x}_{\text{emb}}) = g_{\theta_1}(\text{Flatten}(\mathbf{x}_{\text{conv}}))^\top g_{\theta_2}(\mathbf{x}_{\text{emb}})$ would be biased towards learning the information in $\mathbf{x}_{\text{conv}}$ only, other than the *mutual* information between X_{conv} and X_{emb}.

To address the problem of dimensionality imbalance, we propose to squeeze $\mathbf{x}_{\text{conv}}$ using some global pooling methods for each channel before MI estimation. In this case, the *critic* becomes

$$f(\mathbf{x}_{\text{conv}}, \mathbf{x}_{\text{emb}}) = g_{\theta_1}(\text{GlobalPool}(\mathbf{x}_{\text{conv}}))^\top g_{\theta_2}(\mathbf{x}_{\text{emb}}). \tag{6.10}$$

Common global pooling operations can be global average pooling, statistics pooling [3], attentive pooling [40, 41], etc. The optimization is the same as (6.9).

Note that $\mathbf{x}_{\text{conv}}$ does not necessarily have to be at the output of the first convolutional layer as shown in Figure 6.1. Instead, it can be the output of any frame-level layers, and it can even be the input acoustic features.

In conclusion, the only difference between the proposed embedding and the DIM regularized embedding in Section 6.3 is that the former applies a channel-wise global pooling operation on the convolutional feature maps instead of flattening them. The squeeze operation facilitates the MI estimation although it introduces some information loss, which may explain the empirical performance improvement in Section 6.6.

6.5 Experimental Setup

We evaluated the performance of squeeze-DIM on the VoxCeleb1 test set (clean) [13] and the VOiCES 2019 development and evaluation sets [132].

6.5.1 Training of Speaker Embedding Extractor

Both VoxCeleb1 development and VoxCeleb2 development data were used for training, which amounts to around 2.1 million utterances from 7,185 speakers. We followed the Kaldi's VoxCeleb recipe to prepare the training data, i.e., using 40-dimensional filter bank features, performing energy-based voice activity detection, implementing augmentation (by adding reverberation, noise, music, and babble to the original speech files), applying cepstral mean normalization with a window of 3 seconds, and filtering out utterances with a duration less than 4 seconds.[1] Totally, we had approximately twice the number of clean utterances for training the embedding network.

The embedding extractor in the upper branch of Figure 6.1 is used as the baseline. The number of output filters of the convolutional layers (or blocks) is 512 except that it is 1,536 for the last convolutional layer. The kernel sizes of the convolutions are 5, 3, 3, 3, and 1, respectively, and the dilation rates are 1, 2, 3, 4, and 1, respectively. The scale and the number of convolutional filters of all three Res2Blocks [34] are 8 and 64, respectively. We used an embedding size of 192. For the MI-enhanced training, we followed the structure in Figure 6.1 and set the number of nodes in all fully-connected layers in the MI estimator (the lower part of Figure 6.1) to 32. I_{InfoNCE} in (6.3) was used as the MI estimator in our experimental setups because we found that it is more stable to optimize than I_{NWJ} (see (6.4)) and other MI estimators [139, 140]. The hyperparameter α for weighting the MI estimation was set to 0.1. We used a global average pooling layer for the squeeze operation in the squeeze-DIM regularized speaker embedding. To further verify the effectiveness of MI regularization, we also

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

included an embedding that uses the SE block [136] with a reduction factor of 8 as a comparison.

The additive margin softmax loss [74] was used for training. The additive margin and the scaling factor were set to 0.25 and 30, respectively. The mini-batch size was set to 128 and there are around 2,337 mini-batches in one epoch. Each mini-batch was created by randomly selecting speech segments of 2 seconds from the training data. We used an Adam [143] optimizer. The learning rate was initialized to 1.0×10^{-3} and it was decayed by half at Epoch 25. At Epoch 50, we increased the learning rate to 1.0×10^{-3} and decreased it by half again at Epoch 75. Totally, the networks were trained for 100 epochs.

6.5.2 PLDA Training

We used Gaussian PLDA backends [11] for both evaluation tasks. For VoxCeleb1, the PLDA model was trained on the x-vectors extracted from the clean utterances in the training set for the embedding network. For VOiCES 2019, we trained the backend on the concatenated speech with the same video session and used utterances augmented with reverberation and noise. Before PLDA training, the x-vectors were projected onto a 192-dimensional space by LDA for VoxCeleb1 and 150-dimensional space by LDA for VOiCES 2019, followed by whitening and length normalization. The LDA projection matrix was trained on the same dataset as for training the PLDA models. For VOiCES 2019, we also applied adaptive score normalization [133]. The cohort was selected from the longest two utterances of each speaker in the PLDA training data .

6.6 Results and Discussions

The main result of MI-enhanced training is shown in the upper part (Rows 1–4) of Table 6.1. We can see that DIM regularized embedding only achieves marginal im-

Table 6.1: Performance on VoxCeleb1, VOiCES19-dev, and VOiCES19-eval. The upper part (Rows 1–4) is the main result of MI-enhanced training, while the lower part (Rows 5–10) shows the ablation study by varying the source of $\mathbf{x}_{\text{conv}}$ in Figure 6.1. The layer in the parenthesis denotes where $\mathbf{x}_{\text{conv}}$ comes from, e.g., '1st conv' means that $\mathbf{x}_{\text{conv}}$ is the output of the first convoulutional layer, etc. DIM and squeeze-DIM represent DIM regularized and squeeze-DIM regularized speaker embedding.

Row	Emb. sys.	# Paras	VoxCeleb1		VOiCES19-dev		VOiCES19-eval	
			EER	minDCF	EER	minDCF	EER	minDCF
1	Baseline	4.70 M	1.89	0.188	2.20	0.275	5.94	0.474
2	DIM	7.98 M	1.94	0.189	1.85	0.236	5.73	0.449
3	squeeze-DIM	4.73 M	**1.60**	**0.161**	**1.79**	**0.229**	**5.12**	**0.399**
4	Baseline + SE [136]	4.77 M	1.74	0.180	1.81	0.236	5.70	0.444
5	squeeze-DIM (input)	4.71 M	1.75	0.185	1.94	0.235	5.75	0.430
6	squeeze-DIM (1st conv)	4.73 M	**1.60**	**0.161**	1.79	**0.226**	**5.12**	**0.399**
7	squeeze-DIM (2nd conv)	4.73 M	1.67	0.178	**1.76**	0.229	5.45	0.412
8	squeeze-DIM (3rd conv)	4.73 M	1.82	0.190	1.97	0.232	5.56	0.420
9	squeeze-DIM (4th conv)	4.73 M	1.97	0.198	2.01	0.230	5.62	0.426
10	squeeze-DIM (5th conv)	4.76 M	1.88	0.192	2.15	0.234	5.71	0.436

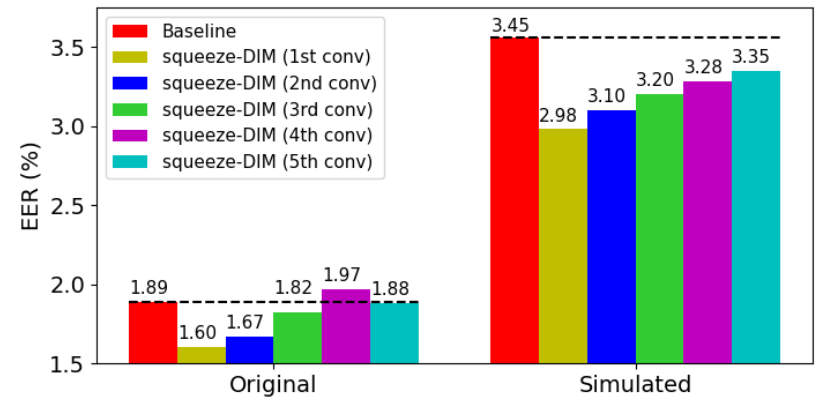
provement over the baseline on VOiCES 2019, whereas the squeeze-DIM regularized embedding remarkably outperforms the DIM regularized version on all tasks. Although DIM regularization should theoretically incorporate more information in the embeddings than the baseline, the practical implementation of the MI estimator can be severely biased towards the information of the convolutional feature maps due to their higher dimensionality than the embeddings. As a result, the MI estimation is unreliable and this limits the regularization effect on the speaker embeddings. In contrast, although the squeeze operation can introduce information loss, it facilitates the MI estimation, which helps feed useful information from the low-level features into the embedding. This verifies the motivation of the proposed squeeze-DIM regularization. By comparing Row 3 and Row 4, we observe that squeeze-DIM is more effective than applying SE, which further verifies the effectiveness of squeeze-DIM.

The lower part (Rows 5–10) of Table 6.1 shows the performance by varying the source of $\mathbf{x}_{\text{conv}}$ in Figure 6.1. In general, we can achieve the best performance if $\mathbf{x}_{\text{conv}}$

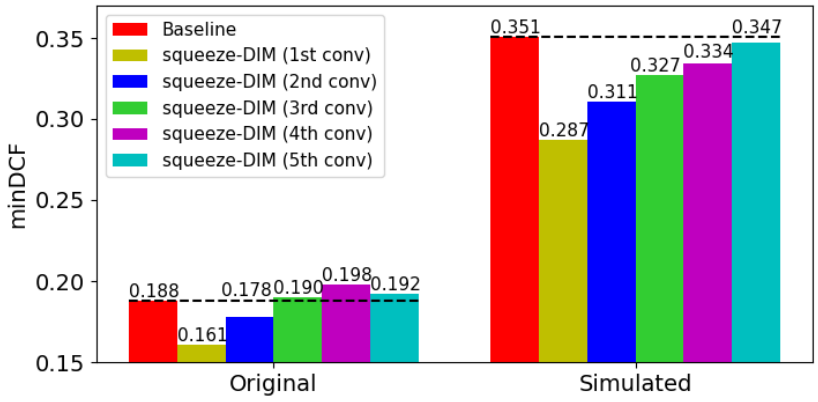
is the output from the first frame-level layer (Row 6). When $\mathbf{x}_{\text{conv}}$ moves from the first layer to upper layers, the performance degrades gradually. This may be because the information becomes more speaker-specific and attenuates gradually while propagating to the upper layers. We also see that all the squeeze-DIM regularized embeddings can achieve better performance than the baseline on VOiCES 2019. However, when $\mathbf{x}_{\text{conv}}$ comes from the fourth and fifth convolutional layers (Row 9 and Row 10), their performance is slightly worse than the baseline on VoxCeleb1. This suggests that MI maximization is more effective on noisy data and has robustness against adverse conditions.

To investigate whether squeeze-DIM is more effective under adverse conditions, we added simulated noise to the original VoxCeleb1-test data and evaluated the performance on the simulated data. Specifically, we followed the augmentation strategy in the Kaldi's recipe and randomly added noise, babble, music, and reverberation to each utterance of VoxCeleb1-test. The results are illustrated in Figure 6.2. We can see that when $\mathbf{x}_{\text{conv}}$ (see Figure 6.1) moves from the first convolutional layer to upper layers, the performance of squeeze-DIM on the simulated data degrades gradually. However, squeeze-DIM outperforms the baseline consistently. In contrast, squeeze-DIM cannot compete with the baseline on the original data when $\mathbf{x}_{\text{conv}}$ moves to the forth and fifth layers. This observation further verifies the robustness of squeeze-DIM under noisy conditions.

Another interesting observation is that maximizing the mutual information between the inputs and the embeddings is less effective than maximizing the mutual information between the immediate convolutional features and the embeddings, which can be verified by comparing Row 5 and Rows 6–7. This means that although the input filter-bank features contain more information than the middle layers, it is difficult to extract speaker information directly from the input layer.



(a)



(b)

Figure 6.2: Comparison of (a) EER and (b) minDCF of squeeze-DIM between the original VoxCeleb1-test data and the simulated VoxCeleb1-test data. In the legend, the layer in the parenthesis denotes where $\mathbf{x}_{\text{conv}}$ comes from, e.g., '1st conv' means that $\mathbf{x}_{\text{conv}}$ is the output of the first convoulutional layer, etc.

Chapter 7

CONCLUSIONS AND FUTURE WORK

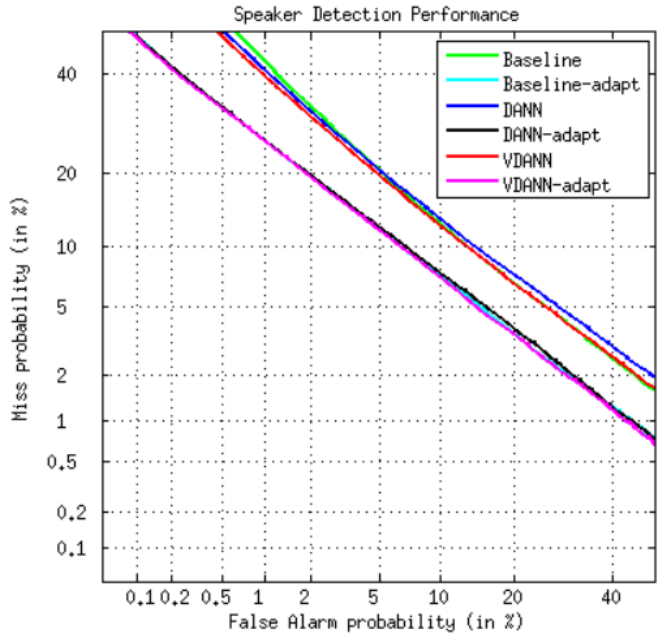
In this thesis, the author introduces three strategies, namely, variational domain adversarial learning, short-time spectral aggregation, and mutual information (MI) enhanced training, to improve the robustness of speaker embeddings against specific adverse environments for SV. In this chapter, we compare the effectiveness of these strategies and give concluding remarks.

7.1 Discussions

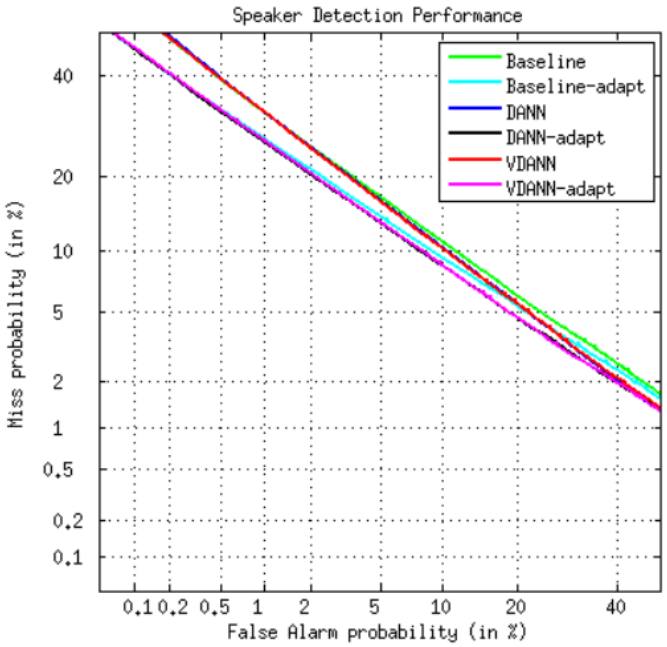
We first investigate which of three strategies is the most effective for SV. Rather than comparing every possible combinations of these strategies, we compare the performance of the representative configurations through DET curves.

7.1.1 Effect of Variational Domain Adversarial Learning

In Section 4.6.1, we have observed that the VDANN-transformed x-vectors can slightly outperform the DANN-transformed x-vectors. This observation suggests that performing Gaussian regularization through a VAE is beneficial to the x-vector/PLDA framework. To further investigate the effect of variational regularization, we compare the x-vectors transformed by VDANN with the original x-vectors on SRE16 and SRE18-CMN2. As shown in Figure 7.1, we observe that there is no significant difference among the original x-vectors, DANN-transformed x-vectors, and VDANN-transformed x-vectors for both SREs. This indicates that the overall improvement of the VDANN/DANN-transformed system over the baseline system is marginal.



(a)



(b)

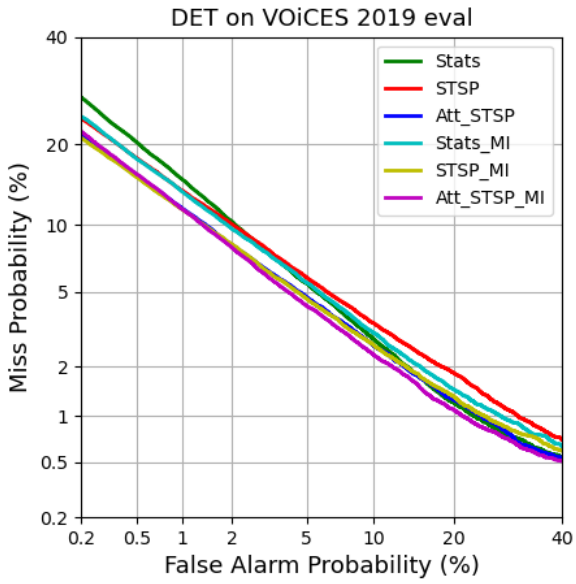
Figure 7.1: Comparison of the original x-vectors, DANN-transformed x-vectors, and VDANN-transformed x-vectors on (a) SRE16-eval and (b) SRE18-CMN2-eval. Baseline refers to the original x-vector system [3] and the networks with "-adapt" means that PLDA adaptation was used as an extra domain adaptation method.

7.1.2 Effect of Spectral Aggregation and Mutual Information Enhanced Training

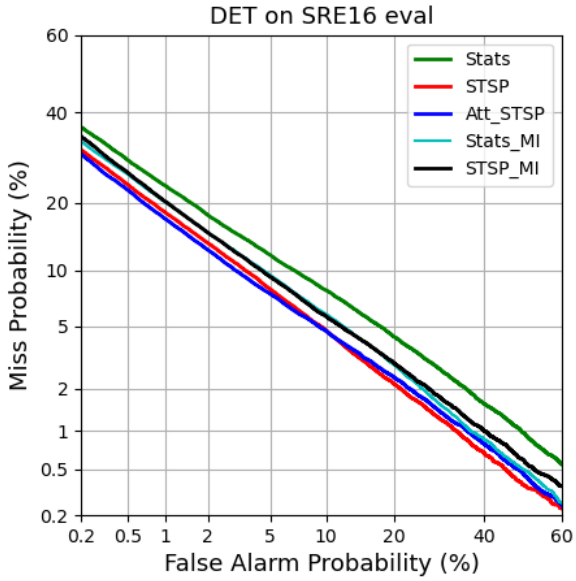
In this section, we investigate the performance of two spectral aggregation strategies and MI-enhanced training on VOiCES19-eval and SRE16-eval. To evaluate the pooling strategies, we used the speaker embedding network detailed in Table 2.1 for VOiCES19-eval, whereas the architecture in the upper branch of Figure 6.1 was used in SRE16-eval. For MI-enhanced training, we used the MI estimator illustrated in the lower branch of Figure 6.1 for both VOiCES19-eval and SRE16-eval. The DET curves are shown in Figure 7.2.

From Figure 7.2(a), we observe that without MI regularization, STSP slightly outperforms statistics pooling, whereas attentive STSP achieves much better performance than both STSP and statistics pooling. When STSP was combined with MI-enhanced training, we can see that MI regularization achieves remarkable performance improvement. However, we do not observe significant performance gain when MI-enhanced training was incorporated into statistics pooling and attentive STSP. Among all these configurations, STSP with MI regularization, attentive STSP, and attentive STSP with MI regularization perform similarly on VOiCES19-eval and these methods largely outperform the other systems. Considering the fact that MI-enhanced training requires an additional MI estimator, which increases the complexity of the speaker embedding system, attentive STSP is recommended in the deployment.

As shown in Figure 7.2(b), without MI-enhanced training, STSP and attentive STSP achieve similar performance on SRE16-eval and both methods substantially outperform statistics pooling. For statistics pooling, performing MI regularization provides further improvement. However, applying MI-enhanced training degrades the performance of STSP. This observation suggests that incorporating MI regularization is not necessarily beneficial when STSP is used. From Figure 7.2, we conclude that in general, attentive STSP provides the largest performance gain among the various combinations between spectral aggregation and MI regularization.



(a)



(b)

Figure 7.2: Comparison between spectral aggregation (STSP and attentive STSP) and MI-enhanced training on (a) VOiCES19-eval and (b) SRE16-eval. Stats and Att_STSP refer to the systems using statistics pooling and attentive STSP as the pooling methods, respectively. The labels with a suffix "MI" represent the systems combining the spectral aggregation strategy with MI-enhanced training.

7.2 Conclusions

In Chapter 4, the author proposed a variational domain adversarial learning framework to jointly address the language mismatch problem and the Gaussianity requirement of the Gaussian PLDA backend. Specifically, two types of DNNs, i.e., the variational domain adversarial neural network (VDANN) and information-maximized VDANN (InfoVDANN), were introduced for unsupervised domain adaptation (DA). The VDANN incorporates a VAE into the conventional DANN to impose a constraint on the distribution of the transformed speaker embeddings so that these embeddings are not only speaker discriminative and domain-invariant, but also conform to a Gaussian distribution. To overcome the potential posterior collapse in VDANNs when training the VAE, the author proposed an InfoVDANN by replacing the VAE with an InfoVAE. InfoVDANN explicitly encourages higher MI between the learned embeddings and the inputs, while simultaneously retaining the advantage of VDANN as a Gaussian distribution regularizer.

Experimental results on SRE16 and SRE18-CMN2 show that both the VDANN and InfoVDANN are capable of reducing domain mismatch through domain adversarial training. Moreover, Gaussianity tests verify the effectiveness of the variational regularization, which validates our motivation. The fact that the InfoVDANN consistently outperforms VDANN suggests that feeding suitable MI into the training of InfoVDANNs is effective for extracting *extra* information for SV. The consistency of the MI estimates on the test datasets also confirms the feasibility of using InfoVDANNs for unsupervised DA.

In Chapter 5, short-time spectral pooling (STSP) and attentive STSP were proposed from a Fourier perspective for better preservation of the speaker information. STSP is able to aggregate both DC components and higher-frequency spectral information into the utterance-level representations, which helps to feed more information into the speaker embeddings. To emphasize on the discriminative segments when com-

puting the spectral representations, attentive STSP was proposed as an extension of STSP. Specifically, attentive STSP exploits two levels of information enhancement strategies during the aggregation process: 1) applying self-attention on the windowed segments of STFT to emphasize on the discriminative information and 2) retaining the low-frequency components in the spectral domain to eliminate the effect of the noisy high-frequency information. Due to these two levels of information preservation, attentive STSP can produce spectral representations with less variations and obtain greater robustness against the non-stationarity in the convolutional feature maps.

Evaluation results on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval show that both STSP and attentive STSP consistently outperform statistics pooling and multi-head attentive pooling, which indicates that it is advantageous to exploit the more stationary spectral statistics during aggregation for robust SV. Moreover, attentive STSP is consistently superior to the vanilla STSP on all evaluation tasks, suggesting that it is beneficial to apply segment-level attention in the spectral domain for SV.

In Chapter 6, MI-enhanced training was proposed to encourage the information flowing from the frame-level feature maps to the speaker embeddings. Rather than directly adopt the DIM framework for regularization, the author introduced a squeeze-DIM regularized embedding to address the problem of dimensionality imbalance between the frame-level features and the embeddings during MI maximization. The evaluation results on both VoxCeleb1 and VOiCES 2019 show that the proposed method outperforms the baseline, DIM regularized embedding, and the SE-integrated embedding, verifying the effectiveness of the proposed method. Besides, the performance gains of squeeze-DIM regularized embedding on VOiCES 2019 are more consistent than those on VoxCeleb1, which suggests that squeeze-DIM regularization is robust under noises and reverberations.

7.3 Future Work

To perform variational regularization, the term $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\right)$ in (3.9) is minimized during the optimization of InfoVDANN. However, the assumption that the prior $p_\theta(\mathbf{z})$ follows a standard Gaussian may be too strong because this means that all $\mathbf{z}$'s drawn from $q_\phi(\mathbf{z}|\mathbf{x})$ would approximate a normal distribution if $q_\phi(\mathbf{z}|\mathbf{x})$ approaches $p_\theta(\mathbf{z})$. One resulting limitation is that posterior collapse may occur while training the VAE, which can lead to non-informative speaker embeddings. This can be addressed by the InfoVDANN method in Chapter 4. Another adverse effect is that the latent variables $\mathbf{z}$'s can only capture single modal information if $p_\theta(\mathbf{z})$ is simply a normal distribution. A possible improvement is to apply a Gaussian mixture model to increase the encoding capacity of the latent features and make the domain transfer more continuous for complicated adaptation. In the future, we will try this idea to find if there are any performance gains through using a mixture of Gaussians as the target distribution in the latent space.

Another possible improvement for variational domain adversarial learning is to try an end-to-end structure instead of transforming the x-vectors. As shown in Table 4.3, we see that end-to-end systems have greater capacity in adapting the speaker embeddings and learning domain invariance. Therefore, end-to-end VDANNs may be a possible solution to further reducing the mismatches among different domains.

In Chapter 5, we have observed that performing aggregation in the spectral domain is more robust against the non-stationarity in the convolutional feature maps than applying temporal aggregation. Nevertheless, STFT is a simple time-frequency transform that cannot achieve good temporal and spectral resolutions simultaneously. Therefore, we may try advanced time-frequency analysis tools such as wavelet analysis to transform the temporal feature maps to the spectral domain, so that better time-frequency details can be captured during utterance-level aggregation.

Finally, we may extend the MI-enhanced training in Chapter 6 by performing

dense squeeze-DIM regularizations on the speaker embeddings. Rather than maximize the MI between the feature maps from a *single* convolutional layer and the speaker embeddings, we simultaneously feed the frame-level information from multiple convolutional layers to the embeddings. In this way, the information fed into the embedding layer would be largely diversified.

BIBLIOGRAPHY

[1] A. Silnova, N. Brümmer, D. Garcia-Romero, D. Snyder, and L. Burget, "Fast variational Bayes for heavy-tailed PLDA applied to i-vectors and x-vectors," in *Proc. Annual Conference of the International Speech Communication Association*, 2018, pp. 72–76.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *International Conference on Learning Representations*, 2014.

[3] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 5329–5333.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, 2015, pp. 448–456.

[5] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 1989, pp. 532–535.

[6] Anil K. Jain and Arun A. Ross, "Introduction to biometrics," in *Handbook of Biometrics*, Anil K. Jain, Patrick Flynn, and Arun A. Ross, Eds., chapter 1, pp. 1–22. Springer, Boston, 2008.

[7] 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.

[8] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2009, pp. 1559–1562.

[9] A. Solomonoff, C. Quillen, and W. M. Campbell, "Channel compensation for SVM speaker recognition," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2004, pp. 57–62.

[10] A. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2006, pp. 1471–1474.

[11] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc. European Conference on Computer Vision*, 2006, pp. 531–542.

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

[13] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, 2020.

[14] C. Richey, M. Barrios, Z. Armstrong, C. Bartels, H. Franco, M. Graciarena, A. Lawson, M. 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. Annual Conference of the International Speech Communication Association*, 2018, pp. 566–1570.

[15] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[16] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "A novel connectionist system for unconstrained handwriting recognition," *Nature*, vol. 521, pp. 436–444, 2015.

[18] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[19] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2016, pp. 5115–5119.

[20] L. Wan, Q. Wang, A. Papir, and I. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 4879–4883.

[21] S. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 171–178.

[22] E. Variani, X. Lei, E. McDermott, I. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2014, pp. 4080–4084.

[23] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 5791–5795.

[24] W. W. Lin, M. W. Mak, and L. Yi, "Learning mixture representation for deep speaker embedding using attention," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2020, pp. 210–214.

[25] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2020, pp. 3830–3834.

[26] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Neural network-based speaker embeddings for end-to-end speaker verification," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 165–170.

[27] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," in *arXiv preprint arXiv:1705.02304*, 2017.

[28] D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, and S. Khudanpur, "The JHU speaker recognition system for the VOiCES 2019 challenge," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2468–2472.

[29] P. Matějka, O. Plchot, H. Zeinali, L. Mošner, A. Silnova, L. Burget, O. Novotný, and O. Glembek, "Analysis of BUT submission in far-field scenarios of VOiCES 2019 challenge," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2448–2452.

[30] W. W. Lin, M. W. Mak, N. Li, D. Su, and D. Yu, "Multi-level deep neural network adaptation for speaker verification using MMD and consistency regularization," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6839–6843.

[31] Y. Z. Tu, M. W. Mak, and J. T. Chien, "Variational domain adversarial learning with mutual information maximization for speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2013–2024, 2020.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[34] S. Gao, M. Cheng, K. Zhao, X. Zhang, M. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, 2021.

[35] S. Yadav and A. Rai, "Frequency and temporal convolutional attention for text-independent speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6794–6798.

[36] Y. Yu and W. Li, "Densely connected time delay neural network for speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2020, pp. 921–925.

[37] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6116–6120.

[38] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[39] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[40] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. Annual Conference of the International Speech Communication Association*, 2018, pp. 2252–2256.

[41] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2018, pp. 3573–3577.

[42] Z. Wang, K. Yao, X. Li, and S. Fang, "Multi-resolution multi-head attention in deep speaker embedding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6464–6468.

[43] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 4305–4309.

[44] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.

[45] Y. Z. Tu and M. W. Mak, "Short-time spectral aggregation for speaker embedding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 6708–6712.

[46] Y. Z. Tu and M. W. Mak, "Aggregating frame-level information in the spectral domain with self-attention for speaker embedding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 944–957, 2022.

[47] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2873–2877.

[48] D. Garcia-Romero, A. McCree, D. Snyder, and G. Sell, "JHU-HLTCOE system for the voxsrc speaker recognition challenge," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 7559–7563.

[49] P. Kenny, "Bayesian speaker verification with heavy tailed priors," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2010.

[50] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Annual Conference of the International Speech Communication Association*, 2011, pp. 249–252.

[51] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2130, 2016.

[52] Y. Z. Tu, M. W. Mak, and J. T. Chien, "Variational domain adversarial learning for speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 4315–4319.

[53] Y. Z. Tu, M. W. Mak, and J. T. Chien, "Information maximized variational domain adversarial learning for speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6449–6453.

[54] I. J. 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.

[55] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," in *International Conference on Learning Representations*, 2019.

[56] S. Zhao, J. Song, and S. Ermon, "Towards deeper understanding of variational autoencoding models," in *arXiv preprint arXiv:1702.08658*, 2017.

[57] Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and A. M. Rush, "Semi-amortized variational autoencoders," in *Proc. International Conference on Machine Learning*, 2018, pp. 2678–2687.

[58] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei, "Avoiding latent variable collapse with generative skip models," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2397–2405.

[59] S. Zhao, J. Song, and S. Ermon, "InfoVAE: balancing learning and inference in variational autoencoders," in *Proc. AAAI Conference on Artificial Intelligence*, 2019, pp. 5885–5892.

[60] R. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[61] Y. Z. Tu and M. W. Mak, "Mutual information enhanced training for speaker embedding," in *Proc. Annual Conference of the International Speech Communication Association*, 2021, pp. 91–95.

[62] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," in *NIPS 2016 End-to-End Learning for Speech and Audio Processing Workshop*, 2016.

[63] M. W. Mak and H. Yu, "A study of voice activity detection techniques for NIST speaker recognition evaluations," *Computer Speech and Language*, vol. 28, pp. 295–313, 2014.

[64] M. Jung, Y. Jung, J. Goo, and H. Kim, "Multi-task network for noise-robust keyword spotting and speaker verification using CTC-based soft VAD and global query attention," in *Proc. Annual Conference of the International Speech Communication Association*, 2020, pp. 931–935.

[65] S. Ding, Q. Wang, S. Chang, L. Wan, and I. Lopez Moreno, "Personal VAD: Speaker-conditioned voice activity detection," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 433–439.

[66] N. Wilkinson and T. Niesler, "A hybrid CNN-BiLSTM voice activity detector," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 6803–6807.

[67] R. Zazo, T. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform CLDNNs for voice activity detection," in *Proc. Annual Conference of the International Speech Communication Association*, 2016, pp. 3668–3672.

[68] W. W. Lin and M. W. Mak, "Wav2Spk: A simple DNN architecture for learning speaker embeddings from waveforms," in *Proc. Annual Conference of the International Speech Communication Association*, 2020, pp. 3211–3215.

[69] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[70] J. Peng, X. Qu, J. Wang, R. Gu, J. Xiao, L. Burget, and J. Černocký, "ICSpk: Interpretable complex speaker embedding extractor from raw waveform," in *Proc. Annual Conference of the International Speech Communication Association*, 2021, pp. 511–515.

[71] J. Jung, H. Heo, J. Kim, H. Shim, and H. Yu, "RawNet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 1268–1272.

[72] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.

[73] H. Muckenhirn, M. Magimai.-Doss, and S. Marcell, "Towards directly modeling raw speech signal for speaker verification using CNNs," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 4884–4888.

[74] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 235–238, 2018.

[75] M. W. Mak, X. Pang, and J. T. Chien, "Mixture of PLDA for noise robust i-vector speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 130–142, 2016.

[76] Aleksandr. Sizov, Kong Aik Lee, and Tomi Kinnunen, "Unifying probabilistic linear discriminant analysis variants in biometric authentication," in *Structural, Syntactic, and Statistical Pattern Recognition*, Pasi Fränti, Gavin Brown, Marco Loog, Francisco Escolano, and Marcello Pelillo, Eds., pp. 464–475. Springer, Berlin Heidelberg, 2014.

[77] Man-Wai Mak and Jen-Tzung Chien, *Machine Learning for Speaker Recognition*, Cambridge University Press, 2020.

[78] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2010, p. Keynote paper.

[79] N. Brummer, A. Silnova, Burget. L., and Stafylakis. T., "Gaussian meta-embeddings for efficient scoring of a heavy-tailed PLDA model," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 349–356.

[80] P. Agrawal, R. Kapoor, and S. Agrawal, "A hybrid partial fingerprint matching algorithm for estimation of equal error rate," in *Proc. In Advanced Communication Control and Computing Technologies*, 2014, pp. 1295–1299.

[81] S. O. Sadjadi, T. Kheyrkhah, and A. Tong, "The 2016 nist speaker recognition evaluation," in *Proc. Annual Conference of the International Speech Communication Association*, 2017, pp. 1353–1357.

[82] A. Martin, G. Doddington, M. Ordowski, and Przybocki. M., "The DET curve in assessment of detection task performance," in *Proc. Eurospeech*, 1997, pp. 1895–1898.

[83] J. Rohdin, T. Stafylakis, A. Silnova, H. Zeinali, L. Burget, and O. Plchot, "Speaker verification using end-to-end adversarial language adaptation," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6006–6010.

[84] Q. Wang, W. Rao, S. Sun, L. Xie, E. S. Chng, and H. Li, "Unsupervised domain adaptation via domain adversarial training for speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 4889–4893.

[85] A. Makhzani, J. Shlens, N. Jaitly, I. J. Goodfellow, and B. Frey, "Adversarial autoencoders," in *International Conference on Learning Representations*, 2016.

[86] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[87] H. Kim and A. Mnih, "Disentangling by factorising," in *Proc. International Conference on Machine Learning*, 2018, pp. 2649–2658.

[88] R. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 2615–2625.

[89] B. Esmaeili, H. Wu, S. Jain, A. Bozkurt, N. Siddharth, B. Paige, D. Brooks, J. Dy, and J. Meent, "Structured disentangled representations," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2525–2534.

[90] A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy, "Fixing a broken ELBO," in *Proc. International Conference on Machine Learning*, 2018, pp. 159–168.

[91] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Science & Business Media, 2008.

[92] K. A. Lee, Q. Wang, and T. Koshinaka, "The CORAL+ algorithm for unsupervised domain adaptation of PLDA," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 5821–5825.

[93] H. Aronowitz, "Inter dataset variability compensation for speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2014, pp. 4002–4006.

[94] M. H. Rahman, A. Kanagasundaram, D. Dean, and S. Sridharan, "Dataset-invariant covariance normalization for out-domain PLDA speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2015, pp. 1017–1021.

[95] J. Alam, G. Bhattacharya, and P. Kenny, "Speaker verification in mismatched conditions with frustratingly easy domain adaptation," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 176–180.

[96] P. Bousquet and M. Rouvier, "On robustness of unsupervised domain adaptation for speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2958–2962.

[97] S. Shon, S. Mun, W. Kim, and H. Ko, "Autoencoder based domain adaptation for speaker recognition under insufficient channel information," in *Proc. Annual Conference of the International Speech Communication Association*, 2017, pp. 1014–1018.

[98] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," in *Advances in Neural Information Processing Systems*, 2006, pp. 513–520.

[99] W. W. Lin, M. W. Mak, and J. T. Chien, "Multi-source i-vectors domain adaptation using maximum mean discrepancy based autoencoders," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 16, no. 12, pp. 2412–2422, 2018.

[100] W. W. Lin, M. W. Mak, Y. Z. Tu, and J. T. Chien, "Semi-supervised nuisance-attribute networks for domain adaptation," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6236–6240.

[101] J. C. Tsai and J. T. Chien, "Adversarial domain separation and adaptation," in *Proc. IEEE International Workshop on Machine Learning for Signal Processing*, Sep. 2017, pp. 1–6.

[102] Z. Meng, J. Li, Z. Chen, Y. Zhao, V. Mazalov, Y. Gong, and B.H. Juang, "Speaker-invariant training via adversarial learning," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 5969–5973.

[103] L. Li, Z. Tang, Y. Shi, and D. Wang, "Gaussian-constrained training for speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6036–6040.

[104] Y. Zhang, L. Li, and D. Wang, "VAE-based regularization for deep speaker embedding," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 4020–4024.

[105] J. T. Chien and K. T. Peng, "Adversarial learning and augmentation for speaker recognition," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 342–348.

[106] M. D. Hoffman and M. J. Johnson, "ELBO surgery: yet another way to carve up the variational evidence lower bound," in *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

[107] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. Conference on Computational Natural Language Learning*, 2016, pp. 10–21.

[108] J. M. Tomczak and M. Welling, "VAE with a VampPrior," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1214–1223.

[109] J. T. Chien and C. W. Wang, "Self attention in variational sequential learning for summarization," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 1318–1322.

[110] NIST, "NIST 2016 speaker recognition evaluation plan," `https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016`, 2016.

[111] NIST, "NIST 2018 speaker recognition evaluation plan," `https://www.nist.gov/itl/iad/mig/nist-2018-speaker-recognition-evaluation`, 2018.

[112] D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey, "Improving speaker recognition performance in the domain adaptation challenge using deep neural networks," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 378–383.

[113] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "Generative adversarial speaker embedding networks for domain robust end-to-end speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6226–6230.

[114] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *The Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[115] M. B. Wilk and R. Gnanadesikan, "Probability plotting methods for the analysis of data," *Biometrika*, vol. 55, no. 1, pp. 1–17, 1968.

[116] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[117] N. M. Razali and Y. B. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests," *Journal of Statistical Modeling and Analytics*, vol. 2, no. 1, pp. 21–33, 2011.

[118] Thomas F. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*, Prentice Hall, 2002.

[119] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 235–238, 1977.

[120] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing (2nd Edition)*, Prentice Hall, 1999.

[121] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[122] B. Scalzo, A. Arroyo, L. Stankovic, and D. Mandic, "Nonstationary portfolios: Diversification in the spectral domain," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 5155–5159.

[123] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proc. International Conference on Machine Learning*, 2019, pp. 5301–5310.

[124] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations*, 2017.

[125] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. Kanwal, T. Maharaj, A. Fischer, A. Courville, and Y. Bengio, "A closer look at memorization in deep networks," in *Proc. International Conference on Machine Learning*, 2017, pp. 233–242.

[126] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, 2006.

[127] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage BERT: A globally normalized BERT model for open-domain question answering," in *Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5878–5882.

[128] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems*, 2020, pp. 17283–17297.

[129] Hynek Hermansky, "Modulation spectrum in speech processing," in *Signal Analysis and Prediction*, A. Procházka, J. Uhlíř, P. W. J. Rayner, and N. G. Kingsbury, Eds., chapter 27, pp. 395–406. Birkhäuser, Boston, 1998.

[130] Mounya Elhilali, "Modulation representations for speech and music," in *Timbre: Acoustics, Perception, and Cognition*, K. Siedenburg, C. Saitis, S. McAdams, A. Popper, and R. Fay, Eds., chapter 12, pp. 335–359. Springer, Cham, 2019.

[131] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.

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

[133] P. Matějka, O. Novotný, O. Plchot, L. Burget, M. Sáchez, and J. Černocký, "Analysis of score normalization in multilingual speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2017, pp. 1567–1571.

[134] F. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[135] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[136] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[137] M. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, and Y. Bengio, "Mutual information neural estimation," in *Proc. International Conference on Machine Learning*, 2018, pp. 531–540.

[138] M. Tschannen, J. Djolonga, P. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," in *International Conference on Learning Representations*, 2020.

[139] B. Poole, S. Ozair, A. van den Oord, A. Alemi, and G. Tucker, "On variational bounds of mutual information," in *Proc. International Conference on Machine Learning*, 2019, pp. 5171–5180.

[140] J. Song and S. Ermon, "Understanding the limitations of variational mutual information estimators," in *International Conference on Learning Representations*, 2020.

[141] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," in *arXiv preprint arXiv:1807.03748*, 2018.

[142] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[143] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

Appendix A

MONTE CARLO ESTIMATE OF MUTUAL INFORMATION

The mutual information (MI) between the latent variable $\mathbf{z}$ and the input $\mathbf{x}$ is defined as the KL divergence between the joint probability distribution $q_\phi(\mathbf{x}, \mathbf{z})$ and the product of their marginal distributions $p_\mathcal{D}(\mathbf{x})q_\phi(\mathbf{z})$

$$\begin{aligned} I_q(\mathbf{x}; \mathbf{z}) &= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{x}, \mathbf{z})}{p_\mathcal{D}(\mathbf{x})q_\phi(\mathbf{z})}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})}\right] \\ &= \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right] \\ &= -\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\left[\mathbb{H}\left[q_\phi(\mathbf{z}|\mathbf{x})\right]\right] - \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right]. \end{aligned} \tag{A.1}$$

In (A.1), $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational posterior in the terminologies of variational autoencoders (VAEs), where ϕ parameterizes the encoder of a VAE; $q_\phi(\mathbf{z})$ is the aggregated posterior, i.e., $q_\phi(\mathbf{z}) = \int_\mathbf{x} p_\mathcal{D}(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})d\mathbf{x}$.

The first term on the right-hand side of (A.1) is the average negative entropy of $\mathbf{z}$'s drawn from $q_\phi(\mathbf{z}|\mathbf{x})$. For a given $\mathbf{x}_s$, we assume $q_\phi(\mathbf{z}|\mathbf{x}_s) = \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_s, \mathrm{diag}\left(\boldsymbol{\sigma}_s^2\right)\right)$, where $\boldsymbol{\mu}_s \equiv \boldsymbol{\mu}(\boldsymbol{x}_s; \phi)$ and $\boldsymbol{\sigma}_s \equiv \boldsymbol{\sigma}(\boldsymbol{x}_s; \phi)$ are the mean and standard deviation outputs of the encoder before the sampling process. Then the negative entropy can be analytically computed as follows:

$$-\mathbb{H}\left[q_\phi(\mathbf{z}|\mathbf{x}_s)\right] = -\frac{1}{2}\sum_{j=1}^{J}\left[1 + \log(2\pi) + \log\sigma_{sj}^2\right], \tag{A.2}$$

where J is the dimension of $\mathbf{z}$.

The second term $\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right]$ can be estimated by Monte Carlo methods as in (4.16):

$$\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{z})\right] \approx \frac{1}{B}\sum_{s=1}^{B}\left[\log \frac{1}{B}\sum_{b=1}^{B} q_\phi\left(\mathbf{z}_s|\mathbf{x}_b\right)\right], \tag{A.3}$$

where B is the mini-batch size and $\mathbf{z}_s$ is a sample from the output of the InfoVAE's encoder given a sample $\mathbf{x}_b$ uniformly sampled from $p_{\mathcal{D}}(\mathbf{x})$, i.e., $\mathbf{z}_s$ is drawn from $\mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_b, \mathrm{diag}\left(\boldsymbol{\sigma}_b^2\right)\right)$.

Finally, the MI is calculated as

$$\hat{I}_q(\mathbf{x};\mathbf{z}) = \frac{1}{B}\sum_{s=1}^{B}\left\{-\frac{1}{2}\sum_{j=1}^{J}\left[1+\log(2\pi)+\log\sigma_{sj}^2\right] - \log\frac{1}{B}\sum_{b=1}^{B} q_\phi\left(\mathbf{z}_s|\mathbf{x}_b\right)\right\}. \tag{A.4}$$

During the testing stage, B can be set to a large value (e.g., 1,024) for an accurate estimate of $I_q(\mathbf{x};\mathbf{z})$.

Alternatively, the MI can be expressed as

$$\begin{aligned} I_q(\mathbf{x};\mathbf{z}) &= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log\frac{q_\phi(\mathbf{x}|\mathbf{z})}{p_{\mathcal{D}}(\mathbf{x})}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z})}\mathbb{E}_{q_\phi(\mathbf{x}|\mathbf{z})}\left[\log q_\phi(\mathbf{x}|\mathbf{z})\right] - \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\mathcal{D}}(\mathbf{x})\right] \\ &= -\mathbb{E}_{q_\phi(\mathbf{z})}\left[\mathbb{H}\left[q_\phi(\mathbf{x}|\mathbf{z})\right]\right] + \log N, \end{aligned} \tag{A.5}$$

where N is the number of test samples uniformly drawn from $p_{\mathcal{D}}(\mathbf{x})$. The term $\log N$ is resulted from the uniform sampling on the data set [106]. As entropy is always positive for random variables, the MI is bounded by $\log N$ from (A.5), which can provide a reference for the MI estimates.

Appendix B

STATISTICAL SIGNIFICANCE TESTS

We conducted McNemar's tests [5] on the SRE performance to test the significance of the performance gain achieved by the InfoVDANN. EER was used as the operating point for hard decisions in the test. As shown in Table B.1, the P-values of the McNemar's tests between both InfoVDANNs and the others are mostly zeros for SRE16 and SRE18-CMN2. This means that the improvement of both InfoVDANNs over VDANN, DANN and the baseline is statistically significant. The "adp" in Table B.1 represents the Kaldi's PLDA adaptation.

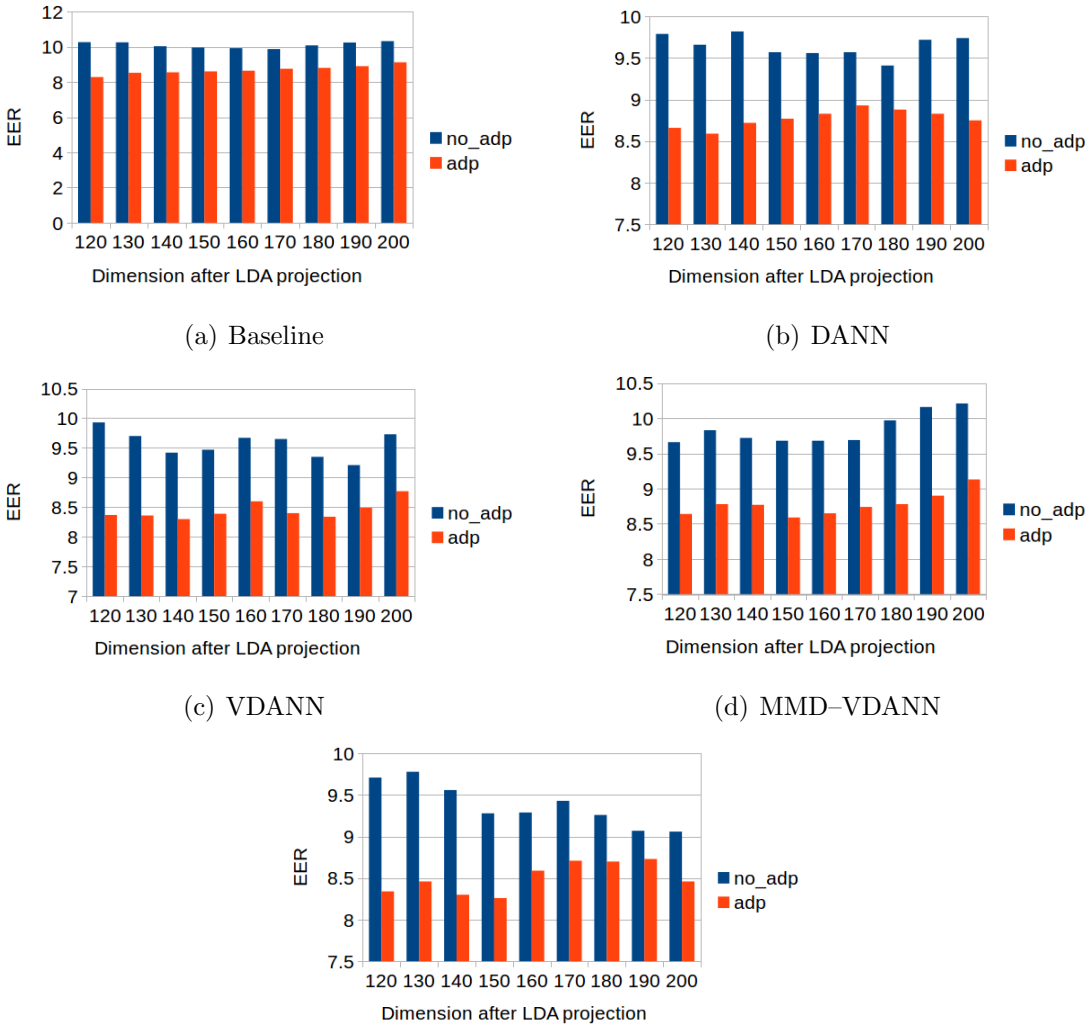
Table B.1: P-values of the McNemar's tests [5]

System1	System2	SRE16, All		SRE18-CMN2	
		w/o adp	w/ adp	w/o adp	w/ adp
MMD–VDANN	baseline	0	0	0	0
AAE–VDANN	baseline	0	0	0	0
MMD–VDANN	DANN	0	0	0	4.44×10^{-16}
AAE–VDANN	DANN	0	0	0	0
MMD–VDANN	VDANN	0	0	0	0
AAE–VDANN	VDANN	0	0	0	2.77×10^{-7}

Appendix C

SELECTION OF THE LDA DIMENSION

The dimension of LDA projection was determined by the EERs evaluated on the development sets. For SRE18-CMN2, we used the SRE18-CMN2 development set to compute the EERs. As shown in Figure C.1, for each system, the optimal dimension is 150 considering the EERs with and without PLDA adaptation. We thus set the dimension of the LDA projection to 150. We have a similar conclusion for SRE16, according to Figure C.2.



(a) Baseline

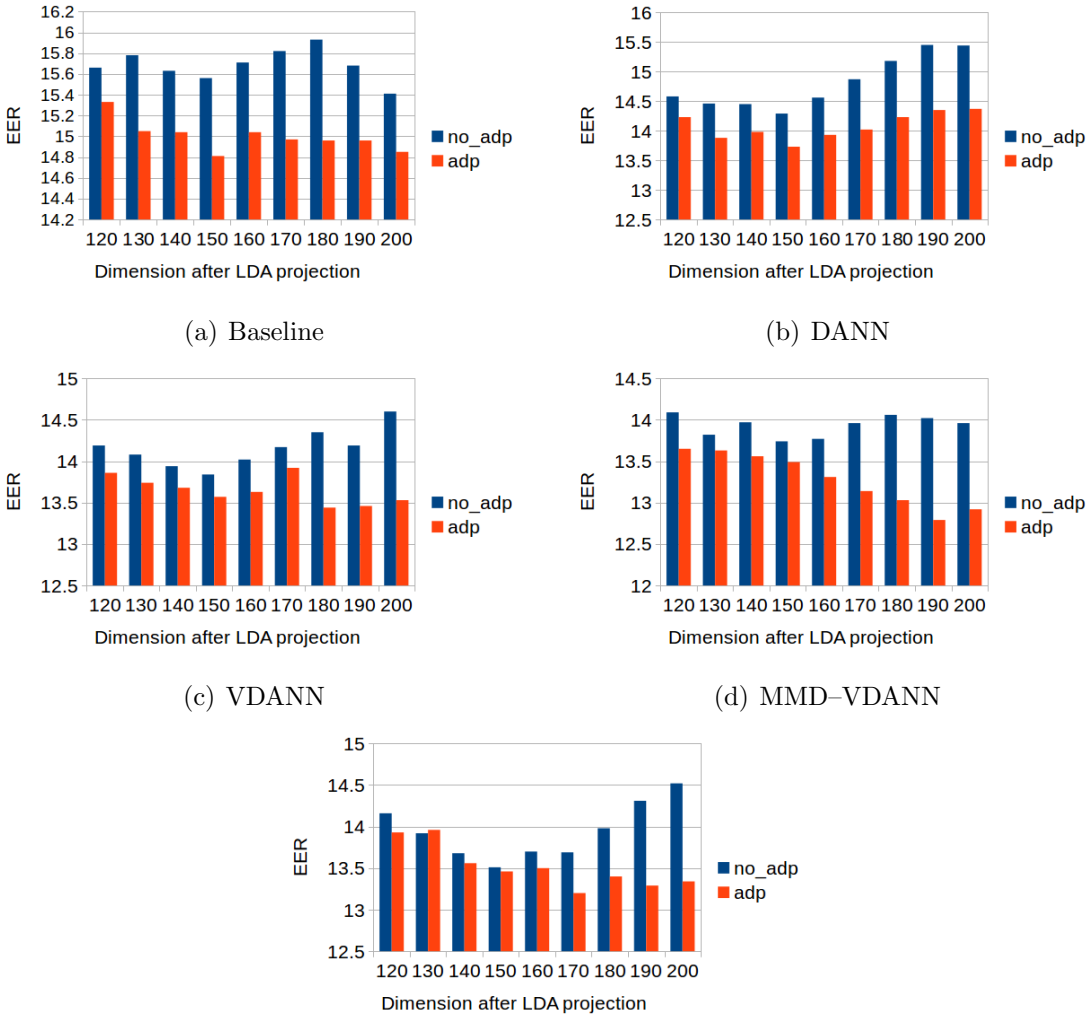
(b) DANN

(c) VDANN

(d) MMD–VDANN

(e) AAE–VDANN

Figure C.1: EERs evaluated on the SRE18-CMN2 development set for different systems



(a) Baseline

(b) DANN

(c) VDANN

(d) MMD–VDANN

(e) AAE–VDANN

Figure C.2: EERs evaluated on the SRE16 development set for different systems