

國立臺灣大學電機資訊學院電機工程學研究所

博士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

進階神經架構於語言、語者暨語音辨識

Advanced Neural Structures for Language, Speaker, and  
Speech Recognition

李鴻欣

Hung-Shin Lee

指導教授: 鄭士康 博士、王新民 博士

Advisors: Shyh-Kang Jeng, Ph.D. and Hsin-Min  
Wang, Ph.D.

中華民國 110 年 10 月

October, 2021



# 國立臺灣大學博士學位論文

## 口試委員會審定書

進階神經架構於語言、語者暨語音辨識

Advanced Neural Structures for Language, Speaker, and  
Speech Recognition

本論文係李鴻欣君（學號 D98921018）在國立臺灣大學電機工程學系完成之博士學位論文，於民國 110 年 10 月 23 日承下列考試委員審查通過及口試及格，特此證明。

口試委員：

鄭士康

王新民

(簽名)

(指導教授)

曹昱

高明達

張智星

李宏毅

系主任

吳忠幟

(簽名)



For my beloved wife, *Hsin-Hua*,  
and adorable daughters, *Annie* and *Carrie*





## Acknowledgements

Thanks be unto God for His *unspeakable* gift.

—2 Corinthians 9:15, KJV

The heavens declare the glory of God;  
and the firmament sheweth his handywork.

Day unto day uttereth *speech*,  
and night unto night sheweth knowledge.

There is no *speech* nor *language*,  
where their *voice* is not heard.

Their line is gone out through all the earth,  
and their *words* to the end of the world.

—Psalms 19:1–4, KJV





# Abstract

This doctoral dissertation focuses on advanced neural structures for phonotactic language recognition, automatic speaker verification, and automatic speech recognition tasks.

The first is a subspace-based neural network (SNN), where each utterance is represented as a linear subspace, and a reasonable manifold-based kernel computation is applied to a orthogonality-constrained layer. SNN is implemented for phonotactic language recognition. The second is a discriminative autoencoder, where the encoder converts the input utterance into an identity embedding, called an identity code that represents the speaker (for speaker recognition) or phone-state (for speech recognition)), and a residual code.

The decoder reconstructs the input acoustic features based on the identity code and the residual code. It is hoped that through training, the speaker or phonetic information is mainly distributed in the identity code. Some experiments conducted on the NIST-LRE 2003, NIST-LRE 2007, NIST-LRE 2009, VoxCeleb-1, NIST-SRE 2016, WSJ, and Aurora-4 datasets show the superiority of our methods to the baselines.

**Keywords:** subspace learning, neural networks, discriminative autoencoders, language recognition, speaker verification, speech recognition







# Contents

	Page
<b>Acknowledgements</b>	<b>5</b>
<b>Abstract</b>	<b>7</b>
<b>Contents</b>	<b>9</b>
<b>List of Figures</b>	<b>13</b>
<b>List of Tables</b>	<b>17</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Subspace-based Neural Networks for Phonotactic Language Recognition</b>	<b>5</b>
2.1 Introduction . . . . .	6
2.2 Phonetic Feature Extraction . . . . .	11
2.3 Subspace Construction . . . . .	12
2.3.1 Phonetic Vector Contextualization . . . . .	13
2.3.1.1 Vector stacking . . . . .	13
2.3.1.2 Dynamic linear modeling . . . . .	14
2.3.2 Orthogonality Pursuit . . . . .	16
2.3.2.1 From a set of stacked vectors to a subspace . . . . .	17
2.3.2.2 From a dynamic linear model to a subspace . . . . .	19
2.3.3 Summary . . . . .	20

2.4	Subspace Learning . . . . .	20
2.4.1	Principal Angles, Projection Kernel, and SVMs . . . . .	21
2.4.2	SNNs . . . . .	23
2.5	Experiment Results and Discussion . . . . .	26
2.5.1	Phone Recognition . . . . .	27
2.5.2	Experiment Flow and Control . . . . .	28
2.5.3	Oracle Experiment on LRE-03 . . . . .	31
2.5.3.1	Effect of sample subspace ratio . . . . .	33
2.5.3.2	Effect of other parameters in PPR-SM-SNN . . . . .	35
2.5.3.3	Learning strategy of PPR-SM-SNN . . . . .	38
2.5.4	Language Recognition on LRE-07 . . . . .	38
2.5.5	Dialect/Accent Identification on LRE-09 . . . . .	41
2.5.6	Comparison with Results in Literature . . . . .	43
2.6	Conclusions . . . . .	45
<b>Chapter 3</b>	<b>Discriminative Autoencoders for Speaker Verification</b>	<b>47</b>
3.1	DcAE based on LDA and i-vectors . . . . .	48
3.1.1	Introduction . . . . .	48
3.1.2	Objectives . . . . .	50
3.1.2.1	Probabilistic linear discriminant analysis . . . . .	50
3.1.2.2	Three PLDA-inspired objective functions . . . . .	51
3.1.3	Realizations . . . . .	53
3.1.4	Experiments and Results . . . . .	55
3.1.4.1	Experiment setup . . . . .	55
3.1.4.2	Results compared with baselines . . . . .	56
3.1.4.3	Observations on DcAE training . . . . .	58

3.1.5	Conclusions . . . . .	59
3.2	DcAE based on Acoustic Features and ResNet . . . . .	60
3.2.1	Introduction . . . . .	60
3.2.2	Proposed Model . . . . .	61
3.2.3	Primitive Experimental Settings and Results . . . . .	62
<b>Chapter 4</b>	<b>Discriminative Autoencoders for Speech Recognition</b>	<b>65</b>
4.1	Exploring the Encoder Layers of DcAE . . . . .	67
4.1.1	Introduction . . . . .	67
4.1.2	Discriminative Autoencoders . . . . .	70
4.1.2.1	Architectures . . . . .	70
4.1.2.2	Objective function . . . . .	72
4.1.3	Experiments . . . . .	74
4.1.3.1	Corpora . . . . .	74
4.1.3.2	Input features . . . . .	76
4.1.3.3	Baseline systems . . . . .	76
4.1.3.4	Proposed systems . . . . .	77
4.1.3.5	Results . . . . .	78
4.1.4	Conclusions and Future Work . . . . .	80
4.2	Roubst DcAE . . . . .	80
4.2.1	Introduction . . . . .	80
4.2.2	Robust Discriminative Autoencoders . . . . .	82
4.2.3	Parallel Robust DcAE (p-DcAE) . . . . .	84
4.2.3.1	Hierarchical robust DcAE (h-DcAE) . . . . .	85
4.2.3.2	Objective functions . . . . .	86
4.2.4	Experiments . . . . .	88



4.2.4.1	The Aurora-4 dataset . . . . .	88
4.2.4.2	Baseline systems . . . . .	89
4.2.4.3	Primitive results . . . . .	90
<b>Chapter 5</b>	<b>Conclusions</b>	<b>93</b>
	<b>References</b>	<b>95</b>
	<b>Appendix A — Subspace Construction</b>	<b>117</b>
	<b>Appendix B — My Publications</b>	<b>121</b>

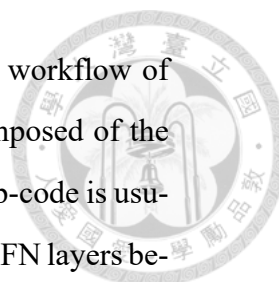




## List of Figures

- 2.1 Phonetic vector extraction for an SLR system with a phonetic repertoire containing only four phones:  $u_1$ ,  $u_2$ ,  $u_3$ , and  $u_4$ . . . . . 11
- 2.2 (Left) Vector stacking for a sequence of  $K$  phonetic vectors with size  $M$ , where the context order is set to 3, and padding with the first vector is adopted. (Right) Dynamic linear modeling for deriving the temporal structure of the phonetic vectors, in terms of linear operators **A** and **C**. . . . . 13
- 2.3 Subspace-based neural networks (SNNs), where 3 subspace inputs are associated with 3 BUT phone recognizers respectively, the output layer is associated with 14 target languages, and  $m$  orthonormal weight maps per input are for kernel computation. . . . . 24
- 2.4 Experiment flowchart, where the orange box with a dashed border shows the procedure of phonotactic modeling, inside which the red rectangle expresses two key steps of our proposed framework based on subspace representation and learning. . . . . 29
- 2.5 EERs on LRE-03 for baselines and two kinds of PPR-SM-based backends with respect to various context orders, sample subspace ratios, and three methods for subspace construction: OLR (2.3), ODL (2.4), and DLM (2.5). 32
- 2.6 EERs on LRE-03 for PPR-SM-SNN in the case of ODL/5/0.6 with respect to various values of  $m$ ,  $\beta$ , and  $\lambda$ . . . . . 33

2.7	Variation of training loss and evaluation EER with respect to the number of training epochs on LRE-03 for PPR-SM-SNN in the case of ODL/5/0.6. The weight map initializer in (a) is orthogonal, while in (b) the Glorot normal initializer is used. Inherited from (a), the learning rate in (c) is reduced from $10^{-4}$ to $10^{-5}$ , while in (d), the drop-based learning rate schedule is adopted. The boxes with dashed edges denote the best evaluation EER, while those with solid edges denote the EER after the last epoch. . . . .	34
2.8	DET curves for different methods, where each method adopts the settings achieving the best performance in Table 2.4. . . . .	39
2.9	Average EERs of stratified 5-fold CV for different methods, (a) the context order $n = 2$ and (b) the context order $n = 5$ . . . . .	40
3.1	Architecture of i-vector based DcAE. . . . .	54
3.2	DET curves of DcAE, PLDA, and cosine for SRE-10. . . . .	57
3.3	Scatter plot of the results by t-SNE for SRE-10. . . . .	57
3.4	EERs of SRE-08 for various settings of hyper-parameters with respect to the training epoch in DcAE. . . . .	58
3.5	Heat map generated from validation results reflecting EERs of SRE-08 in various settings of $\alpha$ and $\beta$ in (3.5). . . . .	59
3.6	Structure of ResNet-based discriminative autoencoders (DcAE) for speaker verification. . . . .	62
4.1	Diagram of the standard GMM-DNN-HMM-based recipe of Kaldi adopted in this study. . . . .	69



4.2 Architectures of different acoustic modeling systems. The workflow of the baseline, denoted by the dotted directional lines, is composed of the input layer, encoder, p-code, and output layer. Note that the p-code is usually the penultimate layer of the baseline, and the additional FFN layers between the p-code and the output layer, representing the triphone states, are optional. The basic version of DcAEs (DcAE-B), expressed by the solid directional lines, adds the r-code, decoder, and output layer for feature reconstruction as its key components. The u-net based DcAEs (DcAE-U) has additional connections between the encoder and decoder layers. . . . 72

4.3 Structure of the parallel robust DcAE (p-DcAE), where P-Code, S-Code, and R-Code denote phoneme-aware code, speaker-aware code, and residual code, respectively. . . . . 83

4.4 Structure of the hierarchical robust DcAE (h-DcAE), where C-Code represents the clean speech information. . . . . 85







## List of Tables

2.1	Summary of our proposed methods for subspace construction. . . . .	21
2.2	Numbers of utterances in different tasks with respect to different languages. TR and TEST denote the training and test data, respectively. . . .	26
2.3	Numbers of utterances used in accent/dialect identification with respect to 8 pairs of languages. . . . .	28
2.4	EERs and $C_{avg}$ 's on LRE-07 for different methods with our own implementations. The lowest values in each column are in bold face. The column "rel. %" shows the relative reduction achieved by PPR-SM-SNN (ODL) over the compared method computed based on their respective best rates among all context orders. . . . .	37
2.5	EERs (%) for LRE'03. #PPRs denotes the number of parallel phone recognizers and the training data sets. . . . .	42
2.6	EERs (%) for LRE'07. LRE'96-05 means LRE'96, 03, and 05. . . . .	44
3.1	Results for SRE-10. The percentages are the relative improvements over PLDA-250. . . . .	56
3.2	Architecture and specifications of ResNet-34, where res denotes the ResNet-based layer, and $B$ , $T$ , $D$ , and $C$ represent the batch size, temporal length, feature dimensionality, and number of training speakers, respectively. . .	63
3.3	Results on the task of VoxCeleb-1, where the models were trained using the training set of VoxCeleb-1. . . . .	64
3.4	Results on the task of NIST SRE-16, where the models were trained using training sets from NIST SRE-04, 05, 06, 08, Fisher, and SwitchBoard. . .	64

4.1	Various models' specifications used in WSJ and MATBN. $D$ , $T$ , $L$ , and $C$ denote the dense, TDNN-based, LSTM-based, and code layers, respectively. The subscript of the layer notation means the skip connection between two layers. For example, $T_{h1}$ and $D_{h1}$ have a common connection $h1$ . The layer-wise context shows the information about splicing indices of TDNN-based layers, which are usually somewhat different between the TDNN and TDNN-LSTM systems. DcAE-B stands for the basic version of DcAEs while DcAE-U means the version utilizing the u-net. . . . .	75
4.2	CERs (%) with respect to two baselines and their relevant DcAE-based variants for MATBN. . . . .	78
4.3	WERs (%) with respect to two baselines and their relevant DcAE-based variants for WSJ. . . . .	79
4.4	WERs (%) on the task of WSJ. The encoder type of DcAE is TDNN-F. . .	91
4.5	WERs (%) on the task of Aurora-4. . . . .	91



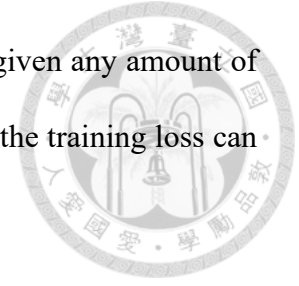
# Chapter 1

## Introduction

There is no doubt that deep learning has revolutionized the development of artificial intelligence (AI), not only in academic but also in commercial applications, by 2021. This must be attributed to the model structure used in deep learning: deep neural networks. Unlike traditional linear or probabilistic models, deep neural networks are able to closely approximate any joint data/target probability distribution or target posterior probability given the data. The data can be sound, pictures, and targets can be categorical objects such as text or numbers of license plates, or regression objects such as more uncluttered speech or clearer pictures.

However, the performance of deep neural networks in machine learning is influenced by two factors that are necessary for AI to excel. First, data. In this era of big data, the development of internet and hardware has made the amount of data transmission and collection more fast and convenient for training data acquisition. The large amount of training data not only reduces the mismatch between training data and test data (because the more training data, the larger possibility that the test data can be covered), but also allows researchers to use neural networks without worrying about the model's capacity

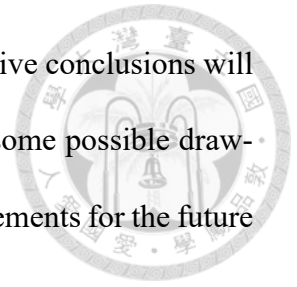
being insufficient for the model to be well trained. In other words, given any amount of training data and suitable capacity and structure of neural networks, the training loss can be effectively and efficiently reduced to nearly zero.



Therefore, in the past decade, researchers have been actively investigating neural networks, especially the use of neural networks to deal with supervised learning problems, i.e., classification problems and regression problems. The current research on neural networks can be broadly divided into three directions. First, the research on the structure, components, and loss functions of neural networks. Compared with the most traditional and simple multi-layer perceptrons (or dense layers), this type of research mainly focuses on creating novel structures and developing novel components, such as the attention mechanism, Transformers, ResNet, etc., or loss functions that are more efficient than the traditional and widely used loss functions (e.g., the softmax plus cross entropy), such as the angular softmax, etc. Second, research on training methods. This type of research focuses on how to make neural network training more efficient and stable, such as other methods different from the traditional stochastic gradient decent, batch normalization, learning rate scheduler, and so on. The third direction is about applications, which tends to use existing neural networks for practical tasks. That is, researchers in this category usually think about how to apply neural networks that have been successful in other fields to their own familiar fields.

Belonging to the second category of researchers, we proposed two novel neural structures, subspace neural networks and discriminative autoencoders, and applied them to three primary tasks of speech processing, i.e., language recognition, speaker verification, and speech recognition. In the remainder of the doctoral dissertation, more theoretical details, including related studies and comparisons, about the two advanced structures will

be illustrated. Relevant experiments, analysis, and a number of tentative conclusions will be described and explored in each chapter. Moreover, we will give some possible drawbacks of our proposed methods and the corresponding way of improvements for the future work.







## Chapter 2

# Subspace-based Neural Networks for Phonotactic Language Recognition

In the recent development of deep learning, the end-to-end (E2E) based neural framework is getting more and more attention and interest. In a strict sense, a learning machine belongs to the E2E framework means that its input can be directly fed by the raw form of data, e.g., the waveform (sampling points) of a speech utterance, and its output straightly associates with the targets, e.g., words, speakers, or yes/no options, without any further pre-processing and post-processing. Although the strict E2E framework is fascinating and easily to be implemented, its power of generalization highly depends on the size of training data. It means, in some situations, the research of the data representation and its corresponding learning mechanism are still necessary and valuable. Therefore, in this study of phonotactic language recognition, we started with representing a speech utterance in a novel way and ended with the construction of its suitable neural structure, instead of building a waveform-in language-out E2E model.

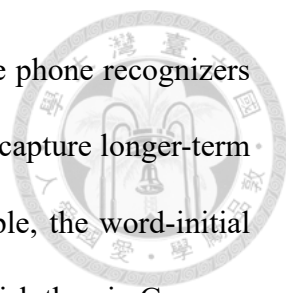
## 2.1 Introduction



Spoken language recognition (SLR) is a branch of audio classification where sound patterns extracted from raw waveform data are typically expressed by discrete symbol sequences or continuous-valued feature vectors. The performance of SLR tasks depends significantly on data representation [80], which can be constructed from three levels of information: acoustic, phonetic, and phonotactic [1, 188]. At the acoustic level, short-time frame-based features, such as mel-frequency cepstral coefficients (MFCCs) [21], linear prediction cepstral coefficients (LPCCs) [91], or shifted delta cepstral (SDC) features [162], are derived from speech signals. Probabilistic models, such as Gaussian mixture models and total variability models [94, 175], are generally used to form a vectorial representation for each utterance to be used by backend classifiers, such as support vector machines (SVMs) [12]. With the great success of deep learning in recent years, acoustic frames are directly fed into deep neural networks (DNNs) [53, 85, 96, 108, 131, 156], recurrent neural networks (RNNs) [64], and long short-term memory [44].

At the phonetic level, phone log-likelihood ratio (PLLR) features are derived from the frame-by-frame phone posteriors provided by a neural network trained for frame-wise phone classification [28]. One of the strengths of the PLLR is that it can directly combine phonetic and acoustic features [26]. In addition, bottleneck features (BNFs), i.e., feature streams generated from the linear bottleneck layer in a deep neural architecture, aim to bridge the gap between acoustic and phonetic levels of information [63, 95, 97, 181]. Evolved BNFs were recently proposed by reformulating the goal to multilabel detection of articulatory attributes. The problem was solved using convolutional RNNs with a maximal figure-of-merit mathematical framework [11, 73].

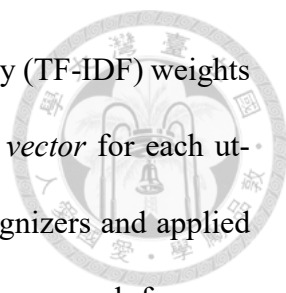




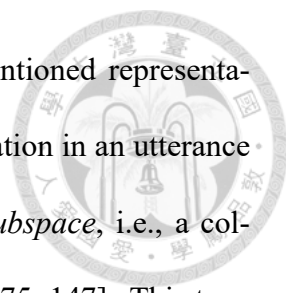
By contrast, phonotactic approaches exploit a single or multiple phone recognizers to convert a speech utterance into one or several phone sequences to capture longer-term and higher-level phonetic information across languages. For example, the word-initial phone /s/ is more typically followed by non-nasal consonants in English than in German. In addition, the issue of rhoticity, whether /r/ is sounded when it follows a vowel, is a phonotactic constraint to identify whether the speaker is from England or the West Country in the United Kingdom [165]. Therefore, if the phonotactic constraint can be sufficiently described for each utterance, the characteristics of each language will be well modeled and better recognition performance will be obtained. In this study, we intend to focus on the use of phonotactic (phonetic-contextual) information, which has been proven to be one of the most effective and convincing clues for SLR [80, 187].

**Distributional and vectorial representations.** Various implementations have been proposed for handling phonotactic information. In these implementations, a phone recognizer is used to tokenize an utterance into a phone sequence. Some researchers have employed phone  $n$ -gram modeling to capture the patterns of phone sequences, where each target language or each utterance can be described as a multinomial *distribution* by using their phone  $n$ -gram statistics. Thus, the similarity between a target language and an utterance can be measured by their negative cross entropy (or KL divergence) [179] or based on the relative positions of  $n$ -grams [23], where discriminative  $n$ -grams are selected to reduce the search space. Rather than using only the best phone sequence, Gauvain *et al.* used the posterior probabilities provided by the phone lattice as soft counts for  $n$ -gram modeling and obtained good results [38].

Stemming from the vector space modeling (VSM) framework in the field of information retrieval, Li *et al.* arranged the phone  $n$ -gram statistics of training and test samples in



terms of  $n$ -gram counts or term frequency-inverse document frequency (TF-IDF) weights into high-dimensional vectors [79]. They built a composite feature *vector* for each utterance by concatenating the vectorized statistics from multiple recognizers and applied the vectors to backend classifiers. Based on the VSM framework, the research focus on phonotactic SLR was shifted to address two main issues: to obtain more discriminative features in the VSM-based representation for classification and to extract a more compact yet informative representation. For the first issue, Richardson and Campbell used SVMs to select key  $n$ -gram terms [132]. Tong *et al.* expanded each existing phone decoder to multiple target-oriented phone recognizers by selecting a subset of phones for each target language that can best discriminate the target language from other languages [161]. Penagarikano *et al.* used time alignment information by considering time-synchronous cross-decoder phone co-occurrences and defined a new concept of multiphone labels to integrate the contributions afforded by multiple decoders in a frame-by-frame manner [121]. For the second issue, some researchers have attempted to apply unsupervised dimension reduction approaches, such as latent semantics analysis (LSA) [79] and principal component analysis (PCA) [99] on the original VSM-based feature vectors to render the classification task more efficient and to avoid the curse of dimensionality when training data are deficient. In addition, owing to the superior performance in the speaker recognition field [24], the idea of i-vectors has been introduced into phonotactic SLR, where each high-dimensional feature vector is projected onto a low-dimensional subspace, forming a compact vector. For instance, inspired by the probabilistic model proposed in [88], Soufifar *et al.* used the subspace multinomial model (SMM) along with the maximum likelihood criterion to effectively represent the information contained in  $n$ -grams [155]. Its latest variant that introduces sparsity constraints is available in [68].



**Subspace-based representations.** In addition to the aforementioned representations, in our previous work, we assumed that the phonotactic information in an utterance can be distilled maximally using phone decoders and borne by a *subspace*, i.e., a collection of mutually orthogonal vectors lying on a specific manifold [75, 147]. This type of data representation was originally proposed by Oja and Kohonen for characterizing a class composed of multiple samples [114]. It has been developed into a broader field that considers the mutuality of subspaces in a kernel or nonkernel manner [163, 178]. Subsequently, it was successfully applied to action recognition [52], image search [172], and visual sequence learning [164]. However, to the best of our knowledge, it has not been applied to speech processing. The subspace-based framework for SLR may be able to more fully utilize phonetic information provided by phone decoders. This is because all phones, not limited to those in the single most likely decoded sequence, can be considered and used to compensate for information loss due to inaccurate phone decoders. This idea is similar to using phone lattices [38] or posterigram-based  $n$ -gram counts [27]. Furthermore, the order of  $n$ -grams or the length of phonotactic constraints are no longer limited by the exponentially increasing memory size required to model phone sequences. Therefore, the contextual relationship among adjacent phones can be sustained with a linear growth rate of space complexity in data representation via vectorial concatenation and matrix factorization [147] or dynamic linear modeling [75] for subspace formulation.

For two subspaces residing in a common Euclidean space, many metrics can be used to evaluate their similarity or distance based on their principal angles (or canonical correlations) [8, 22, 190]. As described in [133], the notion of principal angles might not be the best geometrical definition of *angles* between two subspaces. However, it can still contribute to useful distance functions, some of which are suitable for kernel machines,

such as kernel linear discriminant analysis [143] and SVMs [49], for classification. At this point, all subspaces must be on the same Grassmann manifold, which is a set of fixed-dimensional subspaces in the Euclidean space. Therefore, the kernel functions used in kernel machines or neural networks must be designed in a Grassmannian manner and characterized by two subspaces.

**Contributions.** In this study, we attempt to extend and rectify our prior works on the NIST LRE corpora [75, 147] and propose a more integrated subspace-based framework for SLR by threading three key components.

- 1) **Phonetic feature extraction** derives as much phonetic information as possible by multiple language-dependent phone recognizers for better robustness and generalization.
- 2) **Subspace construction** transforms an utterance into subspace-based representations by considering its phonotactic information. In addition to orthogonality of a subspace, which is a necessary condition that can be easily achieved through singular value decomposition (SVD), we introduce a supplementary constraint of sparsity for *orthogonal dictionary learning*.
- 3) **Subspace learning** involves two types of classifiers, namely SVMs and a newly developed model called the subspace-based neural networks (SNNs). The final score fusion is no longer needed because the subspaces of an utterance generated by parallel phone recognizers are used as multiple inputs by the SNNs. Any two nearly identical input subspaces will get similar outputs. This means that the mapping between the input layer and output layer of the SNNs is mathematically functional, so it is feasible to optimize the entire model using back propagation [137].

**Organization.** The remainder of this chapter is organized as follows. In Section

2.2, we demonstrate the method to leverage phone decoders for gathering phonotactic information in each utterance. Section 2.3 presents several methods for subspace construction. Section 2.4 describes the learning mechanisms for subspaces. We describe our experiments and report the results on three NIST LRE corpora in Section 2.5. Section 2.6 concludes the chapter.

## 2.2 Phonetic Feature Extraction

For the sequence of acoustic vectors,  $\mathbf{o}_1, \dots, \mathbf{o}_T$ , of an utterance, a phone decoder yields the best phone sequence  $p_1, \dots, p_K$  and phone/frame alignment information. The basic idea of our phonotactic data representation is to use the best phone sequence provided by the phone decoder as a type of clustering in the acoustic vectors in a phonetic manner.

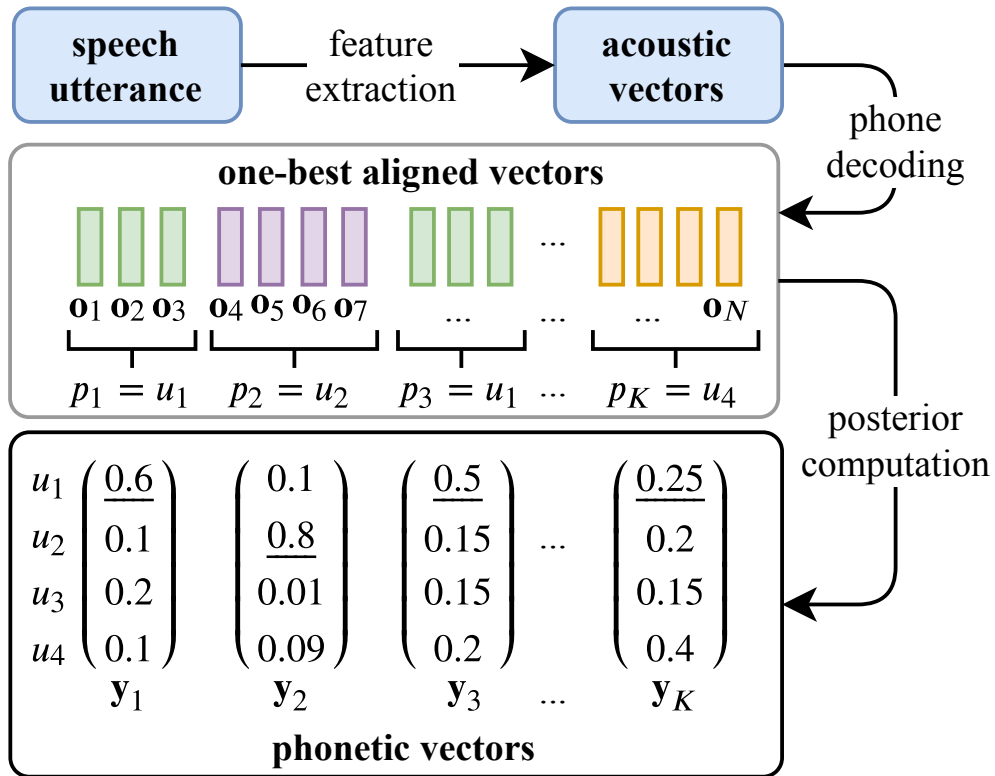
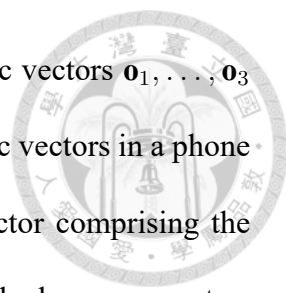


Figure 2.1: Phonetic vector extraction for an SLR system with a phonetic repertoire containing only four phones:  $u_1$ ,  $u_2$ ,  $u_3$ , and  $u_4$ .



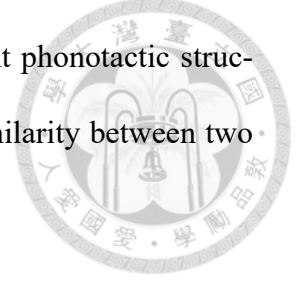
As shown by the example in Fig. 2.1, after phone decoding, acoustic vectors  $\mathbf{o}_1, \dots, \mathbf{o}_3$  are aligned to phone  $u_1$ , whereas  $\mathbf{o}_4, \dots, \mathbf{o}_7$  belong to  $u_2$ . The acoustic vectors in a phone segment are further used to derive a more meaningful phonetic vector comprising the posterior probabilities of individual phone models. Consequently, each phone segment  $p_k$  is represented by a phonetic vector  $\mathbf{y}_k$ , whose dimension is the size of the phone set.

The concept of phonetic vectors is that each utterance is compressed in length from  $T$  acoustic frames to  $K$  phonetic vectors ( $K < T$ ). Each decoder need not be a highly accurate phone recognizer, but a *sound* tokenizer. Moreover, the *sound* boundary need not be consistent with the ground truth if it exists, even though the decoded *sound* pattern might be inexplicable. The task of utilizing uncertainty information and learning patterns of phonotactic constraints is consigned to the procedure of contextualizing phonetic vectors, which is one of the key components in constructing a subspace.

## 2.3 Subspace Construction

A traditional subspace-based learning method is based on separately extracting the most characteristic properties of each class. Specifically, a subspace is represented or spanned by a set of vectors constructed from the feature vectors of each class [114]. The learning mechanism depends on the similarity (or distance) measure between a class subspace and a sample vector. However, most common subspace-relevant methods for SLR, such as the i-vector and PCA, do not belong to this definition. On the contrary, their goal is to derive a coordinate representation (or a vectorial point) for each utterance in a single or joint lower-dimensional linear space. In our proposed framework, each utterance is a subspace. Using techniques such as *phonetic vector contextualization* and *orthogonal*

*pursuit*, each subspace can be constructed by considering the salient phonotactic structure and the orthogonality, which is necessary for measuring the similarity between two subspaces.



## 2.3.1 Phonetic Vector Contextualization

We propose two methods to characterize the contextual relationship between adjacent phonetic vectors in an utterance.

### 2.3.1.1 Vector stacking

Different methods can be used to acquire acoustic dynamics from filter-bank outputs by concatenating adjacent vectors into a larger one that contains longer-term information [7, 74]. Analogously, we vertically stacked each phonetic vector up with its proximal

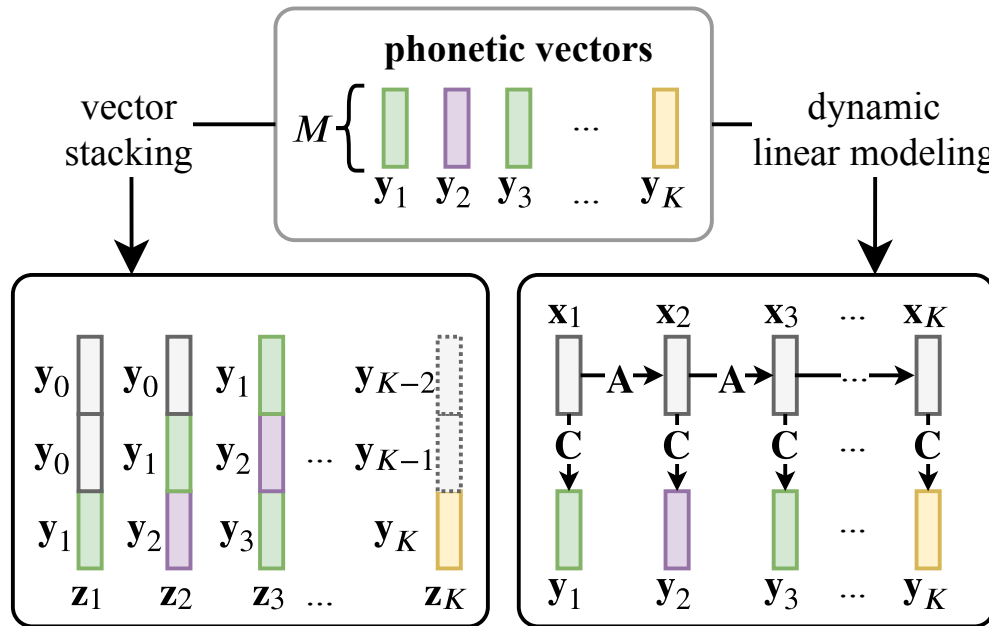
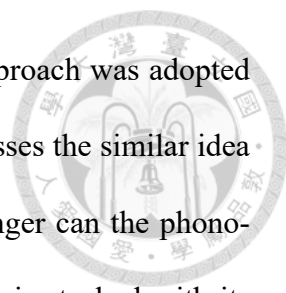


Figure 2.2: (Left) Vector stacking for a sequence of  $K$  phonetic vectors with size  $M$ , where the context order is set to 3, and padding with the first vector is adopted. (Right) Dynamic linear modeling for deriving the temporal structure of the phonetic vectors, in terms of linear operators  $A$  and  $C$ .



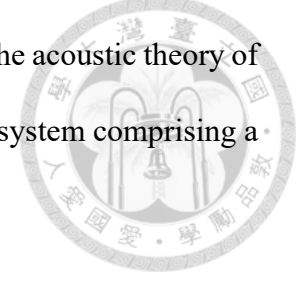
neighbors, forming a higher-dimensional vector [147]. A similar approach was adopted in [50]. As shown in Fig. 2.2 for example, the context order  $n$  possesses the similar idea of the notation  $n$  in “ $n$ -gram,” i.e., the larger the value of  $n$ , the longer can the phonotactic pattern be described. If we set  $n$  to 3, the phonetic vector  $\mathbf{y}_k$  is stacked with its two preceding vectors  $\mathbf{y}_{k-2}$  and  $\mathbf{y}_{k-1}$ , forming a context-dependent super-vector  $\mathbf{z}_k$ . Regarding the phonetic vectors near the beginning of the utterance, such as  $\mathbf{y}_1$  and  $\mathbf{y}_2$ , the non-existing historical vectors can be filled up by zero-padding (used in our experiments) or a duplication of the first vector. Therefore, for a phone set of size  $M$ , the total number of phonotactic features (i.e., number of *trigram* combinations) to represent an utterance with  $K$  phones will reach  $M^3$  under the  $n$ -gram-based framework. However, in vector stacking, it will only be  $3 \times M \times K$ , which is much smaller than  $M^3$ , as most speech utterances in practice are less than 30 s.

### 2.3.1.2 Dynamic linear modeling

Another method to capture the phonetic dynamics is to use state space models (SSMs), which are similar to hidden Markov models, except that the hidden states in SMMs are continuous [102]. Discrete-time dynamic linear models (DLMs) with Gaussian noise are simple versions of SSMs. It is assumed that the state of a process can be summarized by a multivariate variable, which cannot be observed directly at each time step but can be linked to observables using a measurement equation, provided that all the conditional random variables are linear-Gaussian distributed [136]. In the last decade, DLMs have been used to characterize acoustic-feature dynamics and track objects in videos [87, 168]. Inspired by these studies, we assume that each utterance can be generated by a DLM-based system, which may be regarded as a sub-system related to an unknown *language*



*production system* in the human brain. This assumption is similar to the acoustic theory of speech production, which assumes that speech production is a linear system comprising a source and a filter [128].



Let  $\{\mathbf{y}_k\}_{k=1}^K$  be a sequence of  $K$  phonetic vectors of an utterance, where  $\mathbf{y}_k \in \mathbb{R}^M$  and  $M$  denotes the number of phonemes, and  $\{\mathbf{x}_k\}_{k=1}^K$  be the corresponding latent variables representing the hidden states of the system, where  $\mathbf{x}_k \in \mathbb{R}^d$ ,  $d$  denotes the system complexity, and  $d \leq \min(M, K)$ . A DLM can be specified by the following two linear equations:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}, \quad \mathbf{w} \sim N(\mathbf{0}, \mathbf{Q}); \quad (2.1a)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}, \quad \mathbf{v} \sim N(\mathbf{0}, \mathbf{R}). \quad (2.1b)$$

In (2.1),  $\mathbf{w} \in \mathbb{R}^d$  and  $\mathbf{v} \in \mathbb{R}^M$  denote two time-independent noises caused during state evolution and observation generation, respectively. They are white-Gaussian distributed and stochastically independent. In addition to the covariance matrices  $\mathbf{Q}$  and  $\mathbf{R}$ , which can be reduced to identity matrices  $\mathbf{I}_d$  and  $\mathbf{I}_M$  for simplicity, respectively, the system comprises two operators that must be estimated, i.e., the state transition matrix  $\mathbf{A} \in \mathbb{R}^{d \times d}$  and the generative matrix  $\mathbf{C} \in \mathbb{R}^{M \times d}$ . In this regard, the DLM transforms an utterance into a pair of linear operators, i.e.,  $\mathbf{A}$  and  $\mathbf{C}$ , as shown in Fig. 2.2.

To learn the dynamic system, i.e., to identify the parameters  $\mathbf{A}$  and  $\mathbf{C}$ , it is typically assumed that the distribution of the input sequence  $\{\mathbf{y}_k\}_{k=1}^K$  is known. Therefore, we can infer the state sequence  $\{\mathbf{x}_k\}_{k=1}^K$  and estimate the parameters using the expectation-maximization (EM) algorithm based on the maximum likelihood estimation or maximum a posteriori [40]. However, owing to its sensitivity to initial parameter estimates, EM does

not always yield satisfactory results, as described in [102, Chapter 18]. Therefore, we adopted an alternative approach that estimates the parameters by the reconstruction error minimization of (2.1). This approach is known as a subspace method [65], [84, Chapter 7]. We aim to search for a state-space realization that does not require noise modeling. The algorithm used to derive the closed-form solutions of  $\mathbf{A}$  and  $\mathbf{C}$  is shown in Algorithm 1 in Appendix A.

### 2.3.2 Orthogonality Pursuit

The subspace mentioned in this section is not merely a bag of fixed-size vectors; instead, it has its own topological meaning that must be clarified by the concept of a manifold. A manifold  $\mathcal{M}$  is a topological space that is locally homeomorphic to the Euclidean space  $\mathbb{R}^D$  for some  $D$ ; this also represents the dimensionality of the manifold. Hence, a bijective mapping from an open set around each point  $m \in \mathcal{M}$  to an open set in  $\mathbb{R}^D$  exists. A manifold endowed with the differentiable property and a Riemannian metric is called a Riemannian manifold, which extends measures such as lengths and *angles* from familiar Euclidean spaces to a curved space [31]. Two of its applicable members, the Stiefel and Grassmann manifolds, with their embeddings in  $\mathbb{R}^D$  might help us understand the reality of subspaces and their metrics.

The set of  $D \times d$  matrices with orthonormal columns forms the Stiefel manifold  $\mathcal{ST}_{D,d}$ , where  $D \geq d \geq 1$ . In the extreme case,  $\mathcal{ST}_{D,d}$  results in the group of all the  $d \times d$  orthogonal matrices denoted by  $\mathcal{O}_d$  if  $D = d$ . We can express  $\mathcal{ST}_{D,d}$  as

$$\mathcal{ST}_{D,d} := \{\mathbf{S} \in \mathbb{R}^{D \times d} | \mathbf{S}^T \mathbf{S} = \mathbf{I}_d\}. \quad (2.2)$$

Owing to the defined inner product and tangents,  $\mathcal{ST}_{D,d}$ , which is regarded as an embedded manifold in the Euclidean space  $\mathbb{R}^D$ , inherits a canonical Riemannian metric in  $\mathbb{R}^{D \times d}$  [31]. This type of metric enables us to define various geometric notions on the manifold, such as the angle between two curves (or linear subspaces in view of  $\mathbb{R}^D$ ) and the length of a curve. In the following, several methods are proposed to derive a set of orthonormal vectors from stacked vectors or the dynamic linear model of an utterance.

### 2.3.2.1 From a set of stacked vectors to a subspace

Suppose  $K \geq D \geq d$ , the simplest method to obtain  $\mathbf{S} \in \mathbb{R}^{D \times d}$  in (2.2) for an utterance is to seek a set of orthonormal bases that compose the column space, where all contextualized phonetic vectors  $\{\mathbf{z}_k\}_{k=1}^K$  defined in Section 2.3.1.1 are spanned approximately. This objective can be achieved by solving the following orthogonal linear regression (OLR) problem in a matrix-factorization manner, which involves obtaining the best orthogonal projections  $\hat{\mathbf{S}}_{olr}$  while minimizing the reconstruction error [49]:

$$\hat{\mathbf{S}}_{olr} = \arg \min_{\mathbf{S}} \|\mathbf{Z} - \mathbf{S}\mathbf{W}\|_F^2 \quad (2.3a)$$

$$\text{subject to } \mathbf{S}^T \mathbf{S} = \mathbf{I}_d, \quad (2.3b)$$

where  $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_K] \in \mathbb{R}^{D \times K}$ , and  $\|\cdot\|_F$  denotes the Frobenius norm. In fact, because  $d \leq D$ , the unbounded loading matrix  $\mathbf{W} \in \mathbb{R}^{d \times K}$  renders (2.3) an unweighted low-rank approximation problem, which can be solved through truncated SVD, while naturally ensuring the satisfaction of the orthogonality constraint. Therefore, we approximated  $\mathbf{Z}$  by  $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ , where  $\mathbf{\Sigma}$  is a  $d \times d$  diagonal matrix containing the largest  $d$  singular values  $\sigma_1, \dots, \sigma_d$  in descending order, and  $\mathbf{U}$  and  $\mathbf{V}$  are matrices with orthonormal columns  $\{\mathbf{u}_i\}_{i=1}^d$  and  $\{\mathbf{v}_i\}_{i=1}^d$  spanning the column and row spaces of  $\mathbf{Z}$ , respectively. Finally, we

express  $\hat{\mathbf{S}}_{olr} = [\mathbf{u}_1, \dots, \mathbf{u}_d]$  as the ultimate subspace-based representation of an utterance.

To further filter out some utterance-dependent noise or irrelevant information varying among the contextualized vectors  $\{\mathbf{z}_k\}$  and within the space spanned by  $\mathbf{S}$ , we considered the sparsity of the loading matrix  $\mathbf{W}$ , where each column vector  $\mathbf{w}_k$  contains less than  $d$  non-zero values [61]. Therefore, (2.3) can be reformulated as solving the orthogonal dictionary learning (ODL) problem as follows:

$$\hat{\mathbf{S}}_{odl} = \arg \min_{\mathbf{S}} \|\mathbf{Z} - \mathbf{S}\mathbf{W}\|_F^2 + \lambda_{odl}^2 \|\mathbf{W}\|_0 \quad (2.4a)$$

$$\text{subject to } \mathbf{S}^T \mathbf{S} = \mathbf{I}_d, \quad (2.4b)$$

where  $\|\mathbf{W}\|_0$  denotes the sparsity measure defined as the number of nonzero entries in  $\mathbf{W}$ , and  $\lambda_{odl} > 0$  is a scalar regularization parameter that balances the trade-off between the reconstruction error and sparsity. Because  $\mathbf{W}$  is a sparse matrix, only a few column bases in  $\mathbf{S}$  contribute to representing some  $\mathbf{z}_k$  in terms of  $\mathbf{w}_k$ ; the remaining column bases contribute to the restoration of unwanted residual information. In fact, if each  $\mathbf{z}_k$  represents a contextualized sound pattern, a relatively large  $\lambda_{odl}$  in (2.4) guarantees that  $\mathbf{z}_k$  is linearly compounded from only a small number of basic components. This corresponds analogously to the attribute-based SLR approach, where each canonical phone is tokenized by only two types of articulatory attributes, i.e., manner and place [148].

In contrast to traditional dictionary learning,  $\mathbf{S} \in \mathbb{R}^{D \times d}$  in (2.4) is *not* over-complete because of  $d \leq D$ , and has an orthogonal constraint on itself. Therefore, we cannot directly follow conventional procedures for ODL [90]. Propositions for solving (2.4) are presented in Appendix A. The detailed procedure is presented in Algorithm 3.

It is noteworthy that the current OLR and ODL-based subspace construction methods

cannot address short utterances. This is because the number of phonetic vectors (i.e., the length of a phone sequence)  $K$  of an utterance must be greater than or equal to the dimension of the stacked vector,  $D$ . Because a 30-s utterance in general contains 300–500 phones and the sizes of phone sets used in our phone recognizers range from 45 to 61, the condition is always satisfied. The issue of handling short utterances will be addressed in our future research.

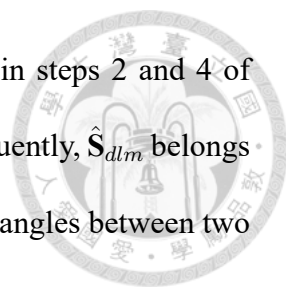
### 2.3.2.2 From a dynamic linear model to a subspace

In [22], Cock and Moor proposed a distance measurement between two dynamic linear models. In this study, we used subspace-based methods to calculate the distance between two DLMs. For the estimates of the state transition matrix  $\hat{\mathbf{A}} \in \mathbb{R}^{d \times d}$  and the generative matrix  $\hat{\mathbf{C}} \in \mathbb{R}^{M \times d}$  derived through Algorithm 1, the corresponding subspace-based representation is defined as

$$\hat{\mathbf{S}}_{dlm} := \left[ \hat{\mathbf{C}}^T, (\hat{\mathbf{C}}\hat{\mathbf{A}})^T, \dots, (\hat{\mathbf{C}}\hat{\mathbf{A}}^{(n-1)})^T \right]^T, \quad (2.5)$$

where  $n$  denotes the context order. Furthermore,  $\hat{\mathbf{S}}_{dlm}$  is named the observability matrix [65] because it characterizes how a state vector  $\mathbf{x}_k$  associated with the  $k$ th sound pattern in an utterance affects the generation of the  $k$ th phonetic vector  $\mathbf{y}_k$  and the next  $n - 1$  phonetic vectors  $\mathbf{y}_{k+1}, \dots, \mathbf{y}_{k+n-1}$ :

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k+1} \\ \vdots \\ \mathbf{y}_{k+n-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{C}} \\ \hat{\mathbf{C}}\hat{\mathbf{A}} \\ \vdots \\ \hat{\mathbf{C}}\hat{\mathbf{A}}^{(n-1)} \end{bmatrix} \mathbf{x}_k. \quad (2.6)$$



Because we have ensured that  $\hat{\mathbf{C}}$  and  $\hat{\mathbf{A}}$  are both orthonormal in steps 2 and 4 of Algorithm 1, we obtained  $\hat{\mathbf{S}}_{dlm}^T \hat{\mathbf{S}}_{dlm} = \mathbf{I}_d$  and  $\text{rank}(\hat{\mathbf{S}}_{dlm}) = d$ . Consequently,  $\hat{\mathbf{S}}_{dlm}$  belongs to a Stiefel manifold, which is necessary for calculating the principal angles between two subspaces.

### 2.3.3 Summary

In summary, three implementations can be performed for subspace construction: OLR (cf. (2.3)), ODL (cf. (2.4)), and DLM (cf. (2.5)). Their goals are summarized in Table 2.1. For OLR and ODL, an utterance is represented by a subspace characterized by  $d$   $D$ -dimensional bases that approximately span the  $K$   $D$ -dimensional stacked phonetic vectors of the utterance. In this case,  $d$  determines the amount of information carried in a subspace. For DLM, the phonetic vectors of an utterance are modeled by a dynamic linear system characterized by  $d$   $D$ -dimensional vectors, where  $d$  denotes the complexity of the system represented by a subspace. Although the physical meanings of  $d$  in both cases differ, the symbol  $d$  reflects a common geometric meaning, i.e., the number of orthogonal bases used to represent an utterance.

In the next section, we present two methods for calculating the distance between two utterances based on their subspace representations.

## 2.4 Subspace Learning

To learn from subspaces, the equivalence of subspaces and their mutual similarity measure must be defined and clarified in advance. Because a subspace may have countless matrix-like expressions, the distance between two subspaces cannot be measured by

Table 2.1: Summary of our proposed methods for subspace construction.

Method	abbr.	Equation
<b>Orthogonal Linear Regression</b> To minimize the reconstruction errors of the augmented phonetic vectors under considerations of the orthogonality and 2-norm regularization.	OLR	(2.3)
<b>Orthogonal Dictionary Learning</b> The same as OLR but with 1-norm regularization.	ODL	(2.4)
<b>Dynamic Linear Modeling</b> To solve the problem of a dynamic linear system by the subspace method.	DLM	(2.5)

matrix-matrix subtraction.

In addition to  $\mathbf{S} \in \mathcal{ST}_{D,d}$ , we assume that  $\mathbf{S} \in \mathcal{G}_{D,d}$ , where  $\mathcal{G}_{D,d}$  denotes the Grassmann manifold defined by

$$\mathcal{G}_{D,d} := \mathcal{ST}_{D,d} / \mathcal{O}_d. \quad (2.7)$$

Here,  $\mathcal{G}_{D,d}$  is the quotient of group  $\mathcal{ST}_{D,d}$  modulo its subgroup  $\mathcal{O}_d$ . Therefore, we can identify an abstract *point* (subspace)  $\mathbf{S} \in \mathbb{R}^{D \times d}$  with orthogonal columns on  $\mathcal{G}_{D,d}$  as an equivalent class under the orthogonal transform of  $\mathcal{ST}_{D,d}$ . Two subspaces  $\mathbf{S}_1, \mathbf{S}_2 \in \mathbb{R}^{D \times d}$  are equivalent if an orthogonal matrix  $\mathbf{Q} \in \mathcal{O}_d$  exists such that  $\mathbf{S}_1 = \mathbf{S}_2 \mathbf{Q}$  [31].

In the following, we present the definition of the principal angle (or canonical correlation) between two subspaces, the corresponding distance function and kernel function applicable to SVMs, and the subspace-based weights for SNNs.

### 2.4.1 Principal Angles, Projection Kernel, and SVMs

For two subspaces  $\mathbf{S}_1 \in \mathbb{R}^{D \times d}$  and  $\mathbf{S}_2 \in \mathbb{R}^{D \times d}$  ( $D \geq d$ ), the  $d$  principal (or minimal) angles between them are recursively and uniquely defined as  $\theta_1, \dots, \theta_d$  in  $[0, \pi/2]$

satisfying

$$\cos \theta_k = \max_{\substack{\|\mathbf{a}\|_2=1 \\ \mathbf{a} \in \text{span}(\mathbf{S}_1)}} \max_{\substack{\|\mathbf{b}\|_2=1 \\ \mathbf{b} \in \text{span}(\mathbf{S}_2)}} \mathbf{a}^T \mathbf{b} = \mathbf{a}_k^T \mathbf{b}_k, \quad (2.8a)$$

$$\text{subject to } \mathbf{a}^T \mathbf{a}_i = \mathbf{b}^T \mathbf{b}_i = 0, i = 1, \dots, d-1. \quad (2.8b)$$

Because the columns of  $\mathbf{S}_1$  and  $\mathbf{S}_2$  form unitary bases of the two subspaces,  $\cos \theta_k$  can be derived by performing SVD on  $\mathbf{S}_1^T \mathbf{S}_2$ , i.e.,  $\mathbf{S}_1^T \mathbf{S}_2 = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ , where  $\mathbf{U} = \{\mathbf{a}_1, \dots, \mathbf{a}_d\}$ ,  $\mathbf{V} = \{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ , and  $\mathbf{\Sigma} = \text{diag}(\cos \theta_1, \dots, \cos \theta_d)$  [8, 43]. The cosine-based similarity between  $\mathbf{S}_1$  and  $\mathbf{S}_2$  can be defined as

$$\text{sim}(\mathbf{S}_1, \mathbf{S}_2) := \sum_{i=1}^d \cos^2 \theta_i. \quad (2.9)$$

Because the Euclidean distance between two vectors can be calculated by the dot product, i.e.,  $\|\mathbf{u} - \mathbf{v}\|^2 = \mathbf{u}^T \mathbf{u} + \mathbf{v}^T \mathbf{v} - 2\mathbf{u}^T \mathbf{v}$ , we can represent (2.9) as a distance function:

$$\begin{aligned} \text{dist}^2(\mathbf{S}_1, \mathbf{S}_2) &:= \text{sim}(\mathbf{S}_1, \mathbf{S}_1) + \text{sim}(\mathbf{S}_2, \mathbf{S}_2) - 2\text{sim}(\mathbf{S}_1, \mathbf{S}_2) \\ &= 2d - 2\|\mathbf{S}_1^T \mathbf{S}_2\|_F^2 \\ &= \|\mathbf{S}_1 \mathbf{S}_1^T - \mathbf{S}_2 \mathbf{S}_2^T\|_F^2 \\ &= \|\Psi(\mathbf{S}_1) - \Psi(\mathbf{S}_2)\|_F^2, \end{aligned} \quad (2.10)$$

where  $\Psi : \text{span}(\mathbf{S}) \mapsto \mathbf{S} \mathbf{S}^T$  maps all subspaces from  $\mathbb{R}^D$  to a higher-dimensional feature space  $\mathbb{R}^{D \times D}$  [13]. For any two subspaces,  $\Psi$  satisfies  $\text{span}(\mathbf{S}_1) = \text{span}(\mathbf{S}_2) \iff \Psi(\mathbf{S}_1) = \Psi(\mathbf{S}_2)$  [49]. Subsequently,  $\text{trace}((\mathbf{S}_a \mathbf{S}_a^T)(\mathbf{S}_b \mathbf{S}_b^T))$ , as described in [59, Section 9.2], can be adopted as the inner product of  $\Psi(\mathbf{S}_1)$  and  $\Psi(\mathbf{S}_2)$ . Using the kernel trick





[142], the subspace-based kernel function can be obtained as

$$\begin{aligned} \text{kern}(\mathbf{S}_1, \mathbf{S}_2) &= \langle \Psi(\mathbf{S}_1), \Psi(\mathbf{S}_2) \rangle = \text{trace}((\mathbf{S}_1 \mathbf{S}_1^T)(\mathbf{S}_2 \mathbf{S}_2^T)) \\ &= \|\mathbf{S}_1^T \mathbf{S}_2\|_F^2. \end{aligned} \quad (2.11)$$

It is easy to verify that  $\text{kern}(\mathbf{S}_1, \mathbf{S}_2) = \text{kern}(\mathbf{S}_1 \mathbf{Q}_1, \mathbf{S}_2 \mathbf{Q}_2)$ , where  $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathcal{O}_d$ . To see the positive definiteness of the kernel, i.e., it can give rise to a positive Gram matrix, we have

$$\begin{aligned} \sum_{i,j} c_i c_j \|\mathbf{S}_i^T \mathbf{S}_j\|_F^2 &= \sum_{i,j} c_i c_j \text{trace}(\mathbf{S}_i \mathbf{S}_i^T \mathbf{S}_j \mathbf{S}_j^T) \\ &= \sum_{i,j} \text{trace}((c_i \mathbf{S}_i \mathbf{S}_i^T)(c_j \mathbf{S}_j \mathbf{S}_j^T)) \\ &= \text{trace}((\sum_i c_i \mathbf{S}_i \mathbf{S}_i^T)^2) = \|\sum_i c_i \mathbf{S}_i \mathbf{S}_i^T\|_F^2 \geq 0. \end{aligned} \quad (2.12)$$

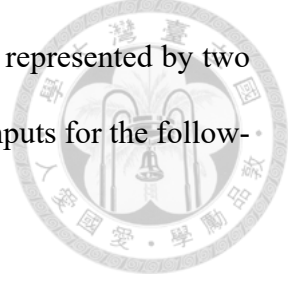
Therefore, (2.11), named *the Projection kernel* [49], enables us to build a precomputed kernel matrix for valid SVM learning for SLR. We refer readers to [143, Chapters 2 and 7.4] for details.

## 2.4.2 SNNs

The SNNs for the SLR, as shown in Fig. 2.3, transform the subspace inputs, contributed by a set of phone recognizers, into a target label for each utterance. There are two hidden layers. One processes subspace-based inputs to generate subspace-aware scores (cf. the kernel layer in Fig. 2.3), whereas the other involves multilayer perceptrons (MLPs) followed by an output layer with a softmax activation function to transform the subspace-aware scores into the decision scores for classification. The most significant differences between SNNs and regular DNNs are as follows:

**Inputs.** The input layer of SNNs simultaneously accepts multiple subspace inputs  $\mathbf{S}_1, \dots, \mathbf{S}_L$  of an utterance, each from one of the  $L$  parallel phone recognizers. The ranks

(widths) of the inputs can be varied. For example, if the utterance is represented by two subspace inputs,  $\mathbf{S}_1 \in \mathbb{R}^{D_1 \times d_1}$  and  $\mathbf{S}_2 \in \mathbb{R}^{D_2 \times d_2}$ , they are allowable inputs for the following weight computation even when  $D_1 \neq D_2$ .



**Orthonormal Weight Maps.** The second layer of SNNs is a kernel function  $f_{lm} : \mathbb{R}^{D_l \times d_l} \rightarrow \mathbb{R}$ , which, in matrix notation, can be expressed as

$$f_{lm}(\mathbf{S}_l) = k_{lm} = \|\mathbf{W}_{lm}^T \mathbf{S}_l\|_F^2 \quad (2.13a)$$

$$\text{subject to } \mathbf{W}_{lm}^T \mathbf{W}_{lm} = \mathbf{I}_{d'_l}, \quad (2.13b)$$

where  $\mathbf{W}_{lm} \in \mathbb{R}^{D_l \times d'_l}$  denotes the  $m$ -th orthonormal weight map or kernel subspace for the  $l$ -th subspace input with  $d'_l$  as its rank, and  $\|\cdot\|_F$  denotes the Frobenius norm used to measure the similarity between a *sample* subspace  $\mathbf{S}_{il} \in \mathcal{G}_{D_l, d_l}$  and a *reference* subspace  $\mathbf{W}_{lm} \in \mathcal{G}_{D_l, d'_l}$ . In fact, (2.9) is not confined to two subspaces lying on the same Grassmann manifold [43]. The similarity between two subspaces  $\mathbf{W} \in \mathcal{G}_{D, p}$  and  $\mathbf{S} \in \mathcal{G}_{D, q}$  can be measured based on their principal angles by  $\sum_{i=1}^q \cos^2 \theta_i = \|\mathbf{W}^T \mathbf{S}\|_F^2$  if  $p \geq q$ . Moreover,

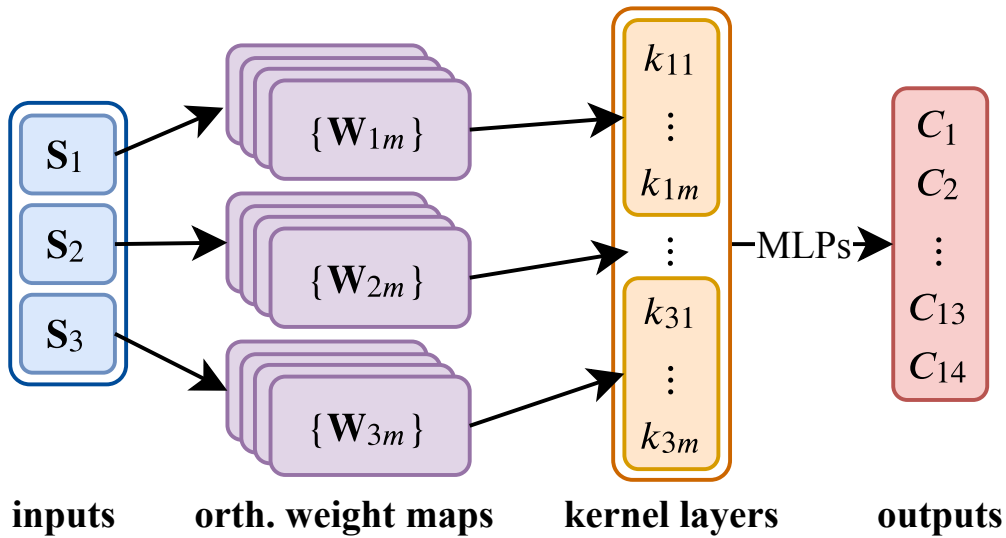
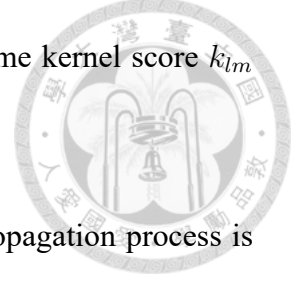


Figure 2.3: Subspace-based neural networks (SNNs), where 3 subspace inputs are associated with 3 BUT phone recognizers respectively, the output layer is associated with 14 target languages, and  $m$  orthonormal weight maps per input are for kernel computation.

it can be easily verified that two *sample* subspaces will have the same kernel score  $k_{lm}$  when they are equivalent in the Grassmannian sense.



**Optimization and Regularization.** To train SNNs, a back-propagation process is implemented from the output layer down through the whole SNNs to adjust all parameters. The gradient of each weight can be easily derived by partial differentiation on the cross-entropy cost function, so that the model can be iteratively updated by using an optimizer based on gradient descent. Equation (2.13a) is differentiable with respect to  $\mathbf{W}_{lm}$ , and the derivative is

$$\begin{aligned} \frac{\partial f_{lm}(\mathbf{S}_{il})}{\partial \mathbf{W}_{lm}} &= \frac{\partial \|\mathbf{W}_{lm}^T \mathbf{S}_{il}\|_F^2}{\partial \mathbf{W}_{lm}} \\ &= \frac{\partial \text{trace}(\mathbf{W}_{lm}^T \mathbf{S}_{il} \mathbf{S}_{il}^T \mathbf{W}_{lm})}{\partial \mathbf{W}_{lm}} = 2\mathbf{S}_{il} \mathbf{S}_{il}^T \mathbf{W}_{lm}. \end{aligned} \quad (2.14)$$

However, the constraint term in (2.13b) renders the parameter estimation intractable. To alleviate this problem, an alternative method is to replace (2.13b) with the following regularizer that allows us to apply penalties on  $\mathbf{W}_{lm}$  as follows:

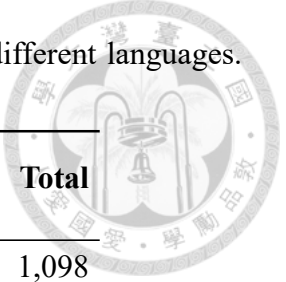
$$\lambda \|\mathbf{W}_{lm}^T \mathbf{W}_{lm} - \mathbf{I}_{d'_l}\|_F^2, \quad (2.15)$$

where the tunable parameter  $\lambda$  controls the degree of orthonormality of  $\mathbf{W}_{lm}$ . These penalties are further incorporated in the final categorical cross-entropy loss (i.e., negative log-likelihood) function with a derivative with respect to  $\mathbf{W}_{lm}$ ,

$$2\lambda \mathbf{W}_{lm} (\mathbf{W}_{lm}^T \mathbf{W}_{lm} - \mathbf{I}_{d'_l}), \quad (2.16)$$

while performing optimization using back propagation.

Table 2.2: Numbers of utterances in different tasks with respect to different languages. TR and TEST denote the training and test data, respectively.

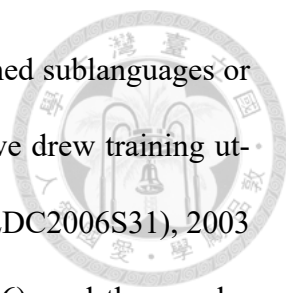


Task Language	LRE-03		LRE-07		Total
	TR	TEST (30 s)	TR	TEST (30 s)	
Arabic	309	80	629	80	1,098
Bengali			240	80	320
Chinese	625	80	5,549	398	6,652
English	1,276	240	6,684	240	8,440
Farsi	318	80	1,253	80	1,731
French	314	80	-	80	474
German	318	80	646	80	1,124
Hindustani	309	80	2,956	240	3,585
Indonesian			-	80	80
Italian			-	80	80
Japanese	315	160	1,702	80	2,257
Korean	312	80	2,014	80	2,486
Punjabi			-	32	32
Russian	-	80	1,323	160	1,563
Spanish	611	80	2,763	240	3,694
Tagalog			-	80	80
Tamil	297	80	823	160	1,360
Thai			515	80	595
Vietnamese	309	80	1,069	160	1,618
<b>Total</b>	5,313	1,280	28,166	2,510	37,269
<b>Positives</b>		1,200		2,158	
<b>Trials</b>		15,360		35,140	

## 2.5 Experiment Results and Discussion

Our proposed methods were evaluated on two tasks: NIST LRE 2007 (LRE-07) [110] for spoken language verification (SLV) and NIST LRE 2009 (LRE-09) [112] for dialect/accident identification.

The first task pertains to the *General LR* test of LRE-07 conducted on 30-s 8k-sampled telephony utterances. Nineteen languages were tested, including 14 target languages and 5 out-of-set languages (French, Indonesian, Italian, Punjabi, and Tagalog) that were not the target languages of each trial and lacked corresponding training data. Four



target languages (Chinese, English, Hindustani, and Spanish) contained sublanguages or dialects. To learn the phonotactic models and backend classifiers, we drew training utterances that were longer than 10 s from NIST LRE 1996 (LRE-96, LDC2006S31), 2003 (LRE-03, LDC2006S31), 2005 (LDC2008S05), 2009 (LDC2014S06), and the supplemental set provided by LRE-07 (LDC2009S05). We used LRE-03 as the validation set to determine the optimal representation setting and model hyperparameters. According to the LRE-03 evaluation plan [109], the test set contains 12 target languages and one out-of-set language (Russian); the training data were only from LRE-96. The number of utterances and the statistics regarding trials in the LRE-07 (for evaluation) and LRE-03 (for validation) tasks are listed in Table 2.2.

The second task involves 8 dialect/accent pairs, as shown in Table 2.3, for evaluating how a system can distinguish cognate or mutually intelligible languages. Each dialect/accent pair, where only 10-s and 30-s recorded telephony utterances were selected from LRE-09, was independently evaluated by binary closed-set classification using stratified five-fold cross-validation.

### 2.5.1 Phone Recognition

We used the well-known BUT phone recognizers<sup>1</sup> developed by the Brno University of Technology for Czech (CZ), Hungarian (HU) and Russian (RU) [144]. The numbers of sound units in the BUT decoders for CZ, HU, and RU were 46, 62, and 53, respectively. Each set includes three nonphonetic units, i.e., /int/ (intermittent noise), /pau/ (short pause), and /spk/ (nonspeech speaker noise). These phone decoders use a three-state model per unit, which means that three posterior probabilities per unit are provided

---

<sup>1</sup><https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context>

Table 2.3: Numbers of utterances used in accent/dialect identification with respect to 8 pairs of languages.

<b>Dialect or Accent Pair</b>	<b>abbr.</b>	<b># Utt.</b>	<b>Total</b>
Mandarin / Cantonese	Man / Can	2,006 / 730	2,736
Portuguese / Spanish	Por / Spa	794 / 770	1,564
Haitian Creole / French	Cre / Fre	646 / 790	1,436
Russian / Ukrainian	Rus / Ukr	1,003 / 776	1,779
Hindi / Urdu	Hin / Urd	1,285 / 756	2,041
Dari / Farsi	Dar / Far	776 / 775	1,551
Bosnian / Croatian	Bos / Cro	710 / 752	1,462
American Eng. / Indian Eng.	Ame / Ind	1,759 / 1,113	2,872

at each frame. Therefore, to form the phonetic vectors for our proposed subspace construction, as shown in Fig. 2.2, a single posterior probability was computed for each unit by adding the posterior probabilities of all the states in the corresponding model.

## 2.5.2 Experiment Flow and Control

As depicted in Fig. 2.4, parallel phone recognition language modeling (PPR-LM) [80], parallel phone recognition vector space modeling (PPR-VSM) [80], and parallel phone recognition subspace multinomial modeling (PPR-IVEC) were implemented as the baselines, where each utterance was represented by multiple multinomial distributions, high-dimensional vectors, and lower-dimensional vectors, respectively. In addition to the sequence-based PPR-LM, we implemented lattice-based PPR-LM [38], where phone  $n$ -gram statistics were derived from the top-five decoded phone sequences, and the average numbers of nodes and edges of a phone lattice were 3,700 and 15,000, respectively. For PPR-VSM, three phone-recognizer-dependent 350-dimensional vectors, each of which was derived through phone recognition,  $n$ -gram counts, TF-IDF weights, and LSA, were evaluated by phone-recognizer-dependent target-dependent one-vs-rest SVMs with a linear kernel [79]. Instead of using NN-based backends such as MLPs, we used SVMs in

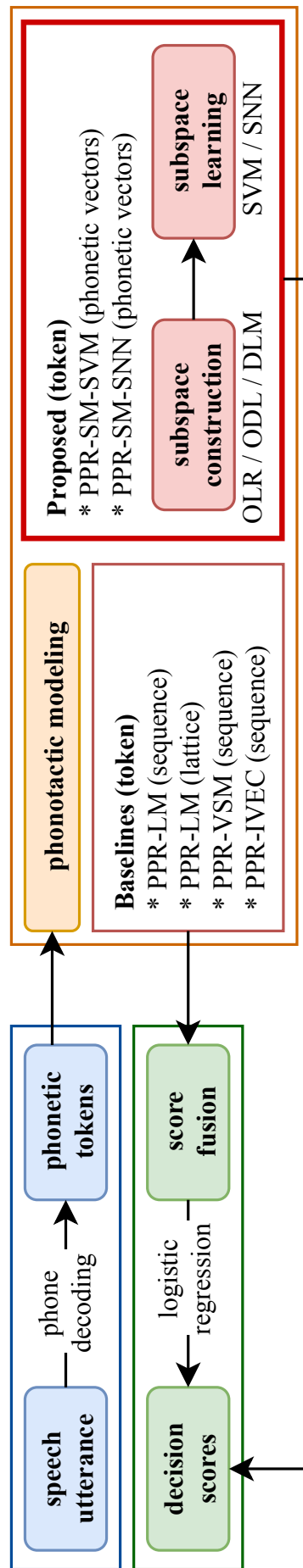
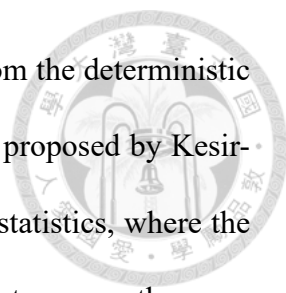


Figure 2.4: Experiment flowchart, where the orange box with a dashed border shows the procedure of phonotactic modeling, inside which the red rectangle expresses two key steps of our proposed framework based on subspace representation and learning.





PPR-VSM because we did not think that MLPs will benefit much from the deterministic vectorial representation. For PPR-IVEC, we adopted the algorithm<sup>2</sup> proposed by Kesiraju *et al.* to represent each utterance as an i-vector using its  $n$ -gram statistics, where the Laplace prior enforced sparsity in the basis matrix [68]. Most parameters were the same as those in [68], except for the dimension of the i-vector of an utterance, which was 900.

Our proposed method, named parallel phone recognition subspace modeling (PPR-SM), uses two classifiers for subspace learning, i.e., the SVMs and SNNs. The corresponding systems are denoted as PPR-SM-SVM and PPR-SM-SNN, respectively. The PPR-SM-SNN, as shown in Fig. 2.3 and (2.13), is mainly composed of orthonormal weight maps and MLPs. Here, the first set of parameters was initially drawn from the  $O(D_l)$  Haar distribution to randomly generate a set of orthonormal matrices [98]<sup>3</sup>, and the parameters of the MLPs were normally initialized by the Glorot process [42]. The adaptive gradient algorithm for updating the PPR-SM-SNN parameters was Adam [69], where default parameters were used, except that the learning rate was initially fixed at  $10^{-3}$ . Moreover, a drop-based learning rate schedule was practiced by reducing the learning rate by half every 10 epochs. The mini-batch size and the maximum number of training epochs were 24 and 200, respectively.

Similar to PPR-VSM, PPR-SM-SVM not only specifies the Projection kernel in (2.11) on SVMs, but also extends our previous versions in [75, 147], where only one phone recognizer developed for a universal phone set is used. For PPR-LM, PPR-VSM, and PPR-SM-SVM, because three BUT phone recognizers and 14 target languages were used in the SLV task,  $3 \times 14 = 52$  decision scores contributed by target-dependent LMs and support

---

<sup>2</sup><https://github.com/skesiraju/smm>

<sup>3</sup>For the sake of performance, we did not use Saxe's method [141], which was recently proposed to fill the weights with a (semi) orthogonal matrix and implemented using existing NN-based toolkits, such as Keras and PyTorch.



vectors, respectively, were derived for each test utterance. To combine the 52 scores for one target language, a systematic method based on multiclass logistic regression was implemented [80]. For a fair comparison, the MLPs in the structure of the PPR-SM-SNN (cf. Fig. 2.3) were implemented with no hidden layers, which can be reegarded as a simplified version of logistic regression.

As for subspace construction, three implementations were performed, i.e., OLR, ODL, and DLM, according to (2.3), (2.4), and (2.5). Their goals are summarized in Table 2.1. For ODL, the regularization parameter  $\lambda_{odl}$  and the number of iterations  $J$  in Algorithm 3 were empirically set to  $10^{-4}$  and 50, respectively.

### 2.5.3 Oracle Experiment on LRE-03

We conducted oracle experiments on LRE-03 to determine the optimal parameters of the phonotactic models, including the penalty factor in SVMs, the regularization strength in logistic regression, and the subspace dimensionality  $d$  in (2.3), (2.4), (2.5), and (2.13). The rank of the sample subspace  $d_{sl}$  was controlled by the sample subspace ratio  $\alpha \in (0, 1)$  as  $d_{sl} = \max(\lfloor \alpha M_l \rfloor, 2)$ , where  $M_l$  denotes the number of phonemes in the  $l$ -th phone recognizer. The rank of the kernel (reference) subspace  $d_{kl}$  was controlled by the reference subspace ratio  $\beta \in [0.5, 1.5]$  as  $d_{kl} = \max(\lfloor \beta d_{sl} \rfloor, 2)$ . Furthermore, for the PPR-SM-SNN, other parameters, such as the orthogonality factor  $\lambda$  in (2.15) and the number of orthonormal weight maps  $m$ , were tuned within a heuristic range through an exhaustive grid search, without learning rate scheduling. It is noteworthy that  $\beta$  was always set to 1 to satisfy (2.12) for PPR-SM-SVM.

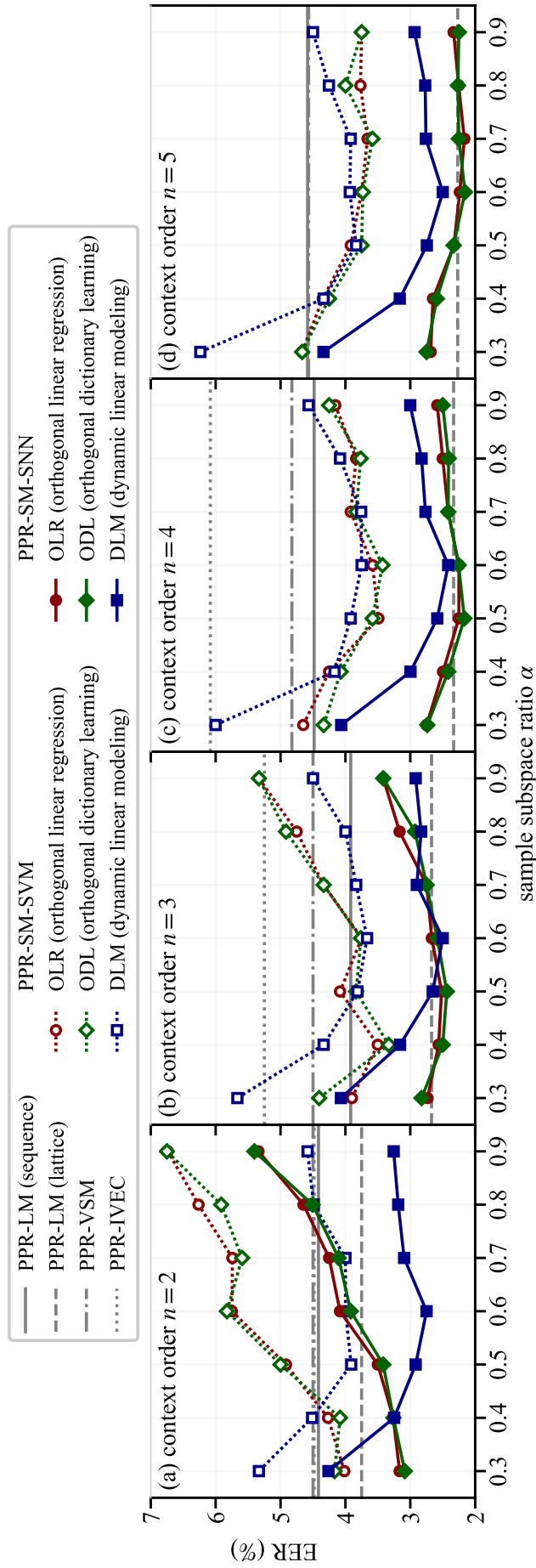


Figure 2.5: EERs on LRE-03 for baselines and two kinds of PPR-SM-based backends with respect to various context orders, sample subspace ratios, and three methods for subspace construction: OLR (2.3), ODL (2.4), and DLM (2.5).





### 2.5.3.1 Effect of sample subspace ratio

Fig. 2.5 shows the equal error rates (EERs) with respect to  $\alpha$  and the context order  $n$ . Each point (for PPR-SM-SVM and PPR-SM-SNN) and each line (for the baseline methods) represent the best result from experiments with some possible parameter combinations. The parameter  $\alpha$  represents the degree that a subspace can describe a set of phonotactic observations. Although a larger  $\alpha$  enables a subspace to retain more phonotactic information, it might yield subtle effects caused by unwanted structural noises owing to the deficiency of phone recognizers. The context order  $n$  represents the length of the phonotactic constraint used for phonotactic modeling. Although  $n = 3$  (trigram) has been shown an appropriate length in some phonotactic learning simulation systems [54], we changed  $n$  to investigate how these systems achieved their best performances with more or less phonotactic information. The four subplots in Fig. 2.5 show the manner in which the context order  $n$  affects the performance. Unlike PPR-VSM, sequence-based PPR-LM, and PPR-IVEC, where a higher context order may cause overfitting thereby increasing the

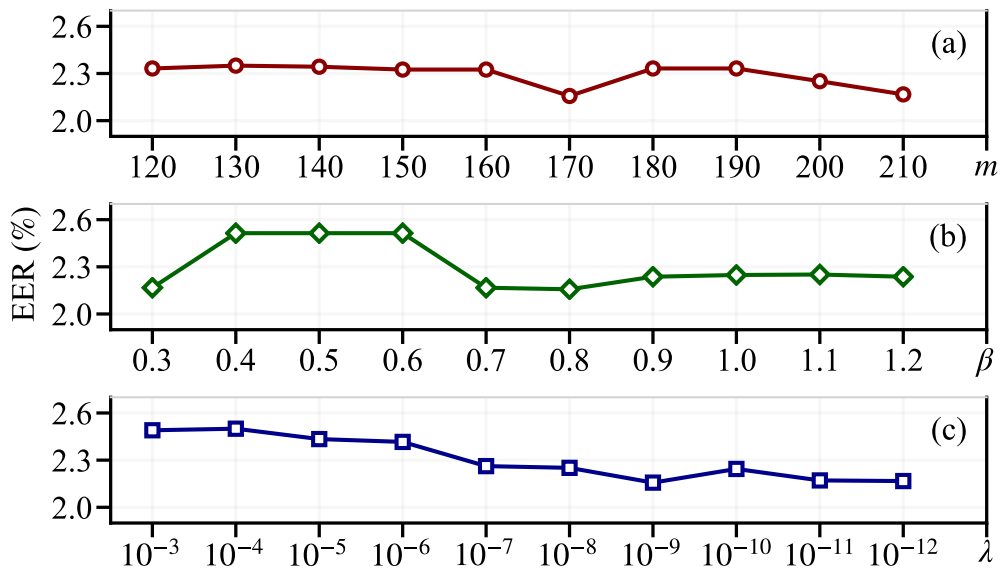
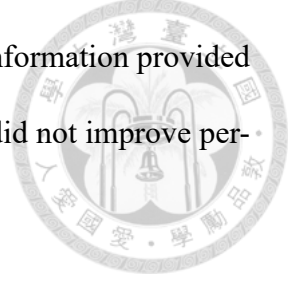


Figure 2.6: EERs on LRE-03 for PPR-SM-SNN in the case of ODL/5/0.6 with respect to various values of  $m$ ,  $\beta$ , and  $\lambda$ .

EERs, the PPR-SM-SNN and lattice-based PPR-LM can utilize the information provided by longer phonotactic constraints. For PPR-SM-SVM, increasing  $n$  did not improve performance (the only exception is increasing  $n$  from 2 to 3).



When focusing on subspace construction, we observed from Fig. 2.5 that the results of OLR and ODL were similar and better than those of DLM when  $n \geq 4$ . For DLM, when  $\alpha \leq 0.6$  (implying a smaller  $d$  in Algorithm 1), significant amounts of irrelevant noise and phonetic information might be lost when estimating  $\mathbf{C}$  and  $\mathbf{X}_{1:K}$  in Algorithm 1

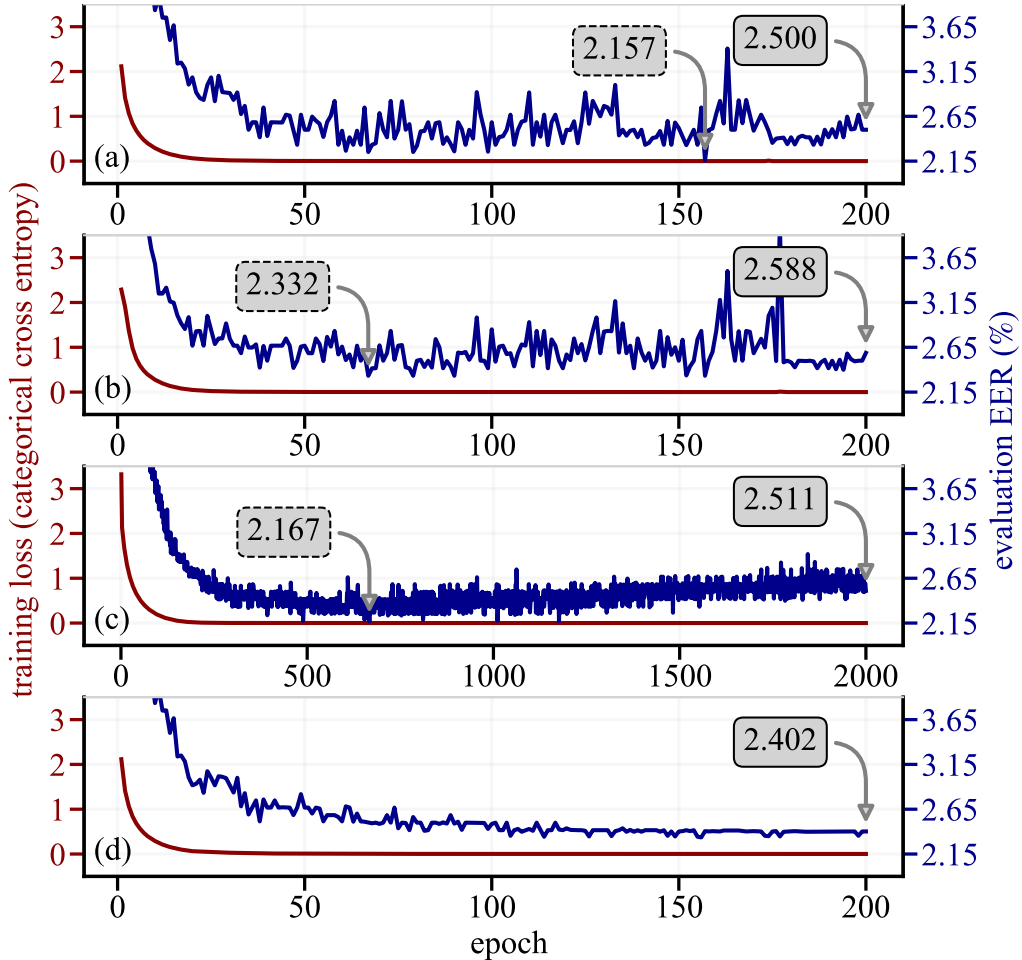


Figure 2.7: Variation of training loss and evaluation EER with respect to the number of training epochs on LRE-03 for PPR-SM-SNN in the case of ODL/5/0.6. The weight map initializer in (a) is orthogonal, while in (b) the Glorot normal initializer is used. Inherited from (a), the learning rate in (c) is reduced from  $10^{-4}$  to  $10^{-5}$ , while in (d), the drop-based learning rate schedule is adopted. The boxes with dashed edges denote the best evaluation EER, while those with solid edges denote the EER after the last epoch.

and  $\mathbf{A}$  in Algorithm 2. Moreover, the results show that an increase in context order is not necessarily good for DLM. One possible reason is that, as a type of first-order Markovian dynamics shown in (2.1), DLM has a limited ability to explicitly *simulate* phonotactic constraints that are longer than 3 in length (i.e.,  $n \geq 3$ ).

Regarding subspace learning, we discovered that regardless of the subspace construction method used, the PPR-SM-SNN is almost always better than PPR-SM-SVM and is slightly better than lattice-based PPR-LM at certain  $\alpha$  values. When the amount of training utterances is relatively large, e.g., exceeding 5,000, as shown in Table 2.2, the PPR-SM-SNN is superior to PPR-SM-SVM. Another limitation of PPR-SM-SVM is the kernel selection [9]. Only two valid metrics exist on the Grassmann manifold, i.e., the Projection kernel and the Binet–Cauchy kernel, which were studied in [49]. The former is more useful for our preliminary experiments. Better kernels for the SLR task will be investigated in future studies.

### 2.5.3.2 Effect of other parameters in PPR-SM-SNN

Next, we discuss the effects of three key parameters, i.e.,  $m$  in (2.13),  $\beta$ , and  $\lambda$  in (2.15), on the performance of the PPR-SM-SNN. We investigated the best case shown in Fig. 2.5, where the subspace construction was based on ODL with  $n = 5$  and  $\alpha = 0.6$  (ODL/5/0.6 for short), and the parameters  $m$ ,  $\beta$ , and  $\lambda$  were 170, 0.8, and  $10^{-9}$ , respectively. In the experiments, each parameter was evaluated separately with the remaining parameters fixed at the best settings described earlier. Based on Fig. 2.6, the following observations were obtained.

1. In Fig. 2.6(a), the best number of weight maps  $m$  was 170, which makes SNNs a

wide neural network.



2. In Fig. 2.6(b), the best  $\beta$  was 0.8 when  $\alpha = 0.6$ , and the corresponding input widths  $\{d_{sl}\}_{l=1}^3$  and weight map sizes  $\{d_{kl}\}_{l=1}^3$  were  $[27, 37, 31]^4$  and  $[21, 29, 24]^5$ , respectively, whereas the numbers of phonemes  $\{M_l\}_{l=1}^3$  for the BUT phone recognizers were  $[46, 62, 53]$ . The result shows that for PPR-SM-SNN,  $d_{sl}$  does not have to be equal to  $d_{kl}$ . However, when  $\beta = 1$ , under the same sample subspace specification, the EER of PPR-SM-SNN was 2.248%, which was still much lower than that of PPR-SM-SVM (3.732%).
  
3. Although the lowest EER occurred when  $\lambda$  was fairly small ( $10^{-9}$ ), and an almost negative correlation between the orthogonality factor  $\lambda$  and EER existed, as shown in Fig. 2.6(c), this does not mean that the orthogonality of the weight maps is less important than expected. We can also observe that when  $\lambda$  becomes smaller ( $\leq 10^{-10}$ ), the EER will increase. This result demonstrates that even if the appropriate value of  $\lambda$  is small, it still has its role. In general, the negative gradients of commonly used regularizers, such as the  $l^2$ -norm, always *equi-dimensionally* point toward  $\mathbf{0}$ . They may not compromise the classification performance but will avoid overfitting the data. Furthermore, this does not mean that the orthonormal initialization of weight maps described in Section 2.4.2 is unnecessary. This issue will be addressed next.

---

<sup>4</sup> $d_{sl} = \alpha M_l$ , so  $[27, 37, 31] \approx 0.6 \times [46, 62, 53]$ .

<sup>5</sup> $d_{kl} = \beta d_{sl}$ , so  $[21, 29, 24] \approx 0.8 \times [27, 37, 31]$ .

Table 2.4: EERs and  $C_{avg}$ 's on LRE-07 for different methods with our own implementations. The lowest values in each column are in bold face. The column "rel. %" shows the relative reduction achieved by PPR-SM-SNN (ODL) over the compared method computed based on their respective best rates among all context orders.

Metric	Context Order $n$ ( $n$ -gram)	Equal Error Rate (EER %)					Average Cost ( $C_{avg}$ )						
		2	3	4	5	rel. %	p-value	2	3	4	5	rel. %	p-value
PPR-LM (seq.) [179]		10.844	10.751	9.880	10.009	52.15	< 0.001	0.152	0.151	0.132	0.127	42.52	< 0.001
PPR-LM (lat.) [38]		8.753	7.265	6.812	6.534	27.64	0.016	0.120	0.107	0.101	0.097	24.74	0.007
PPR-VSM [79]		8.886	8.990	9.368	10.056	46.80	< 0.001	0.117	0.120	0.124	0.133	37.61	< 0.001
PPR-IVEC [68, 155]		10.978	12.326	12.604	12.462	56.93	< 0.001	0.152	0.165	0.168	0.163	51.97	< 0.001
PPR-SM-SVM (OLR) [147]		7.222	6.670	6.759	6.627	28.66	0.019	0.099	0.094	0.094	0.095	22.34	0.029
PPR-SM-SVM (ODL)		7.222	6.534	6.534	6.673	27.64	0.020	0.099	0.091	0.093	0.094	19.78	0.034
PPR-SM-SVM (DLM) [75]		7.001	7.403	7.611	7.786	32.47	0.005	0.098	0.102	0.105	0.106	25.51	0.008
PPR-SM-SNN (OLR)		<b>5.974</b>	<b>4.912</b>	4.904	4.819	1.89	0.864	<b>0.087</b>	<b>0.077</b>	0.075	0.075	2.67	0.928
PPR-SM-SNN (ODL)		6.349	5.007	<b>4.858</b>	<b>4.728</b>	-	-	0.092	<b>0.077</b>	<b>0.074</b>	<b>0.073</b>	-	-
PPR-SM-SNN (DLM)		6.252	5.746	5.940	6.164	17.72	0.123	0.089	0.083	0.086	0.087	12.05	0.200



### 2.5.3.3 Learning strategy of PPR-SM-SNN

We first discuss the need to consider orthogonality when initializing the weight maps of the PPR-SM-SNN. As shown in Figs. 2.7(a) and (b), when the initial random weight maps were set with the Glorot normal initializer without considering orthogonality [42], the *best* evaluation EER increased by approximately 8% (from 2.157% to 2.332%), compared with the case involving the orthogonal initializer. Moreover, its *final* (epoch 200) EER converged to 2.588%, which is worse than 2.5%, as shown in Fig. 2.7(a).

Next, we discuss the drop-based learning rate schedule. As shown in Fig. 2.7(a), without using a learning rate schedule, the curve for evaluating the EER fluctuated significantly; hence, we could not determine the stopping epoch using an additional development set or through cross-validation. Even when the learning rate was reduced from  $10^{-4}$  to  $10^{-5}$ , as shown in Fig. 2.7(c), the EER measure still appeared unreliable. Furthermore, we observed overfitting in Fig. 2.7(c). However, with the drop-based learning rate schedule, when the number of epochs increases to a sufficiently large number, the learning rate will decay and barely affect the training loss and model. Hence, the EER evaluated after the last training epoch might not be the lowest, but should still be acceptable. As an example, after the 200th epoch shown in Fig. 2.7(d), where the learning rate finally decays to  $2 \times 10^{-6}$ , the evaluation ERR (2.402%) is only 11.4% higher than the best EER (2.157%) shown in Fig. 2.7(a).

### 2.5.4 Language Recognition on LRE-07

In the LRE-07 task, the model configuration, including subspace specifications and hyperparameters, such as the regularization penalty and learning rate, was predetermined



by the LRE-03 task. Therefore, we used the settings for the best results in Fig. 2.5. As shown in Table 2.4<sup>6</sup> and Fig. 2.8, lattice-based PPR-LM is superior to the other three baseline methods and comparable to PPR-SM-SVM in terms of both the EER and average cost ( $C_{avg}$ ). Both the lattice-based PPR-LM and PPR-SM-SNN can use richer phone-contextual information to distinguish languages than the other methods. However, the performance of the PPR-SM-SNN was better than that of lattice-based PPR-LM. Regardless of the value of  $n$ , the EERs and  $C_{avg}$  values of the PPR-SM-SNN were significantly lower than those of lattice-based PPR-LM. For example, when  $n = 5$ , PPR-SM-SNN (ODL) achieved a 27.64% reduction in the EER (from 6.534% to 4.728%) and a 24.74% reduction in  $C_{avg}$  (from 0.097 to 0.073), compared with lattice-based PPR-LM. According to Welch's t-test, the PPR-SM-SNN significantly outperformed all the other methods (with p-values  $< 0.05$ ), as shown in Table 2.4. For PPR-SM-SVM, as performed in the LRE-03

<sup>6</sup>The baseline results may be worse than those evaluated on the same dataset reported in the corresponding papers (e.g., [1]) because we have less training data available for phonotactic modeling. For example, as described in [80], many systems used additional CallHome and CallFriend corpora for phonotactic modeling, but these corpora were not used in this study.

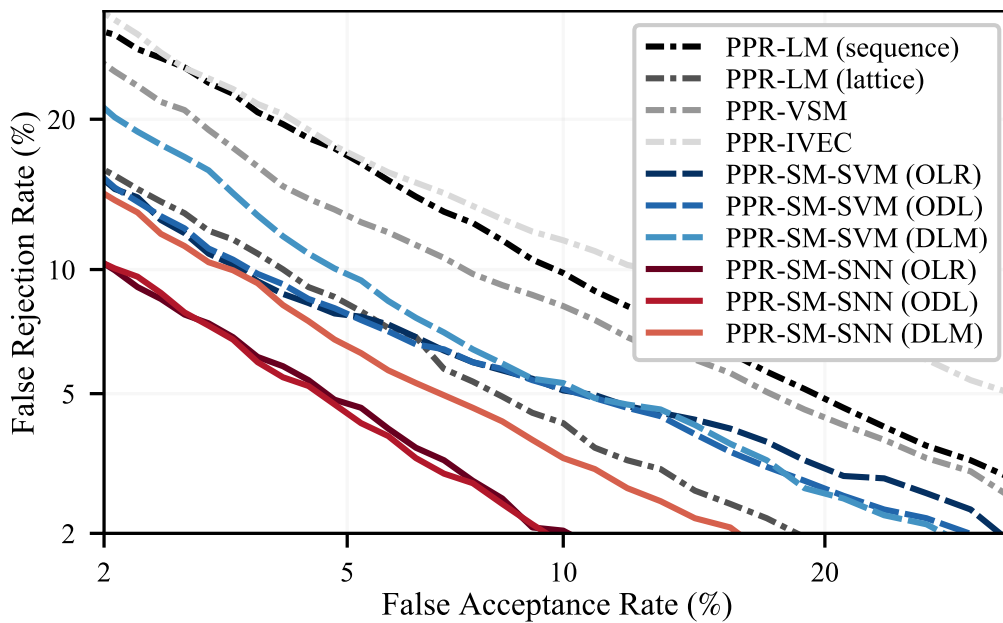


Figure 2.8: DET curves for different methods, where each method adopts the settings achieving the best performance in Table 2.4.

task, increasing  $n$  did not improve the performance. The only exception was an increase in  $n$  from 2 to 3. The results demonstrate that PPR-SM-SVM did not perform well on larger and more challenging datasets. It can be concluded that SNNs are more suitable for subspace learning than SVMs, because the *reference* subspaces used in the weight maps of SNNs are not as restricted as the *sample* subspaces used for kernel computation in SVMs.

Although LRE-07 and LRE-03 are two different tasks, we discovered that the performance trends of all the compared methods on the two tasks were generally consistent. Two new observations obtained in this study are as follows:

1. When  $n = 2$ , DLM generally outperformed OLR and ODL in the LRE-03 task, as shown in Fig. 2.5(a). However, in the LRE-07 task, DLM did not show an

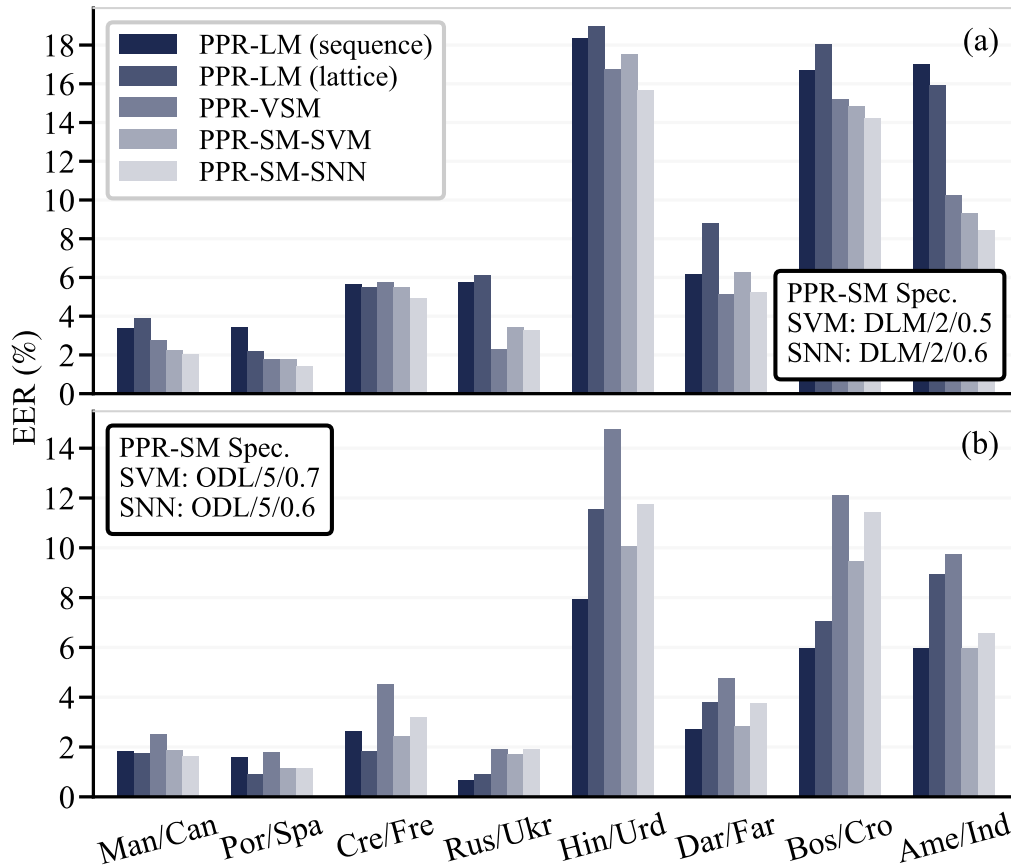


Figure 2.9: Average EERs of stratified 5-fold CV for different methods, (a) the context order  $n = 2$  and (b) the context order  $n = 5$ .

advantage over OLR and ODL.

2. Although ODL and OLR seemed to be neck and neck in most cases in Fig. 2.5 and Table 2.4, PPR-SM-SNN (ODL) slightly outperformed PPR-SM-SNN (OLR) in Fig. 2.8.

### 2.5.5 Dialect/Accent Identification on LRE-09

In the LRE-09 task, similar to the LRE-07 task, the model configuration inherits the LRE-03 settings from the best results shown in Fig. 2.5. The average EERs of the stratified five-fold cross-validation are shown in Fig. 2.9. PPR-IVEC was not implemented for the task owing to its poor performance in the first two tasks. The followings were observed from Fig. 2.9.

1. The last five spoken language pairs are generally considered to be mutually understandable [112], i.e., two persons who speak different languages can understand each other and communicate easily. However, the result shows that, compared with Hin/Urd, Bos/Cro, and Ame/Ind, the subtle differences in the phonotactics of Rus/Ukr and Dar/Far are easier to detect.
2. Surprisingly, in most cases, sequence-based PPR-LM outperformed lattice-based PPR-LM, especially in the last five pairs, when longer phonotactic constraints were used (i.e.,  $n = 5$ ). When  $n = 2$ , PPR-VSM was superior to sequence-based PPR-LM and lattice-based PPR-LM, but when  $n = 5$ , PPR-VSM was inferior. Overall, PPR-SM-SVM and PPR-SM-SNN outperformed the three baseline methods in most cases, especially when  $n = 2$ .
3. Contrary to the previous results, PPR-SM-SVM was better than PPR-SM-SNN

Table 2.5: EERs (%) for LRE'03. #PPRs denotes the number of parallel phone recognizers and the training data sets.

Methods	EER	#PPRs	Training Sets
PPR-SM-SNN	2.157	3 (BUT)	LRE'96 (10 s, 30 s)
PPR-LM (lattice) impl. by us	2.269	3 (BUT)	LRE'96 (10 s, 30 s)
PPR-LM (lattice) impl. by Gauvain [38]	4.000	3 (Switchboard, CallHome)	LRE'96 (3 s, 10 s, 30 s)

when  $n = 5$ , except for the first two pairs. One possible reason is that when the training sample set is not large enough (e.g., less than 3,000 in this task), more complex learning machines with higher capacity (such as NN-based models) will be prone to overfitting. At the representation level, more informative phonotactic tokens, such as soft-count statistics and our proposed subspaces, may not be suitable for small data volume situations.

4. When  $n = 2$ , PPR-VSM achieved the best results for the pairs Rus/Ukr and Dar/Far. Moreover, when  $n = 5$ , PPR-SM-SVM outperformed PPR-SM-SNN except for the first two pairs. The results may indicate that PPR-VSM and PPR-SM-SVM, which both use SVMs as the backend classifier, have their advantages in binary classification problems. It is also obvious that the subspace-based representation is more useful than the VSM-based representation, especially when the length of the phonotactic constraint is longer.
5. When  $n = 2$ , regardless of the difficulty of these pairs, PPR-SM-SNN showed its superiority and stability in distinguishing dialects and accents.

## 2.5.6 Comparison with Results in Literature



Table 2.5 shows the results evaluated on LRE'03. We compared the results of our proposed PPR-SM-SNN and the PPR-LM (lattice) implemented by us with the result of PPR-LM (lattice) from Gauvain's paper [38]. The main difference between our PPR-LM (lattice) and Gauvain's PPR-LM (lattice) is the use of different parallel phone recognizers. As shown in the table, our implementation of PPR-LM (lattice) is reliable; under similar conditions, its EER is lower than that of Gauvain's PPR-LM (lattice). In addition, the proposed PPR-SM-SNN method is superior to the two PPR-LM (lattice) methods. We did not find any references with reliable results of PPR-LM (lattice) performed on LRE'07 under similar training conditions.

For PPR-VSM, we referred to Tong's work [161] and Ambikairajah's work [1]. The results evaluated on LRE'07 are shown in Table 2.6. In [161], Tong constructed seven phone recognizers (PRs), conducted PPR-VSM based on each PR, and hence obtained EERs ranging from 8.39% to 13.96%. The EER of our PPR-VSM method was 8.886%. We believe that the performance is reasonable. It is noteworthy that only one (Russian) among three languages of our PPRs (BUT) appeared in the 14 target languages in LRE'07; however, all seven languages in Tong's PRs overlapped with the 14 target languages in LRE'07. In addition, as shown in Table 2.6, our PPR-VSM is worse than Ambikairajah's PPR-VSM. The reasons are twofold. First, compared with our PPR-VSM, Ambikairajah's PPR-VSM used more matched languages in PPRs. Second, the CallFriend corpus, which is a much larger training set than LRE'09 (selected), was used in Ambikairajah's work. In summary, the performance of our PPR-VSM is reasonable. It is clear that the proposed PPR-SM-SNN method is superior to our PPR-VSM baseline.

Table 2.6: EERs (%) for LRE'07. LRE'96-05 means LRE'96, 03, and 05.

Methods	EER	#PPRs	Training Sets
PPR-SM-SNN	4.728	3 (BUT)	LRE'96-05, LRE'07 (dev), LRE'09 (selected)
PPR-VSM impl. by us	8.886	3 (BUT)	LRE'96-05, LRE'07 (dev), LRE'09 (selected)
PPR-VSM impl. by Tong [161]	8.390- 13.960	7 (IIR-LID, OGI-MLTS etc.)	LRE'96-05, LRE'07 (dev), CallFriend
PPR-VSM impl. by Ambikairajah [1]	4.380	7 (IIR-LID, OGI-MLTS etc.)	LRE'07 (dev), CallFriend

In this study, we focused on phonotactic SLR and investigated how PPRs can help distinguish languages. Therefore, we only compared the proposed methods with state-of-the-art phonotactic methods. Comparing the proposed methods with nonphonotactic systems may not be suitable for the purpose of this study. Even though the phonotactic approaches cannot compete with state-of-the-art methods based on i-vectors [96] and x-vectors [150], they are still worthy of investigation. However, in-depth investigation and comparison of these fundamentally different methods should be performed. Moreover, because the proposed phonotactic methods and i-/x-vectors are derived based on different concepts and may contain complementary information, we will investigate unified frameworks that combine these two classes of methods in future studies.

## 2.6 Conclusions



Herein, we proposed a new phonotactic representation of an utterance based on the concept of linear subspace, which is neither equivalent nor reducible to a distributional or vectorial representation. It increases the capacity of the representation for containing more phonotactic information. In particular, it can utilize uncertain information produced by unreliable phone recognizers by forming phonetic vectors. In addition to the previously proposed orthogonal linear regression and dynamic linear modeling, we proposed a dictionary learning-based method for subspace construction. The resulting subspace formed under Laplace's prior condition can restore the phonotactic structure underlying an utterance; hence, it is more capable of capturing the most salient features.

In addition to SVMs, the proposed SNNs were investigated as a solution for subspace learning and experimentally proven to outperform four phonotactic baselines in most experiments, i.e., sequence-based PPR-LM, lattice-based PPR-LM, PPR-VSM, and PPR-IVEC. The input layer of the SNNs accepted subspace-shaped samples, and the intermediate weight maps were specifically designed with a specific and differentiable subspace similarity based on the principal angles. Using SNNs, the PPR-SM-SNN achieved a relative reduction of 27% in EER on the LRE-07 language detection task, compared with the SVM-based subspace learning counterpart (PPR-SM-SVM).

Although we did not quantitatively analyze the time complexity, with the assistance of GPUs, the training process of SNNs was much faster than those of other lattice- or SVM-based methods compared in this study. This was particularly true when the context order exceeded 3. However, the development of SNNs can be improved further. We believe that when training SNNs, certain GPU-based realizations for setting constraints

on the weight maps can stabilize the movement of the training loss and the performance of the development data. For example, the recently proposed parameterization based on the Lie group theory via an exponential map might be worth investigating [78].

According to the parlance of phonology, a learning approach for SLR is regarded as *phonotactic* in that the smallest speech event to be addressed is a phoneme rather than an acoustic frame. Typically, the former is longer than the latter. In future studies, we will include other *frame-based* statistics or information in the formation of *phoneme-based* phonetic vectors, such as the PLLR [26, 28] and frame-based features [50]. The integration of the proposed subspace approaches with the PLLR features will be an interesting future direction.





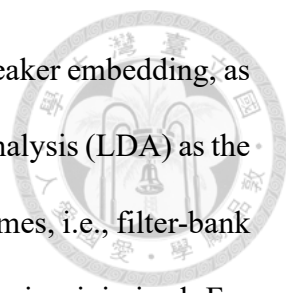
## Chapter 3

# Discriminative Autoencoders for Speaker Verification

Traditionally speaking, the task of text-independent speaker recognition is a classification problem, which can be defined as an utterance level “sequence-to-one” learning issue. In this task, we have to identify which speaker spoke the test utterance (speaker identification) or to verify whether two utterances spoken by the same speaker or different speakers (speaker verification).

In this chapter, we introduce a novel neural structure, namely discriminative autoencoders (DcAE), to the task of speaker verification. Different from most traditional speaker models that only focus on discriminating speakers, the DcAE generates utterances while discriminating them in the training stage, so that the information used for discriminating speakers is purer. Moreover, in the evaluation (inference) stage, the time complexity of the DcAE is the same as the baseline model; that is, the decoder part of the DcAE is not required for inference.

In the remainder of the chapter, we will explore two types of DcAE used for speaker



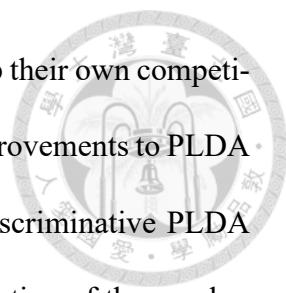
verification. The first one takes i-vectors, an kind of well-prepared speaker embedding, as the input/output and uses the objective similar to linear discriminant analysis (LDA) as the loss function. The other replaces the i-vector with acoustic feature frames, i.e., filter-bank features, and the cross-entropy along with a softmax activation function is minimized. For the second type of DcAE, some issues, such as the pooling mechanism and the design of layers for restoring acoustic features, have to be addressed.

## 3.1 DcAE based on LDA and i-vectors

### 3.1.1 Introduction

Even though the methodology of deep neural networks (DNNs) has seemingly become more and more popular in the field of speaker recognition and obtained some gains in performance [67, 77, 131, 140, 167, 177], either total variability modeling (i-vector) [24] or probabilistic linear discriminant analysis (PLDA) [66], as well as their modifications, are still indispensable and robust ingredients in most of current speaker verification systems. The aim of *i-vector* is to represent variable-length speech signals by fixed-size vectorial tokens while the session/channel variabilities induced by various sources are compensated and the speaker characteristics are abundantly and ulteriorly preserved [76]. Given two i-vectors, the task of *PLDA* is to linearly discriminate between speakers in a low-rank subspace and give a reasonable metric to measure their decision score in a probabilistic sense [32, 83].

Actually, going through the latest three ICASSP proceedings (2014-16), more than three-fourths of papers dealing with speaker verification use PLDA as one of their scoring



backends, among which there are much fewer efforts to either develop their own competitive algorithms for backend speaker discrimination or make some improvements to PLDA by standing on its shoulders. For example, Rohdin *et al.* gave a discriminative PLDA training algorithm, where some constraints are imposed on the derivation of the speaker variability matrix [134]. Similar to the work by Lee *et al.* [76], Cumani and Laface employed pairwise support vector machines (SVMs) to efficiently classify pairs of i-vectors as belonging or not to the same speaker even with large-scale datasets [18]. Nautsch *et al.* proposed a PLDA-alike approach with restricted Boltzmann machines (RBM), which aims at suppressing channel effects and recovering speaker-discriminative information on a small dataset [107]. Most recently, Heigold *et al.* used DNNs and long short term memory (LSTM) to represent utterances and directly map each trial set of utterances to a decision score for verification [56].

In this study, we replace the role of PLDA with an *autoencoder* and tweak its objectives for speaker discrimination. The autoencoder is a symmetric neural network that is trained to approximately copy its input to the output [45]. Besides the reconstruction error, which makes the autoencoder analogous to a generative model that benefits to unsupervisedly learn most salient and useful properties of the data, two more objective functions are concerned in our proposed framework. They attempt to ensure that utterances spoken by the same speaker would have similar identity codes (i-codes) in the speaker-discriminative subspace represented by the middlemost hidden layer, where the scatterness of all i-codes are also maximized to some extent to avoid the effect of over-concentration. Finally, the decision score of each utterance pair is simply computed by cosine similarity of their i-codes. To our best knowledge, despite the autoencoder has been widely applied to many speech processing tasks, such as speech enhancement [2, 122], acoustic novelty detec-

tion [93], and reverberant speech recognition [100], much less papers used it directly for speaker recognition.

Most important of all, our contributions are two-fold. First, we present a kind of neural network-based discriminant analysis, which consumedly and nonlinearly extends the capability of PLDA. Second, our proposed model is immune to computational intractability, e.g., matrix inversion when the training set becomes very large [32, 66].

Our proposed framework, its objectives and analogy with PLDA, and its realization and evaluation, are given and reported in the remainder of this study.

### 3.1.2 Objectives

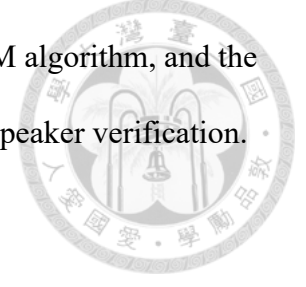
#### 3.1.2.1 Probabilistic linear discriminant analysis

The PLDA model assumes that the  $j$ -th utterance (i-vector) of the  $i$ th speaker is described by the following process [32]:

$$\mathbf{x}_{ij} = \underbrace{\boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i}_{\text{Identity}} + \underbrace{\mathbf{G}\mathbf{w}_{ij} + \boldsymbol{\epsilon}_{ij}}_{\text{Noise}}, \quad (3.1)$$

where  $\boldsymbol{\mu}$  and  $\boldsymbol{\epsilon}_{ij}$  denote the global mean and the residual following a Gaussian distribution with zero mean and diagonal covariance  $\Sigma$ , respectively. By (3.1), each  $\mathbf{x}_{ij}$  is factorized into two parts. In the identity part, the matrix  $\mathbf{F}$  denotes the subspace, where the utterances that belong to the same speaker would have the same projective location or speaker identity, characterized by a hidden variable  $\mathbf{h}_i$ . As for the noise part, all other information irrelevant to speaker discrimination is thrown into the subspace  $\mathbf{G}$  and locates in a noise factor  $\mathbf{w}_{ij}$ . Both  $\mathbf{h}_i$  and  $\mathbf{w}_{ij}$  are standard Gaussian distributed.

The model parameters  $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{F}, \mathbf{G}\}$ , can be estimated by the EM algorithm, and the likelihood of a set of i-vectors can be used as the decision score for speaker verification.



### 3.1.2.2 Three PLDA-inspired objective functions

By the framework of PLDA, we have two kinds of sights. First, PLDA is a generative model, equipped with an inference procedure for computing the distribution of latent variables given an observation, and providing a generative procedure for stochastically generating a copy of an observation. Second, given two utterances  $\mathbf{x}_{11}$  and  $\mathbf{x}_{21}$  that belong to different speakers, PLDA seems *not* to explicitly ensure that their speaker identities  $\mathbf{h}_1$  and  $\mathbf{h}_2$  will be consumedly different. This might increase the number of false alarms in verification tasks. Therefore, we design three kinds of objective functions for speaker discrimination in order to take the essence of PLDA and make up for its deficiency.

**1) Reconstruction Error** Suppose our proposed model  $\mathcal{M}$  contains a pair of deterministic mappings  $f(\cdot)$  and  $g(\cdot)$ , which are responsible for latent variable inference and observation generation in the terminology of Bayesian inference, respectively. Given a set of training data  $\mathcal{X}$  ready to go through the inference-generation process by  $\mathcal{X} \xrightarrow{f} \mathcal{H} \xrightarrow{g} \mathcal{X}$ , where  $\mathcal{H}$  is the internal representations residing in a latent subspace, the average reconstruction error based on the residual sum of squares between  $\mathbf{x} \in \mathcal{X}$  and its reconstruction  $\mathbf{y} = g(f(\mathbf{x}))$  is given by

$$F_r(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_2^2, \quad (3.2)$$

where  $\|\cdot\|_2$  is the 2-norm operator and  $|\mathcal{X}|$  is the sample size.

Like a copy machine,  $\mathcal{M}$  is usually restricted in ways that allow it *to copy only approximately, and to copy only input that resembles the training data. Because the model*

is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data.



**2) The speaker identity loss** According to the above italic description quoted from [45], we can suppose  $\mathcal{H}$  contains salient and useful features, certainly, including untreated speaker characteristics, by minimizing (3.2). Without loss of generality, we split  $\mathcal{H}$  into two parts,  $\mathcal{H}_s$  and  $\mathcal{H}_n$ , where  $\mathcal{H}_s$  is supposed to contain all of information for speaker discrimination, and  $\mathcal{H}_n$  possesses the residual content. Suppose  $\mathcal{H}_s = \{\mathcal{H}_{s1}, \dots, \mathcal{H}_{sm}\}$ , where  $\mathcal{H}_{si}$  corresponds to those training data that belong to the  $i$ -th speaker of  $m$  training speakers, and the loss function related to speaker identity is described as follows:

$$F_s(\mathcal{H}_s) = \frac{1}{m} \sum_{i=1}^m \left( \frac{1}{|\mathcal{H}_{si}|} \sum_{\mathbf{h} \in \mathcal{H}_{si}} \|\mathbf{h} - \bar{\mathbf{h}}_{si}\|_2^2 \right), \quad (3.3)$$

where  $\bar{\mathbf{h}}_{si}$  denotes the empirical mean vector of  $\mathcal{H}_{si}$ . Apparently, the term in parentheses in (3.3) measures the average within-speaker *compactness*. Therefore, to minimize (3.3) implies to increase the similarity score between two utterances that belong to the same speaker if the score metric is 2-norm-related. In this way, the number of false rejects in verification tasks will conceivably decrease to some extent.

**3) The internal dispersion** The last objective function to be minimized is about the dispersion of *total* internal representations, which is given by

$$F_d(\mathcal{H}_s) = -\frac{1}{|\mathcal{H}_s|} \sum_{\mathbf{h} \in \mathcal{H}_s} \|\mathbf{h} - \bar{\mathbf{h}}_s\|_2^2, \quad (3.4)$$

where  $\bar{\mathbf{h}}_s$  denotes the empirical mean vector of  $\mathcal{H}_s$ . There are two reasons to support the necessity of (3.4). First, the minimum of  $F_s(\mathcal{H}_s)$  in (3.3) might naturally become *zero* when the model makes all of  $\mathbf{h} \in \mathcal{H}_s$  turn out to be the same. Second, analogically speaking, the

goal of linear discriminant analysis (LDA) is to maximize the Rayleigh quotient given by the *total* scatterness over the within-class scatterness [35]. The expression is equivalent to the traditional attempt to maximize the between-class scatterness in the whitened space [74]. Therefore, the cooperation of (3.3) and (3.4) can keep a distance between utterances that belong to different speakers so as to reduce the errors caused by false alarms.

Finally, by combining (3.2), (3.3), and (3.4), the new goal of the training process is to find an optimal  $\mathcal{M}$  by minimizing

$$F_r(\mathcal{X}) + \alpha (\beta F_s(\mathcal{H}_s) + (1 - \beta) F_d(\mathcal{H}_s)) + \lambda \|\mathcal{M}\|_2^2, \quad (3.5)$$

where  $\alpha > 0$  controls the relative importance of the last two objective functions to the reconstruction error,  $\beta \in [0, 1]$  adjusts the ratios between the speaker identity loss and the internal dispersion that we want to emphasis, and  $\lambda$  is a regularization parameter that controls the complexity of the model. The treatment of  $\alpha$  and  $\beta$  is akin to that in ElasticNet [189].

### 3.1.3 Realizations

We use an artificial neural network-based autoencoder, called the discriminative autoencoder (DcAE), to realize  $\mathcal{M}$  along with the objective function (3.5). In contrast with another work that trains the autoencoder in a discriminative way in [129], the cost function of DcAE is not only determined on the output layer but also highly affected by the middlemost code, i.e., the internal representation. The detailed structure and definition of the autoencoder can be referred to [6], [170], and [45].

The architecture of DcAE is depicted in Figure 3.1. It maps an input  $\mathbf{x}$  to an output

y through an internal representation or code  $\mathbf{h}$ , which is split into  $\mathbf{h}_s$  and  $\mathbf{h}_n$  as described in Section 2.2.2. The encoder  $f(\cdot)$  and the decoder  $g(\cdot)$  are built up of full-connected hidden layers and their corresponding weights and biases. The activation flows through the hidden layers and the code layer by means of the hyperbolic tangent function, until it reaches the final layer and linearly gives the output.

To train DcAE, a back-propagation process is implemented from the output layer down through the whole DcAE to adjust all parameters. The gradient of each parameter in  $\mathcal{M}$  can be easily derived by partial differentiation on the cost function (3.5), so that the model can be iteratively updated by using an optimizer based on gradient descent.

To generate the decision score  $s_{1,2}$  of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  for speaker verification, we simply use the cosine similarity of their i-codes, i.e.,  $(\mathbf{h}_{s1} \cdot \mathbf{h}_{s2})/|\mathbf{h}_{s1}||\mathbf{h}_{s2}|$ .

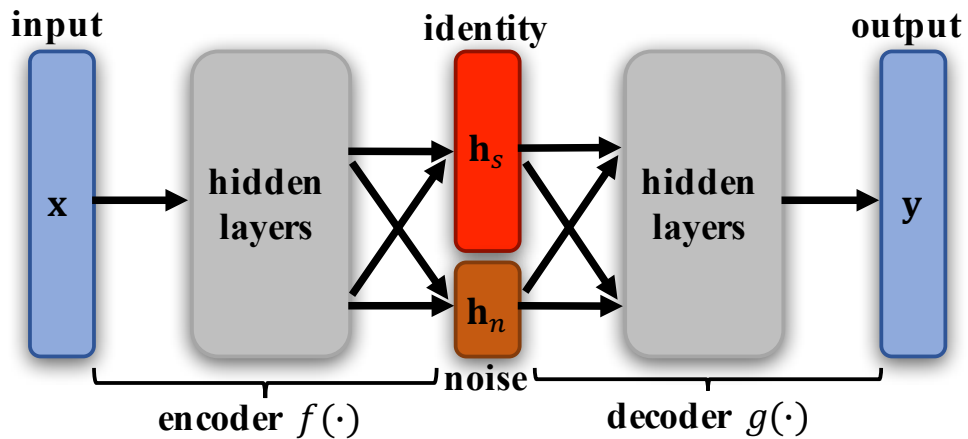


Figure 3.1: Architecture of i-vector based DcAE.



### 3.1.4 Experiments and Results



#### 3.1.4.1 Experiment setup

All the experiments were carried out on the male portion of the core task in NIST SRE-10 (core-core/condition-5) for evaluation and NIST SRE-08 (short2-short3/condition-6) for model validation based on the equal error rate (EER), where each session is an excerpt of five-minutes telephone speech [111, 113]. With the frame length of 25 ms and the frame shift of 10 ms, speech parameters were represented by a 60-dimensional feature vector of Mel-frequency cepstral coefficients (MFCCs) with first and second derivatives appended using a 2-frame window, followed by data distribution-based feature warping with a 300-frame window in order to compensate for the effects of environmental mismatch [120].

A gender-dependent UBM consisting of 1,024 Gaussian components with diagonal covariance matrices, the total variability model (for i-vectors) with rank 400, the PLDA model, as well as our proposed DcAE model, were trained with 8,511 utterances spoken by 413 speakers, drawn from NIST SRE-04 and SRE-05, Switchboard II-Phase 1, 2 and 3, and Switchboard Cellular Part 1 and 2. The i-vectors were length-normalized prior to PLDA and DcAE training [37].

For DcAE, we set the tunable numbers of nodes in the identity and noise layers to 300 and 100, respectively. The number of hidden nodes is 400 for all hidden layers. Initial weights are uniformly sampled by the Glorot process that is fit for the tanh activation function [42]. The adaptive gradient algorithm adopted to update the model parameters is AdaGrad, which scales the learning rate by dividing with the square root of accumulated

Table 3.1: Results for SRE-10. The percentages are the relative improvements over PLDA-250.

Method	EER	NMDC
Cosine	10.99	0.47
PLDA-250	6.20	0.29
DcAE-0	<b>3.94 (36%)</b>	<b>0.22 (24%)</b>



squared gradients [30].

### 3.1.4.2 Results compared with baselines

The DET curves with respect to various baselines and our proposed methods are shown in Figure 3.2, where “PLDA-300” stands for the PLDA model with  $\mathbf{F}$  of rank 300 in (3.1), “cosine” means the cosine kernel presented in [24], and “DcAE-1” represents the DcAE model with one hidden layer in both encoder and decoder parts. Obviously, our proposed method performs much better than PLDA whatever the cost parameters are. Table 3.1 also shows that, compared with PLDA-250, DcAE-0 achieves 36% and 24% relative improvements in EER and normalized minimum detection cost (NMDC).

On the other side, it can be seen that DcAE-1 and DcAE-2 do not outperform DcAE-0 as expected, although their results are acceptable while compared with PLDA. Actually, for the sake of convenience, all of the parameters for training the structure of DcAE with hidden layers are copied from the well-tuned parameters based on DcAE-0. Besides, the training set with less than 10,000 samples dose not seem to be enough for a more complicated structure.

Moreover, to demonstrate the effectiveness of speaker discrimination of DcAE, we arbitrarily single out two sets of sessions from SRE-10, which belong to two different speakers, to be visualized by t-Distributed Stochastic Neighbor Embedding (t-SNE) [166].

Figure 3.3 illustrates the results that t-SNE maps the corresponding 400-dimensional i-vectors and the 300-dimensional i-codes into a 2-dimensional plane.

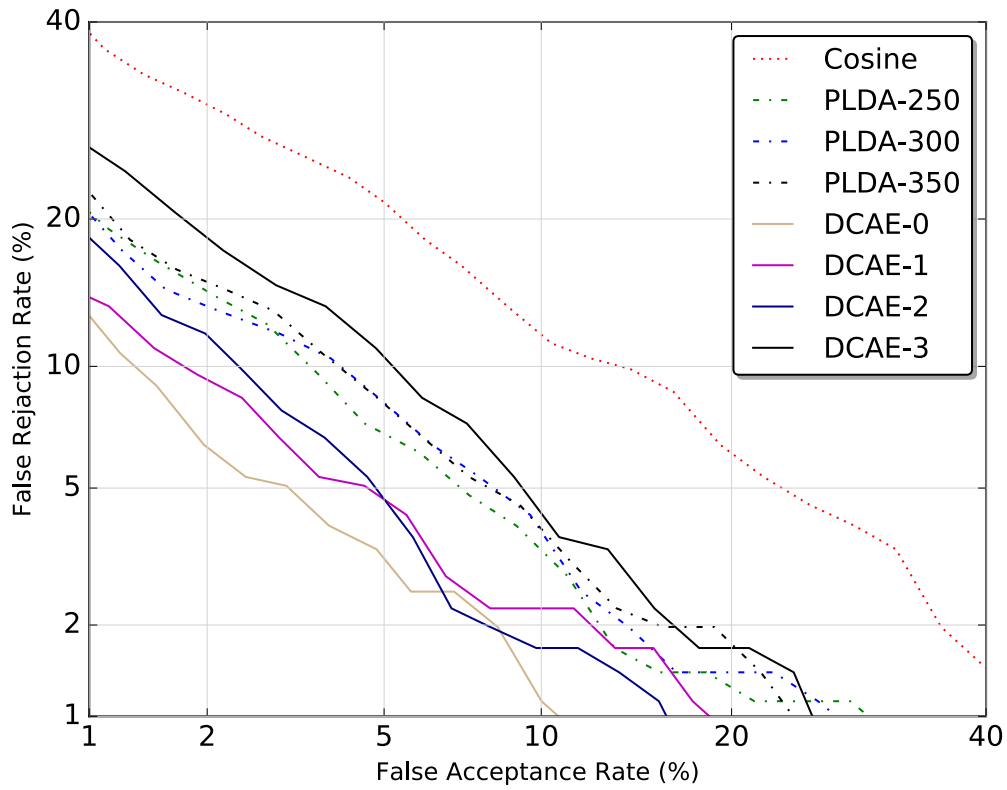
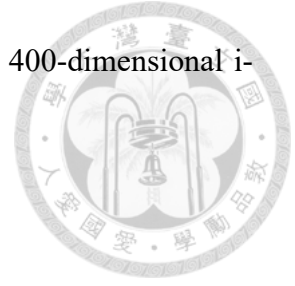


Figure 3.2: DET curves of DcAE, PLDA, and cosine for SRE-10.

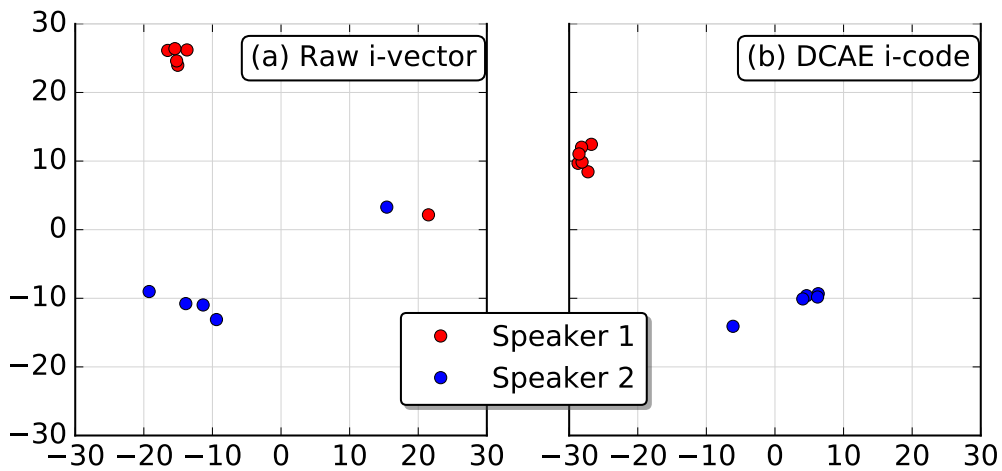


Figure 3.3: Scatter plot of the results by t-SNE for SRE-10.



### 3.1.4.3 Observations on DcAE training

We made some observations from the training process of DcAE-0. First, *ceteris paribus*, we recorded the validation results in each training epoch to see the trend of different learning rates, batch sizes, and L2 weight regularization penalties (i.e.,  $\lambda$  in (3.5)). From Figure 3.4, it can be seen that, in most cases, the best parameter usually occurs within 10 training epochs. This property for DcAE training makes the tuning of hyper-parameters much more tractable.

Second, we were also interested in the relationship among the three objective functions proposed in Section 2. As depicted in Figure 3.5, most of lower EERs occur when  $\alpha$ , the weightiness of speaker identity loss and internal dispersion, is relatively small. This

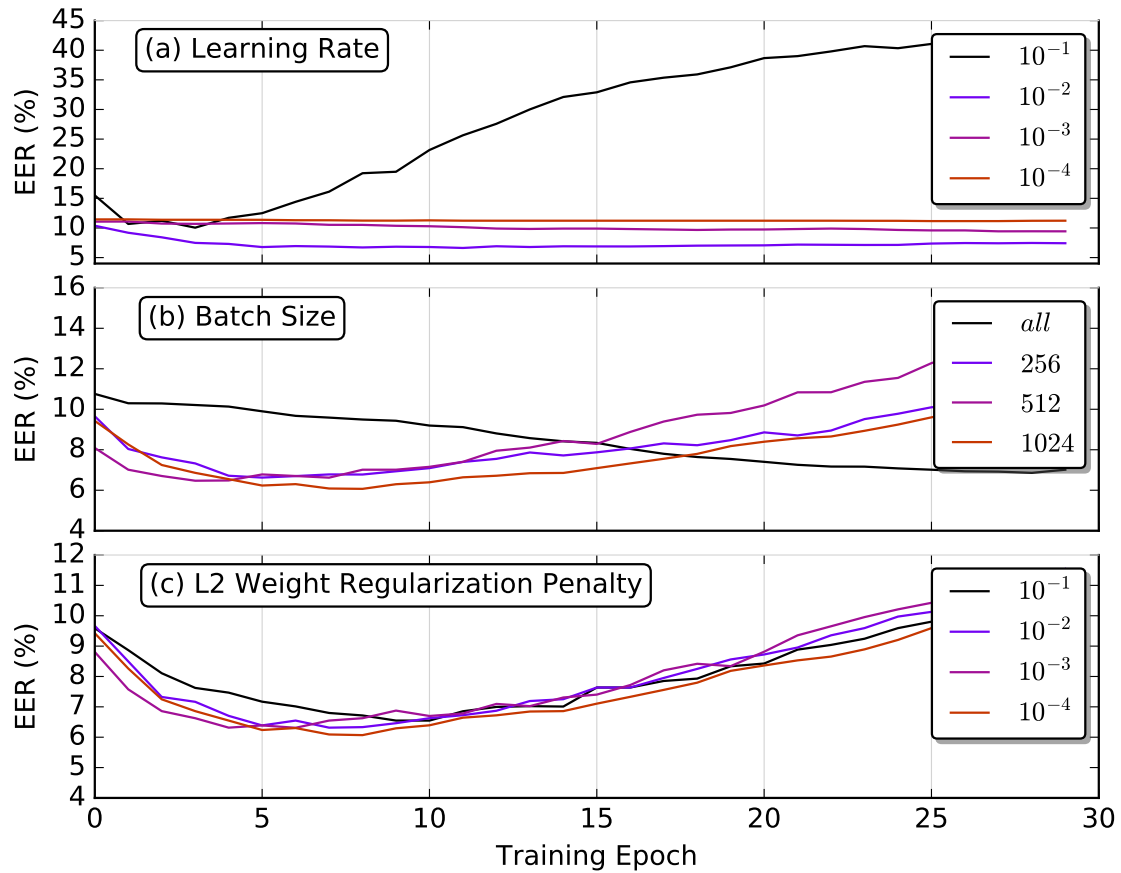


Figure 3.4: EERs of SRE-08 for various settings of hyper-parameters with respect to the training epoch in DcAE.

implies that the underestimation of the generative aspect of DcAE dose not necessarily help increase the discriminative power of the learning machine.



### 3.1.5 Conclusions

In this study, we have proposed a framework based on three kinds of objective functions, which cooperate to make our model more like a generative model that possesses discriminative power for speaker verification. The framework has been realized by a neural network-based autoencoder, where the implementation is tractable for big data. The experiment results demonstrated the potential of the framework.

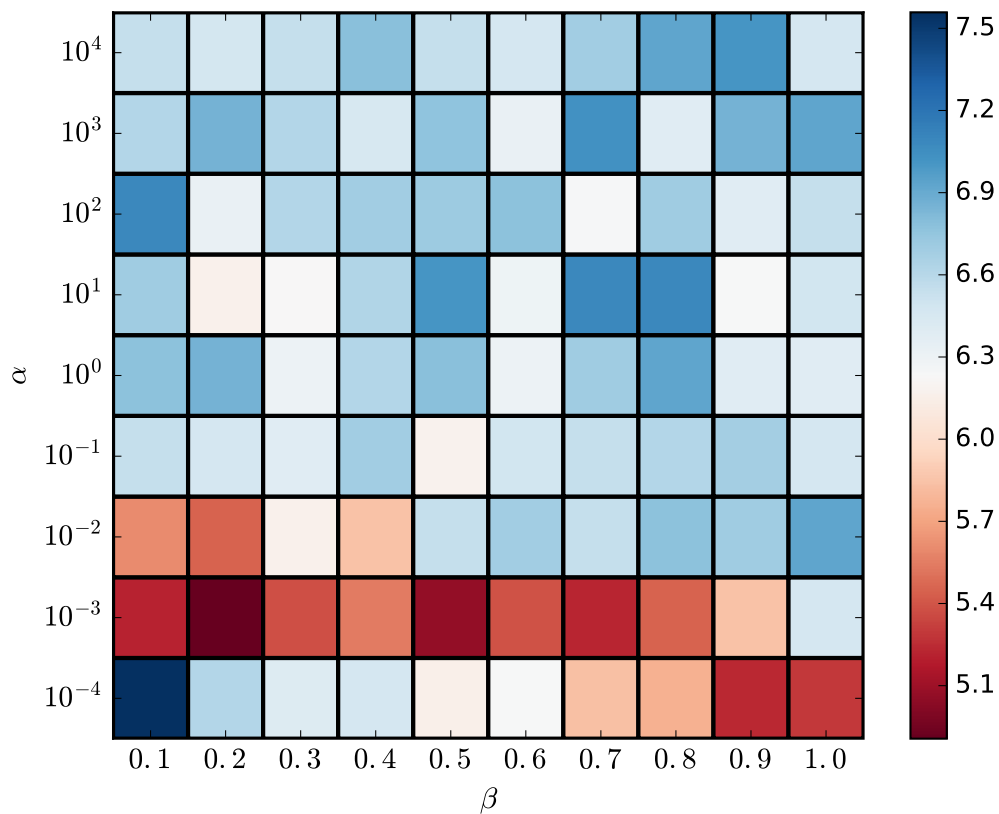


Figure 3.5: Heat map generated from validation results reflecting EERs of SRE-08 in various settings of  $\alpha$  and  $\beta$  in (3.5).

## 3.2 DcAE based on Acoustic Features and ResNet

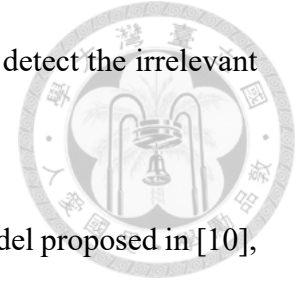


### 3.2.1 Introduction

In recent years, x-vectors [152], using the time-delay neural network (TDNN) architecture, and their subsequent improvements [151] have consistently provided state-of-the-art results on the task of speaker verification. The neural networks, called speaker models, are usually trained on the speaker identification task first. After the training stage, vectorial speaker embeddings can be extracted from the penultimate or third from bottom layer preceding the output layer to characterize the speaker in the input recording. Speaker verification can be accomplished by comparing the two speaker embeddings corresponding with an enrollment and a test recording to accept or reject the hypothesis that both recordings come from the same speaker. The cosine measurement or PLDA score can be used for this comparison.

The rising popularity of the x-vector system has resulted in significant architectural improvements and optimized training procedures over the original approach. The topology of the system was improved by incorporating elements of the popular ResNet architecture [55]. Adding residual connections between the frame-level layers has been shown to enhance the embeddings. Additionally, residual connections enable the back-propagation algorithm to converge faster and help avoid the vanishing gradient problem. The statistics pooling layer in the x-vector system projects the variable-length input into a fixed-length representation by gathering simple statistics of hidden node activations across time. The authors in [115, 186] introduced a temporal self-attention system to this pooling layer which allows the network to only focus on frames it deems important. It can also be in-

interpreted as a Voice Activity Detection (VAD) preprocessing step to detect the irrelevant non-speech frames.



In this study, we followed the implementation of the speaker model proposed in [10], and took the ResNet architecture as the encoder part of DcAE. For the decoder part of DcAE, we used several deconvolutional layers to restore the acoustic features.

We will not go into details about the experiment settings and model structures, but give primitive ideas and results.

### 3.2.2 Proposed Model

As shown in Fig. 3.6, the proposed DcAE is composed of three parts: the encoder, codes (S-Code and R-Code), and decoder. The encoder is realized by a ResNet, where several convolutional blocks sequentially transform frame-based acoustic features into two disjoint feature maps, i.e., the speaker code (S-Code) and the residual code (R-Code). S-Code represents speaker-related information in an utterance while R-Code covers speaker-irrelevant factors. During training, the decoder, implemented by several deconvolutional layers, is expected to restore the original acoustic features from S-Code and R-Code and to minimize the mean square error  $L_{MSE}$  between predicted features and true acoustic features.

The separation of S-Code and R-Code relies on the minimization of two loss functions. The first one  $L_S$  is the categorical cross entropy between the true speaker labels and the speaker likelihood scores derived from S-Code. The second one  $L_R$  is the binary cross entropy to have the speaker likelihood scores derived from R-Code be nearly uniformly distributed, i.e., making R-Code contain less information to distinguish speakers. Note

that the classifier (Linear) shares the same weights for both codes.

The objective function to be minimized is expressed by

$$L = \alpha L_{MSE} + (1 - \alpha)(\gamma L_R + (1 - \gamma)L_S), \quad (3.6)$$

where  $\alpha$  and  $\gamma$  are weighting factors that need to be tuned.

### 3.2.3 Primitive Experimental Settings and Results

For the convenience of implementation, we used the convolutional neural network (CNN) based architecture similar to that described in [10] for speaker modeling in our proposed scheme. As shown in Table 3.2, after feed-forwarding through 5 ResNet-based layers (res1 to res5) and an operation of average pooling, a 256-dimensional speaker

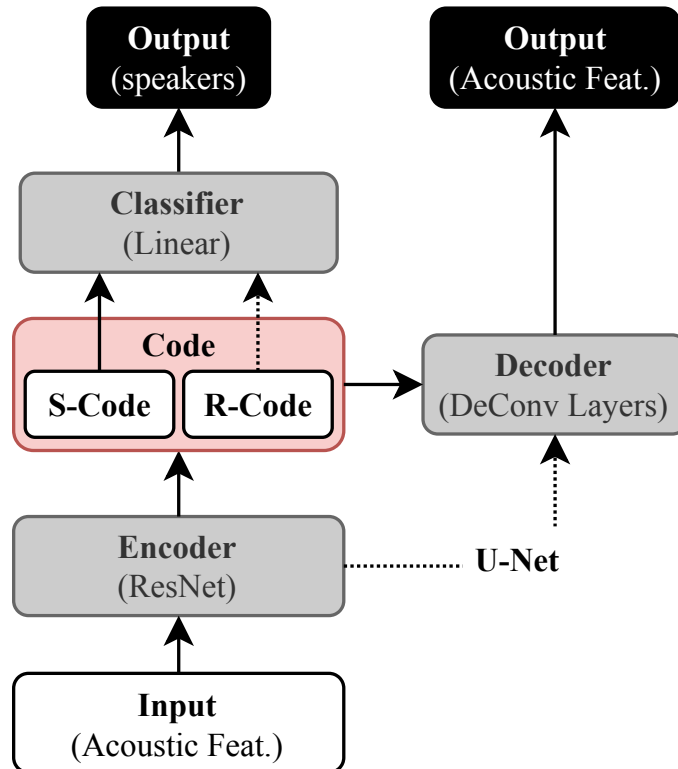


Figure 3.6: Structure of ResNet-based discriminative autoencoders (DcAE) for speaker verification.





Table 3.2: Architecture and specifications of ResNet-34, where res denotes the ResNet-based layer, and  $B$ ,  $T$ ,  $D$ , and  $C$  represent the batch size, temporal length, feature dimensionality, and number of training speakers, respectively.

Layer	Feature Size	Downsample	# Blocks
input	$B \times 1 \times T \times D$	-	-
conv	$B \times 16 \times T \times D$	False	-
res1	$B \times 16 \times T \times D$	False	3
res2	$B \times 32 \times \frac{T}{2} \times \frac{D}{2}$	True	4
res3	$B \times 64 \times \frac{T}{4} \times \frac{D}{4}$	True	6
res4	$B \times 128 \times \frac{T}{8} \times \frac{D}{8}$	True	3
res5	$B \times 256 \times \frac{T}{16} \times \frac{D}{16}$	True	3
pooling	$B \times 256$	-	-
linear1	$B \times 256$	-	-
linear2	$B \times C$	-	-

embedding was derived in the linear1 layer while the linear2 layer was the penultimate layer for speaker classification. To train the ResNet-34 model, the utterances were chopped into segments of 3 to 8 seconds, then the 64-dimensional Mel-filter banks were extracted with the frame length of 0.025 seconds and the frame shift of 0.01 seconds as the input.

As shown in Table 3.3, results on the task of VoxCeleb-1, where the models were trained using the training set of VoxCeleb-1, proves that the best of our proposed model outperform the baseline with 11.88% and 10.92% relative EER reductions, when using softmax/temporal average pooling and a-softmax/statistical pooling, respectively.

The models were also conducted on the task of NIST SRE-16, where the models were trained using training sets from NIST SRE-04, 05, 06, 08, Fisher, and SwitchBoard.

Table 3.3: Results on the task of VoxCeleb-1, where the models were trained using the training set of VoxCeleb-1.

Method	Loss Type	Aggregation	EER %	rel. %
Baseline	softmax	temporal average pooling	5.05	
	a-softmax	statistical pooling	4.03	
DcAE (B)	softmax	temporal average pooling	4.75	5.94
	a-softmax	statistical pooling	3.73	7.44
DcAE (U)	softmax	temporal average pooling	4.79	5.15
	a-softmax	statistical pooling	3.73	7.44
DcAE (R)	softmax	temporal average pooling	<b>4.45</b>	11.88
	a-softmax	statistical pooling	<b>3.59</b>	10.92

Table 3.4: Results on the task of NIST SRE-16, where the models were trained using training sets from NIST SRE-04, 05, 06, 08, Fisher, and SwitchBoard.

Method	Loss Type	Aggregation	EER %	rel. %
Baseline	softmax	temporal average pooling	10.9	
	a-softmax	statistical pooling	10.43	
DcAE (B)	softmax	temporal average pooling	10.71	1.74
	a-softmax	statistical pooling	9.92	4.89
DcAE (U)	softmax	temporal average pooling	10.55	3.21
	a-softmax	statistical pooling	<b>9.71</b>	6.90
DcAE (R)	softmax	temporal average pooling	<b>10.47</b>	3.94
	a-softmax	statistical pooling	9.96	4.51

As shown in 3.4, the best of our proposed model outperform the baseline with 3.94% and 4.51% relative EER reductions, when using softmax/temporal average pooling and a-softmax/statistical pooling, respectively.



## Chapter 4

# Discriminative Autoencoders for Speech Recognition

The introduction of discriminative autoencoders (DcAE) in automatic speech recognition originated in our proposed paper of Interspeech 2017 [180].

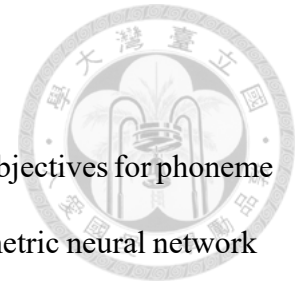
Building rich generative models, which are capable of extracting or preserving useful, salient, and high-level latent information from the high-dimensional, context-embedded sensory input, lies at the core of solving many machine learning tasks [159]. These models capture the underlying structure of data by defining flexible probability distributions over high-dimensional data as part of a complex, partially observed system. Recently, some of the successful generative models, such as variational autoencoders (VAE) [70, 158] and generative adversarial networks (GAN) [46], were proposed to discover meaningful and high-level latent representations. In light of these developments, we attempted to propose a new framework, named discriminative autoencoders (DcAE), for acoustic modeling based on autoencoder neural networks. With a proper use of sample labels, not only can the optimization of DcAEs be handled in a discriminative fashion, but the whole

model inherits the aforementioned merits of generative models.

We replaced the role of DNN with an autoencoder and tweak its objectives for phoneme classification and speaker discrimination. The autoencoder is a symmetric neural network that is trained to approximately copy its input to the output. In addition to the reconstruction error, which makes the autoencoder analogous to a generative model that benefits to unsupervisedly learn most salient and useful properties of the data, three additional objective functions are also considered in our proposed framework. The first one is used to estimate the categorical distribution, i.e., the posterior probabilities of the context-dependent phone states, in the middlemost code layer to form a phoneme code vector (p-vector) for each acoustic frame. The other two objectives attempt to ensure that utterances spoken by the same speaker would have similar representations in the speaker-discriminative subspace represented in the code layer. Finally, the acoustic score of each acoustic frame is derived by its p-vector. To our best knowledge, although autoencoders have been widely applied to many speech processing tasks, such as speech enhancement [2], acoustic novelty detection [93], and robust feature extraction in ASR [39, 138, 185], fewer papers used them directly for acoustic modeling.

Unlike other variants of autoencoder neural networks, our framework is able to isolate phonetic components from a speech utterance by simultaneously taking two kinds of objectives into consideration, which cooperate to make our model behave like a generative model that possesses discriminative power for ASR. The experimental results demonstrated that our proposed framework outperforms the conventional DNN-based methods.

In the following sections, we will introduce two kinds of developments of DcAE. One explores more complicated encoder layers and the other deals with the robust ASR.



Each variants of DcAE can be counted as a big progress of the originally proposed DcAE in [180].



## 4.1 Exploring the Encoder Layers of DcAE

### 4.1.1 Introduction

Due to the introduction of deep neural networks (DNNs) and the fact that computing power has advanced by leaps and bounds over the past decade, automatic speech recognition (ASR) has gained more attention from various artificial intelligence (AI)-related fields and communities. DNNs have also become mainstream architectures for acoustic modeling in large vocabulary continuous speech recognition (LVCSR) instead of traditional GMM-HMM (hidden Markov model with Gaussian mixture model emissions) based models [25, 57]. What have followed naturally are many free-to-use toolkits supporting the usage of GPUs for building ASR systems, such as Microsoft's CNTK<sup>1</sup>, RWTH Aachen University's RASR<sup>2</sup>, and Kaldi developed primarily by Daniel Povey and his research team [124].

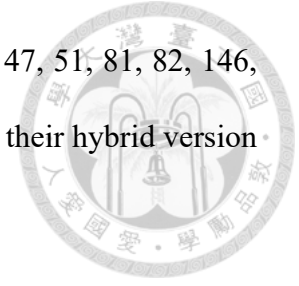
There is no doubt that Kaldi is one of the most popular open-source toolkits, providing industrial engineers and academic researchers working in the speech processing area with the most advanced and regularly updated training recipes and a fair ground for comparison. Based on the topology of HMMs and weighted finite-state transducers (WFSTs), Kaldi also generates high-quality word/phone lattices that are not only sufficiently efficient for real-time decoding but also more effective than other potential and promising end-to-end

---

<sup>1</sup><https://www.microsoft.com/en-us/cognitive-toolkit>

<sup>2</sup><https://www-i6.informatik.rwth-aachen.de/rwth-asr>

approaches, such as connectionist temporal classification (CTC) [3, 47, 51, 81, 82, 146, 153], sequence-to-sequence attention-based models [4, 14, 157], and their hybrid version [20].



In Kaldi, a standard recipe for GMM-DNN-HMM-based ASR generally consists of three steps, as shown in Figure 4.1. In the first step of preparation, users must provide a pronunciation lexicon and language models corresponding to the target language or output texts, and specify the speech utterances and their corresponding transcripts for training and evaluation. In this step, training data augmentation techniques, such as speed/volume perturbations [71] and multi-condition training with noise and reverberation additions [72, 149], can be considered. In the second step, based on the defined speech units (e.g., phonemes or sub-syllables), the GMM-HMM system is initialized by monophone model training and extended to a context-dependent triphone system. In the last step, a more accurate acoustic model is developed by DNNs based on the frame-level triphone alignment generated by the pre-trained GMM-HMM system. The acoustic model built up by DNNs is suitable for integrating higher-resolution acoustic features and i-vectors, which contain abundant speaker-related information [24].

As for DNN modeling, there are currently four setups in Kaldi, namely `nnet1`, `nnet2`, `nnet3`, and `chain`. The main difference between `nnet1` and `nnet2` is that the former uses DNN-generated alignment and mini-batch stochastic gradient descent (SGD), while the latter uses GMM-based alignment and parallel SGD with periodic model averaging. `nnet3` is an extension of `nnet2` designed to support a wider variety of networks than simple feed-forward networks (FFNs) [19]. `chain`, based on `nnet3`, uses a 3 times smaller frame rate at the output of the neural network, and uses additional lattice-free maximum mutual information (LF-MMI) for training [92].

In this study, according to the standard recipe mentioned above, we attempt to improve the two existing acoustic models implemented through the library of `nnet3` without the training criterion of LF-MMI. One is time-delay neural networks (TDNNs), which are also known as one-dimensional convolutional neural networks (1-d CNNs). It has been shown that they can effectively learn the temporal dynamics of signals from short-term feature representations [118, 171]. The other, built up of interleaving TDNN layers and long short-term memory (LSTM) layers [58], is the TDNN-LSTM structure [119], which has a wide temporal context and performs as well as bidirectional LSTM networks with less latency [48]. Following the work of discriminative autoencoders (DcAEs) for acoustic modeling in [180], where the encoder layers were only implemented by deep FFNs with temporarily augmented fMLLR features, we separately use TDNNs and TDNN-LSTM networks as encoders to see if the advantages of DcAEs can be sustained and developed. As described in [45], an autoencoder can help to preserve the most salient information. Furthermore, as a feature regularizer, it has the advantage of improving generalization of the learned acoustic models by increasing the ability to reconstruct the input features from

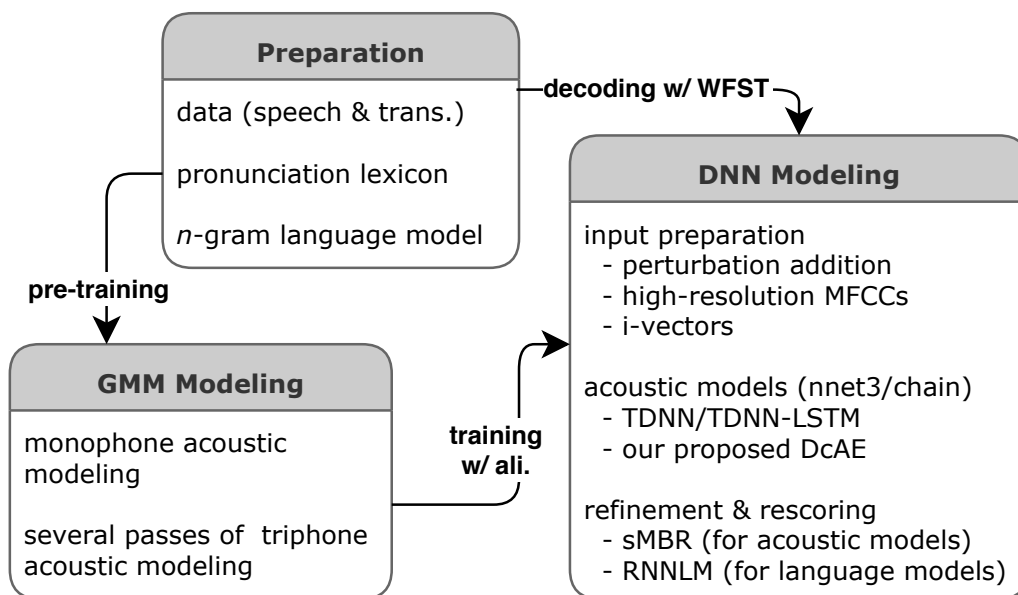
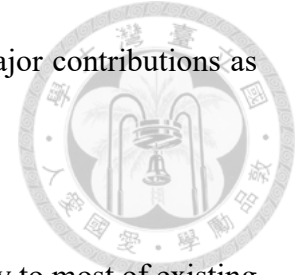


Figure 4.1: Diagram of the standard GMM-DNN-HMM-based recipe of Kaldi adopted in this study.

the frame embeddings. Our work in this thesis has at least three major contributions as follows:



1. The structure of our proposed model can be applied effortlessly to most of existing or prospective neural network models without increasing the decoding costs and size of the inference model, demonstrating its flexibility and universality. Based on our modified nnet3 setup and recipe<sup>3</sup>, comparison with other models and application to other corpora can be realized easily in Kaldi.
2. The mechanism of skip connections (or the u-net) is included in DcAEs for better output regularization [135].
3. The results of LVCSR experiments on two corpora, namely the MATBN Mandarin Chinese corpus and the WSJ English corpus, indicate that our proposed method can be applied across languages.

## 4.1.2 Discriminative Autoencoders

### 4.1.2.1 Architectures

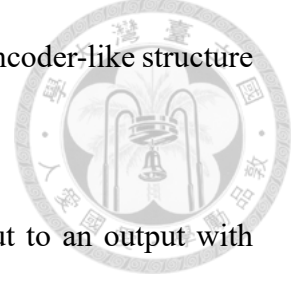
For standard ASR tasks built with the Kaldi nnet3 setup, acoustic features are typically fed into the TDNN or TDNN-LSTM model frame-wisely to extract phoneme-related information for triphone state prediction, as shown in Figure 4.2. It is worth noting that the actual output vector is a one-hot vector representing a categorical distribution over a set of events, i.e., triphone states. To help the extraction of phoneme-related information, we attached FFN layers as a decoder to the penultimate layer of the TDNN or TDNN-LSTM

---

<sup>3</sup><https://github.com/melodyhpt/Kaldi-DcAE>



model and treated the baseline model as an encoder, creating an autoencoder-like structure (cf. DcAE-B in Figure 4.2).



Autoencoders are learning models designed to convert an input to an output with minimal distortion. The embedded internal representation, namely the code layer, consists of the most critical information that minimizes the reconstruction error. To obtain accurate acoustic scores for ASR, we divided the code layer into two sub-representations: the phoneme-aware layer (p-code) and the residual layer (r-code). The p-code consisted of phonetic information and was connected to the original output layer of an ASR task. The r-code carried information within the acoustic frame that was unrelated to phonetic information, which could include environmental noise, speaker identity, and any other possible information that was considered useless for the task. The p-code and r-code were then concatenated as the input of the decoder for reconstructing the acoustic frame.

On top of the DcAE-B architecture, we further added skip connections between the encoder and the decoder (cf. DcAE-U in Figure 4.2). Based on the concept of momentum [126], which is a common optimization method that helps speed up SGD in the relevant direction and reduce oscillations by adding a fraction of the update vector from the past time step to the current update vector, we wish to use the information from the encoder as a kind of momentum to guide the training of the autoencoder with skip connections. The implemented structure is similar to the u-net [135], which has achieved great success in medical image segmentation tasks. We connected the outputs of the encoder layer with those of the decoder layer, producing new inputs for the next layer of the decoder as

$$D_{j+1}^{in} = E_i^{out} \oplus \beta D_j^{out}, \quad (4.1)$$

where  $E_i^{out}$  and  $D_j^{out}$  denote the outputs of the  $i^{th}$  layer of the encoder and  $j^{th}$  layer of the decoder, respectively,  $D_{j+1}^{in}$  denotes the input of the  $j + 1^{th}$  layer of the decoder,  $\beta$  is used to control the strength of the momentum, and  $\oplus$  denotes the connection operation. In this study, we use two kinds of skip connections, one by appending and the other by summation.

#### 4.1.2.2 Objective function

The objective function used in the proposed framework includes two parts: the mean square error for feature reconstruction and the phoneme-aware cross-entropy for acoustic modeling.

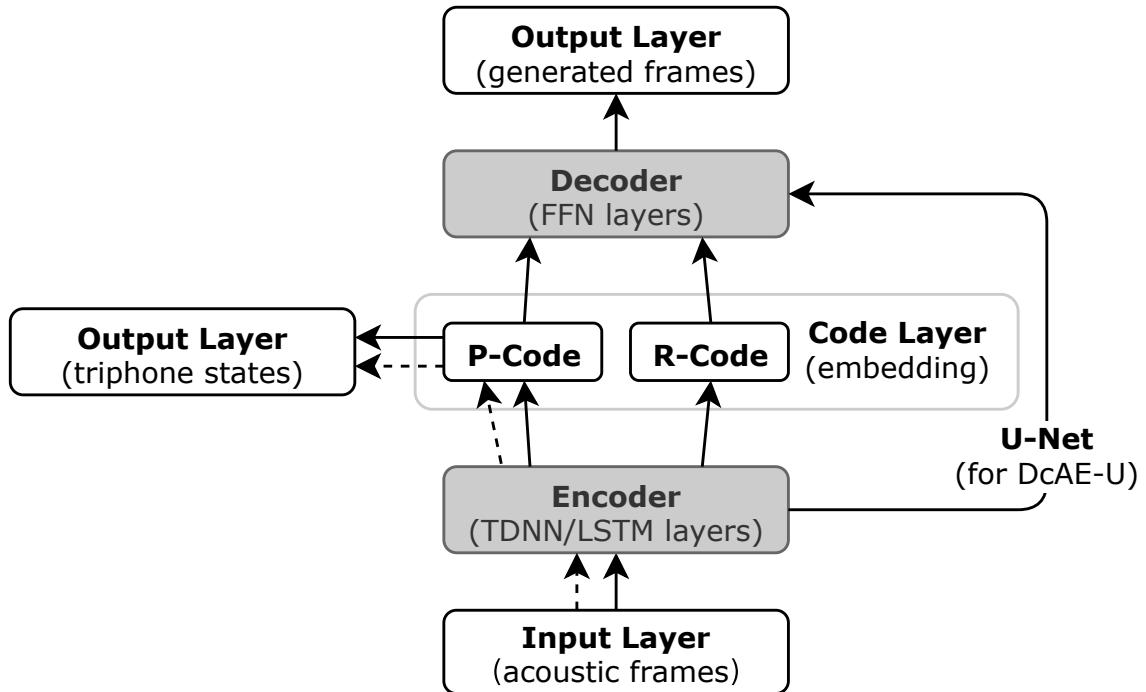
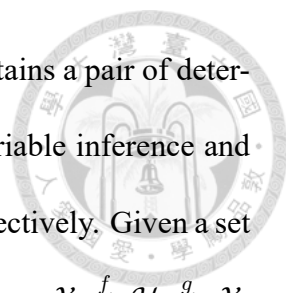


Figure 4.2: Architectures of different acoustic modeling systems. The workflow of the baseline, denoted by the dotted directional lines, is composed of the input layer, encoder, p-code, and output layer. Note that the p-code is usually the penultimate layer of the baseline, and the additional FFN layers between the p-code and the output layer, representing the triphone states, are optional. The basic version of DcAEs (DcAE-B), expressed by the solid directional lines, adds the r-code, decoder, and output layer for feature reconstruction as its key components. The u-net based DcAEs (DcAE-U) has additional connections between the encoder and decoder layers.



**The reconstruction error** Suppose our proposed model  $\mathcal{M}$  contains a pair of deterministic mappings  $f(\cdot)$  and  $g(\cdot)$ , which are responsible for latent variable inference and observation generation in the terminology of Bayesian inference, respectively. Given a set of training data  $\mathcal{X}$  ready to go through the inference-generation process  $\mathcal{X} \xrightarrow{f} \mathcal{H} \xrightarrow{g} \mathcal{X}$ , where  $\mathcal{H}$  is the internal representation residing in a latent subspace, i.e., the code layer shown in Figure 4.2, the average reconstruction error, denoted by  $\mathcal{L}_{mse}$ , based on the residual sum of squares between  $\mathbf{x} \in \mathcal{X}$  and its reconstruction  $\mathbf{x}' = g(f(\mathbf{x}))$  is given by

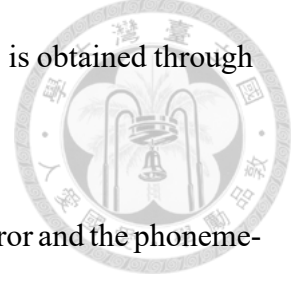
$$\mathcal{L}_{mse}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x}' - \mathbf{x}\|_2^2, \quad (4.2)$$

where  $\|\cdot\|_2^2$  is the 2-norm operator and  $|\mathcal{X}|$  is the sample or mini-batch size. Like a copy machine,  $\mathcal{H}$  is usually restricted in ways that allow it to copy only approximately, and to copy only input that resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data [45].

**The phoneme-aware cross-entropy** One of the straightforward strategies to interpret the phonetic information of an sensory input is to leverage a categorical distribution over the predefined context-dependent phonetic states. In Kaldi, the HMM state of a triphone is assigned with a pdf-id, which is used as an index for the probability distribution function (pdf). These distinct and contiguous pdf-ids are used as targets for acoustic modeling. To realize the representation, we define an objective, expressed by  $\mathcal{L}_{xent}$ , to maximize the total logarithmic posterior probability of the target pdf-ids over all training samples  $\mathcal{X}$  (or minimizing the cross-entropy between the ground truth and the predicted values):

$$\mathcal{L}_{xent}(\mathcal{X}) = - \sum_{\mathbf{x} \in \mathcal{X}} \log p(q_{\mathbf{x}}|\mathbf{x}), \quad (4.3)$$

where  $q_x$  denotes the context-dependent HMM state for  $\mathbf{x}$ , and  $p(q_x|\mathbf{x})$  is obtained through a softmax activation function.



**The final objective function** By combining the reconstruction error and the phoneme-aware cross-entropy, the objective function to be minimized becomes

$$(1 - \alpha)\mathcal{L}_{xent} + \alpha\mathcal{L}_{mse}, \quad (4.4)$$

where  $\alpha$  is an adjustable weight between  $\mathcal{L}_{xent}$  and  $\mathcal{L}_{mse}$ .

We implemented this framework in Kaldi by modifying its nnet3 neural network library. We added an additional output layer called output\_ae as the output layer of the DcAE. The target of this layer was set to the input feature. The weight  $\alpha$  was set by adjusting the parameter learning-rate-factor of each output layer.

### 4.1.3 Experiments

#### 4.1.3.1 Corpora

We evaluated our proposed framework with two speech corpora of different languages, namely Wall Street Journal (WSJ) and Mandarin Chinese Broadcast News (MATBN) [173].

**Wall Street Journal (WSJ)** For WSJ, we used both WSJ0 (LDC93S6B) and WSJ1 (LDC94S13B) as the training set, consisting of 81 hours and known as train\_si284 in most Kaldi recipes. Besides, we used dev93 and eval92 as the test sets.

**Mandarin Chinese Broadcast News (MATBN)** MATBN is a broadcast news corpus collected in Taiwan [173]. Each episode has been segmented into separate stories and

Table 4.1: Various models' specifications used in WSJ and MATBN.  $D$ ,  $T$ ,  $L$ , and  $C$  denote the dense, TDNN-based, LSTM-based, and code layers, respectively. The subscript of the layer notation means the skip connection between two layers. For example,  $T_{h1}$  and  $D_{h1}$  have a common connection  $h1$ . The layer-wise context shows the information about splicing indices of TDNN-based layers, which are usually somewhat different between the TDNN and TDNN-LSTM systems. DcAE-B stands for the basic version of DcAEs while DcAE-U means the version utilizing the u-net.

Model (for MATBN)	Architecture	Layer-wise context			
TDNN	$TTTTD$	$\{-2,-1,0,1,2\}$	$\{-1,0,1\}$	$\{-3,0,3\}$	$\{-6,-3,0\}$
TDNN-DcAE-B	$TTTTTCDDDDDD$				
TDNN-DcAE-U	$T_{h1}T_{h2}TTTCDDDDDD_{h2}D_{h1}$				
TDNN-LSTM	$TTLLTLLTLD$	$\{-2,-1,0,1,2\}$	$\{-1,0,1\}$	$\{-3,0,3\}$	$\{-3,0,3\}$
TDNN-LSTM-DcAE-B	$TTLLTLLTLCDD$				
TDNN-LSTM-DcAE-U	$T_{h1}T_{h2}TTLLTLLCLCD_{h2}D_{h1}$				
Model (for WSJ)	Architecture	Layer-wise context			
TDNN	$TTTTD$	$\{-2,-1,0,1,2\}$	$\{-1,0,1\}$	$\{-3,0,3\}$	$\{-6,-3,0\}$
TDNN-DcAE-B	$TTTTTCDDDD$				
TDNN-DcAE-U	$T_{h1}T_{h2}TTTCDD_{h2}D_{h1}$				
TDNN-LSTM	$TTLLTLLTLD$	$\{-2,-1,0,1,2\}$	$\{-1,0,1\}$	$\{-3,0,3\}$	$\{-3,0,3\}$
TDNN-LSTM-DcAE-B	$TTLLTLLTLCDDDD$				
TDNN-LSTM-DcAE-U	$T_{h1}T_{h2}TTLLTLLCLCDDDD_{h2}D_{h1}$				



manually transcribed. Each story contains the speech of one studio anchor, as well as several field reporters and interviewees. In our experiments, we used a subset of 25-hour speech data for training the model and tested on two testing sets, namely Dev and Test, each consisting of 1.4 hours of speech.

#### 4.1.3.2 Input features

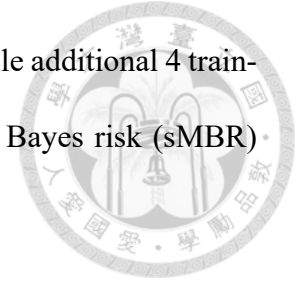
To extract acoustic features, spectral analysis was applied to a 25 ms frame of speech waveform every 10 ms. For each frame, 40 high-resolution MFCCs, derived by DCT conducted on 40 Mel-frequency bins and normalized by utterance-based mean subtraction, were used as the input to the NN-based acoustic models. Since Mandarin is a tonal language, 3 pitch-related features were concatenated to the 40-dimensional MFCCs [41] for the Mandarin ASR task. Moreover, for both tasks, we appended a 100-dimensional i-vector to each acoustic frame [24].

#### 4.1.3.3 Baseline systems

The GMM-HMM system was trained to generate frame-to-state (or pdf) alignments for subsequent neural network training. We used the fifth round triphone system (i.e., `tri5`) in MATBN and the fourth round triphone system (i.e., `tri4`) in WSJ to decode and generate the alignment for each utterance.

The TDNN-based system was built with 6 hidden layers, each containing 650 hidden nodes. Their output layer was a softmax-activated layer with 3,376 nodes for WSJ and 4,272 for MATBN, and the maximum change in the parameters per mini-batch was set to 1.5. The detailed splicing indices are given in Table 4.1. The mini-batch sizes were 256 and 128. The initial and final effective learning rates were set to 0.0015 and 0.00015,

respectively, and the total number of training epochs was set to 3, while additional 4 training epochs were used for fine tuning with the state-level minimum Bayes risk (sMBR) criterion [169].



The TDNN-LSTM-based system was constructed by 3 continuous groups of layers, where each group consisted of 2 TDNN-based layers followed by an LSTM-based layer, and a dense layer. Each layer had 520 hidden nodes. We used a delay factor of -3 and decay time of 20 for all LSTM-based layers. The mini-batch sizes were 128 and 64, and the number of training epochs was set to 6. The other hyper-parameters were set to the same values as the TDNN-based system.

#### 4.1.3.4 Proposed systems

The dimensions of the residual layer (r-code) and the decoder layers were the same as the number of nodes of the baseline model's penultimate layer (p-code). The number of decoder layers was given as  $D$  in the column named Architecture in Table 4.1. For the proposed u-net version of DcAE (DcAE-U), additional skip-connections were added symmetrically between the second layer and the penultimate layer as well as the third layer and the third to the last layer. In DcAE-U with the appending operation, the dimensions of the decoder layers, which were linked with skip-connections and appended with the corresponding encoder layers, were doubled. Take TDNN-DcAE-U for example, as shown in Table 4.1, the dimension of the last two dense layers was 1,300 ( $650 \times 2$ ) for the append version.

Table 4.2: CERs (%) with respect to two baselines and their relevant DcAE-based variants for MATBN.

Model	Dev	Test	+sMBR	
			Dev	Test
TDNN	7.88	7.64	7.54	7.45
DcAE-B	7.45	<b>7.39</b>	7.20	<b>7.07</b>
DcAE-U (append)	<b>7.43</b>	<b>7.39</b>	7.21	7.22
DcAE-U (sum)	7.51	<b>7.39</b>	<b>7.17</b>	7.12
TDNN-LSTM	8.37	8.26	8.05	7.92
DcAE-B	8.05	7.92	7.64	7.68
DcAE-U (append)	<b>7.78</b>	7.91	<b>7.28</b>	<b>7.37</b>
DcAE-U (sum)	8.00	<b>7.86</b>	7.72	7.62

#### 4.1.3.5 Results

Table 4.1 presents various models' architectures and layer-wise contexts used in this study, where DcAE-B represents DcAE without highway connections, and DcAE-U represents DcAE with highway connections. For DcAE-U, both implementation methods (appending and summing) have the same architecture. The layer-wise contexts for the DcAE models were the same as their baseline TDNN or TDNN-LSTM model.

Tables 4.2 and 4.3 show the error rates achieved with TDNN and TDNN-LSTM as well as the proposed DcAE-based systems for MATBN and WSJ. For MATBN,  $\alpha$  for TDNN was set to  $5 \times 10^{-10}$  and  $\beta$  was set to  $10^{-1}$ ;  $\alpha$  for TDNN-LSTM was set to  $5 \times 10^{-9}$  and  $\beta$  was set to  $8 \times 10^{-1}$ . Note that in actual implementation,  $\mathcal{L}_{mse}$  in Eq. (4.4) is not an average as shown in Eq. (4.2) but is orders of magnitude larger than  $\mathcal{L}_{xent}$ , so it is not surprising that  $\alpha$  is a very small value. Compared to the TDNN baseline system, DcAE-B before and after fine-tuning with sMBR achieved relative error rate reductions of 3% and 5%, respectively. With the TDNN-LSTM-based architecture, DcAE-U achieved a relative error reduction of 4% over DcAE-B both before and after fine-tuning with sMBR.



Table 4.3: WERs (%) with respect to two baselines and their relevant DcAE-based variants for WSJ.

Model	dev93	eval92	+sMBR	
			dev93	eval92
TDNN	6.62	3.92	5.93	3.51
DcAE-B	<b>6.23</b>	<b>3.58</b>	6.07	<b>3.23</b>
DcAE-U (append)	6.52	3.79	5.94	3.46
DcAE-U (sum)	6.36	3.67	<b>5.88</b>	3.44
TDNN-LSTM	6.29	4.00	6.39	3.99
DcAE-B	6.45	3.90	6.23	<b>3.62</b>
DcAE-U (append)	6.56	4.09	6.25	3.92
DcAE-U (sum)	<b>6.21</b>	<b>3.77</b>	<b>6.01</b>	3.7

For WSJ,  $\alpha$  for TDNN was set to  $10^{-10}$  and  $\beta$  was set to  $5 \times 10^{-1}$ ;  $\alpha$  for TDNN-LSTM was set to  $10^{-10}$  and  $\beta$  was set to  $3 \times 10^{-1}$ . The DcAE-based systems generally outperformed the baseline models with relative error rate reductions of 5% to 10%.

From the results, we can conclude that DcAE-B in general outperforms its baseline model with relative error rate reductions of 5% to 10% for both corpora. The results show that the reconstruction error  $\mathcal{L}_{mse}$  can indeed help extract the most salient information, making it easier to separate out the phonetic information. The results also indicate that, despite having a great success in the field of image recognition, adding skip-connections does not necessarily benefit ASR. A possible reason is due to the different types of information to be learned by different systems. For image recognition tasks, the knowledge learned by the model is space-wise information extracted from an image. However, for ASR tasks, the goal is to learn time-wise information, which is sequential.



#### 4.1.4 Conclusions and Future Work

This study presented two kinds of discriminative autoencoders (DcAEs), namely DcAE-B and DcAE-U, which were implemented on two existing acoustic models, namely TDNN and TDNN-LSTM, on the basis of the nnet3 setup of Kaldi. The results showed that DcAE-based systems achieved error rate reductions over their baseline systems for both Mandarin and English ASR tasks. We also provided the DcAE recipe for WSJ, allowing rapid reproduction of our results. We will implement DcAEs on other outstanding models such as TDNN-F [123] and CNN-DNN [139] as well as other larger data sets. In addition, inspired by the concept of sequence to sequence learning [157] and deconvolutional networks [184], the relationship of adjacent phonetic embeddings (p-code) will be taken into account for the design of the decoder in DcAEs. We will also implement DcAE-based models on the Kaldi chain setup [125].

## 4.2 Roubst DcAE

### 4.2.1 Introduction

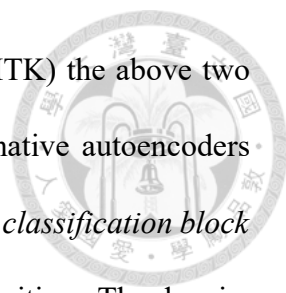
With the advent and breakthroughs of deep learning, current automatic speech recognition (ASR) systems perform significantly well under relatively clean conditions. However, due to the mismatch between train and test conditions, noise is a harmful factor that can greatly degrade performance in practical real-world application scenarios. Many techniques have been proposed to reduce the ill influence of noise on acoustic modeling, which arises from interfering background events (such as babble or street noise), reverberation generated from non-ideal room acoustics, or channel distortion caused by different

microphone characteristics [101]. These techniques mainly fall into two categories: multi-condition training and speech enhancement-based pre-processing [33, 89, 103, 176, 183].

In the case of multi-condition training, it has been shown in the literature that acoustic models based on neural networks can be adapted well if degraded data is used to train these models. This strategy is widely adopted to achieve model robustness against additive noise, channel mismatch [29, 145, 183], and reverberation [17, 60]. As stated in [182], adding extra noisy speech to the training data is a form of regularization that can provide better generalization capabilities. In addition, multi-condition training has been shown to provide better performance than the enhancement-based front-end under the condition of unseen degradation [29, 145]. However, the performance under unseen conditions still lags behind the performance obtained in seen conditions [29, 127].

On the other hand, inheriting the achievements in speech enhancement in [174], Narayanan and Wang first proposed the joint training of time-frequency (T-F) mask-based front-end and DNN-based acoustic model back-end. The front-end was used to estimate clean features from noisy features, and the back-end was used to classify phoneme labels. They were connected as a combined neural network, and the model parameters were simultaneously optimized with back-propagation [105]. Related extensions can be seen in [36, 104, 106]. In addition, Soni and Panda recently proposed another version of the framework in [154], which could be used without parallel clean-noisy speech training data and without pre-trained feature enhancement modules.

The above integrated topology of speech enhancement and phoneme classification can be simply expressed by *input features / enhancement block / classification block / output targets*. A detailed description can be seen in Figure 2 in [34], which shows two



network structures for joint training (JT) and multi-task learning (MTK) the above two blocks. In this study, we extend our previous work, i.e., discriminative autoencoders [62, 180], and propose an alternative neural architecture, in which the *classification block* is wrapped in the *enhancement block* to achieve robust speech recognition. The denoising autoencoders (DAEs) have been a popular DNN structure that can learn the mapping between noisy and clean speech in the frequency or time domain [116, 130, 160] from single-channel or multi-channel observations [2, 86]. Our work in this study has at least two major contributions:

1. We proposed two kinds of robust DcAE. What they have in common is that the enhancement block not only removes unwanted noise but also restores the original signal. Their major difference lies in the way that the enhancement block reconstructs noisy and clean speech: in parallel or hierarchically. Therefore, robust DcAE not only has merits to extract cleaner and more discriminative embeddings for phone-state classification but also has the advantage of improving the generalization ability<sup>4</sup>.
2. To the best of our knowledge, this is the first work to introduce the lattice-free MMI criterion into *joint training* for robust acoustic modeling [125]. Better ASR performance can be expected.

## 4.2.2 Robust Discriminative Autoencoders

An autoencoder is a learning model designed to convert an input into a reconstructed output with minimal distortion. The embedded internal representation, namely the code

---

<sup>4</sup>In DcAE, the decoder layers themselves are a kind of regularizer, as mentioned in [62]

layer, contains the most critical information that can be used to minimize the reconstruction error. In our previous work, to obtain more accurate acoustic scores for ASR, we divided the code layer into two sub-representations, namely the phoneme-aware layer (P-Code) and the residual layer (R-Code), to form the basic version of discriminative autoencoder (hereafter referred to as DcAE-B) [62]. P-Code contained phonetic information and was connected to the original output layer of the acoustic model of an ASR system. R-Code carried information that was unrelated to phonetic information, which could include environmental noise, speaker identity, and any other information deemed useless for the ASR task. P-Code and R-Code were then concatenated as the input of the decoder to reconstruct the original acoustic frame. DcAE-B was implemented with the TDNN and TDNN-LSTM acoustic models in the Kaldi nnet3 setup. In the following subsections, we will introduce two versions of DcAE that can make the acoustic model more robust in noisy environments when the training data containing clean/noisy speech pairs are suffi-

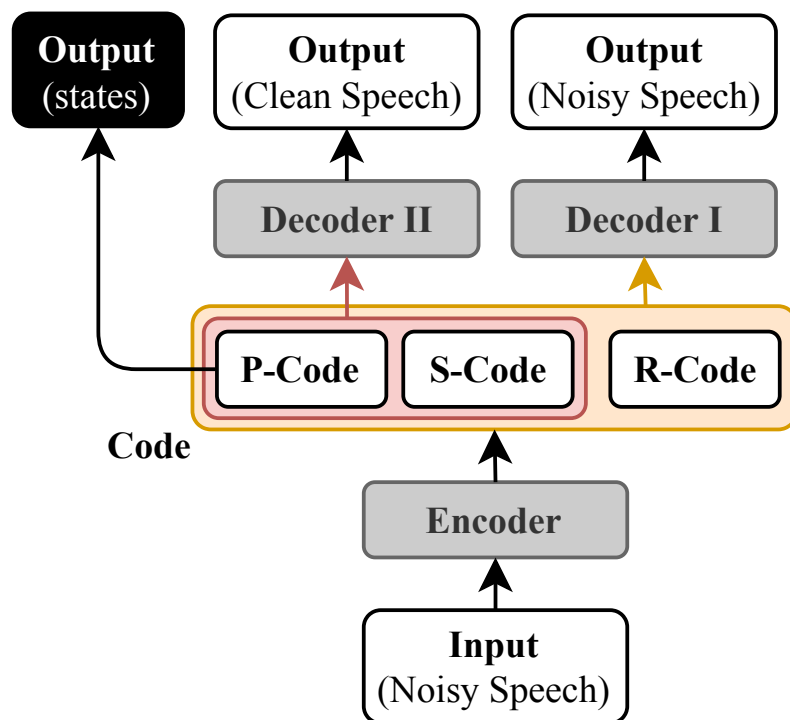


Figure 4.3: Structure of the parallel robust DcAE (p-DcAE), where P-Code, S-Code, and R-Code denote phoneme-aware code, speaker-aware code, and residual code, respectively.

cient.



### 4.2.3 Parallel Robust DcAE (p-DcAE)

The first version of robust DcAE is the parallel robust DcAE (hereafter referred to as p-DcAE). As shown in Figure 4.3, the architecture of p-DcAE can be simply described as *Input / Encoder / Code / Decoder  $\times 2$  / Output  $\times 3$* . A noisy acoustic frame and its corresponding speaker embedding (e.g., the i-vector) are fed into the encoder, which can be implemented with various networks, such as TDNN-F [123], TDNN-LSTM, and CNN-TDNN. In the code layer (Code), the code nodes are divided into three parts. The phoneme-aware code (P-Code) that is designed to contain as much phonetic information as possible is directly connected to the phone-state output layer. The output scores are associated with the reference phone-state labels using the cross entropy and lattice-free MMI (LF-MMI) in the objective function. S-Code contained the speaker information while R-Code carried other remaining information, especially unwanted noise. Decoder II is used to reconstruct the corresponding clean acoustic frame of the input noisy acoustic frame according to the concatenated P-Code and S-Code. Decoder I is used to reconstruct the noisy acoustic frame according to the concatenated P-Code, S-Code and R-Code. Both decoders are implemented with full-connected networks.

It is noteworthy that S-Code is not formulated by training with actual speaker labels. S-Code means the speaker code, since ideal clean speech can be regarded as a signal compound lying only in phonetic and speaker dimensions which are orthogonal to each other.



#### 4.2.3.1 Hierarchical robust DcAE (h-DcAE)

The second version of robust DcAE is the hierarchical robust DcAE (hereafter referred to as h-DcAE). As shown in Figure 4.4, the architecture of h-DcAE can be simply described as *Input / Encoder / Code / Encoder / Code / Decoder / Output*, where the clean acoustic frame is hierarchically distilled through two-leveled code layers from the noisy acoustic frame. Actually, there are two autoencoders lurking in h-DcAE. The first is *Input (Noisy Speech) / Encoder I / Code I / Decoder I / Output (Noisy Speech)*. By restoring the original noisy speech, it assure the code layer (Code I) contains salient components of speech. The second is *C-Code / Encoder II / Code II / Decoder II / Output (Clean Speech)*,

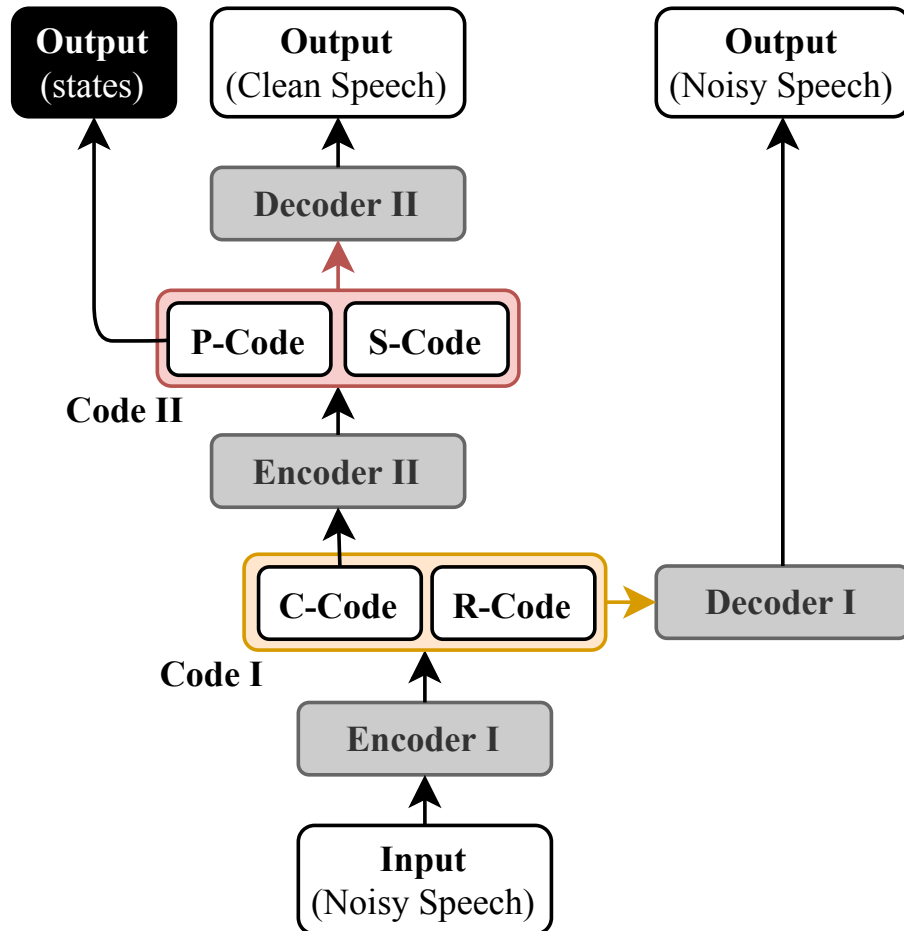


Figure 4.4: Structure of the hierarchical robust DcAE (h-DcAE), where C-Code represents the clean speech information.

where C-Code, a link-up between the two autoencoders, is defined as the embedding of a clean acoustic frame. Similar to p-DcAE, both decoders in h-DcAE are implemented with full-connected networks.



The main differences between p-DcAE and h-DcAE are twofold. In p-DcAE, all code layers are respectively generated by the same encoder for a one-shot operation, while in h-DcAE, each distinct encoder has its specific functionality. Thus, the model capacity of h-DcAE might be larger. Secondly, the treatment of clean speech in h-DcAE while training is more complicated.

#### 4.2.3.2 Objective functions

The objective function used in the proposed framework involves reconstruction and restoration errors and phoneme-aware criteria, including the cross-entropy (XENT) and maximum mutual information (MMI). The former ones are to be minimized while the later are to be maximized in the training phase.

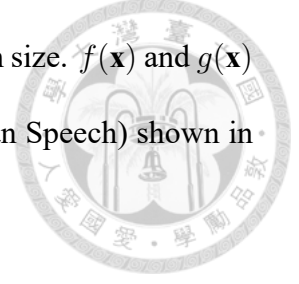
**Reconstruction and restoration errors** Suppose our proposed model  $\mathcal{M}$  contains a deterministic mapping  $f(\cdot)$  and  $g(\cdot)$ , which are responsible for observation reconstruction and restoration, respectively. Given a set of noisy data  $\mathcal{X}$  and its corresponding clean data  $\mathcal{Y}$  ready to go through the inference-generation processes  $\mathcal{X} \xrightarrow{f} \mathcal{X}$  and  $\mathcal{X} \xrightarrow{g} \mathcal{Y}$ , the average reconstruction and restoration errors, denoted by  $\mathcal{L}_{rc}$  and  $\mathcal{L}_{rs}$ , are respectively given by

$$\mathcal{L}_{rc}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \|f(\mathbf{x}) - \mathbf{x}\|_2^2, \quad (4.5)$$

$$\mathcal{L}_{rs}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}} \|g(\mathbf{x}) - \mathbf{y}\|_2^2, \quad (4.6)$$



where  $\|\cdot\|_2^2$  is the 2-norm operator and  $|\mathcal{X}|$  is the sample or mini-batch size.  $f(\mathbf{x})$  and  $g(\mathbf{x})$  correspond to the blocks of Output (Noisy Speech) and Output (Clean Speech) shown in Figures 4.3 and 4.4, respectively.



**Phoneme-aware criteria** There are two kinds of phoneme-aware criteria related to the block of Output (states) in Figures 4.3 and 4.4 in our acoustic models. One, denoting by  $\mathcal{F}_{xent}$ , is the expected cross-entropy between the distribution represented by the reference labels and the predicted distribution. The other, denoting by  $\mathcal{F}_{mmi}$ , is the MMI criterion between the distributions of the observation and word sequences. For an overall description of the LF-MMI training, also named as the “chain” modeling in the Kaldi toolkit<sup>5</sup>, the reader is directed to [125, 169].

By combining the criterion based on noisy feature reconstruction, clean feature restoration errors, the phoneme-aware cross-entropy, and MMI, the objective function to be minimized becomes

$$-\mathcal{F}_{mmi} - 5 \times \mathcal{F}_{xent} + \alpha \mathcal{L}_{rc} + \beta \mathcal{L}_{rs}, \quad (4.7)$$

where the coefficient “5” of  $\mathcal{F}_{xent}$  is fixed according most of recipes for “chain” modeling, and  $\alpha$  and  $\beta$  are adjustable weights to increase or decrease regularization strengths of  $\mathcal{L}_{rc}$  and  $\mathcal{L}_{rs}$ , respectively. Actually, for the NN-based acoustic model as a whole,  $\mathcal{F}_{xent}$ ,  $\mathcal{L}_{rc}$ , and  $\mathcal{L}_{rs}$  can be taken as three regularizers.

<sup>5</sup><https://kaldi-asr.org/doc/chain.html>

## 4.2.4 Experiments



### 4.2.4.1 The Aurora-4 dataset

We evaluated our proposed framework with the Aurora-4 database. Aurora-4 is a medium vocabulary database used to evaluate the noise robust continuous speech recognition task. Based on the Wall Street Journal (WSJ) corpus [117], it contains microphone recordings with noise variation. There are two training sets. The clean training set consists of 7,138 speech utterances recorded by the primary Sennheiser microphone. The other one is the time-synchronized multi-conditioned training set, in which half of the utterances were recorded by the primary Sennheiser microphone while the other half were recorded by one of the secondary microphones. Both halves contain clean speech (893 utterances) and speech corrupted by one of the six types of noise (i.e., street, train station, car, babble, restaurant, and airport) under SNR conditions of 10-20 dB (2,676 utterances in total). There are two test sets; one was recorded by the primary microphone and the other was recorded by one of the secondary microphones. Each test set consists of 330 utterances from 8 speakers. Each test set was then corrupted by the same six types of noise used in the training set under SNR conditions of 5-15 dB, resulting in a total of 14 test sets. These 14 test sets were grouped into 4 subsets: clean (Set 1, denoted by A), noisy (Set 2 to Set 7, denoted by B), clean with channel distortion (Set 8, denoted by C), and noisy with channel distortion (Set 9 to Set 14, denoted by D). Moreover, 100 utterances were selected from the development set of Aurora-4 as the validation set for tuning the parameters associated with the proposed models. These utterances were recorded and corrupted by the same conditions as the test set, so a total of 1,400 validation utterances were obtained.

#### 4.2.4.2 Baseline systems

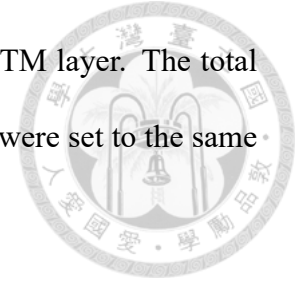


Our ASR experiments were conducted using the Kaldi toolkit [124]. The GMM-HMM system was trained to generate the frame-to-state (or pdf) alignments for subsequent neural network training. The GMM system was achieved by a speaker adaptive training (SAT) system that splices 7 frames (3 on each side of the current frame) and uses linear discriminant analysis (LDA) to project down to 40 dimensions, together with maximum likelihood linear transform (MLLT). We evaluated 3 neural network based acoustic models, namely TDNN-F [123], TDNN-LSTM, and CNN-TDNN.

**TDNN-F** Time Delay Neural Network (TDNN), also known as one-dimensional Convolutional Neural Network (1-d CNN), has been shown to effectively learn the temporal dynamics of signals from short-term feature representations. The factorized form of TDNN (TDNN-F) is structurally the same as the TDNN where the layers are compressed via singular value decomposition (SVD), but is trained by random initialization with semi-orthogonal constraint in one of the two factors of each matrix. The TDNN-F system was built with 13 hidden layers, each containing 1024-dimensional hidden nodes and 128-dimensional linear bottleneck nodes. In addition, the parameters of the final layer are also factorized using a 192-node linear bottleneck layer. The mini-batch sizes were 128 and 64. The initial and final effective learning rates were set to 0.0005 and 0.00005, respectively, and the total number of training epochs was set to 10.

**TDNN-LSTM** The TDNN-LSTM structure is built by stacking the long short-term memory (LSTM) layers [58] over TDNN. The TDNN-LSTM system was constructed by 4 TDNN layers and 1 LSTM layer. Each TDNN layer consisted of 448 hidden nodes. The cell dimension was set to 384 and the bottleneck dimension was set to 256 for the LSTM

layer. We used a delay factor of -3 and decay time of 20 for the LSTM layer. The total number of training epochs was set to 6. The other hyper-parameters were set to the same values as the TDNN-F system.



**CNN-TDNNF** The CNN-TDNNF system contained 15 layers. The first six layers are convolutional and the remaining nine layers are TDNN-F. The time offsets and height offsets were set to “-1, 0, 1” for all convolutional layers. The numbers of filters for the six convolutional layers were 48, 48, 64, 64, 64, and 128, respectively. The parameters of the TDNN-F layers were set to the same value as the TDNN-F system. The total number of training epochs was set to 8. The other hyper-parameters were set to the same values as the TDNN-F system.

#### 4.2.4.3 Primitive results

As shown in Table 4.4, results on the task of WSJ proves that our proposed models, DcAE (B) and DcAE (U), outperform the baseline with 10.41% and 11.09% relative EER reductions, respectively, in dev93. In eval92, DcAE (B) and DcAE (U), outperform the baseline with 0.09% and 10.28% relative EER reductions, respectively.

In the case of robust ASR, the models were conducted on the task of Aurora-4. As shown in 4.5, our proposed models, p-DcAE and h-DcAE, outperform the baseline with 4.0% and 5.9% relative EER reductions, when using CNN-TDNNF as the encoder, respectively.



Table 4.4: WERs (%) on the task of WSJ. The encoder type of DcAE is TDNN-F.

Method	dev93	rel. %	eval92	rel. %
Baseline	4.42		2.53	
DcAE (B)	3.96	10.41	2.3	9.09
DcAE (U)	3.93	11.09	2.27	10.28

Table 4.5: WERs (%) on the task of Aurora-4.

Encoder	Method	A	B	C	D	Average	rel. %
TDNN-F	Baseline	1.91	3.69	3.42	10.31	6.38	
	p-DcAE	1.70	3.50	3.16	10.22	6.23	2.4
	h-DcAE	1.61	3.50	3.14	10.19	6.21	2.7
CNN-TDNNF	Baseline	1.72	3.19	3.40	9.48	5.80	
	p-DcAE	1.57	3.19	3.03	9.04	5.57	4.0
	h-DcAE	1.63	3.21	2.67	8.82	5.46	5.9
TDNN-LSTM	Baseline	2.35	5.07	4.22	13.57	8.46	
	p-DcAE	2.02	4.79	4.30	13.17	8.15	3.7
	h-DcAE	1.98	4.77	4.15	13.16	8.12	4.0





## Chapter 5

## Conclusions

This doctoral dissertation focuses on advanced neural structures for phonotactic language recognition, automatic speaker verification, and automatic speech recognition tasks.

The first is a subspace-based neural network (SNN), where each utterance is represented as a linear subspace, and a reasonable manifold-based kernel computation is applied to a orthogonality-constrained layer. Three types of methods for subspace construction were proposed; thereinto, orthogonal dictionary learning and dynamic linear modeling are original with the author. For subspace learning, although the Projection kernel has been used in the subspace-based SVM, it is the author who first introduced the kind of kernel into deep learning. SNN is implemented for phonotactic language recognition. In the future, we will explore to open the black box of SNN, by explaining the physical meaning of each orthogonal weight map and figuring out its role in phonotactics. Moreover, we will search for other opportunities that SNN can be applied to other research fields or directly used with more low-level information, such as speech waveforms.

The second is a discriminative autoencoder (DcAE), where the encoder converts the input utterance into an identity embedding, called an identity code that represents the

speaker (for speaker recognition) or phone-state (for speech recognition)), and a residual code. The decoder reconstructs the input acoustic features based on the identity code and the residual code. It is hoped that through training, the speaker or phonetic information is mainly distributed in the identity code. The kind of neural structure, combining the concepts of generative and discriminative models, is also originated by the author. In the future, we will implement the DcAE on the end-to-end platform for ASR, and the state-of-the-art speaker model, such as ECAPA-TDNN, will be taken as the encoder part of the DcAE for speaker verification or its robust applications to achieve higher performance.

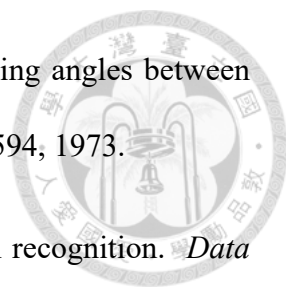
Some experiments conducted on the NIST-LRE 2003, NIST-LRE 2007, NIST-LRE 2009, VoxCeleb-1, NIST-SRE 2016, WSJ, and Aurora-4 datasets show the superiority of our methods to the baselines.







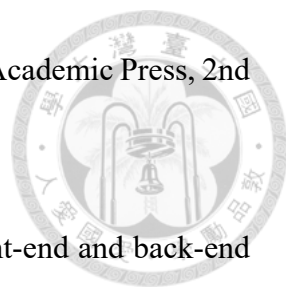
## References

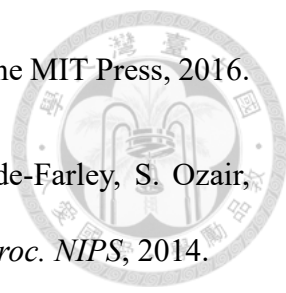
- [1] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu. Language identification: A tutorial. *IEEE Circuits Syst. Mag.*, 11(2):82–108, 2011.
- [2] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani. Exploring multi-channel features for denoising-autoencoder-based speech enhancement. In *Proc. ICASSP*, 2015.
- [3] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo. Direct acoustics-to-word models for english conversational speech recognition. In *Proc. Interspeech*, 2017.
- [4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.
- [5] C. Bao, J.-F. Cai, and H. Ji. Fast sparsity-based orthogonal dictionary learning for image restoration. In *Proc. ICCV*, 2013.
- [6] Y. Bengio. Learning deep architectures for AI. *FNT in Machine Learning*, 2(1):1–127, 2009.
- [7] K. Beulen, L. Welling, and H. Ney. Experiments with linear feature extraction in speech recognition. In *Proc. Eurospeech*, 1995.

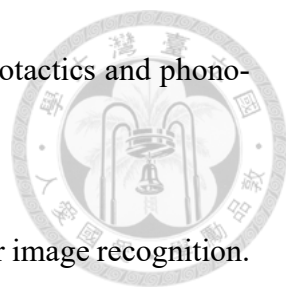
- 
- [8] A. Bjorck and G. H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.
- [9] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [10] W. 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*, 2018.
- [11] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 25(6):1291–1303, 2017.
- [12] W. Campbell, J. Campbell, D. Reynolds, E. Singer, and P. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Comput. Speech and Language*, 20:210–229, 2006.
- [13] Y. Chikuse. *Statistics on Special Manifolds*. Springer Science & Business Media, 2003.
- [14] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *Proc. NIPS*, 2014.
- [15] M. Chu. On the statistical meaning of truncated singular value decomposition. Technical report, Dept. Mathematics, North Carolina State University, 2001.
- [16] M. Chu and N. Trendafilov. The orthogonally constrained regression revisited. *J. Computational and Graphical Stat.*, 10(4):746–771, 2001.
- [17] L. Couvreur, C. Couvreur, and C. Ris. A corpus-based approach for robust ASR in reverberant environments. In *Proc. ICSLP*, 2000.

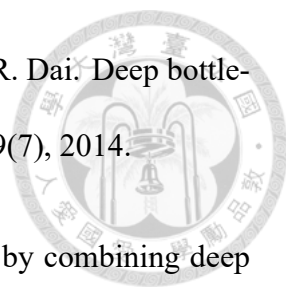
- 
- [18] S. Cumani and P. Laface. Training pairwise support vector machines with large scale datasets. In *Proc. ICASSP*, 2014.
- [19] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio, Speech, Lang. Process.*, 20(1):30–42, 2012.
- [20] A. Das, J. Li, R. Zhao, and Y. Gong. Advancing Connectionist Temporal Classification with Attention. In *Proc. ICASSP*, 2018.
- [21] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Audio, Speech, Lang. Process.*, 28(4):357–366, 1980.
- [22] K. De Cock and B. De Moor. Subspace angles between ARMA models. *Syst. & Control Lett.*, 46(4):265–270, 2002.
- [23] R. de Córdoba, L. F. D’Haro, F. Fernández-Martínez, J. M. Guarasa, and J. Ferreiros. Language identification based on n-gram frequency ranking. In *Proc. Interspeech*, 2007.
- [24] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Trans. Audio, Speech, Lang. Process.*, 19(4):788–798, 2011.
- [25] L. Deng and X. Li. Machine learning paradigms for speech recognition : An overview. *IEEE Trans. Audio, Speech, Lang. Process.*, 21(5):1060–1089, 2013.
- [26] L. D’Haro, R. de Córdoba, C. S. Palacios, and J. D. Echeverry. Extended phone

- 
- log-likelihood ratio features and acoustic-based i-vectors for language recognition. In *Proc. ICASSP*, 2014.
- [27] L. D'Haro, O. Glembek, O. Plchot, P. Matejka, M. Soufifar, R. Cordoba, and J. Cernocky. Phonotactic language recognition using i-vectors and phoneme posterigram counts. In *Proc. Interspeech*, 2012.
- [28] M. Diez, A. Varona, M. Penagarikano, L. J. Rodriguez-Fuentes, and G. Bordel. On the use of phone log-likelihood ratios as features in spoken language recognition. In *Proc. IEEE SLT*, 2012.
- [29] J. Du, Q. Wang, T. Gao, Y. Xu, L. Dai, and C.-h. Lee. Robust speech recognition with speech enhanced deep neural networks. In *Proc. Interspeech*, 2014.
- [30] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.
- [31] A. Edelman, T. Arias, and S. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Applicat.*, 20(2):303–353, 1998.
- [32] L. El Shafey, C. McCool, R. Wallace, and S. Marcel. A Scalable Formulation of Probabilistic Linear Discriminant Analysis: Applied to Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1788–1794, 2013.
- [33] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Trans. Audio, Speech, Signal Process.*, 33(2):443–445, 1985.
- [34] M. Fujimoto and H. Kawai. One-pass single-channel noisy speech recognition using a combination of noisy and enhanced features. In *Proc. Interspeech*, 2019.

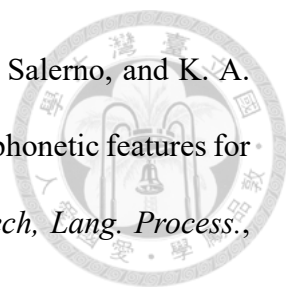
- 
- [35] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.
- [36] T. Gao, J. Du, L. R. Dai, and C. H. Lee. Joint training of front-end and back-end deep neural networks for robust speech recognition. In *Proc. ICASSP*, 2015.
- [37] D. Garcia-Romero and C. Y. Espy-Wilson. Analysis of i-vector length normalization in speaker recognition systems. *Proc. Interspeech*, 2011.
- [38] J.-L. Gauvain, A. Messaoudi, and H. Schwenk. Language recognition using phone lattices. In *Proc. Interspeech*, 2004.
- [39] J. Gehring, Y. Miao, F. Metze, and A. Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *Proc. ICASSP*, 2013.
- [40] Z. Ghahramani and G. Hinton. Parameter estimation for linear dynamical systems. Technical report, Dept. Computer Science, University of Toronto, 1996.
- [41] P. Ghahremani, B. Babaali, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur. A Pitch Extraction Algorithm Tuned for Automatic Speech Recognition. In *Proc. ICASSP*, 2014.
- [42] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, 2010.
- [43] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU Press, 3rd edition, 2012.
- [44] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzales-Rodriguez, and P. J. Moreno. Automatic language identification using long short-term memory recurrent neural networks. In *Proc. Interspeech*, 2014.

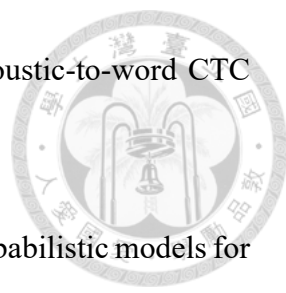
- 
- [45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.
- [46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014.
- [47] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks. In *Proc. ICML*, 2006.
- [48] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur. End-to-end speech recognition using lattice-free MMI. In *Proc. Interspeech*, 2018.
- [49] J. Hamm and D. Lee. Grassmann discriminant analysis: A unifying view on subspace-based learning. In *Proc. ICML*, 2008.
- [50] K. J. Han and J. W. Pelecanos. Frame-based phonotactic language identification. In *Proc. IEEE SLT*, 2012.
- [51] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2014.
- [52] M. Harandi, C. Sanderson, S. Shirazi, and B. Lovell. Kernel analysis on Grassmann manifolds for action recognition. *Pattern Recognition Lett.*, 34(15):1906–1915, 2013.
- [53] V. Hautamaki, S. M. Siniscalchi, H. Behravan, V. M. Salerno, and I. Kukanov. Boosting universal speech attributes classification with deep neural network for foreign accent characterization. In *Proc. Interspeech*, 2015.

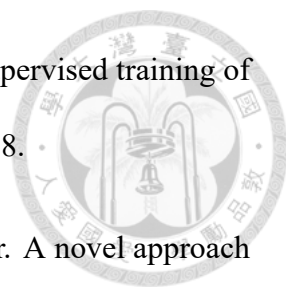
- 
- [54] B. Hayes and C. Wilson. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440, 2008.
- [55] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.
- [56] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer. End-to-end text-dependent speaker verification. In *Proc. ICASSP*, 2016.
- [57] G. Hinton, L. Deng, D. Yu, G. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
- [58] S. Hochreiter and S. Jurgens. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.
- [59] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice-Hall, 2nd edition, 1971.
- [60] R. Hsiao, J. Ma, W. Hartmann, M. Karafiat, F. Grezl, L. Burget, I. Szoke, J. Cernocky, S. Watanabe, Z. Chen, S. Mallidi, H. Hermansky, S. Tsakalidis, and R. Schwartz. Robust speech recognition in unknown reverberant and noisy conditions. In *Proc. IEEE ASRU*, 2015.
- [61] K. Huang and S. Aviyente. Sparse representation for signal classification. In *Proc. NIPS*, 2006.
- [62] P.-T. Huang, H.-S. Lee, S.-S. Wang, K.-Y. Chen, Y. Tsao, and H.-M. Wang. Exploring the encoder layers of discriminative autoencoders for LVCSR. In *Proc. Interspeech*, 2019.

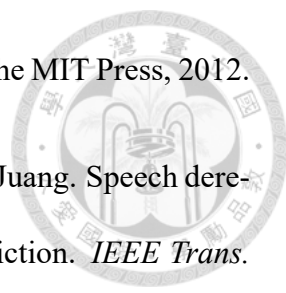
- 
- [63] B. Jiang, Y. Song, S. Wei, J. H. Liu, I. V. McLoughlin, and L. R. Dai. Deep bottle-neck features for spoken language identification. *PLoS ONE*, 9(7), 2014.
- [64] Y. Jiao, M. Tu, V. Berisha, and J. Liss. Accent identification by combining deep neural networks and recurrent neural networks trained on long and short term features. In *Proc. Interspeech*, 2016.
- [65] T. Katayama. *Subspace Methods for System Identification*. Springer, 1996.
- [66] P. Kenny. Bayesian speaker verification with heavy-tailed priors. In *Proc. Odyssey*, 2010.
- [67] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam. Deep neural networks for extracting Baum-Welch statistics for speaker recognition. In *Proc. Odyssey*, 2014.
- [68] S. Kesiraju, L. Burget, I. Szoke, and J. Černocký. Learning document representations using subspace multinomial model. In *Proc. Interspeech*, 2016.
- [69] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.
- [70] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014.
- [71] T. Ko, V. Peddinti, D. Povey, S. Khudanpur, H. Noah, and H. Kong. Audio augmentation for speech recognition. In *Proc. Interspeech*, 2015.
- [72] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *Proc. ICASSP*, 2017.




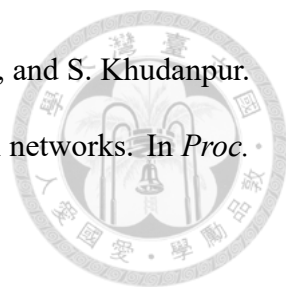
- 
- [73] I. Kukanov, T. Trong, V. M. Hautamaki, S. M. Siniscalchi, V. Salerno, and K. A. Lee. Maximal figure-of-merit framework to detect multi-label phonetic features for spoken language recognition. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 28:682–695, 2020.
- [74] H.-S. Lee and B. Chen. Linear discriminant feature extraction using weighted classification confusion information. In *Proc. Interspeech*, 2008.
- [75] H.-S. Lee, Y.-C. Shih, H.-M. Wang, and S.-K. Jeng. Subspace-based phonotactic language recognition using multivariate dynamic linear models. In *Proc. ICASSP*, 2013.
- [76] H.-S. Lee, Y. Tsao, H.-M. Wang, and S.-K. Jeng. Clustering-based i-vector formulation for speaker recognition. In *Proc. Interspeech*, 2014.
- [77] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Proc. ICASSP*, 2014.
- [78] M. Lezcano-Casado and D. Martínez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *Proc. ICML*, 2019.
- [79] H. Li, B. Ma, and C.-H. Lee. A vector space modeling approach to spoken language identification. *IEEE Trans. Audio, Speech, Lang. Process.*, 15(1):271–284, 2007.
- [80] H. Li, B. Ma, and K. A. Lee. Spoken language recognition: From fundamentals to practice. *Proc. IEEE*, 101(5):1136–1159, 2013.
- [81] J. Li, M. L. Seltzer, X. Wang, R. Zhao, and Y. Gong. Large-scale domain adaptation via teacher-student learning. In *Proc. Interspeech*, 2017.


- 
- [82] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong. Advancing acoustic-to-word CTC Model. In *Proc. ICASSP*, 2018.
- [83] P. Li, Y. Fu, U. Mohammed, J. H. Elder, and S. J. D. Prince. Probabilistic models for inference about identity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(1):144–157, 2012.
- [84] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 2nd edition, 1999.
- [85] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno. Automatic language identification using deep neural networks. In *Proc. ICASSP*, 2014.
- [86] X. Lu, Y. Tsao, S. Matsuda, and C. Hori. Speech enhancement based on deep denoising autoencoder. In *Proc. Interspeech*, 2013.
- [87] T. Ma, S. Srinivasan, G. Lazarou, and J. Picone. Continuous speech recognition using linear dynamic models. *Int. J. Speech Tech.*, 17(1):11–16, 2014.
- [88] A. Maas and A. Ng. A probabilistic model for semantic word vectors. In *Proc. NIPS*, 2010.
- [89] A. L. Maas, Q. V. Le, T. M. O. Neil, O. Vinyals, P. Nguyen, and A. Y. Ng. Recurrent neural networks for noise reduction in robust ASR. In *Proc. Interspeech*, 2012.
- [90] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proc. ICML*, 2009.
- [91] J. Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63(4):561–580, 1975.


- 
- [92] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur. Semi-supervised training of acoustic models using lattice-free MMI. In *Proc. ICASSP*, 2018.
- [93] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks. In *Proc. ICASSP*, 2015.
- [94] D. Martínez, O. Plchot, L. Burget, O. Glembek, and P. Matejka. Language recognition in iVectors space. In *Proc. Interspeech*, 2011.
- [95] P. Matejka, L. Zhang, T. Ng, S. H. Mallidi, O. Glembek, J. Ma, and B. Zhang. Neural network bottleneck features for language identification. In *Proc. Odyssey*, 2014.
- [96] A. McCree, G. Sell, and D. Garcia-Romero. Augmented data training of joint acoustic/phonotactic DNN i-vectors for NIST LRE15. In *Proc. Odyssey*, 2016.
- [97] M. McLaren, L. Ferrer, and A. Lawson. Exploring the role of phonetic bottleneck features for speaker and language recognition. In *Proc. ICASSP*, 2016.
- [98] F. Mezzadri. How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*, 54(5):592–604, 2006.
- [99] T. Mikolov, O. Plchot, O. Glembek, L. Burget, and J. Cernocky. PCA-based feature extraction for phonotactic language recognition. In *Proc. Odyssey*, 2010.
- [100] M. Mimura, S. Sakai, and T. Kawahara. Deep autoencoders augmented with phone-class feature for reverberant speech recognition. In *Proc. ICASSP*, 2015.
- [101] V. Mitra, H. Franco, C. Bartels, J. v. Hout, M. Graciarena, and D. Vergyri. Speech recognition in unseen and noisy channel conditions. In *Proc. ICASSP*, 2017.

- 
- [102] K. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [103] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B. H. Juang. Speech dereverberation based on variance-normalized delayed linear prediction. *IEEE Trans. Audio, Speech, Lang. Process.*, 18(7):1717–1731, 2010.
- [104] A. Narayanan, A. Misra, and K. Chin. Large-scale, sequence-discriminative, joint adaptive training for masking-based robust ASR. In *Proc. Interspeech*, 2015.
- [105] A. Narayanan and D. Wang. Joint noise adaptive training for robust automatic speech recognition. In *Proc. ICASSP*, 2014.
- [106] A. Narayanan and D. Wang. Improving robustness of deep neural network acoustic models via speech separation and joint adaptive training. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 23(1):92–101, 2015.
- [107] A. Nautsch, H. Hao, T. Stafylakis, C. Rathgeb, and C. Busch. Towards PLDA-RBM based speaker recognition in mobile environment: Designing stacked/deep PLDA-RBM systems. In *Proc. ICASSP*, 2016.
- [108] R. W. Ng, M. Nicolao, and T. Hain. Unsupervised crosslingual adaptation of tokenisers for spoken language recognition. *Comput. Speech and Language*, 46:327–342, 2017.
- [109] NIST. The 2003 NIST Language Recognition Evaluation Plan, 2003.
- [110] NIST. The 2007 NIST Language Recognition Evaluation Plan, 2007.
- [111] NIST. The NIST Year 2008 Speaker Recognition Evaluation Plan, 2008.
- [112] NIST. The 2009 NIST Language Recognition Evaluation Plan, 2009.

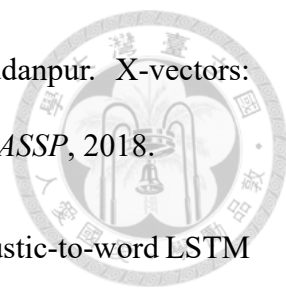
- 
- [113] NIST. The NIST Year 2010 Speaker Recognition Evaluation Plan, 2010.
- [114] E. Oja and T. Kohonen. The subspace learning algorithm as a formalism for pattern recognition and neural networks. In *Proc. ICNN*, 1988.
- [115] K. Okabe, T. Koshinaka, and K. Shinoda. Attentive statistics pooling for deep speaker embedding. In *Proc. Interspeech*, 2018.
- [116] S. Pascual, A. Bonafonte, and J. Serra. SEGAN: speech enhancement generative adversarial network. In *Proc. Interspeech*, 2017.
- [117] D. B. Paul and J. M. Baker. The design for the wall street journal-based CSR corpus. In *Proc. DARPA Speech and Natural Language Workshop*, 1992.
- [118] V. Peddinti, D. Povey, and S. Khudanpur. A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts. In *Proc. Interspeech*, 2015.
- [119] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur. Low latency acoustic modeling using temporal convolution and LSTMs. *IEEE Signal Process. Lett.*, 25(3):373–377, 2018.
- [120] J. Pelecanos and S. Sridharan. Feature warping for robust speaker verification. In *Proc. Odyssey*, 2001.
- [121] M. Penagarikano, A. Varona, L. Rodriguez-Fuentes, and G. Bodel. Improved modeling of cross-decoder phone co-occurrences in SVM-based phonotactic language recognition. *IEEE Trans. Audio, Speech, Lang. Process.*, 19(8):2348–2363, 2011.
- [122] O. Plchot, L. Burget, H. Aronowitz, and P. Matejka. Audio enhancing with DNN autoencoder for speaker recognition. In *Proc. ICASSP*, 2016.

- 
- [123] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohamadi, and S. Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Proc. Interspeech*, 2018.
- [124] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The Kaldi speech recognition toolkit. In *Proc. IEEE ASRU*, 2011.
- [125] D. Povey, V. Peddinti, D. Galvez, P. Ghahramani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur. Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *Proc. Interspeech*, 2016.
- [126] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.
- [127] Y. Qian, M. Yin, Y. You, and K. Yu. Multi-task joint-learning of deep neural networks for robust speech recognition. In *Proc. IEEE ASRU*, 2015.
- [128] L. Rabiner and R. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall, 1978.
- [129] S. Razakarivony and F. Jurie. Discriminative autoencoders for small targets detection. In *Proc. ICPR*, 2014.
- [130] D. Rethage, J. Pons, and X. Serra. A wavenet for speech denoising. In *Proc. ICASSP*, 2018.
- [131] F. Richardson, D. Reynolds, and N. Dehak. Deep neural network approaches to speaker and language recognition. *IEEE Signal Process. Lett.*, 22(10):1671–1675, 2015.


- 
- [132] F. S. Richardson and W. M. Campbell. Language recognition with discriminative keyword selection. In *Proc. ICASSP*, 2008.
- [133] I. B. Risteski and K. G. Trenchevski. Principal values and principal subspaces of two subspaces of vector spaces with inner product. *Beiträge zur Algebra und Geometrie*, 2001.
- [134] J. Rohdin, S. Biswas, and K. Shinoda. Constrained discriminative PLDA training for speaker verification. In *Proc. ICASSP*, 2014.
- [135] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015.
- [136] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Comput.*, 11(2):305–345, 1999.
- [137] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6):533–536, 1986.
- [138] T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *Proc. ICASSP*, 2012.
- [139] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for LVCSR. In *Proc. ICASSP*, 2013.
- [140] A. K. Sarkar, C.-T. Do, V.-B. Le, and C. Barras. Combination of cepstral and phonetically discriminative features for speaker verification. *IEEE Signal Process. Lett.*, 21(9):1040–1044, 2014.
- [141] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proc. ICLR*, 2014.

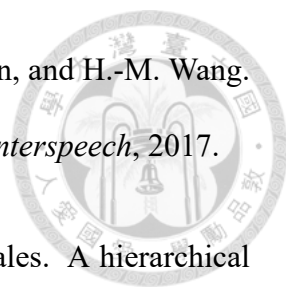
- 
- [142] B. Schölkopf. The kernel trick for distances. In *Proc. NIPS*, 2000.
- [143] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.
- [144] P. Schwarz. *Phoneme recognition based on long temporal context*. PhD thesis, Brno University of Technology, 2008.
- [145] M. L. Seltzer, D. Yu, and Y. Wang. An investigation of deep neural networks for noise robust speech recognition. In *Proc. ICASSP*, 2013.
- [146] A. Senior, H. Sak, F. d. C. Quiry, T. Sainath, and K. Rao. Acoustic Modelling with CD-CTC-SMBR LSTM RNNS. In *Proc. IEEE ASRU*, 2015.
- [147] Y.-C. Shih, H.-S. Lee, H.-M. Wang, and S.-K. Jeng. Subspace-based feature representation and learning for language recognition. In *Proc. Interspeech*, 2012.
- [148] S. M. Siniscalchi, J. Reed, T. Svendsen, and C.-H. Lee. Exploring universal attribute characterization of spoken languages for spoken language recognition. In *Proc. Interspeech*, 2009.
- [149] D. Snyder, G. Chen, and D. Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015.
- [150] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur. Spoken language recognition using x-vectors. In *Proc. Odyssey*, 2018.
- [151] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur. Speaker recognition for multi-speaker conversations using x-vectors. In *Proc. ICASSP*, 2019.



- 
- [152] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. In *Proc. ICASSP*, 2018.
- [153] H. Soltau, H. Liao, and H. Sak. Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition. In *Proc. Interspeech*, 2017.
- [154] M. Soni and A. Panda. Label driven time-frequency masking for robust continuous speech recognition. In *Proc. Interspeech*, 2019.
- [155] M. Soufifar, M. Kockmann, L. Burget, O. Plchot, O. Glembek, and T. Svendsen. iVector approach to phonotactic language recognition. In *Proc. Interspeech*, 2011.
- [156] B. M. L. Srivastava, H. Vydana, A. K. Vuppala, and M. Shrivastava. Significance of neural phonotactic models for large-scale spoken language identification. In *Proc. IJCNN*, 2017.
- [157] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, 2014.
- [158] S. Tan and K. C. Sim. Learning utterance-level normalisation using Variational Autoencoders for robust automatic speech recognition. In *Proc. IEEE SLT*, 2016.
- [159] Y. Tang, N. Srivastava, and R. R. Salakhutdinov. Learning generative models with visual attention. In *Proc. NIPS*, 2014.
- [160] N. Tawara, T. Kobayashi, M. Fujieda, K. Katagiri, T. Yazu, and T. Ogawa. Adversarial autoencoder for reducing nonlinear distortion. In *Proc. APSIPA ASC*, 2018.
- [161] R. Tong, B. Ma, H. Li, and E. S. Chng. A target-oriented phonotactic front-end for spoken language recognition. *IEEE Trans. Audio, Speech, Lang. Process.*, 17(7):1335–1347, 2009.

- 
- [162] P. Torres-Carrasquillo, E. Singer, M. Kohler, R. Greene, D. Reynolds, and J. Deller. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *Proc. ICSLP*, 2002.
- [163] K. Tsuda. Subspace classifier in the Hilbert space. *Pattern Recognition Lett.*, 20(5):513–519, 1999.
- [164] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2273–2286, 2011.
- [165] C. Upton. English Dialect Study: an Overview, 2012.
- [166] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008.
- [167] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *Proc. ICASSP*, 2014.
- [168] A. Veeraraghavan, A. Roy-Chowdhury, and R. Chellappa. Matching shape sequences in video with applications in human movement analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(12):1896–1909, 2005.
- [169] K. Vesely, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. In *Proc. Interspeech*, 2013.
- [170] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

- 
- [171] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.
- [172] G. Wang and K. C. Sim. Sequential classification criteria for NNs in automatic speech recognition. In *Proc. Interspeech*, 2011.
- [173] H.-M. Wang, B. Chen, J.-W. Kuo, and S.-S. Cheng. MATBN: A Mandarin Chinese broadcast news corpus. *International Journal of Computational Linguistics & Chinese Language Processing*, 10(2):219–236, 2005.
- [174] Y. Wang and D. Wang. Towards scaling up classification-based speech separation. *IEEE Trans. Audio, Speech, Lang. Process.*, 21(7):1381–1390, 2013.
- [175] E. Wong and S. Sridharan. Methods to improve gaussian mixture model based language identification system. In *Proc. ICSLP*, 2002.
- [176] Y. Xu, J. Du, L. R. Dai, and C. H. Lee. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal Process. Lett.*, 21(1):65–68, 2014.
- [177] T. Yamada, L. Wang, and A. Kai. Improvement of distant-talking speaker identification using bottleneck features of DNN. In *Proc. Interspeech*, 2013.
- [178] O. Yamaguchi, K. Fukui, and K.-i. Maeda. Face recognition using temporal image sequence. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1998.
- [179] Y. Yan and E. Barnard. An approach to automatic language identification based on language-dependent phone recognition. In *Proc. ICASSP*, 1995.

- 
- [180] M.-H. Yang, H.-S. Lee, Y.-D. Lu, K.-Y. Chen, Y. Tsao, B. Chen, and H.-M. Wang. Discriminative autoencoders for acoustic modeling. In *Proc. Interspeech*, 2017.
- [181] T. F. Yap, J. Epps, E. Ambikairajah, E. H. C. Choi, and S. Wales. A hierarchical framework for language identification. In *Proc. ICASSP*, 2016.
- [182] S. Yin, C. Liu, Z. Zhang, Y. Lin, D. Wang, J. Tejedor, T. Fang, and Y. Li. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015.
- [183] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide. Feature learning in deep neural networks: Studies on speech recognition tasks. In *Proc. ICLR*, 2013.
- [184] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional Networks. In *Proc. CVPR*, 2010.
- [185] X. Zheng, Z. Wu, H. Meng, and L. Cai. Contrastive auto-encoder for phoneme recognition. *Proc. ICASSP*, 2014.
- [186] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey. Self-attentive speaker embeddings for text-independent speaker verification. In *Proc. Interspeech*, 2018.
- [187] M. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Trans. Audio, Speech, Lang. Process.*, 4(1):31–44, 1996.
- [188] M. A. Zissman and K. M. Berkling. Automatic language identification. *Speech Commun.*, 35(1-2):115–124, 2001.
- [189] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005.

- [190] G. Zucon, L. A. Azzopardi, and C. J. van Rijsbergen. Semantic spaces: Measuring the distance between different subspaces. In *Proc. International Symposium on Quantum Interaction (QI)*, 2009.







## Appendix A — Subspace Construction

---

**Algorithm 1** The Subspace Method for DLMs

---

**Input:** The observation matrix  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_K]$ .

**Output:** The estimates  $\hat{\mathbf{A}}$  and  $\hat{\mathbf{C}}$ .

1. Perform truncated SVD on  $\mathbf{Y}$ , such that  $\mathbf{Y} \approx \mathbf{U}\Sigma\mathbf{V}^T$  [15].  $\Sigma$  is a  $d \times d$  diagonal matrix containing the largest  $d$  positive singular values.  $\mathbf{U} \in \mathbb{R}^{M \times d}$  and  $\mathbf{V} \in \mathbb{R}^{K \times d}$  are the corresponding eigen-matrices.
2. Set  $\hat{\mathbf{C}}$ , the estimate of  $\mathbf{C}$ , to be  $\mathbf{U}$ .
3. Set  $\hat{\mathbf{X}}_{1:K}$ , the estimate of the state matrix  $\mathbf{X}_{1:K}$ , to be  $\Sigma\mathbf{V}^T$ , where  $\hat{\mathbf{X}}_{1:K}$  stands for  $[\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_K] \in \mathbb{R}^{d \times K}$ .
4. Estimate  $\hat{\mathbf{A}}$  by solving the following equation:

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{k=1}^{K-1} \|\hat{\mathbf{x}}_{k+1} - \mathbf{A}\hat{\mathbf{x}}_k\|^2 \quad (\text{A.1a})$$

$$\text{subject to } \mathbf{A}^T \mathbf{A} = \mathbf{I}_d, \quad (\text{A.1b})$$

where  $\|\cdot\|$  denotes the  $l^2$ -norm.

---

Algorithm 1 differs from the algorithms in our previous study [75] and other related studies [49, 84] in that an orthonormal constraint on  $\mathbf{A}$  is added in the last step. Such a constraint is necessary for the pursuit of orthogonality presented in Section 2.3.2. In fact, (A.1) is an orthogonal procrustes problem, and its solution is an orthonormal matrix  $\hat{\mathbf{A}}$  that most closely maps  $\hat{\mathbf{X}}_{1:(K-1)}$  to  $\hat{\mathbf{X}}_{2:K}$ . By referring to [16, Section 1], the process of



solving (A.1) can be summarized as Algorithm 2.

---

**Algorithm 2** The Orthogonal Procrustes Problem

---

**Input:** The state matrix  $\hat{\mathbf{X}}_{1:K}$  estimated by Algorithm 1.

**Output:** The estimate  $\hat{\mathbf{A}}$ .

1. Perform full SVD on  $\hat{\mathbf{X}}_{2:K}\hat{\mathbf{X}}_{1:(K-1)}^T$ , such that  $\hat{\mathbf{X}}_{2:K}\hat{\mathbf{X}}_{1:(K-1)}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ , where  $\mathbf{U}$ ,  $\mathbf{\Sigma}$ , and  $\mathbf{V}$  are all  $d \times d$  matrices.
  2. Set  $\hat{\mathbf{A}}$ , the estimate of  $\mathbf{A}$ , to be  $\mathbf{U}\mathbf{V}^T$ .
- 

To solve (2.4) iteratively, two propositions must be provided:

**Proposition 1.** *Given  $\mathbf{S}^T\mathbf{S} = \mathbf{I}_d$ ,  $\hat{\mathbf{W}} = O_{\lambda_{odl}}(\mathbf{S}^T\mathbf{Z})$  is the unique solution of the following minimization problem:*

$$\min_{\mathbf{W}} \|\mathbf{Z} - \mathbf{S}\mathbf{W}\|_F^2 + \lambda_{odl}^2 \|\mathbf{W}\|_0. \quad (\text{A.2})$$

$O_{\lambda_{odl}}(\cdot)$  is a threshold operator.  $[O_{\lambda_{odl}}(\mathbf{X})]_{ij} = \mathbf{X}_{ij}$  if the absolute value of the  $(i, j)$ -th entry of  $\mathbf{X}$  is larger than  $\lambda$ , and  $[O_{\lambda_{odl}}(\mathbf{X})]_{ij} = 0$  otherwise.

**Proposition 2.**  $\hat{\mathbf{S}}_{odl} = \mathbf{P}\mathbf{Q}^T$  is the unique solution of the following minimization problem:

$$\min_{\mathbf{S}} \|\mathbf{Z} - \mathbf{S}\hat{\mathbf{W}}\|_F^2 \quad (\text{A.3a})$$

$$\text{subject to } \mathbf{S}^T\mathbf{S} = \mathbf{I}_d. \quad (\text{A.3b})$$

$\mathbf{P}$  and  $\mathbf{Q}$  denote the orthonormal matrices defined in the following truncated SVD of  $\mathbf{Z}\hat{\mathbf{W}}^T$ :

$$\mathbf{Z}\hat{\mathbf{W}}^T = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T. \quad (\text{A.4})$$

The mathematical proof is provided in [5]. We performed some minor modifications to suit to our study, but the proof is still valid. The process of solving (2.4) is provided in



Algorithm 3.

---

**Algorithm 3** The Orthogonal Dictionary Learning

---

**Input:** The contextualized phonetic matrix  $\mathbf{Z} \in \mathbb{R}^{D \times K}$ .

**Output:** The subspace  $\hat{\mathbf{S}}_{odl} \in \mathbb{R}^{D \times d}$ .

1. Set a feasible initial guess  $\mathbf{S}^{(0)}$ .
  2. For  $j = 0, 1, \dots, J$ ,
    - (a)  $\mathbf{W}^{(j)} \leftarrow O_{\lambda_{odl}}(\mathbf{S}^T \mathbf{Z})$ ;
    - (b) Perform truncated SVD on  $\mathbf{Z}\mathbf{W}^{(j)T} = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$ ;
    - (c)  $\mathbf{S}^{(i+1)} \leftarrow \mathbf{P}\mathbf{Q}^T$ .
  3.  $\hat{\mathbf{S}}_{odl} \leftarrow \mathbf{S}^{(J+1)}$ .
- 

In this study,  $\mathbf{S}^{(0)}$  was horizontally stacked by  $d$  randomly sampled column vectors of an identity matrix  $\mathbf{I}_D \in \mathbb{R}^{D \times D}$ .







## Appendix B — My Publications

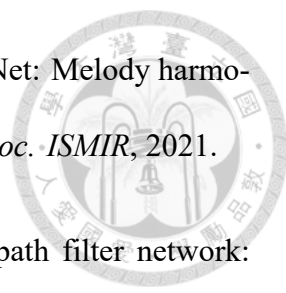
Papers related to the doctoral dissertation are marked with an asterisk (\*). My name (H.-S. Lee) is in boldface.

### Journal Articles


1. \* **H.-S. Lee**, Y. Tsao, S.-K. Jeng, and H.-M. Wang, “Subspace-based representation and learning for phonotactic spoken language recognition,” in *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 2020, vol. 28, pp. 3065–3079.
2. \* **H.-S. Lee**, P.-Y. Chen, I Chien, J.-S. Roger Jang, and H.-M. Wang, “Speech-enhanced and noise-aware acoustic modeling for robust speech recognition,” submitted to *IEEE Signal Process. Lett.*.
3. \* C.-L. Wu, H.-P. Hsu, Y.-D. Lu, Yu Tsao, **H.-S. Lee**, and H.-M. Wang, “A replay spoofing detection system based on discriminative autoencoders,” in *International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP)*, 2017, vol. 22, no. 2, pp. 63–72.

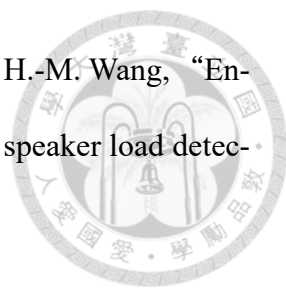
### Conference Proceedings

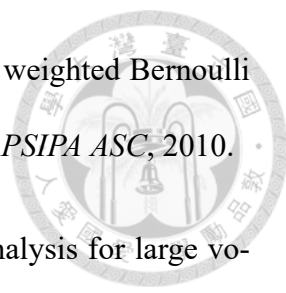
1. Y.-H. Peng, **H.-S. Lee**, P.-T. Huang, and H.-M. Wang, “Generation of Speaker Representations Using Heterogeneous Training Batch Assembly,” in *APSIPA ASC*, 2021.

- 
2. Y.-W. Chen, **H.-S. Lee**, Y.-H. Chen, and H.-M. Wang, “SurpriseNet: Melody harmonization conditioning on user-controlled surprise contours,” in *Proc. ISMIR*, 2021.
  3. F.-L. Wang, Y.-H. Peng, **H.-S. Lee**, and H.-M. Wang, “Dual-path filter network: Speaker-aware modeling for speech separation,” in *Proc. Interspeech*, 2021.
  4. Y.-F. Cheng, **H.-S. Lee**, and H.-M. Wang, “AlloST: Low-resource speech translation without source transcription,” in *Proc. Interspeech*, 2021.
  5. Y.-C. Wu, C.-H. Hu, **H.-S. Lee**, Y.-H. Peng, W.-C. Huang, Yu Tsao, H.-M. Wang, and T. Toda, “Relational data selection for data augmentation of speaker-dependent multi-band MelGAN vocoder,” in *Proc. Interspeech*, 2021.
  6. C.-E. Sun, Y.-W. Chen, **H.-S. Lee**, Y.-H. Chen, and H.-M. Wang, “Melody harmonization using orderless NADE, chord balancing, and blocked Gibbs sampling,” in *Proc. ICASSP*, 2021.
  7. \* **H.-S. Lee**, Y.-H. Peng, P.-T. Huang, Y.-C. Tseng, C.-H. Wu, Yu Tsao, H.-M. Wang, “The Academia Sinica systems of speech recognition and speaker diarization for the CHiME-6 challenge,” in *Proc. CHiME*, 2020.
  8. H. Yen, P.-J. Ku, M.-C. Yen, **H.-S. Lee**, and H.-M. Wang, “Joint training of guided learning and mean teacher models for sound event detection,” in *Proc. DCASE*, 2020.
  9. P.-Y. Chen, C.-H. Wu, **H.-S. Lee**, S.-K. Tsao, M.-T. Ko, and H.-M. Wang, “Using Taigi dramas with Mandarin Chinese subtitles to improve Taigi speech recognition,” in *Proc. O-COCOSDA*, 2020.
  10. X.-B. Chen, Y.-T. Lee, **H.-S. Lee**, J.-S. R. Jang, and H.-M. Wang, “Mandarin mispronunciation detection and diagnosis feedback using articulatory attributes based multi-

task learning,” in *Proc. O-COCOSDA*, 2019.

- 
11. Y.-T. Lee, X.-B. Chen, **H.-S. Lee**, J.-S. R. Jang, and H.-M. Wang, “Multi-task learning for acoustic modeling using articulatory attributes,” in *Proc. APSIPA ASC*, 2019.
  12. S.-B. Luo, **H.-S. Lee**, K.-Y. Chen, and H.-M. Wang, “Spoken multiple-choice question answering using multimodal convolutional neural networks,” in *Proc. IEEE ASRU*, 2019.
  13. \* P.-T. Huang, **H.-S. Lee**, S.-S. Wang, K.-Y. Chen, Y. Tsao, and H.-M. Wang, “Exploring the encoder layers of discriminative autoencoders for LVCSR,” in *Proc. Interspeech*, 2019.
  14. \* M.-H. Yang, **H.-S. Lee**, Y.-D. Lu, K.-Y. Chen, Yu Tsao, B. Chen, H.-M. Wang “Discriminative autoencoders for acoustic modeling,” in *Proc. Interspeech*, 2017.
  15. \* **H.-S. Lee**, Y.-D. Lu, C.-C. Hsu, Y. Tsao, H.-M. Wang, and S.-K. Jeng, “Discriminative autoencoders for speaker verification,” in *Proc. ICASSP*, 2017.
  16. C.-J. Ray Chang, **H.-S. Lee**, H.-M. Wang, and J.-S. Roger Jang, “Speaker diarization based on i-vector PLDA scoring and using GMM-HMM forced alignment,” in *Proc. ROCLING*, 2017.
  17. **H.-S. Lee**, Yu Tsao, C.-C. Lee, H.-M. Wang, W.-C. Lin, W.-C. Chen, S.-W. Hsiao, and S.-K. Jeng, “Minimization of regression and ranking losses with shallow neural networks on automatic sincerity evaluation,” in *Proc. Interspeech*, 2016.
  18. **H.-S. Lee**, Y. Tsao, H.-M. Wang, and S.-K. Jeng, “Clustering-based i-vector formulation for speaker recognition,” in *Proc. Interspeech*, 2014.

- 
19. H. Jing, T.-Y. Hu, **H.-S. Lee**, W.-C. Chen, C.-C. Lee, Yu Tsao, and H.-M. Wang, “Ensemble of machine learning algorithms for cognitive and physical speaker load detection,” in *Proc. Interspeech*, 2014.
  20. K.-Y. Chen, **H.-S. Lee**, H.-M. Wang, B. Chen, and H.-H. Chen, “I-vector based language modeling for spoken document retrieval,” in *Proc. ICASSP*, 2014.
  21. **H.-S. Lee**, Y. Tsao, Y.-F. Chang, H.-M. Wang, and S.-K. Jeng, “Speaker verification using kernel-based binary classifiers with binary operation derived features,” in *Proc. ICASSP*, 2014.
  22. \* **H.-S. Lee**, Y.-C. Shih, H.-M. Wang, and S.-K. Jeng, “Subspace-based phonotactic language recognition using multivariate dynamic linear models,” in *Proc. ICASSP*, 2013.
  23. K.-Y. Chen, **H.-S. Lee**, C.-H. Lee, H.-M. Wang, and H.-H. Chen, “A study of language modeling for chinese spelling check,” in *Proc. SIGHAN*, 2013.
  24. \* Y.-C. Shih, **H.-S. Lee**, H.-M. Wang, and S.-K. Jeng, “Subspace-based feature representation and learning for language recognition,” in *Proc. Interspeech*, 2012.
  25. J.-C. Wang, **H.-S. Lee**, H.-M. Wang, and S.-K. Jeng, “Learning the similarity of audio music in bag-offrames representation from tagged music data,” in *Proc. ISMIR*, 2011.
  26. M.-S. Wu, **H.-S. Lee**, and H.-M. Wang, “Exploiting semantic associative information in topic modeling,” in *Proc. IEEE SLT*, 2010.
  27. **H.-S. Lee**, H.-M. Wang, and B. Chen, “A discriminative and heteroscedastic linear feature transformation for multiclass classification,” in *Proc. ICPR*, 2010.

- 
28. J.-C. Wang, **H.-S. Lee**, S.-K. Jeng, and H.-M. Wang, “Posterior weighted Bernoulli mixture model for music tag annotation and retrieval,” in *Proc. APSIPA ASC*, 2010.
29. **H.-S. Lee** and B. Chen, “Likelihood ratio based discriminant analysis for large vocabulary continuous speech recognition,” in *Proc. ROCLING*, 2009.
30. **H.-S. Lee** and B. Chen, “Generalized likelihood ratio discriminant analysis,” in *Proc. IEEE ASRU*, 2009.
31. **H.-S. Lee** and B. Chen, “Empirical error rate minimization based linear discriminant analysis,” in *Proc. ICASSP*, 2009.
32. **H.-S. Lee** and B. Chen, “Improved linear discriminant analysis considering empirical pairwise classification error rates,” in *Proc. ISCSLP*, 2008.
33. **H.-S. Lee** and B. Chen, “Linear discriminant feature extraction using weighted classification confusion information,” in *Proc. Interspeech*, 2008.
34. S.-H. Liu, F.-H. Chu, S.-H. Lin, **H.-S. Lee**, and B. Chen, “Training data selection for improving discriminative training of acoustic models,” in *Proc. IEEE ASRU*, 2007.



*Love* never ends.

As for prophecies, they will pass away;

as for tongues, they will cease;

as for *knowledge*, it will pass away.

—1 Corinthians 13:8, ESV