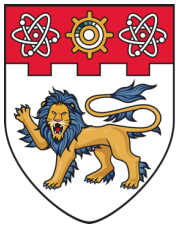
Enhancing Spoken Language Identification and Diarization for Multilingual Speech



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Liu Hexin

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

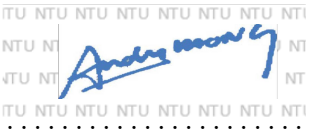
10th August 2022

Date

Liu Hexin

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.



10th August 2022

Date

A/P. Andy W. H. Khong

Authorship Attribution Statement

This thesis contains material from four papers published in the following peer-reviewed journal(s) and accepted at conferences in which I am listed as an author.

Chapter 3 is published as H. Liu, L. P. G. Perera, A. W. H. Khong, J. Dauwels, S. J. Styles, S. Khudanpur, "Enhancing Language Identification Using Dual-Mode Model with Knowledge Distillation," in Proc. Speaker Lang. Recognit. Workshop (Odyssey), 2022, pp. 248-254

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. Dr. Leibny Paola Garcia Perera, Prof A. W. H. Khong, Prof. Justin Dauwels, and Prof. Sanjeev Khudanpur assisted me in revising the manuscript draft.
- I co-designed the method and experiments with Dr. Leibny Paola Garcia Perera.
- Dr. Leibny Paola Garcia Perera assisted me with experiments regarding x-vector baseline, I performed all other experiments.
- Prof. Suzy J. Styles provided the insight of analysis regarding clip-wise linguistic variability and lexical integrity;

Chapter 4 is published as H. Liu, L. P. G. Perera, A. W. H. Khong, E.S. Chng, S. J. Styles, S. Khudanpur. (2022) Efficient Self-supervised Learning Representations for Spoken Language Identification. IEEE J. Sel. Topics Signal Process., to appear

The contributions of the co-authors are as follows:

- I wrote the manuscript draft and performed all experiments. The manuscript draft was revised with the help of Dr. Leibny Paola Garcia Perera, Prof Andy W. H. Khong, Prof. Suzy J. Style, and Prof. Sanjeev Khudanpur.
- Prof. Eng Siong Chng provided the idea of dimension-wise scaling method.
- Dr. Leibny Paola Garcia Perera and I proposed other methods and designed all experiments.

Chapter 5 is published as H. Liu, L. P. G. Perera, A. W. H. Khong, S. J. Styles, S. Khudanpur, "PHO-LID: A Unified Model Incorporating Acoustic-Phonetic and Phonotactic Information for Language Identification," in Proc. Interspeech, 2022, to appear

The contributions of the co-authors are as follows:

- I performed all experiments and prepared the manuscript.
- Dr. Leibny Paola Garcia Perera, Prof. Andy W. H. Khong, and I designed the whole study.
- The manuscript was revised and polished with the help from Dr. Leibny Paola Garcia Perera, Prof. Andy W.H. Khong, and Prof. Sanjeev Khudanpur.
- Prof. Suzy J. Styles helped us analyze the results regarding phoneme boundary detection.

Chapter 6 is published as H. Liu, L. P. G. Perera, X. Zhang, J. Dauwels, A. W. H. Khong, S. Khudanpur, and S. J. Styles, "End-to-end language diarization for bilingual code-switching speech," in Proc. Interspeech, 2021, pp. 1489–1493

The contributions of the co-authors are as follows:

- I co-designed this study with Dr. Leibny Paola Garcia Perera and we performed all the experiments with the help of Dr. Xinyi Zhang.
- I prepared the manuscript drafts.
- Dr. Leibny Paola Garcia Perera, Prof Andy W. H. Khong, Prof. Justin Dauwels, Prof. Sanjeev Khudanpur, and Prof. Suzy J. Styles assist me in revising the manuscript draft.

10th August 2022
.....
Date

.....
Liu Hexin

Acknowledgements

I would like to first express my greatest gratitude to my advisors, Dr. Andy W. H. Khong and Dr. Leibny Paola Garcia Perera, for their invaluable supervision, guidance, and support throughout my Ph.D. journey. Their immense knowledge and selfless help have encouraged me to grow as a research scientist and an enthusiastic person.

I would like to thank Dr. Sanjeev Khudanpur, Dr. Chng Eng Siong, and Dr. Suzy J. Styles for their insightful and generous comments and suggestions on my research. I wish to extend my thanks to Dr. Justin Dauwels for offering me the scholarship that allows me to pursue this degree.

I would also like to thank my team members of the NRF and ISSAC projects for a cherished time spent together. I enjoy our stimulating discussion and collaboration in our meetings. Many thanks to all people at Nanyang Technological University who help me to complete my Ph.D. program.

I want to thank my friends, whose warm concern and lovely chat have accompanied me these years.

Finally, I would like to express my special thanks to my family, especially my mom, for their consistent love and support.

I am aware of how dull and difficult I can be. I just want to say, in my way, I love you all.

Abstract

Spoken language identification (LID) refers to the automatic process of determining the identity of the language spoken in a speech signal. It has been widely employed as preprocessing in multilingual speech signal processing systems. While existing approaches have shown high performance for general LID, it is still challenging to perform well on speech of various durations. In addition, general LID methods only employ a single type of language cue. Since language cues depict language information from different perspectives, the incorporation of language cues is expected to exhibit higher performance compared to using a single language cue. Therefore, in this thesis, an x-vector self-attention LID (XSA-LID) model is proposed to achieve robustness to speech duration. Two approaches are next introduced to improve and incorporate the language cues, respectively. Finally, LID is performed in a more complex scenario—language diarization via an end-to-end LD model.

To achieve robustness against performance degradation due to varying duration, a dual-mode framework on the XSA-LID model with knowledge distillation (KD) is proposed. The dual-mode XSA-LID model is trained by jointly optimizing both the full and short modes with their respective inputs being the full-length speech and its short clip extracted by a specific Boolean mask before KD is applied to further boost the performance on short utterances. In addition, the impact of clip-wise linguistic variability and lexical integrity for LID is investigated by analyzing the variation of LID performance in terms of the lengths and positions of the mimicked speech clips.

To enhance LID from the perspective of language cues, two methods are next introduced through which language cues can be utilized efficiently and effectively. The first method investigated efficient methods to compute reliable representations and discard redundant information for LID using a pre-trained multilingual wav2vec 2.0 model. To determine an optimal basic system, the performance of the wav2vec features extracted from the different inner layers of the context network are compared. For this approach, the XSA-LID model forms the backbone used

to discriminate between distinct languages. Two mechanisms are then employed to reduce the irrelevant information of the representations in LID—the first being the attentive squeeze-and-excitation (SE) block for dimension-wise scaling and the second being the linear bottleneck (LBN) block that reduces the irrelevant information by nonlinear dimension reduction. These two methods are incorporated within the XSA-LID model, named AttSE-XSA and LBN-XSA respectively.

In the second approach, a novel LID model is proposed to hierarchically incorporate phoneme and phonotactic information without requiring phoneme annotations for training. In this model, named PHO-LID, a self-supervised phoneme segmentation task and a LID task share a convolutional neural network (CNN) module, which encodes both *language* identity and sequential *phonemic information* in the input speech to generate an intermediate sequence of "phonotactic" embeddings. These embeddings are then fed into transformer encoder layers for utterance-level LID. This architecture is called CNN-Trans.

Finally, LID is extended for a code-switching scenario language diarization. In this work, two end-to-end neural configurations are proposed for language diarization on bilingual code-switching speech. The first, a BLSTM-E2E architecture, includes a set of stacked bidirectional LSTMs to compute embeddings and incorporates the deep clustering loss to enforce grouping of languages belonging to the same class. The second, an XSA-E2E architecture, is based on an x-vector model followed by a self-attention encoder. The former encodes frame-level features into segment-level embeddings while the latter considers all those embeddings to generate a sequence of segment-level language labels.

All proposed approaches are evaluated on standard datasets including NIST LRE 2017, OLR, SEAME, and WSTCSMC 2020. Compared with the baseline systems, the proposed approaches exhibits significant performance improvement on their corresponding language identification and diarization tasks.

Contents

List of Figures

List of Tables

Symbols and Acronyms

Symbols

AvgPool	the average pooling operation
$\mathrm{CE}(\cdot)$	the cross entropy loss function
$\delta(\cdot)$	the ReLU activation function
$\sigma(\cdot)$	the sigmoid activation function
$\exp(\cdot)$	the exponential function
$\mathbf{E}$	the language embeddigs
$\mathbf{H}$	the hidden outputs
$\mathbf{Head}_j^i$	the i-th attention head output of the j-th transformer encoder layer
$\mathrm{KL}(\cdot, \cdot)$	the KL divergence
L	the loss
$\log(\cdot)$	the logarithm
$\mathbf{Mask}$	the attention mask matrix used during training
$\mathcal{N}(a, b)$	Normal distribution with mean a and variance b
$\mathrm{Mean}(\cdot)$	the mean vector of input
$\mathbb{R}^n$	the n-dimensional real vector space
$\mathbf{S}_n$	the n-th short segment partitioned from speech features
$\mathrm{Scale}(\cdot)$	the element-wise product
$\mathrm{SelfAtten}(\cdot, \cdot)$	the computations performed within self-attention encoder layers
$\mathrm{sim}(\cdot, \cdot)$	the similarity between two vectors
$\mathrm{Softmax}(\cdot)$	the softmax operation
$\mathrm{Std}(\cdot)$	the standard deviation vector of input
$\mathrm{TDNN}(\cdot)$	the computations performed within TDNN layers
$\mathbf{Temp}$	the distillation temperature
$\mathbf{W}$	the linear projection

$\mathbf{X}$	the features of the speech sample
$\widetilde{\mathbf{X}}$	the processed features of the speech sample
$\mathbf{Y}$	the language label of the speech samples
$\widehat{\mathbf{Y}}$	the language prediction of the speech sample
$\mathbf{Z}_j$	the output of j-th transformer encoder layer

Acronyms

ASR	Automatic Speech Recognition
BLSTM	Bidirectional Long Short Term Memory
BNBS	Broadcast Narrow Band Speech
BNF	Bottleneck Feature
Cavg	Average Detection Cost
CE	Cross Entropy
CNN	Convolutional Neural Network
CS	Code-Switching
CTC	Connectionist Temporal Classification
CTS	Conversational Telephone Speech
DC	Deep Clustering
DNN	Deep Neural Network
E2E	End-to-End
EER	Equal Error Rate
EM	Expectation Maximization
FA	False Alarm
Fbank	Filter Bank Energy
FC	Fully-Connected
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
KD	Knowledge Distillation
KL	Kullback–Leibler
LBN	Linear Bottleneck
LD	Language Diarization
LDA	Linear Discriminative Analysis
LR	Logistic Regression
LID	Language identification
LRE	Language Recognition Evaluation
LM	Language Model
LSTM	Long Short Term Memory
MFCC	Mel Filterbank Cepstral Coefficient
MLS	Multi-Language Speech
MTL	Multi-task Learning

NCE	Noise-Contrastive Estimation
NIST	National Institute for Standards in Technology
OLR	Oriental Language Recognition
PCA	Principal Component Analysis
PER	Phoneme Error Rate
PHO	Phonetic and Phonotactic
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SA	Self-Attention
SAD	Speech Activity Detection
SE	Squeeze-and-Excitation
SID	Speaker Identification
SSL	Self-Supervised Learning
SVM	Support Vector Machine
TDNN	Time-Delay Neural Network
UBM	Universal Background Model
VAD	Voice Activity Detection
VAST	Video Annotation for Speech Technologies
W2V	Wav2Vec
XSA	X-vector Self Attention

Chapter 1

Introduction

1.1 Motivation

1.1.1 Spoken language identification

Spoken language identification (LID) refers to the process through which the language identity of a speech sample can automatically be determined [1]. This process aims to replicate human intelligence to discriminate different spoken languages and is widely employed as a front end in multilingual speech signal processing tasks such as multilingual automatic speech recognition (ASR) [2,3]. Systems based on a two-stage process such as i-vector and x-vector methods were introduced by [4,5], respectively, and have been dominant in this task. This type of system first extracts a language representation vector for a given speech sample using a language encoder before feeding the vector to a classifier.

However, the language encoder and classifier are optimized independently. For the i-vector approach, the language encoder is an unsupervised Gaussian mixture model (GMM) [4, 6], while the classifier requires ground-truth language labels during training; for the x-vector approach, the time-delay neural network-based (TDNN) language encoder is trained on fixed-length speech chunks while the classifier is trained on language representations extracted from various-duration utterances [7]. The mismatch between the front and back ends may degrade the performance of a two-stage LID system. The end-to-end LID system, in contrast, can tackle these problems by jointly optimizing the encoder and classifier.

Moreover, recent research suggests that deep neural network-based (DNN) models benefit more from extensive training speech resources than the traditional i-vector approach [8]. Therefore, developing end-to-end neural LID systems is important to improve state-of-the-art (SOTA) LID performance.

1.1.2 Duration Robustness

Notwithstanding that existing methods have shown promising results on general LID tasks, achieving LID on short utterances while maintaining high performance on long speech is still challenging. Plenty of works for Language Recognition Evaluation (LRE) held by National Institute for Standards in Technology (NIST) exhibit performance degradation on short speech (i.e., 3 s) compared to that on long speech (i.e., 10 s, 30 s) [5, 9, 10]. Recent deep neural network-based approaches were proposed for short-utterance LID and have shown to be more effective than conventional i-vector approaches for short utterances [11, 12]. Nevertheless, these approaches focus primarily on short utterances and are not developed to achieve good performance on long utterances. Practical applications of the LID system, however, should pursue good results on speech of varying duration. Developing duration-robust LID, therefore, is crucial for LID.

1.1.3 Language cues

Language discriminative information varies across perceptual cues in different abstraction levels. Early experiments pertaining to human listening suggests that both prelexical and lexical information can help humans recognize spoken languages [13], where prelexical language cues comprise short-term acoustic features and long-term phonotactics and prosody features. Since the acoustic and phonotactic features have been proven to be the most effective language cues for spoken language identification [1, 14, 15], these features have been widely investigated. Acoustic features such as Mel-frequency cepstral coefficients (MFCCs) and filter bank energies (Fbanks) are extracted using a sliding window that is defined as shorter than a phoneme. They are the most commonly used language cues in recent LID systems [4, 16, 17].

As opposed to an acoustic unit that is defined as shorter than a phoneme, a phonotactic unit can cover several phonemes. Therefore, phonotactic information can exhibit superiority for long-utterance LID [15,18] while acoustic information shows high performance on short speech. However, a phonotactic LID system requires phoneme annotations of speech samples during training. The annotation process is usually time-consuming and requires significant resources and domain expertise. In contrast, acoustic approaches require only digitized speech samples and their language labels—they exhibited higher convenience in terms implementation over recent years compared to phonotactic approaches. Nevertheless, acoustic and phonotactic cues depict the language identities of a speech signal in different granularities and perspectives and it is therefore natural to consider both jointly to achieve high LID performance.

In addition, as a paradigm of unsupervised learning, self-supervised learning (SSL) has been applied to learn speech representations from unlabeled speech data. In contrast to the language cues mentioned above, SSL representations such as wav2vec features [19] encode the audio information in high-dimensional vectors through a contrastive learning task and have shown to be effective in a number of speech signal processing tasks. Therefore, the method through which SSL features can be utilized efficiently in the LID task is worth exploiting.

1.1.4 Spoken language diarization

Code-switching refers to the switching of languages within a spontaneous multilingual recording. Language diarization (LD), as a special case of language identification, involves partitioning a code-switching speech signal into homogeneous segments before determining their language identities. The term diarization was originally used to describe the task of determining audio segments associated with the same speaker in a multi-speaker recording. It was subsequently extended to cover language diarization. Note that language diarization cannot be achieved using LID methods due to the one-language-per-audio-sample assumption. Conventional LD systems consist of multiple stages which are trained separately [20]. Considering the end-to-end LD model can intrinsically address the independent optimization problem, it is worth investigating to improve the state-of-the-art LD performance.

1.2 Objectives and Contributions

This thesis proposes two novel methods for the aforementioned problems in LID. The first method aims to develop a dual-mode LID model to enhance the LID performance of the single-mode model for both long and short utterances. The second method incorporates acoustic and phonotactic features in a unified model to improve the state-of-the-art LID performance. Apart from the phonotactic language cues, the third proposed method applies the SSL features in LID in an efficient manner. Furthermore, two end-to-end configurations are proposed for LD that is under a more complex code-switching (CS) scenario compared to LID.

1.2.1 Dual-mode LID model

A dual-mode LID model based on knowledge distillation (KD) and the x-vector self-attention end-to-end (XSA-E2E) model is proposed to enhance the LID performance on the speech of various duration [21, 22]. Apart from the performance enhancement in LID, the proposed dual-mode LID approach possesses several desirable properties. Firstly, the proposed method employs the KD method without introducing an additional model or parameters. Secondly, instead of preprocessing using speech segmentation, a Boolean mask is applied to acquire features more flexibly with lower computational complexity. In addition, the masking operation in the proposed model mimics short speech clips. Various types of linguistic information introduced by the mask serve as data augmentation and lead to higher LID performance. Lastly, since the Boolean mask is applied to the transformer encoder layers in the proposed model [23], the approach can easily be transferred.

1.2.2 Efficient self-supervised speech representations

This thesis investigates efficient methods to compute reliable representations and discard redundant information for language identification (LID) using a pre-trained multilingual wav2vec 2.0 model [24]. Two mechanisms are proposed as the front-end of the end-to-end x-vector-self-attention LID (XSA-LID) model to reduce the irrelevant information of the representations in LID. The first is the attentive squeeze-and-excitation (SE) block for dimension-wise scaling and the second is the

linear bottleneck (LBN) block that reduces the irrelevant information by nonlinear dimension reduction. The proposed LBN-XSA model not only achieves performance improvement compared to the XSA-LID model, but also outperforms the latter which adopts the fine-tuned features.

1.2.3 Incorporating acoustic and phonotactic features

A phonetic and phonotactic LID (PHO-LID) method is proposed to incorporate phonetic and phonotactic information hierarchically via a convolutional neural network-transformer encoder (CNN-Trans). Apart from the high LID performance resulting from the complementary characteristics of acoustic-phonetic and phonotactic features, the proposed method has two contributions. Firstly, as opposed to existing phoneme or phonotactic-aware methods, the proposed approach does not require phoneme annotation or transcription due to self-supervised training. Secondly, the proposed PHO-LID model combines different language cues in an end-to-end manner resulting in lower complexity compared to the fusion of several subsystems with respect to different language cues [1].

1.2.4 End-to-end language diarization

In this study, an x-vector-self-attention end-to-end (XSA-E2E) configuration is proposed for LD. The proposed LD approach possesses three desirable properties. Firstly, the LD task in this work is defined as a multi-label classification problem. The proposed models generate a sequence of segment-level labels for each test recording, where a segment consists of several adjacent frames, allowing the model to identify different languages within an utterance, detect the language change point, and tag the silences in a unified manner. Secondly, as opposed to existing approaches that require pre-processing for speech-activity detection (SAD), the proposed models perform SAD implicitly by defining silence as an output label. Finally, hierarchical processing is employed in the proposed model with multi-objective training. This hierarchical processing captures local language information in each segment before establishing global dependency between input and output, which benefits language diarization.

1.3 Thesis Organization

This chapter provides the motivation, objectives, and contributions of this thesis. Chapter 2 first introduces the background of spoken language identification and diarization with reviewing their respective recent advances.

Chapter 3 first illustrates the methodology of the proposed dual-mode LID model to enhance the LID performance for both long and short utterances. Next, the data, experiment setup, and experimental results are given, followed by the ablation studies. The influence of different Boolean masks for mimicking short input speech is discussed at the end of this chapter.

Chapter 3 introduces a short-utterance language identification approach with the aim of developing a duration-robust language identification system. As opposed to Chapter 3 belonging to Part I, Part II consists of Chapters 4 and 5 and focuses on enhancing and incorporating language cues. Chapter 4 describes the attentive squeeze-and-excitation and linear bottleneck blocks to compute efficient SSL wav2vec 2.0 features for LID. Data preparation, detailed model configurations, and training strategies are given before presenting the model performance. Other feature extraction attempts are next discussed followed by the conclusion.

As opposed to Chapter 3 where the focus is on single type of language cue, in Chapter 5, a unified PHO-LID model is proposed to incorporate acoustic and phonotactic information. Detailed experiment setup and ablation study are given to demonstrate the effectiveness of this method. In addition, visualization of confusion matrices imply the superiority of the proposed model on languages of the same cluster. A comparison between predicted phoneme boundaries and corresponding audio spectrograms illustrates the leveraging of phoneme information for LID.

In Chapter 6, two end-to-end LD configurations are proposed for bilingual code-switching speech. This chapter then introduces two datasets where the experiments are conducted, one of which is from a workshop and the other is simulated using monolingual speech from a code-switching corpus. Lastly, the experiment results are presented, followed by the discussion and analysis.

Finally, Chapter 7 concludes this thesis and discusses the future work.

Chapter 2

Literature Review

This chapter reviews literature of language identification and diarization, and illustrates datasets and metrics used in this thesis to evaluate LID and LD models.

Section 2.1 introduces existing LID approaches in terms of the knowledge abstraction level of language cues employed in these LID systems. The diarization task is then introduced in Section 2.2 by extending the monolingual speech sample in LID to a code-switching case. Corpora on which the experiments are conducted in this thesis are described in Section 2.3. Performance metrics are finally described in Section 2.4.

2.1 Spoken language identification systems

A spoken language can be characterized by prelexical and lexical language information. Prelexical language cues comprise acoustic, phonotactic, and prosodic features, while lexical language cues include words and syntax. In addition, self-supervised speech representations have shown to be significantly superior in many speech signal processing tasks than the aforementioned features.

2.1.1 Acoustic features and approaches

One language generally differs from another in terms of phones and the frequency of their occurrence. Under the assumption that the phonemic information is encoded

in the audio, acoustic features are extracted from the audio spectrum through a sliding window with an overlap, where the window is usually defined as between 10 ms to 50 ms that is shorter than a phoneme (e.g., 25 ms), while the overlap can be 10 ms to compensate for the distortion due to sidelobe of the window. Typical acoustic features include Mel-frequency cepstral coefficients (MFCCs), filterbank energies (Fbanks) are widely employed as the input of LID systems [4,17,25]. The two-stage models such as i-vector and x-vector dominate in the LID task. Recently, although the end-to-end approaches are more preferred due to their structural advantage and suitability for the extensive training resources, the i-vector and x-vector methods are still more commonly applied baselines in LID evaluations compared to the end-to-end approaches.

2.1.1.1 GMM-UBM i-vector method

The i-vector approach was first proposed for speaker verification in [6], and next transferred to LID in [4]. This approach encapsulates speech variability including language identity in a vector which is of hundreds of dimensions in general. A universal background model (UBM), which is conventionally a Gaussian mixture model (GMM) of C mixtures, is first trained via the expectation maximization (EM) algorithm on a large amount of training speech data. We can then represent an input speech sample as an FC-dimensional vector $\mathbf{M}$ via

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{w} + \epsilon, \tag{2.1}$$

where $\mathbf{m}$ is the concatenation of F-dimensional mean vectors of all C Gaussian components in the UBM, $\mathbf{T}$ is the $FC \times W$ total variability matrix that is trained through the EM algorithm to encode language and channel information. The variable ϵ denotes the residual noises following a distribution $\mathcal{N}(0, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the $FC \times FC$ residual variability matrix that captures the information that is not encoded in $\mathbf{T}$. The i-vector $\mathbf{w}$ is W-dimensional random vector following a standard normal distribution $\mathcal{N}(0, I)$.

Given a speech sample and its acoustic features $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F | t = 1, ..., T)$, where T is the number of frames and F denotes the frame-level feature dimension. The corresponding i-vector $\mathbf{w}$ can be defined by its posterior distribution conditioned

to the Baum-Welch statistics that are computed as

$$\mathbf{N}_c = \sum_{t=1}^{T} P\left(c|\mathbf{x}_t, \Omega\right), \tag{2.2}$$

$$\widetilde{\mathbf{F}}_c = \sum_{t=1}^{T} P\left(c|\mathbf{x}_t, \Omega\right)\left(\mathbf{x}_t - \mathbf{m}_c\right). \tag{2.3}$$

Here $\mathbf{N}_c$ and $\widetilde{\mathbf{F}}_c$ are the zero and centered first order Baum-Welch statistics of the speech sample, $\mathbf{m}_c$ is the mean vector of the c-th Gaussian mixture component, and Ω denotes the UBM. The i-vector is then computed via

$$\mathbf{w} = \left(I + \mathbf{T}^t\boldsymbol{\Sigma}^{-1}\mathbf{N}\left(u\right)\mathbf{T}\right)^{-1}\mathbf{T}^t\boldsymbol{\Sigma}^{-1}\widetilde{\mathbf{F}}\left(u\right). \tag{2.4}$$

Here, the $\mathbf{N}\left(u\right)$ is defined as a diagonal matrix whose diagonal blocks are $\mathbf{N}_c I$, $\widetilde{\mathbf{F}}\left(u\right)$ is a vector achieved by concatenating $\widetilde{\mathbf{F}}_c$.

The extracted i-vectors are next fed into a language classifier. The language classifier can either be scoring methods transferred from speaker verification such as probabilistic linear discriminant analysis (PLDA) and cosine distance scoring, or logistic regression (LR), Gaussian classifier, and support vector machine (SVM). In addition, pre-processing is generally applied to the i-vectors before feeding them into a classifier. The pre-processing procedures consist of whitening, length normalization, and linear discriminant analysis (LDA). Early research found that although the i-vectors are assumed to have a standard normal distribution, they may exhibit non-Gaussian characteristics [26]. Therefore, whitening and length normalization are used to make the i-vectors obey the Gaussian distribution. The LDA is applied to reduce the feature dimensions. It can also maximize the between-class variability while minimizing the intra-class variability. These processing procedures can empirically improve the LID performance.

In addition, since the DNN has shown to be effective in speech signal processing tasks, [27, 28] proposed to replace UBM with a DNN pre-trained on the ASR task to derive the Baum-Welch statistics using the posteriors of the DNN. Apart from language-related information, the ASR-DNN model can learn phoneme-aware information. The DNN-based i-vector approach thus achieves higher LID performance compared to the GMM-UBM i-vector approach.

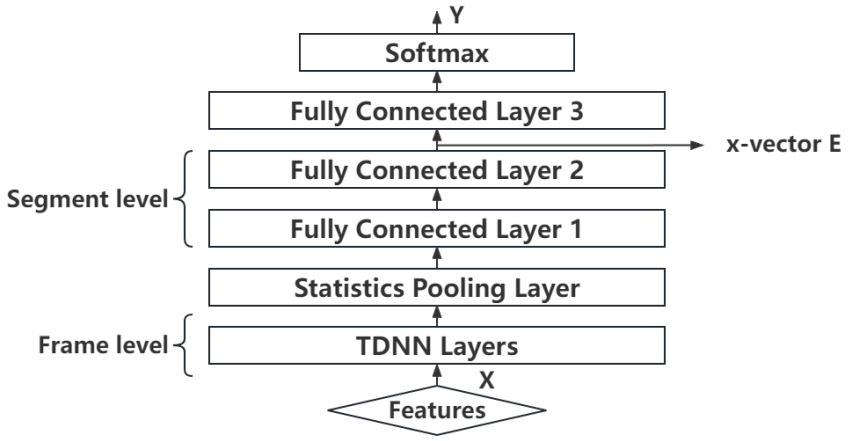


FIGURE 2.1: TDNN-based x-vector model.

Notwithstanding the above, plenty of attempts to develop DNN-based LID approaches have been made due to their success in the ASR and SID tasks. One typical model is the TDNN-based x-vector model that is also the most commonly used baseline model in recent years.

2.1.1.2 TDNN-based x-vector method

The TDNN-based x-vector was originally developed for speaker recognition in [8], and was adapted to LID in [5]. An x-vector model comprises three modules: a frame-level language encoding module, a statistics pooling layer, and an utterance-level representation module. As shown in Figure 2.1, the frame-level language encoder consists of five TDNN layers and the utterance-level representation module comprises three fully-connected layers with ReLU activation.

Given a speech sample with the corresponding acoustic features $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F | t = 1, ..., T)$, the output features of the TDNN layers are given by $\mathbf{H} = (\mathbf{h}_t \in \mathbb{R}^D | t = 1, ..., T')$, where D is the number of output frames. Output features $\mathbf{H}$ are subsequently fed into the statistics pooling layers to compute an utterance-level representation vector via

$$\mu = \frac{1}{T'} \sum_{t=1}^{T'} \mathbf{h}_t, \tag{2.5}$$

$$\sigma = \sqrt{\frac{1}{T'} \sum_{t=1}^{T'} (\mathbf{h}_t - \mu)^2}, \tag{2.6}$$

where μ and σ are the mean and standard deviation vectors. The output vector of the statistics pooling layer is achieved by concatenating μ and σ, and is of dimensions $2D$. The output dimension of this model is L that is the number of target languages. During training, the model is optimized via a cross-entropy loss as

$$Loss^{CE} = -\sum_{n}^{B}\sum_{l}^{L} d_{nl} \log P\left(\text{lang}_l|\mathbf{Y}\right), \tag{2.7}$$

where B is the batch size, d_{nl} is 1 if the prediction corresponds to the language label, otherwise it is 0. Here $\mathbf{Y}$ is the output of the model, while lang_l is a one-hot language label. The x-vector is then defined as the output of the first FC layer.

During inference, the extracted x-vectors are used similarly to i-vectors. Whitening, length normalization and dimension reduction are applied to the x-vectors prior to the inference to improve the LID performance. Moreover, the TDNN x-vector model can achieve LID directly using the model output. In [5], the authors illustrated the comparison between direct LID with the embedding-based two-stage process. Apart from the higher LID performance, the embedding-based method is more flexible since it allows expanding to new languages without retraining the model.

Inspired by the success of the x-vector approach, models built upon TDNN were proposed and achieved high performance for speaker and language recognition. The extended x-vector approach employs a slightly wider temporal context in the TDNN layers and interleaves dense layers between the TDNN layers [29, 30]. This architecture shows high performance in the VOiCES 2019 challenge [31, 32]. The ECAPA-TDNN-based x-vector approach improves the TDNN layer by incorporating squeeze-and-excitation block [33, 34]. The ECAPA-TDNN approach outperformed the extended-x-vector method and further improved state-of-the-art speaker verification performance.

Compared to the i-vector approach, the x-vector method exhibits its superiority in terms of LID performance especially when extensive training data are available, while it shows higher performance improvement on short utterances than on long utterances under the same training condition. In addition, these two approaches appear to be complementary when being fused in the LID task [5], and such fusion has been broadly employed when building LID systems [10].

2.1.1.3 End-to-end acoustic LID approaches

Although the end-to-end approaches are not as applicable as embedding-based approaches in the open-set LID task, effective end-to-end approaches with various architectures have shown state-of-the-art LID performance for close-set LID. In addition, the two-stage LID method comprises separate language encoder, classifier, and processes prior to feeding language embeddings into classifier being complicated in the model-building process. In contrast, end-to-end approaches simplify this process by integrating language encoder and classifier in a neural network. Typical end-to-end LID models are structured similarly to the x-vector model comprising language encoder, classifier modules, and a pooling layer between them to generate utterance-level language representation. It is useful to note that the x-vector approach can be also employed in an end-to-end manner. However, as reported in [35], the end-to-end x-vector method has shown lower performance than its two-stage counterpart in speaker and language recognition when there are limited indomain data available for training. Therefore, it is not discussed in this section.

Early works utilized a single convolutional neural network (CNN) or recurrent neural network (RNN) to encode language identities. [36] proposed a CNN-based LID model, where the input is a $T \times F$-dimensional feature map $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F | t = 1, ..., T)$. The output of the CNN layers is of shape $CH \times W \times H$, CH denotes the number of channels, while W and H denote the width and height of the output feature map. The output feature map is shrunk to 1×1 after convolution and pooling layers and is projected to L dimensions, where L is the number of target languages. In [37], a stacking LSTM-based LID model was proposed to encode language information in the input features $\mathbf{X}$. The attention mechanism is then applied to compute utterance-level representation using the hidden states and the corresponding language category embedding $\mathbf{l}_k$ in a look-up table as

$$s_t = \text{Score}\left(\mathbf{l}_k, \mathbf{h}_t\right), \tag{2.8}$$

$$a_t = \frac{\exp\left(s_t\right)}{\sum_{i=1}^{T} \exp\left(s_i\right)}, \tag{2.9}$$

$$\mathbf{E} = \sum_{t=1}^{T} a_t \mathbf{h}_t. \tag{2.10}$$

Here, s_t is a score regarding t-th hidden state computed by Score $(\cdot, \cdot)$, where the score can be either cosine distance or the linear projection by a learnable matrix. a_t is the attention weight for $\mathbf{h}_t$, and $\mathbf{E}$ denotes the utterance-level representation and is fed into the classification module to derive the posteriors $\mathbf{Y}$.

[17] introduced a CNN-LSTM-based LID model to incorporate the advantages of CNN and LSTM, where the CNN module is employed to capture local information in $\mathbf{X}$ followed by stacking Bidirectional LSTM layers that encode the global dependencies. In addition, the authors proposed to replace the conventional pooling layer with a self-attentive pooling layer. The self-attentive pooling layer adopts the same attention mechanism but replaces the respective language category embeddings in (2.8) with a learnable vector, while the hidden state $\mathbf{h}_t$ is the linear projection of the output of BLSTM layers followed by a tanh activation. The utterance representation $\mathbf{E}$ is finally utilized to compute the posteriors.

Apart from the stats pooling and SAP layers, other pooling strategies have also been investigated in recent research. Inspired by the GMM supervector, a widely applied learnable dictionary encoding (LDE) layer was proposed in [38]. Given $\mathbf{X}$ and the output of the language encoding module being $\mathbf{H} = (\mathbf{h}_t \in \mathbb{R}^D | t = 1, ..., T')$, $\mathbf{H}$ is then fed into the LDE layer to compute the utterance representation.

The LDE layer performs the aggregation of frame-level features like GMM supervector. Assuming the LDE layer comprises C dictionary components with their learnable center vectors $\mathbf{U} = (\mathbf{u}_c \in \mathbb{R}^D | t = 1, ..., C)$, the weight $w_{t,c}$ associated with the t-th feature $\mathbf{h}_t$ and c-th center vector $\mathbf{u}_c$ is computed as

$$w_{t,c} = \frac{\exp\left(-s_c \left\|\mathbf{x}_t - \mathbf{u}_c\right\|^2\right)}{\sum_{i=1}^{C} \exp\left(-s_i \left\|\mathbf{x}_t - \mathbf{u}_i\right\|^2\right)}, \tag{2.11}$$

where s_c is a learnable smoothing factor. Similar to the GMM supervector, the utterance representation $\mathbf{E}$ output by the LDE layer is a concatenation of C sub-representations with each sub-representation $\mathbf{e}_c$ being

$$\mathbf{e}_c = \frac{\sum_{t=1}^{T'} w_{t,c} \cdot (\mathbf{x}_t - \mathbf{u}_c)}{T'}. \tag{2.12}$$

The utterance embedding $\mathbf{E}$ is next used to perform the language classification.

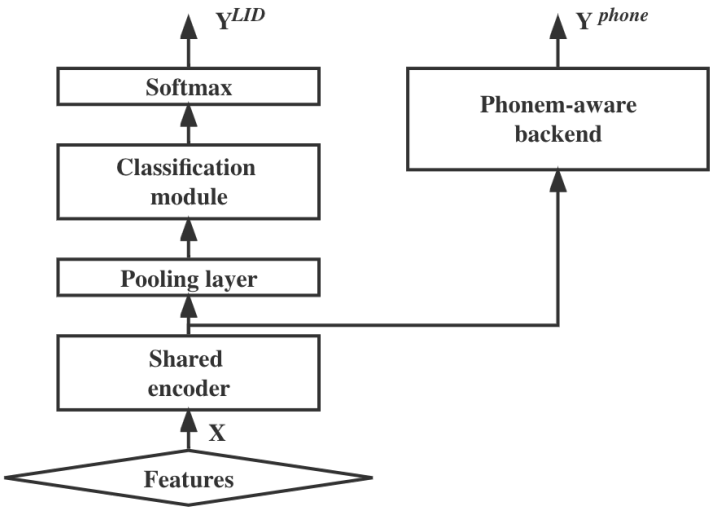


FIGURE 2.2: End-to-end phoneme-aware LID model.

In addition, although different languages may share the same phonemes, each language comprises a unique set of phonemes. Therefore, a LID model can be enhanced by utilizing phonemic information. Such an acoustic-phonetic LID system is trained on acoustic features and incorporates phoneme information by jointly optimizing the LID task and a phoneme-related task such as automatic speech recognition and phoneme classification [39, 40].

As shown in Fig. 2.2, in existing phoneme-aware LID methods, the LID and phoneme-related tasks share the same language encoding module in which the language identities and phoneme information are learned during training. As a result, the phoneme-related branch requires phoneme annotations or transcriptions. During the inference phase, only $\mathbf{Y}^{LID}$ is used to generate language decisions.

Although acoustic features have shown to be effective in the LID task and appropriate to NN-based approaches, the acoustic feature extraction primarily focuses on low-frequency components leading to information loss. Therefore, acoustic features are usually applied in conjunction with features of various granularities to achieve high LID performance [41].

2.1.2 Phonotactic features and approaches

Phonotactics involves the permissible combinations of phonemes in languages. As opposed to an acoustic-phonetic unit that is shorter than a phoneme, a phonotactic unit can cover several phonemes. It is also useful to note that while phonemes can be shared across languages, statistics of their sequential patterns differ from one language to another [1].

Conventional phonotactic LID methods comprise a phone recognition module followed by the language modeling (LM) or vector space modeling (VSM) for target languages [15]. The phone recognizer module can either comprise a universal or several language-specific models. When using LM as its back-end, the former is defined as phone recognition language modeling (PRLM) while the latter is parallel phone recognition language modeling (PPRLM). For a PPRVSM LID system, the phone n-gram statistics form a high-dimensional bag-of-sounds (BOS) vector. A back-end classifier such as SVM then performs LID on the concatenation of those BOS vectors for each utterance.

A PPRLM LID model is shown in Fig. 2.3 as an example to illustrate the phonotactic LID approach. Historically, the phone recognizer is, in general, a hidden Markov model (HMM) trained on multilingual speech data [15]. The language-specific phone n-gram LM works similarly to a word n-gram LM but dedicates to phone sequences instead of word sequences. Each language model captures sequential patterns of the recognized phones followed by scoring for its corresponding target language. The scores computed by each branch are next fused, and the language with the highest score is defined as the language decision.

Since conventional phonotactic LID approaches are primarily hybrid, early research focused on optimizing either the phone recognizer or language model. The authors of [42] suggested that the phone recognizer is the most crucial component in a phonotactic LID system and proposed to replace the HMM with a neural network to improve the LID performance. In [43], the authors proposed to select the most informative phones when discriminating each target language from the phone repository of a pre-trained phone recognizer, each set of selected phones is then used to construct a target-oriented phone tokenizer for its corresponding language. Two criteria were studied to perform the phone selection. The first criterion is the contribution of a phone to the separation margin for a one-versus-rest SVM

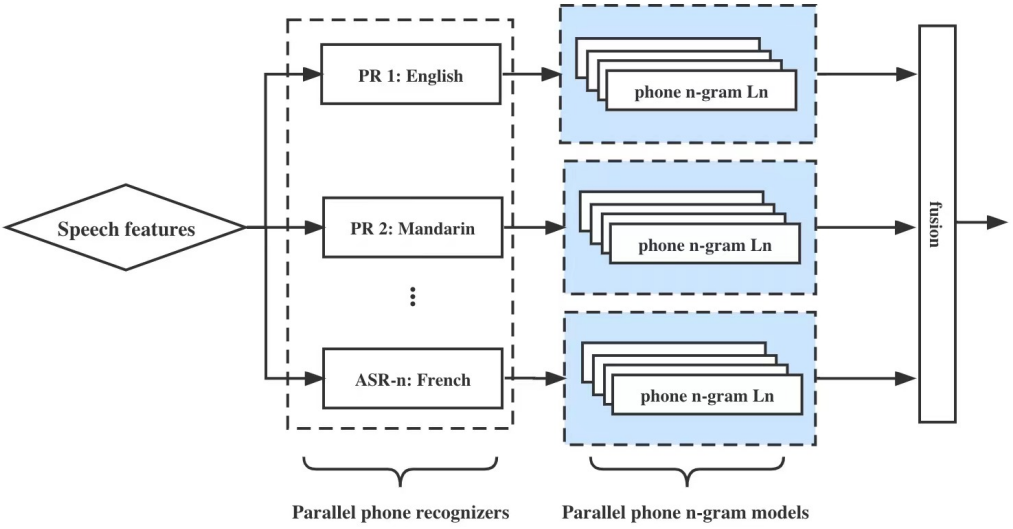


FIGURE 2.3: Parallel phone recognition language modeling model.

classifier, where one corresponds to a target language, while the second is the mutual information of each phone regarding each language. To further enhance the LID performance using the target-aware language information, the authors subsequently proposed to replace the hard decision on selecting phones with a soft decision in [18]. The soft decision is achieved by reflecting the discriminative ability of phones to LID in the training of the target language model instead of the phone recognizers.

In addition to the development of the phone recognizers, recent phonotactic LID methods adopted the merits of neural networks and further investigated NN-based back-ends. In [44], an RNNLM was applied to perform language modeling after the phone recognition. The authors of [45] introduced a novel PPRVSM system, where the n-gram units are achieved by WordPiece tokenizer and Byte-Pair encoding (BPE) [46, 47]. These units are vectorized after a trainable embedding layer and serve as phonotactic representations. These representations are next fed into a transformer encoder followed by a mean pooling and several linear layers to generate language decision. This process is similar to NN-based acoustic LID.

Notwithstanding the above, a phonotactic LID system faces the same challenge as the phoneme-aware LID method—it requires phoneme annotations of speech samples during training. However, the process of annotation is usually time-consuming and requires significant resources and domain expertise. In contrast, acoustic approaches require only digitized speech samples and their language labels; they

have since gained popularity over recent years. In addition, the phonotactic LID approaches, in general, exhibit high LID performance on long speech but perform poorly on short speech due to the need for sufficient phoneme statistics.

2.1.3 Prosodic features and approaches

Prosody is generally defined as suprasegmental features in running speech. The features include duration, pitch, fundamental frequency (F0), and intensity [1, 48], and are believed to be effective in distinguishing languages of different broad classes (i.e., Mandarin and English). In [49], the duration and pitch information are encoded in their corresponding GMM models followed by scoring for C target languages. The score vectors of C dimensions representing the target languages are then used to train a back-end classifier. The authors of [50] proposed a prosodic attribute model (PAM), where the unigram and bigram-level prosodic attributes are computed as the BOS followed by language classification. In addition, [51] investigated the LID performance using prosodic features extracted over different orders of n-gram units, and suggested that the long-range prosodic system can achieve higher LID performance compared to the short-range prosodic system.

However, experiments in the literature suggested that an individual prosodic LID model may not achieve high LID performance. This is because the prosodic features are of low granularity compared with acoustic features. Fusing prosodic LID models with other systems achieves negligible performance improvement. In addition, reliable prosodic feature extraction is still challenging. Consequently, since prosodic LID approach is not as effective as acoustic and phonotactic approaches, this approach is not further discussed in this thesis.

2.1.4 Lexical approaches

The lexical LID approaches are based on the knowledge that each language has its distinctive phonological and syntactic systems. In contrast to phonotactics that rules the permissible phoneme sequences with phone n-gram modeling, syntax denotes the way that words and phrases are structured to form sentences. Since a syntactic system works similarly to a word n-gram modeling [52], lexical approach is naturally considered for LID.

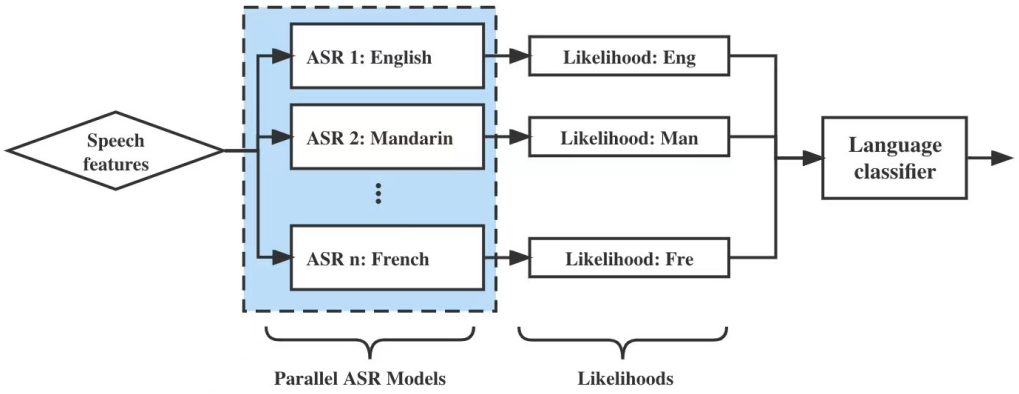


FIGURE 2.4: Parallel large vocabulary continuous speech recognition model.

As shown in Fig. 2.4, one typical lexical LID approach is to employ parallel large vocabulary continuous speech recognition (LVCSR) models, each of which corresponds to a target language [53, 54]. Acoustic features are fed into the parallel LVCSR models before generating language-specific likelihoods. The language corresponding to the model that gives the highest likelihood is considered the decision.

Nevertheless, an LID model is generally employed as a front-end in multilingual speech processing systems. Therefore, once a multilingual LVCSR system comprising parallel monolingual LVCSR models is trained and discriminative to languages, the LID task is redundant. In addition, training an ASR model leads to higher complexity and cost compared with training a LID model. Consequently, the lexical LID approach has not been widely adopted.

2.1.5 DNN-based speech representation

Apart from conventional acoustic features, DNN-based speech representations have been proven to be effective in speech signal processing tasks [55–57]. They are usually of higher dimensions compared to acoustic features being applied similarly to the latter.

2.1.5.1 Bottleneck features

Early research proposed to extract speech representations from a hidden layer (i.e., the penultimate layer) of a DNN model. This DNN model can be pre-trained on

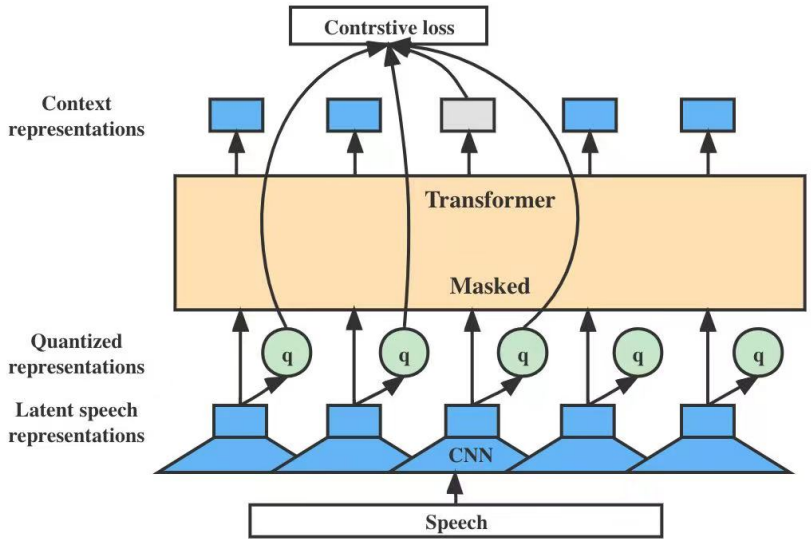


FIGURE 2.5: Wav2vec 2.0 framework which jointly learns contextualized speech representations and an inventory of discretized speech units.

either the ASR or phone recognition task using acoustic features [58, 59]. Since the hidden layer is usually of lower dimensions compared to other hidden layers, it is defined as the bottleneck layer and the extracted representations are bottleneck features (BNFs).

While BNFs have been widely applied in the LID task [5], this approach requires speech transcriptions or phoneme annotations during the fully-supervised training. Therefore, self-supervised learning was proposed to learn speech representations from unlabeled speech data [19, 24, 56, 60].

2.1.5.2 Self-supervised learning of speech representations

An example of self-supervised learning of speech representations is shown in Fig. 2.5, where the wav2vec 2.0 model input is a raw waveform that is first digitized before being normalized to zero mean and unit variance. The wav2vec 2.0 model then employs a convolutional neural network serving as a feature encoder to the raw waveform. The feature encoder generates latent speech representations. A proportion of these latent representations are masked before being fed into the context network. The context network consists of transformer encoder layers and aims to build contextualized representations. The entire model is trained via a contrastive task, where the true latent representations are distinguished from distractors which

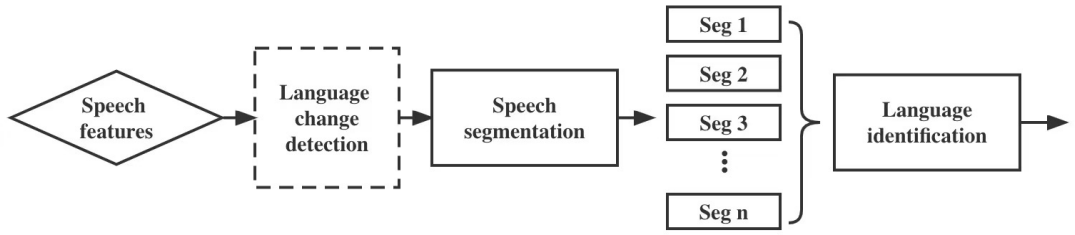


FIGURE 2.6: The workflow of a language diarization model.

are quantized representations of other time steps [24]. Compared to the conventional acoustic features, the wav2vec 2.0 speech representations embed the audio information in their high-dimensional vectors—they suffer from lower information loss and, therefore, are more effective for downstream tasks.

To apply SSL features to the LID tasks, the authors of [61] explored wav2vec 2.0 features on speaker and language identification by employing an average pooling layer and a linear layer on top of the wav2vec 2.0 model. In [57], the authors also adapted the wav2vec 2.0 model to the LID task by adding a pooling layer and a linear layer on top of the wav2vec 2.0 model. As opposed to the former work, this paper investigated the LID performance with no pre-training, pre-training on monolingual speech data, and pre-training on multilingual speech data. The results suggest that the features extracted from the XLSR model, which is a wav2vec 2.0 model pre-trained on multilingual data, show significantly higher performance compared with those extracted from other models.

Although DNN-based speech representations have achieved great success in the LID task, the pre-training of DNN models requires a large amount of data resources for high LID performance. With respect to the SSL features, all audio information is encoded in the high-dimensional vectors. While this reduces the information loss, the encoded irrelevant information (i.e., speaker identities) to LID may limit the LID performance.

2.2 Spoken language diarization systems

The term diarization was originally used to describe the task of determining audio segments associated with the same speaker in a multi-speaker recording and subsequently extended to cover language diarization, a special case of LID. While LID is

under the assumption that only one language exists in a given speech sample, language diarization is proposed to identify the languages and the corresponding time stamps in a more challenging scenario where languages alternate (i.e., code-switch) in a multilingual speech sample.

As shown in Fig. 2.6, the LD process comprises language change detection and segmentation before language identification, where the context dependencies of the speech segments are taken into consideration. Therefore, it is natural to develop an LD approach upon an LID. [62] evaluated acoustic features, phonetic features, and prosodic features for LD. Phonetic features are extracted using an LVCSR model before being fed into a LID system in conjunction with acoustic and prosodic cues. The LID system comprises a quantized acoustic model, a duration model, and a syllabic language model corresponding to acoustic, prosodic, and phonetic features, respectively. The authors subsequently proposed an LD model that is a fusion of phonetic and phonotactic models [63]. The phonetic model comprises a phone recognizer, a GMM-based classifier, and an SVM classifier. The phone segments are identified by the phone recognizer before being fed into a language-specific GMM to generate log likelihoods corresponding to each target language. The likelihoods are then combined with those of the adjacent segments being used to generate segment-level language decisions via an SVM. The phonotactic model employs a phone recognizer to generate phone segments before feeding their statistics in conjunction with the contextual statistics into the conditional random fields.

An SVM-based approach was proposed in [64] to detect language changes. Here, frame-level features of the given speech signal are separated into positive and negative samples. Positive samples are the frames around the code-switch points while others are negative samples. The SVM is optimized to discriminate between positive and negative samples during training. In the inference phase, the frames identified to be positive are defined as where the code-switches happen.

In addition, LD approaches can be adapted from speaker diarization methods. Traditional speaker diarization methods, which aim to segment the speech signal and group together segments belonging to the same speaker, are based on a speaker encoding front-end, such as x-vectors [8], and a clustering back-end model [65, 66]. These are inherently unsupervised methods, in the sense that they neither require nor leverage examples of segmented-and-labeled multi-speaker recordings. End-to-end speaker diarization methods, by contrast, require (and learn from)

labeled multi-speaker audio to jointly train an integrated neural module for speaker encoding and clustering [67].

The authors of [68] proposed an end-to-end BLSTM speaker diarization configuration. The model comprises BLSTM and linear layers. Given a speech sample and the corresponding features $\mathbf{X}$, the first two BLSTM layers encoders speaker identities in the hidden outputs before feeding them into the following three BLSTM and linear layers to output speaker labels $\mathbf{Y} = (\mathbf{y}_t | t = 1, ..., T)$, where $\mathbf{y}_t = [y_{t,c} \in \{0, 1\} | c = 1, 2, ..., C]$ and C denotes the number of speakers. Deep clustering loss is applied to encourage these hidden outputs to be speaker-discriminative representations [69]. Since the predicted speakers do not correspond to any fixed class, this model can meet permutation problem. The permutation-invariant training loss is thus utilized to achieve the permutation which has the minimum error compared with the ground-truth speaker labels. This method transforms the diarization task to a multi-class classification problem and is thus easy to be adapted for LD.

Although traditional methods exhibited their efficacy, a few drawbacks limit their performance for language diarization. The fusion model proposed in [56] requires phoneme annotations during training and a phone recognizer during inference. However, the process of annotation for large data is usually time-consuming, requires significant resources and domain expertise. The SVM-based code-switch detection approach performs identification at the frame level which is not desirable. In addition, although speaker diarization methods could be adapted to language diarization, the speaker diarization methods perform clustering but not identification after speech segmentation. As opposed to the clustering in speaker diarization, the identification process in language diarization can be challenging due to diverse accents. Therefore, a direct adaptation of the speaker diarization approach to language diarization may not show high performance.

2.3 Corpora

In this section, the datasets, which are commonly used in language identification and diarization literature, are described.

TABLE 2.1: NIST LRE 17 target languages and language clusters

Language cluster	**Target language**	**Language code**
Arabic	Egyptian Arabic, Iraqi Arabic, Levantine Arabic, Maghrebi Arabic	ara-arz, ara-acm, ara-apc, ara-ary
Chinese	Mandarin, Min Nan	zho-cmn, zho-nan
English	British English, General American English	eng-gbr, eng-usg
Slavic	Polish, Russian	qsl-pol, qsl-rus
Iberian	Caribbean Spanish, European Spanish, Latin American Continental Spanish, Brazilian Portuguese	spa-car, spa-eur, spa-lac, por-brz

2.3.1 Language identification

2.3.1.1 NIST 2017 Language recognition evaluation

The National Institute for Standards in Technology (NIST) has organized plenty of language recognition evaluations (NIST LRE) and released the corresponding NIST LRE data after these competitions officially ended. These datasets are usually applied for evaluating the performance of LID systems.

Since the NIST LRE data is the most commonly used dataset in the language identification task, the baseline and proposed LID models in this thesis are evaluated on the NIST LRE 2017 data. The NIST LRE 2017 data consist of the Fisher corpus [70], Switchboard corpora [71], a narrow-band (8 KHz) training set (TRN17), a development set (DEV17), and an evaluation set (EVAL17) [9, 72]. TRN17 is built from previous LRE data with over 2000 hours of audio data including 16,205 recordings, where 13,956 are conversational telephone speech (CTS) and 2,249 are broadcast narrow band speech (BNBS) recordings. The DEV17 and EVAL17 comprise narrow-band Multi-language Speech (MLS14) data containing and wide-band (16 KHz) Video Annotation for Speech Technologies (VAST) data, where MLS14 data consist of 3 s, 10 s, and 30 s duration levels and the VAST data comprise segments with speech duration ranging from 10 to 600 s. There are 1,999 CTS and 788 BNBS recordings belonging to MLS14 and 874 AfV recordings corresponding to VAST in DEV17, while the EVAL17 contains 15,018 CTS, 2,002 BNBS, and 3,521 AfV recordings.

TABLE 2.2: Target languages and the corresponding country and language code in OLR challenge

Country	**Target language**	**Language code**
China	Cantonese, Kazakh, Mandarin, Tibetan, Uyghur	ct-cn, ka-cn, zh-cn, ti-cn, uy-id
Indonesia	Indonesian	id-id
Japan	Japanese	ja-jp
Korea	Korean	ko-kr
Russia	Russian	ru-ru
Vietnam	Vietnamese	vi-vn

As shown in Table. 2.1, there are fourteen languages from five language clusters in total within the dataset. The NIST LRE 2017 data are challenging for LID due to significantly unbalanced data in terms of duration resulting in different LID performance on languages.

2.3.1.2 Oriental Language Recognition challenge

The Oriental Language Recognition (OLR) challenge was organized to meet the request for multilingual research on oriental languages. As shown in Table. 2.2, there are ten languages exist in the OLR data from six countries. AP17-OLR comprises AP16-OL7, AP16-OL3, AP17-OL3, THCHS30, and AP17-OLR-test [73]. The AP16-OL7 database consists of AP16-OL7-train/dev and AP16-OL7-test, both of which contain ct-cn, zh-cn, id-id, ja-jp, ru-ru, ko-kr and vi-vn with approximately 9,000 utterances in total for each language. AP17-OL3 includes train and dev sets containing ka-cn, ti-cn, and uy-id with their respective number of utterances being 4,200, 11,100, and 5,800. AP17-OLR-test contains all ten languages with 1,800 utterances for each language.

Compared with AP17-OLR, AP19-OLR consists of additional AP18-OLR-test and AP19-OLR-test sets [74]. The AP18-OLR-test was recorded under a similar condition to AP17-OLR-test and also contained 1,800 utterances for each of ten languages. To evaluate LID systems in different scenarios, AP19-OLR-test was split into three tasks: short-utterance LID, cross-channel LID, and zero-resource LID

TABLE 2.3: Detailed duration and language information of SEAME dataset

	Speakers	**Hours**	**Duration Ratio**		
			Man	**Eng**	**CS**
train	134	101.13	16%	16%	68%
dev_{man}	10	7.49	14%	7%	79%
dev_{sge}	10	3.93	6%	41%	53%

with their respective sets being AP19-OLR-short, AP19-OLR-channel, and AP19-OLR-zero, each of which comprises ten languages with 1,800 utterances for each language. It is useful to note that all OLR speech data except for AP19-OLR-channel were recorded by mobile phones, with a 16 KHz sampling rate and a 16 bits sample size.

As opposed to NIST LRE data, languages used in the OLR challenge are from geographically close countries. Therefore, these languages may influence each other leading to complex development patterns in terms of phonetics and linguistics. In addition, each language has around 7500 recordings of around 10 hours. Therefore, there are only 100 hours of audio in this dataset. These make the LID task on oriental languages fairly challenging.

2.3.2 Language diarization

2.3.2.1 SEAME dataset

SEAME is a Mandarin-English code-switching speech corpus in south-east Asia [75]. The speech data were collected from Singapore and Malaysia for spontaneous code-switching speech comprising interviews and conversations without scripts. Each domain consists of both monolingual and code-switching speech including intra-sentential and inter-sentential code-switches. The recording was conducted in a quiet office room by close-talk microphones with a sampling rate of 16 KHz and a sample size of 16 bits.

The SEAME dataset comprises 290 recordings consisting of 110,145 utterances out of which 24,438 are Mandarin, 28,655 are English, and 57,052 are code-switching utterances. It is officially split into train, dev_{man}, and dev_{sge} sets. Table 2.3 shows

the detailed duration and language information of this division [76], where Man, Eng, and CS denote Mandarin, English, and code-switching, respectively. dev_{man} and dev_{sge} are defined as the test sets corresponding to Mandarin speech and Southeast Asian accent English, respectively. Each test set has ten speakers with balanced genders. In addition, although sentence-level time stamps were annotated with corresponding language labels ZH, EN, and CS denoting Mandarin, English, and code-switching, respectively, the time stamps regarding where intra-sentence code-switches happen were not provided. Since the speech data in SEAME are recorded in Southeast Asia, the natural accent and code switches lead to language confusion. This dataset is thus challenging for speech recognition and language diarization.

2.3.2.2 WSTCSMC 2020

The First Workshop on Speech Technologies for Code-switching in Multilingual Communities (WSTCSMC 2020) was held by Microsoft [77]. The released speech data comprise phrasal and conversational speech containing Tamil-English, Telugu-English, and Gujarati-English language pairs with a sampling rate of 44.1 KHz and a sample size of 16 bits. Two shared tasks were released to evaluate the submitted LID systems, where the shared task A is the utterance-level identification of monolingual and code-switched utterances and the shared task B is the frame-level identification of language in a code-switched utterance.

Language diarization is performed on the shared task B in this workshop. The detailed information regarding duration is provided via Table. 2.4, where gu-en, ta-en, and te-en correspond to Gujarati-English, Tamil-English, and Telugu-English language pairs, respectively. A language label including four languages and silence is given for each 200 ms speech unit.

TABLE 2.4: Duration (hours) of each language in the shared task B in WSTC-SMC 2020

Language pair	**Training**				**Development**			
	Num. Record.	gu/ta/te	en	silence	Num. Record.	gu/ta/te	en	silence
gu-en	8982	10.6	2.1	3.0	1135	1.3	0.3	0.4
ta-en	8226	10.9	1.8	3.1	1047	1.4	0.2	0.4
te-en	8620	10.9	1.8	3.0	1080	1.4	0.2	0.4

2.4 Performance evaluation

2.4.1 Performance measure for language identification

In this thesis, the LID systems are evaluated by employing accuracy (Acc.), equal error rate (EER) and the average cost (C_{avg}) in accordance to the LRE 2007 evaluation plan [78]. There exists two types of errors in the LID task. For Q languages in an LID task, each language can be seen as a target language L_T and the other $L_N = Q - 1$ languages are non-target languages denoted as L_N. The first type of error is defined as the miss error and occurs when the model misclassifies the target language as non-target. The second type of error is known as false alarm when the model misclassifies an impostor (non-target language) as the target. Pair-wise recognition performance is computed for all target/non-target language pairs by employing the miss and false alarm (FA) rates, and these quantities are combined into a single cost performance as

$$\begin{aligned} C\left(L_T, L_N\right) =& \mathrm{C}_{\text{miss}} \cdot \mathrm{P}_{\text{Target}} \cdot P_{miss}\left(L_T\right) + \\ & \mathrm{C}_{\text{FA}} \cdot \left(1 - \mathrm{P}_{\text{Target}}\right) \cdot P_{FA}\left(L_T, L_N\right), \end{aligned} \tag{2.13}$$

where $\mathrm{C}_{\text{miss}} = \mathrm{C}_{\text{FA}} = 1$ and $P_{Target} = 0.5$ are predefined parameters, $P_{miss}\left(L_T\right)$ denotes the miss rate of L_T, and $P_{FA}\left(L_T, L_N\right)$ denotes the false alarm rate given L_T and L_N. The average cost is finally computed via

$$\begin{aligned} C_{avg} =& \frac{1}{\mathrm{Q}} \sum_{L_T} \{\mathrm{C}_{\text{miss}} \cdot \mathrm{P}_{\text{Target}} \cdot P_{miss}\left(L_T\right) + \\ & \sum_{L_N} \mathrm{C}_{\text{FA}} \cdot \left(1 - \mathrm{P}_{\text{Target}}\right) \cdot P_{FA}\left(L_T, L_N\right)\}. \end{aligned} \tag{2.14}$$

In addition, EER is defined as the threshold at which the false alarm rate P_{FA} and miss rate P_{miss} are equal. Accuracy in this work denotes the ratio between the number of correctly classified utterances and the total number of test utterances.

2.4.2 Performance measure for language diarization

Two metrics accuracy and EER are employed to evaluate language diarization models in this thesis [77]. As opposed to those applied for LID evaluation, accuracy and EER are computed in segment-level. In addition, EER is defined as

$$EER = \frac{P_{miss}\left(L_T\right) + P_{FA}\left(L_T, L_N\right)}{2}, \tag{2.15}$$

where $P_{miss}\left(L_T\right)$ and $P_{FA}\left(L_T, L_N\right)$ denote segment-level miss and false alarm rates, respectively. Each segment corresponds to 200 ms speech without overlap in this thesis.

2.5 Summary

The literature on language identification and diarization are reviewed in this chapter. The LID approaches were illustrated in a knowledge abstraction level order including the x-vector model which serves as the baseline model in this thesis. Existing LD methods were described together with typical speaker diarization models. The corpora and metrics used to evaluate LID and LD models were finally introduced.

Part I

Improving Duration Robustness for Language Identification

Chapter 3

Dual-mode X-vector Self-Attention LID Model

This chapter firsts proposes the x-vector self-attention (XSA-LID) model for LID and enhance its performance on various-duration speech by employing a dual-mode framework with knowledge distillation (KD).

In Section 3.1, the existing short-utterance LID approaches and their drawbacks regarding the duration robustness are discussed. The streaming ASR task are also explained in this section. Section 3.2 describes a dual-mode model proposed for streaming ASR which inspires this dual-mode LID work. Section 3.3 and Section 3.4 present the XSA-LID and the dual-model LID models, respectively. The data, model configuration, and detailed experiments are described in Section 3.5. Results and ablation studies are discussed in Section 3.6, and conclusion of this work is presented in Section 3.8.

3.1 Introduction

Recent LID methods have shown high LID performance on long utterances. However, practical LID systems are expected to achieve high performance on both short and long utterances. Several DNN-based approaches have been proposed to achieve short-utterance LID since they have shown to be more effective than conventional i-vector approaches for short utterances [11, 12]. For instance, the authors of [79] proposed to train a convolution deep neural network (CDNN)-based mechanism on short utterance, i.e., 3 s speech . A bidirectional long short-term memory (BLSTM) network has also been proposed to model the temporal dependencies between the past and future frames in short utterances [80]. In addition, Peng et al. proposed several methods based on knowledge distillation (KD) or compensation on x-vector to transfer the knowledge of a pre-trained long-utterance LID model to a short-utterance LID model [81–83]. These approaches, however, focus primarily on short utterances and are not developed to achieve good performance on long utterances. Practical applications of the LID system, however, should cater to speech of varying duration. Moreover, some existing methods require an additional data preprocessing step such as speech segmentation and a pre-trained model or additional parameters. These components are necessary to facilitate the distillation of the knowledge from the model trained on long utterances to the target short-utterance LID model [81–83].

Aside from LID, streaming automatic speech recognition (ASR) aims to generate each hypothesized word as quickly and accurately as possible without the availability of future frames [84, 85]. In [85], a unified dual-mode ASR model has been proposed to improve the streaming ASR with full-context modelling, in which the streaming ASR model shares the same parameters with the full-context ASR model and future information is masked in the streaming mode.

Challenges posed by the availability of only a portion of speech in streaming ASR is similar to that of short-utterance LID. Inspired by the success of the dual-mode ASR system in streaming ASR, this chapter proposes an x-vector self-attention (XSA-LID) model for LID and employs a dual-mode framework and KD to enhance its performance on speech of various duration [21]. Apart from the performance enhancement in LID, the proposed dual-mode LID approach possesses several desirable properties. Firstly, the proposed method uses the KD method

without introducing an additional model or parameters. Secondly, instead of pre-processing using speech segmentation, a Boolean mask is applied to acquire features more flexibly with lower computational complexity. In addition, different from the use of mask in dual-mode ASR which aims to remove the future frames from the streaming mode, the masking operation in the proposed model mimics short speech clips. Various types of linguistic information introduced by the mask serves as data augmentation and leads to higher LID performance. Lastly, since the Boolean mask is applied to the transformer encoder layers in the proposed model, the proposed approach can easily be transferred.

3.2 Related work

In the dual-mode ASR system [85], the full-context ASR model applies a Boolean mask to mimic the streaming ASR model. These models share the same weights and are jointly trained with connectionist temporal classification (CTC) and KD losses. In contrast to the dual-mode ASR model which focuses more on the streaming ASR task, this work aims to improve the LID performance on both long and short utterances. Therefore, the dual-mode framework is adapted to the LID task by modifying the mask to flexibly acquire linguistic information. In addition, considering the success of KD in both short-utterance LID and streaming ASR [82, 85], the KD method is adopted to improve the LID performance on short utterances.

3.3 X-vector self-attention LID model

As depicted in Fig. 3.2, $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F | t = 1, ..., T)$ is defined as input of the XSA-LID model. As opposed to the original x-vector model proposed for language identification that comprises five TDNN layers, the third and fifth layers are removed for the TDNN module in the proposed XSA-LID model. This is because the speech features $\mathbf{X}$ are partitioned into short segments $\mathbf{S}_n = (\mathbf{x}_{k,n} \in \mathbb{R}^F | n = 1, ..., T/K; k = 1, ..., K)$ comprising K frames, where $\mathbf{x}_{k,n}$ is the speech representation at the k-th time step in segment n. Each $\mathbf{S}_n$ has less context frames compared to $\mathbf{X}$. The x-vector embedding module, which consists of three TDNN layers before

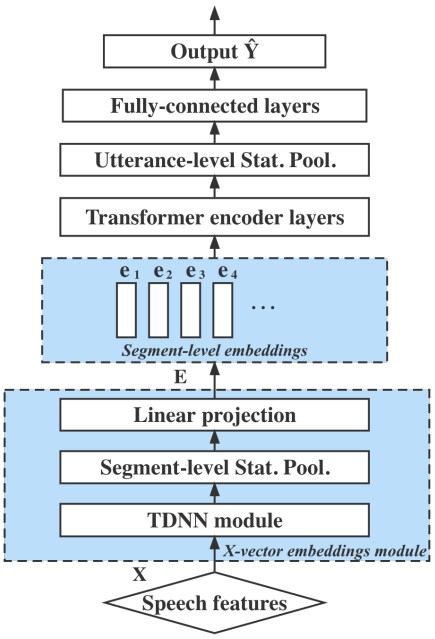


FIGURE 3.1: X-vector Self-Attention LID (XSA-LID) model.

a segment-level statistics pooling layer, is adopted to encode the local information in each segment in a high-dimensional vector $\mathbf{e}_n$ via

$$\mathbf{H}_n = \text{TDNN}(\mathbf{S}_n), \tag{3.1}$$

$$\mathbf{e}_n = \mathbf{W}_{\text{Proj}}\text{Concat}(\text{Mean}(\mathbf{H}_n), \text{Std}(\mathbf{H}_n)), \tag{3.2}$$

where $\mathbf{H}_n = (\mathbf{h}_k \in \mathbb{R}^{D/2} | k = 1, ..., K')$ denotes the TDNN output and K' is the number of output frames, $\text{TDNN}(\cdot)$ is defined as computations within TDNN layers, $\mathbf{W}_{Proj}$ is a linear projection, respectively. The operations $\text{Mean}(\cdot)$ and $\text{Std}(\cdot)$ correspond to mean and standard deviation of inputs, respectively. It is useful to note that the statistics pooling layer computes the mean and standard deviation vectors of its input before concatenating these values to form a new vector as the output. It was concluded that both mean and standard deviation pooling outperform max pooling in speaker identification and verification [86]. Since speaker identification is similar to language identification in terms of the NN-based model configuration, training, and evaluation, the statistics pooling layer is expected to achieve higher LID performance than the max pooling layer.

The resultant segment-level embeddings $\mathbf{E} = (\mathbf{e}_n \in \mathbb{R}^D | n = 1, ..., T/K)$ are subsequently fed into the self-attention encoder module. Since the transformer encoder

layers have shown to be effective in capturing global dependencies, they are employed to learn the global dependencies among these segment-level features. The input of each multi-head attention encoder block of transformer encoder layer is $\mathbf{E}_{j-1}$, where j is the index of the encoder block. The input $\mathbf{E}_0$ of the first encoder layer is transformed from $\mathbf{E}$ by the positional encoding [23] and the layer normalization [87]. Each head of an encoder block j computes query vectors, key vectors and value vectors matrix using dot products $\mathbf{E}_{j-1}\mathbf{W}_j^{(Q,i)}$, $\mathbf{E}_{j-1}\mathbf{W}_j^{(K,i)}$ and $\mathbf{E}_{j-1}\mathbf{W}_j^{(V,i)}$, respectively. The attention weight matrix and the output of head i, and the output of the multi-head self-attention layer are then computed via

$$\mathbf{Q}_j^i = \mathbf{E}_{j-1}\mathbf{W}_j^{Q,i}, \tag{3.3}$$

$$\mathbf{K}_j^i = \mathbf{E}_{j-1}\mathbf{W}_j^{K,i}, \tag{3.4}$$

$$\mathbf{V}_j^i = \mathbf{E}_{j-1}\mathbf{W}_j^{V,i}, \tag{3.5}$$

$$\mathbf{Head}_j^i = \text{Softmax}\left(\frac{\mathbf{Q}_j^i\left(\mathbf{K}_j^i\right)^\intercal}{\sqrt{D}}\right)\mathbf{V}_j^i, \tag{3.6}$$

$$\mathbf{A}_j = \text{Concat}\left(\mathbf{Head}_j^1, \cdots, \mathbf{Head}_j^H\right), \tag{3.7}$$

where $\mathbf{Q}_j^i$, $\mathbf{K}_j^i$, and $\mathbf{V}_j^i$ are $\mathbb{R}^{T\times D}$ matrices, Softmax $(\cdot)$ is the softmax function, Concat $(\cdot)$ is a concatenation operation, $\mathbf{head}_j^i$ denotes the output of the i-th head, $\mathbf{A}_j$ is defined as the output of the multi-head self-attention layer, and H is the number of self-attention heads. The output of the j-th encoder block is subsequently computed by projecting the concatenated outputs of all heads into a $\mathbb{R}^{T\times D}$ matrix. A residual connection and layer normalization is then applied to outputs of the multi-head attention and the position-wise feed-forward layers. This process is computed by

$$\mathbf{Pff}_j^{\text{in}} = \text{LayerNorm}\left(\mathbf{A}_j + \mathbf{E}_{j-1}\right), \tag{3.8}$$

$$\mathbf{Z}_j = \text{LayerNorm}\left(\mathbf{Pff}_j^{\text{in}} + \mathbf{Pff}_j^{\text{out}}\right), \tag{3.9}$$

where $\mathbf{Pff}_j^{\text{in}}$ and $\mathbf{Pff}_j^{\text{out}}$ are the input and output of j-th position-wise feed forward layer, $\mathbf{Z}_j$ is the output of j-th encoder block, and LayerNorm $(\cdot)$ denotes the layer normalization.

In addition, during implementation, given a batch of training samples, the utterances (which are shorter than the longest one in a batch) are padded to be of the equal length so that they can be fed into the model. These padded components

in the matrix corresponding to the "False" in **Mask** are set to a significantly low value (e.g., -10^9) such that they do not contribute to the subsequent computations after the softmax function. The self-attention computations mentioned above can then be described via

$$\mathbf{Z}_J = \text{SelfAtten}\left(\mathbf{E}, \mathbf{Mask}\right), \tag{3.10}$$

where $\text{SelfAtten}\left(\cdot, \cdot\right)$ denotes computations performed within the self-attention encoder module, and **Mask** is defined as the attention mask applied to the attention weight matrix, i.e., the dot product between the query and key matrices [23]. A statistics pooling layer is subsequently employed to generate an utterance-level representation for the input speech signal. This is achieved by aggregating the segment-level output $\mathbf{Z}_J$ of the encoder module. The following fully-connected layers generate class scores $\widehat{\mathbf{Y}}$ of the target languages through

$$\widehat{\mathbf{Y}} = \mathbf{W}_3\delta\left(\mathbf{W}_2\delta\left(\mathbf{W}_1\left(\text{Concat}\left(\text{Mean}\left(\mathbf{Z}_J\right), \text{Std}\left(\mathbf{Z}_J\right)\right)\right)\right)\right), \tag{3.11}$$

where J is the total number of encoder blocks applied in this model, and $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ denote three linear layers, δ is the ReLU activation function. Details of the configuration will be described in Section 3.5.

3.4 Dual-mode language identification

3.4.1 Dual-mode framework

To enhance the performance of the XSA-LID model, the dual-mode architecture, which was originally proposed for streaming ASR, is next adopted. With reference to Fig. 3.2, during training, the full mode corresponds to the XSA-LID model with the input full-length speech being $\mathbf{X}$. On the other hand, since features corresponding to the mimicked short speech clip of the full-length speech are extracted from $\mathbf{X}$ via a Boolean mask applied in (3.10), the input of the short mode is also $\mathbf{X}$. Here, the input of the short mode is equivalently given by $\mathbf{X}_S = (\mathbf{x}_t \in \mathbb{R}^{K\times F} | t = s, ..., s + T_S - 1)$, where $T_S < T$ is the number of segments in the speech clip. These two modes are optimized jointly in this proposed dual-mode LID model. The full mode, therefore, achieves general LID, while the

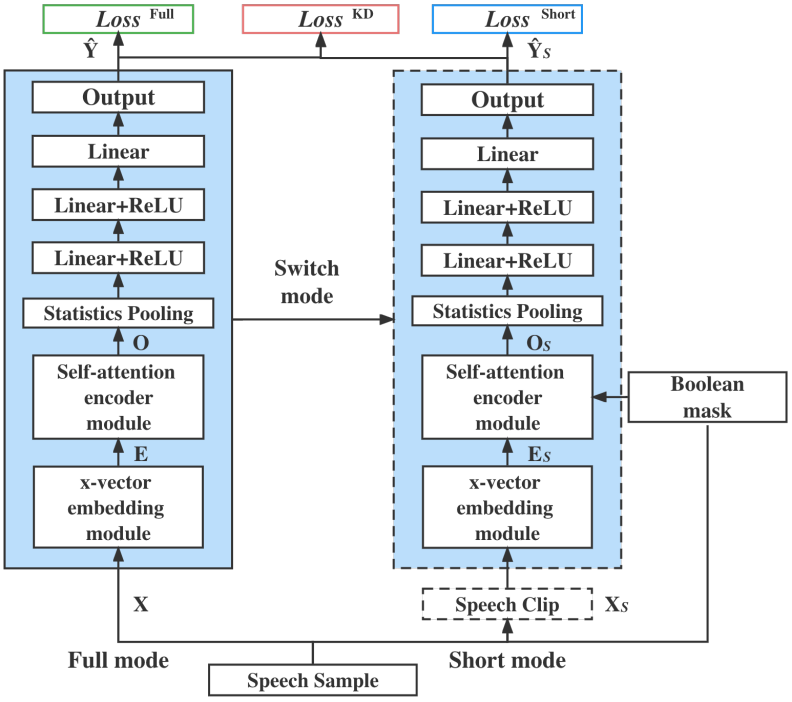


FIGURE 3.2: Dual-mode LID system with XSA-LID model and knowledge distillation. The speech clip within the dotted box is mimicked from the complete speech sample via the use of the Boolean mask. No speech segmentation is needed during training.

short mode focuses on the local information in the speech clips. Hence, the dual-mode method can achieve good LID performance for both full-length speech and short speech.

To achieve good system performance for the LID task without the use of additional dataset, one possible choice is to train the system with the data augmentation using speech clips segmented from the full-length speech. While the model trained with such augmentation can achieve good performance on short utterances, having larger number of short clips in the updates generally results in performance degradation on long utterances. As opposed to data augmentation using short speech clips, features of the mimicked short speech clips in the short mode of the proposed model are being represented by the full-length speech in the full mode in the current update. The short mode better grasps a portion of local information of the full-length speech and the utterance-level embedding is generated by aggregating the segment-level statistics in the statistics pooling layer. Therefore, the enhanced ability to capture local language information results in further

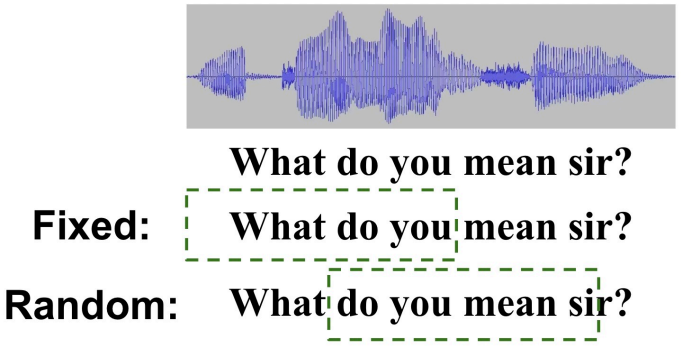


FIGURE 3.3: Two types of Boolean mask for acquiring linguistic information. The text bounded by the green dotted box corresponds to the mimicked speech clip.

performance improvement on long utterance compared to short utterances. Consequently, by minimizing the sum of the losses of these two modes, the proposed model is able to achieve significant performance improvement on long utterances with a modest improvement on short utterances.

3.4.2 Knowledge distillation loss

To improve the short-utterance LID performance on the dual-mode framework, the KD loss is employed to minimize the difference between prediction probability distributions of the full-length speech and the corresponding short clip. The KD loss possesses two important characteristics for LID. Firstly, compared to the CE loss in the dual-mode framework, the KD loss utilizes the output of the full mode which incorporates valuable information pertaining to non-target languages, e.g., dialects. This allows the short mode to learn rich discriminative information from the full mode. Secondly, due to the minimization of the difference between outputs of the two modes, the introduction of KD loss to the optimization may reduce the LID performance for long utterances. However, this performance degradation for long utterances can be offset by the improvement resulted by the dual-mode framework which is optimized via CE losses. With the above, the proposed model can achieve higher performance on speech of varying duration due to the trade-off between KD loss and CE losses.

It is useful to note that two independent utterances can differ from each other in terms of language-unrelated identities such as speaker and lexical information. These identities, however, affect the LID performance. The proposed approach

applies the KD loss to the full-length speech and its short clip. Since they share the same speaker identities and lexical information, the effect of duration-unrelated information is intrinsically reduced when computing the KD loss. Thus, the KD loss which focuses on the difference of time duration is computed via [21]

$$P(\mathbf{X}) = \text{Softmax}\left(\frac{\mathbf{Y}}{\mathbf{Temp}}\right), \tag{3.12}$$

$$L^{\text{KD}} = \text{KL}\left(\log\left(P(\mathbf{X})\right), P(\mathbf{X}_S)\right), \tag{3.13}$$

where $\mathbf{Y}$ denotes the system outputs and **Temp** is the distillation temperature. The softmax operation is defined as $\text{Softmax}(\cdot)$, $\text{KL}(\cdot, \cdot)$ denotes the KL divergence, and $P(\cdot)$ denotes the output probability for language classes given input speech features. During training, the proposed model is optimized by minimizing the weighted sum of KD loss and the CE losses of the full-mode and short-mode. This loss is given by

$$L^{\text{Dual}} = \alpha L^{\text{Full}} + \beta L^{\text{Short}} + (1 - \alpha - \beta) L^{\text{KD}}, \tag{3.14}$$

where α and β are parameters to compensate for any trade-off in performance between short and long utterances. In particular, to achieve a balanced performance on long and short utterances, α, β and $(1 - \alpha - \beta)$ are set to be equal. Since the performance improvement on short utterances is mainly attributed to the KD loss, higher β and $(1 - \alpha - \beta)$ are adopted in the presence of frequently occurring short utterances in the target data. Conversely, higher α and β will be required to achieve high LID performance for long utterances.

3.4.2.1 The Boolean mask for mimicking short speech clips

To mimic the short speech clip and acquire various clip-wise linguistic information, with reference to (3.10), the Boolean mask is utilized. The Boolean mask was originally applied to the transformer encoder layers of the XSA-LID model during training to handle input utterances of varying durations. In contrast to the Boolean mask in the dual-mode ASR which masks future frames, two types of masking strategies are proposed for short-utterance LID: 1) the fixed-location mask through which features of the first several seconds speech of the full-length speech are extracted and 2) the random-location mask through which features of

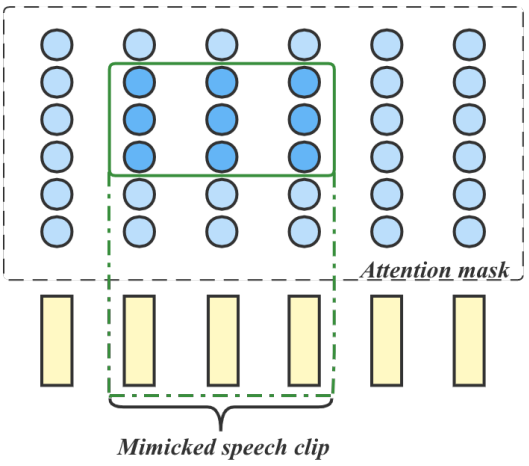


FIGURE 3.4: The random-location Boolean mask for the self-attention encoder module. In this figure, the mimicked short speech clip starts from the second segment to the fourth segment. The shallow blue circles are filled with significantly small values before the softmax operation when computing the attention weight matrix.

a specified window duration from a random time step of the full-length speech are extracted.

To illustrate the above, the utterance "What do you mean sir" shown in Fig. 3.3 is considered. The text bounded by the green dotted box corresponds to the mimicked speech clip. Since it is more likely that the random-position masking cuts off the voiced speech compared to the fixed-position masking, the speech clip "What do you" mimicked by the fixed-location mask contains more clip-wise lexical integrity than the clip "do you mean si" mimicked by the random-location mask. Nonetheless, the random mask can derive different short clips for each utterance over the training epochs, while the fixed mask always locates in the first several seconds. Therefore, the random mask is expected to introduce richer clip-wise linguistic variability for training, while the fixed mask provides better clip-wise lexical integrity to the mimicked short clips. In addition, the model with a random-location mask is expected to achieve higher performance on short utterances since richer short-duration information is exploited for the training, while the model trained with better clip-wise lexical integrity in the fixed-location mask may exhibit higher improvement for long utterances.

The random-location Boolean mask is illustrated in Fig. 3.4, where components circled by the green bounding box correspond to the mimicked short clip. The

Algorithm: Pseudocode of the dual-mode language identification

```
# Load a batch of inputs speech features x and language labels y
for x, y in data_loader:
    # full-mode: inputs are full-length x
    with dual_mode_model.mode("full"):
        # compute full-length prediction given x and y
        full_pred = dual_mode_model.forward(x, y, atten_mask)
        # compute cross-entropy loss for full-mode
        loss_full = CE(full_pred, y)
    # short-mode: inputs are clips of x, one clip for each sample in x
    with dual_mode_model.mode("short"):
        # built a Boolean mask give the length of the speech clip
        short_mode_mask = attention_mask(speech_clip_length)
        # compute short-utterance prediction given x, y and the mask
        short_pred = dual_mode_model.forward(x, y, short_mode_mask)
        # compute cross-entropy loss for short-mode
        loss_short = CE(short_pred, y)
    # compute the KD loss, teacher is the full-mode
    loss_KD = knowledge-distillation-loss(full_pred, short_pred)
    # compute the dual-mode loss and optimize the model
    loss_dual_mode = α loss_full + β loss_short + (1 − α − β) loss_KD
    loss.backward()
```

mimicked short speech clip starts from the second segment to the fourth segment. The shallow blue circles are filled with significantly small values before the softmax operation when computing the attention weight matrix. Considering that the x-vector module in the XSA-LID model represents segments as vectors, no modification was made to the TDNN layers. The Boolean mask is then employed in the self-attention computation. By replacing the **Mask** in (3.10) in the full mode with this Boolean mask, segment-level embeddings which are unrelated to the speech clip are filtered out—the remaining segment-level embeddings therefore correspond to the mimicked speech clip. Optimization with the Boolean mask is shown in the pseudocode of the dual-mode LID.

3.5 Dataset, Experiment, and Model Configuration

3.5.1 Dataset and feature extraction

To ensure fair evaluation, the same features have been used for all systems. 80-dimensional bottleneck features (BNFs) are extracted from an ASR-DNN model that is trained on the Fisher corpus and Switchboard corpora [70, 71]. The input

features of the ASR-DNN model are 13-dimensional Mel frequency cepstral coefficients (MFCCs) extracted from a 25 ms window with a 10 ms shift. Silent frames are removed using an energy-based voice activity detector. I trained all systems on TRN17 and DEV17 and tested them on the MLS14 data in EVAL17 to compare their performance on the speech of different duration levels. To reduce the length of frame sequences for the transformer and conformer encoders, every 20-frame BNFs are concatenated into a new unit and the 1600-dimensional BNFs are subsequently fed into these two models during both training and inference. Feature extraction is performed using the Kaldi toolkit [88].

3.5.2 Model configuration

In terms of the configuration of systems in Table 3.1, the x-vector approach follows that presented in [5] and is trained by modifying the SRE16 recipe in the Kaldi toolkit with a back-end logistic regression classifier. The transformer and conformer models refer to the encoder blocks presented in [23, 89]. These models consist of four self-attention encoder layers followed by a statistics pooling layer and three linear layers with ReLU activation in the first two linear layers.

As shown in Fig. 3.2, the XSA-LID model comprises an x-vector embedding module followed by the self-attention encoder module, an utterance-level statistic pooling layer and three linear layers with ReLU activation in the first two linear layers. The x-vector embedding module consists of three TDNN layers followed by a statistics pooling layer and a linear layer. The input dimension F=80 corresponds to the dimension of the BNFs. The TDNN layers are conv1d layers with kernel size $(5, 5, 1)$, dilation $(1, 2, 1)$, and output dimensions $(512, 512, 512)$. These layers generate an embedding for each segment of duration twenty frames with each segment being approximately 200 ms. The linear layer projects the 1024-dimensional output of the segment-level statistics pooling layer to 64 dimensions. Two transformer encoder layers in the self-attention encoder module follow [23], where each encoder layer has eight attention heads with $d_{model} = 512$ and $d_{ff} = 2048$. The utterance-level statistics pooling layer then generates the 1024-dimensional output which is finally projected by three linear layers to the number of target languages. The output dimensions of these three linear layers are $(512, 512, 14)$.

TABLE 3.1: Results of different models on the MLS14 data in NIST LRE 2017 by employing Accuracy (%), EER (%) and C_{avg} and ablation study

Method	**3 s**			**10 s**			**30 s**			**Overall Avg.**		
	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓
X-vector	-	**11.79**	**0.1159**	-	7.81	0.0748	-	6.84	0.0654	-	8.81	0.0854
Transformer	44.47	19.66	0.1998	69.60	9.00	0.0792	79.58	5.81	0.0487	64.55	11.49	0.1123
Conformer	49.29	16.84	0.1627	73.30	7.76	0.0684	82.57	4.83	0.0393	68.39	9.81	0.0901
XSA-LID	54.18	15.47	0.1685	74.90	7.39	0.0739	84.09	4.46	0.0406	71.06	9.11	0.0943
XSA-Aug	**58.28**	13.23	0.1308	75.28	7.32	0.0674	81.05	5.29	0.0470	71.53	8.61	0.0817
DualNoKD	55.35	15.94	0.1648	77.17	6.51	0.0641	86.13	**3.88**	**0.0370**	72.88	8.78	0.0886
Dual-LID-Fix	57.00	14.30	0.1420	78.97	6.46	0.0595	**87.02**	4.00	0.0372	74.33	8.25	0.0796
Dual-LID-Ran	58.25	13.82	0.1361	**80.08**	**6.25**	**0.0580**	86.65	4.09	0.0372	**74.99**	**8.05**	**0.0771**

The proposed dual-mode LID model is based on the XSA-LID model. Variables $\alpha = \beta = 0.33$ in (3.14) were chosen to assign nearly equal importance to the two modes and KD loss. A 3 s Boolean mask is applied in this work, which covers 15 segments. The distillation temperature is set to 2. These NN-based models are all trained for 20 epochs using the Adam optimizer with an initial learning rate of 10^{-4} and cosine annealing learning rate decay after 24000 warm-up steps. A batch size of 32 is use for the transformer, conformer and the XSA-LID models, and 16 for the proposed dual-mode LID model.

3.6 Results

3.6.1 Results of different LID systems and ablation study

3.6.1.1 Results of different LID systems

Results of different LID systems evaluated on the MLS14 data in EVAL17 are shown in Table 3.1 across three different test speech durations. An ablation study pertaining to the proposed dual-mode LID system is also presented. Since the number of test speech across three durations are equal, their results are averaged to provide an indication of the overall performance.

Compared to other approaches, the x-vector model exhibits the highest performance on 3 s speech but the lowest performance on 30 s utterances. It is not surprising that the transformer encoder, conformer encoder, and XSA-LID models show higher performance on 30 s test speech against the TDNN-based x-vector model since the transformer-based models have been proven to be successful in modeling long-term dependency. The proposed XSA-LID model achieves the highest performance among these three models. This suggests that the ability to capture local information provided by the x-vector embedding module helps improve the LID performance. In addition, due to the superior performance of the x-vector model on 3 s speech, although the XSA-LID model achieves higher performance on 10 s and 30 s test speech, the x-vector model shows higher overall performance.

With regards to the proposed Dual-mode LID method, the models with 3 s speech clips mimicked by random-location and fixed-location masks are denoted as Dual-LID-Ran and Dual-LID-Fix, respectively. The Dual-LID-Ran model achieves 19.23%, 21.52%, and 8.37% relative improvement in C_{avg} on 3 s, 10 s, and 30 s speech, respectively, compared with the XSA-LID model, and the best overall performance among all systems in terms of metrics under consideration.

3.6.1.2 Ablation study

To analyze the effects of the short clips, dual-mode structure and KD loss, ablation studies are performed using the results of XSA-Aug and DualNoKD systems. The XSA-Aug is the XSA-LID model which is trained with data augmentation by the first 3 s speech clips of utterances in TRN17 and DEV17. Therefore, XSA-Aug utilizes the same data as the Dual-LID-Fix without the dual-mode framework nor the KD loss. We note from Table 3.1 that XSA-Aug exhibits higher performance on 3 s and 10 s speech than the XSA-LID model. However, although the XSA-Aug system achieves higher performance on 3 s speech performance than the Dual-LID-Fix, its performance on longer speech, especially for 30 s speech, is lower. This indicates that the augmentation by short speech clips, when enhancing the performance on 3 s and 10 s utterances, degrades the performance on 30 s speech. In addition, these results show that the improvement of LID performance is not attributed only to the data but also by the proposed dual-mode LID method.

TABLE 3.2: Comparison of the performance of dual-mode LID systems with different Boolean masks by employing Accuracy (%), EER (%) and C_{avg}

Mask		**3 s**			**10 s**			**30 s**		
Location	**Length**	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓
Random	1 s	56.89	14.36	0.1399	79.17	6.51	0.0601	86.77	4.24	0.0394
	2 s	57.10	14.13	0.1409	78.63	6.43	0.0596	86.14	4.34	0.0402
	3 s	**58.25**	**13.82**	**0.1361**	80.08	**6.25**	0.0580	86.65	4.09	0.0372
	4 s	57.46	14.17	0.1376	79.25	6.38	0.0573	87.22	4.34	0.0369
	5 s	57.95	14.20	0.1422	**80.11**	6.28	0.0575	**87.24**	4.24	**0.0362**
	6 s	57.84	14.15	0.1391	79.51	**6.25**	**0.0557**	87.03	4.19	0.0376
Fixed	3 s	57.00	14.30	0.1420	78.97	6.46	0.0595	87.02	**4.00**	0.0372

The importance of the dual-mode framework and KD loss is next verified using the DualNoKD model—the same model as the Dual-LID-Ran model without KD loss. Compared to the XSA-LID model, the DualNoKD system achieves 2.2%, 13.26% and 8.87% relative improvement in C_{avg} on 3 s, 10 s and 30 s test data, respectively. This is consistent with my assumption described in Section 3.4 that the dual-mode framework can achieve higher performance improvement on long utterances than short utterances.

We also note from the above results that, after applying the KD loss to the Dual-NoKD system—the Dual-LID-Ran model achieves 17.42%, 9.52% relative improvement in C_{avg} on the 3 s and 10 s speech utterances, respectively, albeit suffering from modest performance degradation on 30 s test speech compared to the Dual-NoKD model. This accords with my supposed characteristics of the KD loss being applied to the dual-mode framework.

3.6.2 Results of the LID models with different Boolean masks

3.6.2.1 Results and analysis of the mask lengths

This next experiment investigates next investigate how short speech clips extracted by different Boolean masks influence the performance of the proposed method via experiments conducted on the MLS14 data in EVAL17. The dual-mode LID models

with random-location Boolean masks of different lengths are first evaluated. From Table 3.2, it is not surprising that the model with the 3 s mask achieves the highest performance on 3 s test speech utterances, and the model with a longer mask achieves, in general, higher performance on 10 s and 30 s testing speech. However, I observed that the 6 s random mask suffers from lower performance on the 30 s speech than that of the 3 s speech. This may indicate that as the mimicked clips get longer, the short mode tends to pay more attention to the contextual dependencies than the local information, the enhancement resulted by better local information thus decreases. When the speech clips are as long as the full-length speech, the dual-mode LID system performs the worst, similar to the XSA-LID model. Consequently, the comparison between masks of different lengths implies that the proposed approach exhibits robustness to various speech durations without restricting to a specific Boolean mask length.

Notwithstanding the above, it is useful to note that an appropriate mask length can optimally improve the overall performance. A mask with a longer length exhibits higher performance on long speech compared to that with a short length in Table 3.2. Therefore, it is possible to achieve high average performance against various durations by utilizing the diverse lengths of short clips during training.

3.6.2.2 Results and analysis of the mask location

Apart from the window length, the position where the Boolean mask is located may also influence the performance of the dual-mode LID model. In Table 3.2, the label "Fixed" denotes the Boolean mask extracting features of the speech in the first several seconds of the full-length speech signal while the label "Random" denotes features of the speech clips being extracted by the mask from a random position of each utterance.

With reference to Table 3.2, compared to the 3 s fixed-location Boolean mask, the 3 s random-location mask exhibits higher performance on 3 s and 10 s test speech, while the former achieves modestly higher performance on 30 s test speech. These results are expected and conform with my proposition highlighted in Section 3.4.2.1. The speech clip extracted by the random-location Boolean mask can vary for each utterance in every epoch during training. Hence, it provides more clip-wise linguistic variability than the first 3 s speech clip. This clip-wise linguistic

variability serves as data augmentation and leads to higher performance on short speech.

On the other hand, these results suggest that the clip-wise lexical integrity plays an important role in reducing the difference between full-length speech and its short clip. Since the KD loss quantifies the difference between two distributions, a lower difference implies a lower KD loss contribution during training. Therefore, the dual-model LID model with the fixed-location Boolean mask suffers from lower performance degradation than that with the random-location mask, whereas it achieves lower performance on short utterances.

3.7 Discussion

Considering that the data may not be balanced across different duration levels, the LID model should cater to the target data. This work shows that the LID performance on long and short utterances can be affected by several factors. Therefore, this work can be adjusted to accommodate the target data once the statistic such as average duration is known.

To promote the performance improvement on long speech, as described in Section 3.4.2, a higher α and β can be adopted in (3.14) during training. In addition, an appropriate length of the Boolean mask is also proven to be effective in boosting the LID performance on long utterances. It is useful to note that although the fixed-location Boolean mask can also help the model achieve higher performance in long-utterance LID, the random-location mask is recommended due to its ability to achieve higher overall performance.

To improve the model performance for short-utterance LID task, a higher weight can be assigned to the KD loss in (3.14). A short Boolean mask is also helpful to achieve higher performance on short speech. It is also useful to note that the x-vector embedding module can be replaced with a more recent architecture like ECAPA-TDNN to improve language identification performance.

3.8 Summary

This work proposed the XSA-LID model and employed a dual-mode framework with knowledge distillation to enhance the LID performance on various-duration speech. Experiments conducted showed that the proposed dual-mode LID model achieves the highest overall performance on the MLS14 set of NIST LRE 2017 data with the random-location Boolean mask. Contributions of the mimicked speech clips, the dual-mode framework, and the KD loss to the performance improvement are evaluated. The influence of different Boolean masks in terms of the mask length and location is also discussed. Experiment results show that the dual-mode framework and fixed-location achieve higher performance improvement on long utterances, while the performance on short utterances is attributed more to the KD loss and random-location mask.

Part II

Enhancing and Incorporating Language Cues for Language Identification

Chapter 4

Efficient Deep Neural Network-based Representations for LID

This chapter investigates efficient methods to compute reliable representations and discard redundant information for LID using a pre-trained multilingual wav2vec 2.0 model.

In Section 4.1, conventional language cues and self-supervised learning representations are described. Section 4.2 introduces the work related to this chapter. The XSA-LID model and the proposed SE-based and LBN blocks are illustrated Section 4.3. The datasets, model configurations and detailed experiments are described in Section 4.4. The results and analysis are presented in Section 4.5. The advantages and limitations of the proposed methods are discussed in Section 4.6. Conclusion is consequently presented in Section 4.7.

4.1 Introduction

Early studies in LID suggested that the acoustic and phonotactic features provide effective language cues [14,15]. Acoustic features such as the Mel-frequency cepstral coefficients (MFCCs) and filterbank energies are usually employed for the LID task [4, 17]. Apart from such acoustic features, bottleneck features (BNFs) have

also been proven to be effective for speaker verification and language recognition [5, 90]. To extract the BNFs, a deep neural network (DNN)-based acoustic model for ASR is first trained using acoustic features in a fully-supervised manner. The BNFs of a given utterance are then extracted from one hidden layer of this trained model in place of the acoustic features as the input of the system. While these acoustic features and BNFs have achieved great success in speech signal processing, acoustic features are generally handcrafted with prior knowledge. In addition, they emphasize low-frequency components and information is inevitably lost during computations.

Self-supervised learning, as a paradigm of unsupervised learning, has been applied to learn speech representations from unlabeled speech data [19, 24, 56]. For instance, the wav2vec 2.0 model first employs a convolutional neural network to a speech signal before masking a proportion of the encoder outputs and feeding them into the context network. The context network builds contextualized representations and the entire model is trained via a contrastive task where the true latent representations are distinguished from distractors [24]. Compared to the conventional acoustic features, the wav2vec 2.0 speech representations embed the audio information in their high-dimensional vectors—they suffer from lower information loss and, therefore, is more effective for downstream tasks. Fine-tuning the pre-trained model on the target task by adding a randomly initialized linear projection to the context network of the wav2vec 2.0 model has also been well studied [24, 55]. The fine-tuned model can be then used as a feature extractor with the extracted representations subsequently being used as input to the downstream model [24, 91].

Although the wav2vec 2.0 speech representations achieved successes in various tasks such as ASR and LID [24, 55, 92], compared to the use of conventional acoustic features (e.g., 39-dimensional MFCCs and 80-dimensional fbanks), they are usually high-dimensional vectors and require higher computational complexity. In addition, the wav2vec 2.0 speech representations extracted from different context network blocks exhibit varied performance in different downstream tasks. This may be attributed to different aspects of information such as language and speaker [55, 93]. Since not all information contributes to the LID task, reducing any information unrelated to the LID task in the representations is expected to lead to better performance. Moreover, although the model fine-tuned on the target task can usually achieve higher performance in the downstream tasks than the pre-trained

model [24], fine-tuning on hundreds or even thousands of hours of speech data is both time and computationally intensive.

This chapter presents a new technique to reduce irrelevant information from speech representations and apply them efficiently without fine-tuning for the LID task. The use of speech representations extracted from different context network blocks of the XLSR-53 model is first compared, where the XLSR-53 is built on the wav2vec 2.0 framework and pre-trained on multilingual speech data. Two methods that serve as the front end of the XSA-LID model are then proposed to reduce redundant information from the representations. These two methods process features from different perspectives with the first being brought about by is the attentive squeeze-and-excitation (SE) block. As opposed to the original SE block, the proposed attentive SE block takes the different importance of time steps into consideration and reduces redundant information by dimension-wise scaling. The second perspective is brought about by the use of the linear bottleneck (LBN) block that was originally proposed in [94, 95]. In contrast to [95] that suggested to avoid the information loss due to the reduction of output dimensions within the LBN block, this information loss can be utilized to reduce any irrelevant information to the LID task. Before employing LBN, principal component analysis (PCA) is employed to explore primary information in the compressed vectors and detect the appropriate-dimensional space which can adequately accommodate the LID-related information. The proposed methods possess several desirable properties. Firstly, with an appropriate setup, the LBN block can effectively suppress any information in the speech representations that is irrelevant to the LID task with only a slight increment of the model parameters. Secondly, comparing these methods helps explain how various aspects of information are represented in the features.

4.2 Related work

4.2.1 The cross-lingual XLSR-53 model

The wav2vec 2.0 model is a self-supervised model trained to learn speech representations from unlabeled data. It randomly masks a proportion of the input speech in a latent space and subsequently recovers it via a contrastive loss between the learnable quantizations and the reconstructed version [24]. The model consists of

a convolution neural network-based feature encoder, a quantization module and a context network that follows the transformer architecture [23].

In [96], the speech representations extracted from a pre-trained XLSR-53 model (which is a multilingual version of the wav2vec 2.0 model [56,97]) have shown to be more effective than those extracted from a pre-trained wav2vec 2.0 model. Therefore, in the context of LID, instead of the monolingual English wav2vec 2.0 model trained on speech data, speech representations are extracted from the context network block of the pre-trained XLSR-53 model.

4.2.2 Bottleneck and squeeze-and-excitation blocks

In addition to XLSR-53, the bottleneck block is applied to LID to reduce irrelevant information. The bottleneck block was originally proposed for the ResNet architecture to reduce the number of parameters with no degradation in image classification accuracy [94]. In MobileNetV2 [95], the LBN block was proposed to prevent nonlinearities from losing information. The LBN block comprises two or more fully-connected layers that first shrink the feature map of the input before expanding it back to the original dimensions. Specifically, the ReLU activation function collapses some channels when zero values are generated for negative inputs. This process, therefore, leads to information loss unless the input manifold lies in a lower-dimensional space than the input space. To mitigate this problem, the $H \times W \times C$ feature matrix $\mathbf{X}$ is expanded to $H \times W \times qC$ by the first layer in LBN, where q is the expansion factor. However, since this work aims to reduce instead of preserving the information, the above property of ReLU activation is utilized to modify the LBN block to adapt it to this work.

For the second model, the SE block is applied for dimension-wise scaling. The SE block was originally proposed for image classification [33] and can be regarded as a channel-wise self-attention. In the squeeze layer, the $H \times W \times C$ feature matrix $\mathbf{X}$ is first embedded into a $1 \times 1 \times C$ dimensional vector by the global average pooling over its spatial $H \times W$ dimensions. The channels are then compressed into C/r followed by ReLU activation. The compressed vector is finally projected back to C dimensions and the channel-wise weights are computed using a sigmoid function. The output is achieved by scaling the channels of the input feature with

the computed channel-wise weight vector via

$$\mathbf{m} = \text{AvgPool}\left(\mathbf{X}\right), \tag{4.1}$$

$$\mathbf{s} = \sigma\Big(\mathbf{W}_2\delta\big(\mathbf{W}_1\mathbf{m}\big)\Big), \tag{4.2}$$

$$\widetilde{\mathbf{x}}_t = \text{Scale}\left(\mathbf{s}, \mathbf{x}_t\right), \tag{4.3}$$

where $\mathbf{m}$ is the mean vector after the global average pooling AvgPool (), $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{r}}$ denote the first and second linear layers, respectively, r is the reduction rate, σ is defined as the sigmoid function, $\mathbf{s}$ is the self-attention weight vector, $\widetilde{\mathbf{x}}_t$ denotes the representation at time step t after scaling, and Scale $(\cdot, \cdot)$ denotes the element-wise product.

4.3 Methodology

4.3.1 XSA-LID model

Given a speech signal, $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F | t = 1, ..., T)$ is defined as the speech representations extracted from the context network block of the pre-trained XLSR-53 model. Here, t is defined as the time index, T is the total number of time steps, and F is the number of output nodes of the transformer encoder layer in the context network of XLSR-53. The speech representations $\mathbf{X}$ are partitioned into short segments $\mathbf{S}_n = (\mathbf{x}_{k,n} \in \mathbb{R}^F | n = 1, ..., T/K; k = 1, ..., K)$ comprising K frames, where $\mathbf{x}_{k,n}$ is the speech representation at the k-th time step in segment n.

As shown in Fig. 3.1, the x-vector embedding module encodes the local information in each segment in a high-dimensional vector $\mathbf{e}_n$. These segment-level embeddings $\mathbf{E} = (\mathbf{e}_n \in \mathbb{R}^D | n = 1, ..., T/K)$ are then fed into two transformer encoder layers. The transformer encoder layers are next applied to capture the global dependencies among these segment-level embeddings. A statistics pooling layer is applied to embed the outputs of the encoder layers to an utterance representation, followed by three fully-connected (FC) layers that perform utterance-level prediction.

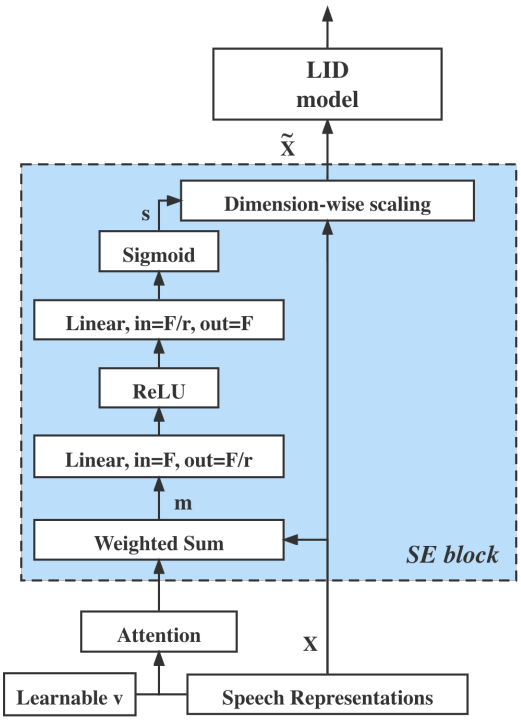


FIGURE 4.1: The XSA-LID model with attentive squeeze-and-excitation block.

4.3.2 Attentive SE block-based dimension-wise scaling

The SE layer was originally proposed to act as channel-wise scaling for image classification and was subsequently modified for speaker verification [33, 34]. It has shown to be more effective in the deep rather than shallow layer of a DNN. Nevertheless, for the first model in this task, the SE block is employed as the dimension-wise attention to determine the discriminative potential of each dimension in speech representations—this block is, therefore, employed between the input and the XSA-LID model where it is relatively shallow.

It is also useful to note that the pooling operation within the squeeze process gives the same weight to all time steps. However, input representations corresponding to some input time steps may be unique and not as significant as others for discriminating languages. Therefore, averaging time steps with equal weighting is not ideal for LID.

To overcome this problem, an attentive SE is proposed to perform the dimension-wise scaling on the wav2vec 2.0 speech representations with reference to the attentive statistics pooling [98]. The attentive SE is incorporated in the end-to-end

LID system as the front-end processing for the input features. The scalar weight for each time step and the mean vector are computed via

$$\mathbf{h}_t = \delta\Big(\mathbf{W}_2^{\text{Att}}\delta\big(\mathbf{W}_1^{\text{Att}}\mathbf{x}_t\big)\Big), \tag{4.4}$$

$$a_t = \frac{\exp\left(\mathbf{v}^\intercal\mathbf{h}_t\right)}{\sum_{i=1}^{T}\exp\left(\mathbf{v}^\intercal\mathbf{h}_i\right)}, \tag{4.5}$$

$$\mathbf{m} = \sum_{i=1}^{T}\left(a_i\mathbf{h}_i\right), \tag{4.6}$$

where $\mathbf{h}_t$ is an F-dimensional vector defined as the nonlinear transformation of the tth time-step speech representation $\mathbf{x}_t$, $\mathbf{W}_1^{\text{Att}} \in \mathbb{R}^{\frac{F}{r}\times F}$ and $\mathbf{W}_2^{\text{Att}} \in \mathbb{R}^{F\times\frac{F}{r}}$ denote the two FC layers in the attention block in Fig 4.1, respectively, r is defined as the reduction rate that is of the same value as that in the SE block. The variable δ is the ReLU function, $\mathbf{v}$ is a learnable vector which has the same dimension as the input representations, a_t denotes the scalar weight of t-th time-step representation, $\mathbf{m}$ denotes the mean vector computed by the weighted sum of the input representations.

Given speech representations $\mathbf{X}$, the attention module which includes two linear layers and a softmax is performed via (4.4) and (4.5) before generating $\mathbf{m}$ through a weighted sum in (4.6). The attentive mean vector $\mathbf{m}$ is subsequently used in (4.2) by replacing (4.1) with (4.4), (4.5), and (4.6). In contrast to (4.6) which performs time-wise scaling, (4.3) is a element-wise product and thus can be equivalent to a dimension-wise scaling. The above process occurs in the attentive SE block as shown in Fig. 4.1.

4.3.3 Linear bottleneck block

The dimension-wise attention mechanism focuses on elements in $\mathbf{x}_t$ that provide high discriminative ability. The attentive SE and SE blocks are expected to improve the LID performance if the discriminative dimensions in the input representations are independent of others, i.e., these dimensions contain predominantly language-related information while others may comprise other information such as speaker identities. In contrast to the attentive SE block, this work propose to employ the bottleneck block when the dimensions are strongly associated with each other such as when several aspects of information is represented across several dimensions.

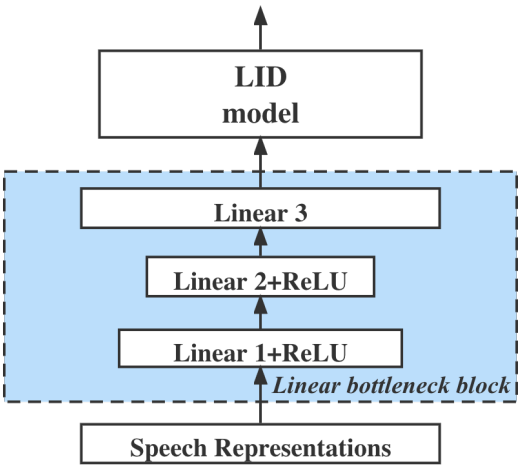


FIGURE 4.2: The XSA-LID model with linear bottleneck block.

As shown in Fig. 4.2, the LBN block is employed as the feature processor of the XSA-LID model. It consists of three linear layers and ReLU activations. The first two layers perform dimension reduction with ReLU activation and are defined as the dimension reduction module. Finally, the third layer linearly projects the outputs of the second layer to the target-dimensional space being speech representations $\widetilde{\mathbf{X}}$. These are performed via

$$\widetilde{\mathbf{X}} = \mathbf{W}_3^{\mathrm{LBN}}\delta\Big(\mathbf{W}_2^{\mathrm{LBN}}\delta\big(\mathbf{W}_1^{\mathrm{LBN}}\mathbf{X}\big)\Big). \tag{4.7}$$

To prevent the output values of the first two linear layers from being compensated by their bias items, the biases in the first two linear layers are manually set to zero. To investigate the appropriate architecture of the LBN block, the first layers are fixed while the output dimensions of the second and third layers are varied. Detailed configurations and experiments of the LBN block and its dimension reduction module are described in Section 4.4. This end-to-end architecture, which comprises the LBN block and XSA-LID model, is referred as LBN-XSA.

The proposed bottleneck block is based on the linear bottleneck block proposed in [95]. In contrast to [95] that employs LBN to minimize information loss by expanding the input channels before feeding the input to the LBN block, this work proposes to utilize wav2vec 2.0 features to the LID task efficiently by suppressing task-unrelated information within the representations. Therefore, the expansion of

the input dimensions is not required.. Considering that the wav2vec 2.0 representations are relatively dense and that they are assumed to approximate zero mean and unit variance after the layernorm [87], there is no low-dimensional subspace that can accommodate the complete speech information of the representations without losing information. This implies that information loss is inevitable if these speech representations are projected into a lower-dimensional subspace below 1024 dimensions accompanied by the ReLU activation. In addition, instead of reducing the height and width of the feature map when applied for image classification, the bottleneck structure is applied on the dimensions of the speech representations. Since the proposed LBN comprises a dimension-reduction process, the LBN block is employed along with by the ReLU activation to reduce the irrelevant information in the input features for LID.

The bottleneck block is typically applied along with the residual connection [94]. Some works emphasize the importance of the residual connection by highlighting the performance improvement from these connections [34, 94]. However, such residual connection is not adopted in this model since information associated with non-LID features in the original representations confuses the model.

4.3.4 Principal component analysis

Before performing dimension reduction within the LBN block, principal information contained within the wav2vec 2.0 representations is first analyzed. To preserve essential information when reducing the dimensions of the representations, the unsupervised PCA is employed to project the representations to low-dimensional space. This is achieved by computing the eigenvalues with the larger eigenvalue indicating a lower reconstruction error after being projected by the corresponding eigenvector. The projection is performed linearly via

$$\widetilde{\mathbf{x}}_t = \Phi^\intercal \mathbf{x}_t, \tag{4.8}$$

where $\mathbf{x}_t$ denotes the G-dimensional representation at t-th time step with $G < F$, Φ is defined as a matrix comprising eigenvectors corresponding to the largest eigenvalues—these eigenvectors represent the most crucial information. The low-dimensional representations $\widetilde{\mathbf{X}} = (\widetilde{\mathbf{x}}_t \in \mathbb{R}^G | t = 1, ..., T)$ are then fed into the XSA-LID model.

By comparing the performance of the transformed features against G, the optimal dimensions are achieved when the performance of the G-dimensional features does not significantly vary with the dimensions. After attaining the appropriate dimensions of the subspace (which can accommodate most task-related information), I construct the dimension reduction module in the LBN block is constructed and the output dimensions of each layer is set according to those learned from PCA.

4.4 Dataset, Experiment, and Model Configuration

4.4.1 Datasets

The proposed approaches are evaluated on the AP19-OLR and NIST LRE 2017. The NIST LRE 2017 dataset consists of the Fisher corpus [70], Switchboard corpora [71], a narrow-band telephony training set (TRN17) built from previous LRE data with over 2000 hours of audio data, a development set (DEV17), and an evaluation set (EVAL17) [9]. The DEV17 and EVAL17 comprise narrow-band MLS14 data and wide-band VAST data, while the MLS14 data consist of 3 s, 10 s, and 30 s duration levels and the VAST data comprise segments with speech duration ranging from 10 to 600 s. There are fourteen languages in total within the dataset. Systems in this work are trained on TRN17 and DEV17 and tested on the MLS14 data in EVAL17 to compare their performance on different duration levels.

The AP19-OLR dataset contains training, development and test sets [74] with the audios sampling rate being 16 kHz. The training set comprises previous year OLR data and THCHS30. The AP19-OLR test set which includes three parts responding to three LID tasks—short-utterance LID, cross-channel LID and zero-resource LID. The proposed systems are trained on the training set without THCHS30 and tested on only the short-utterance LID set consisting of utterances as long as 1 s among ten languages.

4.4.2 Feature extraction

For the NIST LRE 2017 data, each audio is segmented into maximum 30 s segments before extracting two types of features. The first feature is the 80-dimensional BNFs computed from an ASR-DNN model trained on Fisher and Switchboard. The input features of the ASR-DNN model are 13-dimensional MFCCs extracted from 25 ms windows with a 10 ms shift. These speech segments are first passed through an energy-based voice activity detection (VAD) to remove the silent frames in the BNFs. The second feature is extracted using a pre-trained XLSR-53 model trained on fifty-three languages and 56,000 hours of speech data from Multilingual LibriSpeech, CommonVoice and Babel [99–101]. The audios for the wav2vec 2.0 feature extraction are first upsampled to 16 kHz before an energy-based VAD is applied to the audio signal. Each audio is then clipped into voiced segments with a maximum duration of 30 s.

For the AP19-OLR data, feature extraction is performed after each audio is segmented into 20 s maximum speech segments. The 39-dimensional MFCCs (13-dimensional MFCCs with their first and second order derivations) are first extracted to train an XSA-LID model, its performance is then compared with the proposed XSA-LID models trained on the wav2vec 2.0 speech representations. No VAD is applied to the OLR data.

The MFCCs are extracted using the librosa package implemented on python language [102], and the extraction of BNFs is performed using the Kaldi toolkit [88]. The wav2vec 2.0 speech representations are extracted using s3prl toolkit [92].

4.4.3 Model configuration

4.4.3.1 X-vector model

The x-vector model follows the framework in [5]; 80-dimensional BNFs were used for the NIST LRE 2017 experiment, and the back-end scoring comprises linear discriminative analysis (LDA) and the logistic regression classifier. The training stage is performed using a modified x-vector recipe based on SRE16 in the Kaldi toolkit [88].

TABLE 4.1: Configuration of the x-vector embedding module in the XSA-LID model

Layer	Context	Tot. context	In & out
TDNN 1	[t-2, t+2]	5	1024, 512
TDNN 2	{t-2, t, t+2}	9	512, 512
TDNN 3	{t}	9	512, 512
Stats. Pooling	[1, 20]	20	512, 1024
Linear+layernorm	-	20	1024, 64

4.4.3.2 XSA-LID model

The XSA-LID model consists of an x-vector embedding module, a transformer encoder module and an utterance-level statistics pooling layer followed by three FC layers. The x-vector module includes three TDNN layers as shown in Table 4.1. The number of input and output channels are based on the convolutional layer implementation. After each TDNN layer, there is a ReLU activation and batch normalization. The transformer encoder module comprises a linear projection layer and a positional encoding layer followed by two multi-head attention encoder layers that embed the global dependencies between segment-level x-vectors. Each encoder layer has eight heads with input and output dimensions being $d_{model} = 512$ and the dimension of inner layer being $d_{ff} = 2048$. The utterance-level statistics pooling layer generates an embedding for each utterance. The three FC layers project the embedding to C classes with the ReLU nonlinearities for the first two FC layers, where C is the number of languages.

4.4.3.3 SE blocks

The only hyper-parameter in the attentive SE block is the reduction rate $r = 4$. In this case, the output channels of the first linear layer are of 256 dimensions.

4.4.3.4 LBN blocks

With respect to the LBN block, PCA is first applied to the AP19-OLR data. Since the optimal dimension may vary across datasets, the objective was not to determine

an optimal value from 1 to 1024. The intention of this work, however, was to validate the performance on three commonly used hidden dimensions 128, 256, and 512. The performance is validated via the AP18-OLR test set (this set serves as the AP19-OLR dev set for the short-utterance LID task) which comprises 18000 utterances to train the PCA model. The PCA transformation was then applied to both training and test utterances. An XSA-LID model is trained and evaluated using those transformed features.

To set up the dimension reduction module, the number of output channels is initially fixed to 512 for the first FC layer. The output dimensions of the second FC layer are then set to be either 128 or 256 to evaluate the dimensions of the appropriate feature space which can accommodate the LID-related information. The output dimensions of the dimension reduction module are chosen with reference to the results of the PCA experiments. The ReLU activation and zero-value biases are adopted by the two layers of the dimension reduction module. The output of the dimension reduction module is linearly projected by the third FC layer—its output dimensions of the third FC layer are first set to be the same as the second layer. After deciding suitable output dimensions of the dimension reduction module, the output dimensions of the third layer are set to 1024 since high-dimensional vectors can be more discriminative.

4.4.3.5 Training

The proposed models are trained on AP19-OLR data for ten epochs and NIST LRE 2017 data for twenty epochs separately with the same batch size of 128. The learning rate linearly increases to 1×10^{-4} in the first three-epoch steps and decays according to the cosine annealing method. Since the shortcut connection is not applied to the proposed architectures, a higher learning rate 1×10^{-3} is adopted instead of 1×10^{-4} for the SE and attentive SE blocks, and the LBN block to compensate for the slight loss. Parameters of the SE and attentive SE blocks stop updating two epochs earlier than the XSA-LID model.

4.4.3.6 Fine-tuning

The pre-trained XLSR-53 model is fine-tuned on the training set of the AP19-OLR using the hugging face [103]. For fine-tuning, the learning rate warms up to 5×10^{-5} in the first four epochs and decay in the remaining six epochs. Since features extracted from the 16th context network block are used in this work, the fine-tuned model comprises the layers before the 17th context network block of the pre-trained XLSR-53 model followed by a classifier. The classifier consists of two linear layers that project the context network's output to ten language classes. Only the classifier module of the model is updated in the warm-up stage. The context network blocks are then updated together with the classifier in the remaining updates. The representations from the 16th context network block were similarly extracted and fed as inputs to the LID model. Both the XSA-LID and LBN-XSA approaches were conducted on the fine-tuned features. The source code for this research has been made publicly available on GitHub. [1]

4.4.4 Performance measure

As described in Section. 2.4, the proposed systems are evaluated by employing accuracy (Acc.), equal error rate (EER) and the average cost (C_{avg}) in accordance to Section 2.4.

4.5 Results and Analysis

To evaluate the proposed methods, the performance of the XSA-LID models, which are trained on the speech representations extracted from different layers of the context network of the pre-trained XLSR-53 model, is first compared. Based on the performance, experiments are then conducted using the speech representations corresponding to the best performance. The dimension reduction methods, SE blocks and the linear bottleneck block were evaluated on the AP19-OLR data to investigate how speech representations can be utilized effectively. The selected approaches are then evaluated on the NIST LRE 2017 data.

[1] $https://github.com/Lhx94As/JSTSP\_w2v\_for\_LID$

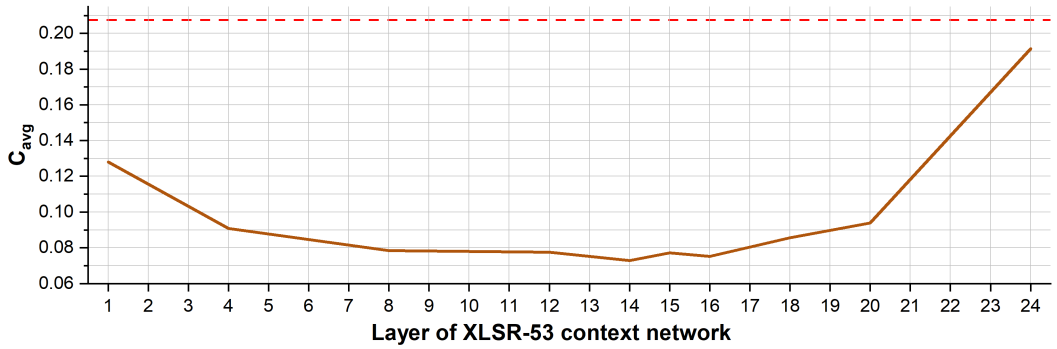


FIGURE 4.3: Comparison between the LID performance of representations extracted from different context network layers on AP19-OLR data by employing C_{avg}, the red dashed line denotes the performance of the model trained on MFCCs

4.5.1 Comparison between representations extracted from different encoder layers of the context network on AP19-OLR data

Other studies have investigated the correlation between performance of the downstream task and representations extracted from different layers of the context network in the wav2vec 2.0 model [55,93]. In this work, the performance of the wav2vec 2.0 features are explored for LID. The XSA-LID model are trained separately on speech representations extracted from different layers of the context network of the pre-trained XLSR-53 model. With reference to [93], we noted that only some layers exhibit reasonable performance for the LID task. Therefore, performance associated with features extracted from the 1st, 4th, 8th, 12th, 16th, 20th and 24th layers are evaluated before selecting that associated with the lowest C_{avg}. As shown in Fig 4.3 and Table 4.2, the XSA-LID model exhibits comparably great results $C_{avg} = 0.0729$ and $C_{avg} = 0.0751$ when the speech representations are extracted from the 14th and 16th layer—they achieve more than 60% reduction in C_{avg} compared with $C_{avg} = 0.2074$ of the model trained on MFCCs. Notwithstanding the above, even the worst performance $C_{avg} = 0.1915$ corresponding to the 24th layer achieves a lower average cost than that of the model trained on (illustrated via the dotted line). Compared with the phoneme classification results in [56], the layer exhibiting the lowest C_{avg} in Fig. 4 is similar to that where the best phoneme error rate (PER) appears. The value $C_{avg} = 0.0784$ corresponding to the 8th-layer features is similar to the best ones, and the performance subsequently degrades

TABLE 4.2: Performance of the XSA-LID models trained on MFCCs and features extracted from different encoder layers of context network on AP19-OLR data by employing Acc. (%), EER (%) and C_{avg}

Features	**Layer**	$C_{avg}\downarrow$	EER$\downarrow$	Acc.$\uparrow$
XLSR-53 w2v	14	**0.0729**	**7.96**	**83.91**
	15	0.0771	8.54	83.26
	16	0.0751	8.20	83.68
	24	0.1915	19.21	56.88
MFCC	-	0.2074	21.12	-

TABLE 4.3: Comparison between applying SE and attentive SE blocks on top of the XSA-LID model on AP19-OLR data by employing Acc. (%), EER (%) and C_{avg}

Method	$C_{avg}\downarrow$	EER$\downarrow$	Acc.$\uparrow$
XSA-LID	**0.0751**	**8.21**	**83.68**
SE-XSA	0.0800	9.09	82.52
AttSE-XSA	0.0763	8.44	83.67

after the 18th layer where $C_{avg} = 0.0856$. These results imply that the model may learn the language-related information simultaneously with the phoneme-related information.

Since the XSA-LID model achieves relatively good performance when training on the speech representations extracted from the 16th-layer, these speech representations are used as the baseline to compare with the other systems.

4.5.2 Results of the XSA-LID model with SE blocks on AP19-OLR data

Results achieved by the SE and attentive SE blocks evaluated on the AP19-OLR data are presented in Table 4.3. SE-XSA and AttSE-XSA have been used to denote the XSA-LID model with the SE block and the attentive SE block, respectively. Both SE-XSA and AttSE-XSA exhibit lower performance than the baseline XSA-LID model, while the AttSE-XSA suffers from lower degradation in performance

TABLE 4.4: Comparison between different dimensions after performing PCA on the SSL speech representations. Models trained on different dimensional SSL features are evaluated on AP19-OLR data by employing Acc. (%), EER (%) and C_{avg}, respectively

Method	**Dim.**	$C_{avg}\downarrow$	EER$\downarrow$	Acc.$\uparrow$
XSA-LID	1024	**0.0751**	**8.21**	**83.68**
PCA-XSA	128	0.0889	9.57	80.67
	256	0.0833	8.91	81.94
	512	0.0816	9.18	82.61

than the SE-XSA. These observations first confirm my proposition that speech representations of different time steps are not identically significant for LID. Secondly, such degradation may imply that the dimensions in the wav2vec 2.0 speech representations collaborate in representing different aspects of information. Therefore, scaling the dimensions, or channels, can lead to the distortion of the LID-related discriminative information. Consequently, instead of the dimension-wise scaling, this work proposes to replace the attentive SE block with the LBN block as the nonlinear feature refiner, and the attentive SE will not be used for all subsequent experiments.

4.5.3 Results and analysis of the LBN-XSA model on AP19-OLR data

Results of the PCA dimension reduction algorithm on AP19-OLR data are shown in Table 4.4 along with the baseline results. The XSA-LID model trained on features processed by the PCA dimension reduction method is denoted as PCA-XSA. Results presented show that such an unsupervised dimension reduction achieves poor performance. This indicates that PCA suppresses language-related information when reducing the dimensions. It is interesting to note that while the model achieves reasonably better results with increasing projected feature dimensions from 128 to 256, further increase from 256 to 512 yields negligible improvement. This suggests that 256-dimensional features can adequately encapsulate the information and 256 is therefore selected as the hidden dimension of the LBN block.

TABLE 4.5: Ablation study of output dimensions of the last two layers and the activations and biases of the first two layers of the LBN block in the LBN-XSA model by employing Acc. (%), EER (%) and C_{avg}

LBN	**Layer2**	**Layer3**	**Activ.**	**Biases**	$C_{avg}\downarrow$	EER$\downarrow$	Acc.$\uparrow$
1	128	128	ReLU	Zero	0.0697	**7.83**	84.82
2	256	256	ReLU	Zero	0.0681	7.87	85.07
3	256	1024	ReLU	Zero	**0.0677**	7.91	**85.18**
4	256	1024	ReLU	Trainable	0.0744	8.79	84.01
5	256	1024	Linear	Zero	0.0797	8.69	82.15

It is also useful to note that further reducing the dimension from 256 to 128 results in performance degradation from $C_{avg} = 0.0833$ to 0.0889 in terms of C_{avg} indicating that LID-related information has been lost. In addition, although 128 is relatively smaller than 1024, the 128-dimensional features can still achieve reasonably good LID performance. This highlights that LID-related information broadly exists in speech representations extracted from the 16th context network layer. Since the optimal dimension may vary across datasets, the intention of using PCA is to validate the performance on three commonly used hidden dimensions 128, 256, and 512. While employing PCA with eigenvalue 256 exhibits the highest performance among all three eigenvalues, it is assumed to be the proper dimension for the feature space.

Results achieved by the LBN-XSA models with different setups are presented in Table 4.5. Apart from different setups of the output dimensions, the influences of the ReLU activations and the biases in the dimension reduction module of the LBN block are also discussed via the ablation study.

As shown in Table 4.5, columns corresponding to Layer 2 and 3 represent their output dimensions. Firstly, the LBN-XSA models 1, 2, and 3 outperform the XSA-LID in terms of all metrics. Model configurations 2 and 3 exhibit higher performance than model configuration 1 in terms of accuracy and C_{avg}. This indicates an output dimension of 256 for the dimension reduction module of the LBN block can achieve good LID performance, which is consistent with experiments conducted for PCA.

In addition, model configurations 2 and 3 achieve comparable performance while model configuration 3 achieves modestly better in accuracy and C_{avg}. However, the higher-dimensional outputs of model configuration 3 introduce approximately 200,000 more parameters to the LBN-XSA model than that of the model configuration 2. With this trade-off, it may not be worth achieving such a modest improvement at the expense of a significant increase in parameters and computations. Consequently, the model configuration 2 is more efficient than model configuration 3.

Compared to model configuration 3, model configuration 4 adopts trainable biases to the first two layers of the LBN block and yields worse performance. Nevertheless, I observe that model configuration 4 exhibits higher performance than the baseline XSA-LID model in terms of accuracy and C_{avg}. This implies that the biases may compensate for the negative outputs, hence they result in fewer number of channel collapses. Consequently, the trainable biases in the first two LBN layers lead to lower amount of irrelevant information loss. The wav2vec 2.0 speech representations are assumed to be zero-mean and unit variance, i.e., these representations distribute in the high-dimensional space as a sphere centered at the origin, while the linear projection generates a similar distribution—the nonzero-value biases results in lower number of negative outputs affecting the information loss. Compared to model configuration 4, model configuration 2 contains fewer parameters but exhibits higher performance. In addition, the proposed AttSE-XSA model, which has more parameters, shows lower performance compared to the vanilla XSA-LID. The above indicates that increasing model parameters and complexity does not necessarily leads to higher performance, which underpins the effectiveness of the proposed LBN block.

Model configuration 5 suffers from higher performance degradation compared to the XSA-LID and model configuration 3. This suggests that the ReLU activation plays an essential role in removing task-unrelated information. As mentioned above, purely linear projections performed by the first and second layers do not significantly influence the performance of the features. Therefore, the bias in the third LBN layer distorts the distribution of the input representations. Since zero-mean and unit-variance inputs, in general, lead to better results, this distortion results in the lower performance.

TABLE 4.6: The performance of the proposed methods with or without using the fine-tuned speech representations in terms of Acc. (%), EER (%) and C_{avg}

Method	**Fine-tuned**	$C_{avg}\downarrow$	EER$\downarrow$	Acc.$\uparrow$
XSA-LID	No	0.0751	8.21	83.68
LBN-XSA	No	**0.0681**	7.87	**85.07**
XSA-LID	Yes	0.0732	**7.27**	83.87
LBN-XSA	Yes	0.0823	9.48	83.40

4.5.4 Fine-tuning on the AP19-OLR data

To compare the performance of the proposed LBN-XSA with the fine-tuning approach, the pre-trained XLSR-53 model is fine-tuned on the AP19-OLR data. Similar to the pre-trained model, the fine-tuned model serves as a feature extractor. The fine-tuned speech representations have also been extracted from the 16th context network layer before being fed to the XSA-LID and LBN-XSA models and results are presented in Table 4.6.

It is not surprising that both models achieve higher performance with the representations extracted from the fine-tuned XLSR-53 model than the pre-trained model. After fine-tuning, the XSA-LID model achieves 11.45% relative improvement in EER and modest improvements in C_{avg} and accuracy. Without using the fine-tuned features, the proposed LBN-XSA model achieves higher performance in C_{avg} and accuracy and suffers from worse performance in EER than the XSA-LID model that is trained on the fine-tuned representations. However, the LBN-XSA model suffers from performance degradation when it is trained on the fine-tuned features. This implies that some discriminative information is lost when passing through the LBN block. As mentioned in Section 4.2.1, the pre-trained XLSR-53 model possesses the ability to filter out the information that is not related to the LID task during the fine-tuning process. Therefore, if the LBN-XSA model is trained on the fine-tuned wav2vec 2.0 features, information that is related to the LID task may be lost within the LBN block resulting in performance degradation.

TABLE 4.7: Evaluation of various models and features on MLS14 data in NIST LRE 2017 dataset by employing Acc. (%), EER (%) and C_{avg}

Method	**Feat.**	**3 s**			**10 s**			**30 s**		
		C_{avg}↓	EER↓	Acc.↑	C_{avg}↓	EER↓	Acc.↑	C_{avg}↓	EER↓	Acc.↑
x-vector	BNFs	0.1159	11.79	-	0.0748	7.81	-	0.0654	6.84	-
XSA-LID	BNFs	0.1685	15.47	54.18	0.0739	7.39	74.90	0.0406	4.46	84.09
XSA-LID	w2v	0.1004	10.71	70.16	0.0432	4.89	84.32	0.0329	3.71	87.20
LBN-XSA	w2v	**0.0879**	**8.79**	**72.90**	**0.0385**	**3.82**	**85.10**	**0.0295**	**2.72**	**89.07**

4.5.5 Evaluations on the MLS14 data in NIST LRE 2017 dataset

Since the AP19-OLR data focuses on oriental languages, the XSA-LID and LBN-XSA models are also evaluated on the NIST LRE 2017 data. We note from Section 4.5.3 that model configuration of model configuration 2 is the most efficient and this setup is then adopted for the LBN-XSA model in this section.

As shown in Table 4.7, the performance of the XSA-LID model and a commonly used baseline x-vector model which are trained using BNFs on the MLS14 data in NIST LRE 2017 is first presented. The performance of the XSA-LID model after replacing the BNFs with wav2vec 2.0 speech representations is then given. I next evaluate the performance of the LBN-XSA model trained on the wav2vec 2.0 representations. The wav2vec 2.0 speech representations are denoted as w2v in the second column of Table 4.7.

Compared to the x-vector model, the XSA-LID model achieves higher performance on longer utterances, i.e., 10 s and 30 s. In particular, for the 30 s test speech, the XSA-LID model yields 37.92% relative improvement compared to the x-vector approach albeit exhibiting a lower performance on the 3 s test speech. This is because the x-vector model is trained on short speech chunks and is expected to perform well on short utterances.

After replacing the BNFs with the w2v features for the XSA-LID model, it achieves significantly higher performance across various durations. Compared to the model trained on BNFs, the XSA-LID trained on w2v features achieves 40.42%, 41.54% and 18.97% relative improvement on 3 s, 10 s and 30 s test speech, respectively, in terms of C_{avg}. This implies that the w2v features are more discriminative for

the downstream tasks. Moreover, compared to the XSA-LID model, the proposed LBN-XSA method further yields 12.45%, 10.88% and 10.33% relative improvements on 3 s, 10 s and 30 s test speech, respectively, in terms of C_{avg}. This indicates that the LBN block extracted more discriminative features for the XSA-LID model. Considering the presence of information loss, this also implies that the LBN block suppresses information that is unrelated to LID resulting in higher performance.

Notwithstanding the above, since the XSA-LID and x-vector models exhibit comparable overall performance in Table VII (9.11% and 8.81% in terms of EER, respectively, by averaging the performance on three duration levels), and that the LBN-XSA w2v achieves 1.32% absolute improvement in EER compared with the XSA-LID, the LBN-XSA w2v is expected to achieve higher overall performance than the x-vector model with wav2vec features, with significantly higher performance on 10 s and 30 s speech.

4.5.6 Feature processing attempts

In addition to performance comparison, additional experiments are performed to optimize speech representations.

4.5.6.1 Padding for various number of time steps

Since the input speech samples vary in duration, I applied either of the two padding strategies before feeding the representations to the model: *a)* zero-value padding, *b)* interpolation. While the first is commonly used, the padding vectors are randomly sampled from the representations of the speech to be padded. I evaluated the performance of the LBN-XSA model on these two types of padding strategies. The zero-padding strategy achieves a 4.13% relative improvement compared to the interpolation in terms of C_{avg}. One possible reason is that the input representations have a zero-value mean. The padding values inevitably influence the results after the layernorm and batchnorm processes [87, 104], yet zero padding may introduce less distortion than interpolations.

4.5.6.2 Sampling for each time step

For a given speech sample, the representation of each time step is extracted from a random layer of the context network in the pre-trained XLSR-53 model. As information associates with features extracted from different layers is related to diverse aspects such as speaker and language identities, this method aims to integrate various information in different layers of the context network. the experiment is performed by specifying the range of alternative layers from 14 to 18 and compare the results of this method with the baseline.

With the above method, the XSA-LID model achieves $C_{avg} = 0.0849$ which has a lower performance compared to that trained on the features extracted from the 16th layer. This suggests that the information variability across the layers may not be better than purely language-related information. This result is consistent with those presented in Section 4.3.3.

4.6 Discussion

With reference to Table 4.2 and [93], it can be observed that the most discriminative information on both phoneme and language exists in the neighborhood region of the 15th context-network encoder layer. Performance associated with features extracted from these layers substantially differs that from the ones extracted from the final layer. In [105], the authors provided the weight analysis of different tasks with respect to the context-network blocks. They showed that the performance of ASR and intent classification benefit more from the last few layers. These observations suggest that selecting an appropriate context network is important for the downstream tasks.

In Table 4.3, the AttSE-XSA model exhibits modestly lower performance compared to the vanilla XSA-LID model. Since bits within the scaling factor vector vary from 0 to 1, a few bits within the speech representations which are scaled by a low-value factor suffer from bit-level information loss. As the assumption made in this chapter is that language-irrelevant information loss could improve language identification performance, this observation indicates that the speech representations after dimension-wise scaling lose language-related information. However, the

attentive squeeze-and-excitation block is trained with the LID model in a supervised manner, it is expected to reduce information that is irrelevant to languages. Therefore, the above implies that dimensions within the self-supervised speech representations work jointly but not separately to present speech information.

In addition, the proposed LBN block can refine the discriminative information in the wav2vec 2.0 speech representations and achieves higher performance than the fine-tuning approach in terms of C_{avg} and accuracy. Nevertheless, to achieve the best performance, appropriate dimension of the subspace that can adequately accommodate the LID-related information should be predetermined. On the other hand, as shown in Table 4.6, since the task-unrelated information has been filtered out in the fine-tuning process, discriminative information is lost within the LBN block when using the fine-tuned features. Hence, the proposed LBN block may not be compatible with the fine-tuning approach which presents a limitation of this method.

Fine-tuning is the process that stacks one linear layer on top of the pre-trained XLSR-53 model and retrains this model on NIST LRE 2017 data for language identification. The supervised training refines the outputs of context blocks within the XLSR-53 model to be more discriminative for language identification. This feature refinement process is similar to the training process of the LBN-XSA model. However, the results shown in Table 4.5 indicate that reducing the output dimension from 256 to 128 leads to moderate performance degradation. Therefore, employing fine-tuning jointly with the proposed LBN block could result in lower performance compared with solely using the LBN block.

4.7 Summary

This work proposed to efficiently apply the wav2vec 2.0 speech representations efficiently for the LID task. Firstly, by replacing the acoustic features or BNFs with the pre-trained wav2vec 2.0 speech representations, the models achieved significant performance improvement on the AP19-OLR and NIST LRE 2017 data. The use of the attentive SE and the LBN blocks was then proposed to suppress information that is not related to the LID task. Compared to the XSA-LID model, while the AttSE-XSA exhibits modestly lower performance, the LBN-XSA method achieves

higher performance by removing the LID-irrelevant information with nonlinearities and dimension reduction. Finally, the pre-trained XLSR-53 model was fine-tuned on the AP19-OLR data. Compared to the pre-trained features, the XSA-LID model achieves higher performance through the use of the fine-tuned features. Moreover, the proposed LBN-XSA model trained on the pre-trained features outperforms the XSA-LID model trained on the fine-tuned features.

Chapter 5

PHO-LID: A Unified Model Incorporating Acoustic-Phonetic and Phonotactic Information for Language Identification

As opposed to Chapter 4 which aims to enhance a single type of language cues, this chapter introduces a novel model that hierarchically incorporates two language cues, phoneme and phonotactic information, for LID without requiring phoneme annotations for training.

In Section 5.1, acoustic and phonotactic features are first introduces, the proposed method that incorporates these two language cues is then briefly described. Section 5.2 introduces the self-supervised phoneme segmentation related to the proposed PHO-LID model. The proposed PHO-LID model is illustrated in Section 5.3, and describe the datasets, model configurations and detailed experiments in Section 5.4. The proposed PHO-LID model is evaluated and results are presented in Section 4.5. A comparison between predicted phoneme boundaries and corresponding audio spectrograms that illustrates the leveraging of phoneme information for LID is shown in Section 5.6. The conclusion of this work is finally given in Section 5.7.

5.1 Introduction

Early studies in LID suggested that acoustic and phonotactic features are the most effective language cues [1, 14, 15], since acoustic-phonetic and phonotactic cues depict the language identities of a speech signal in different granularities, it is natural to consider both jointly to achieve high LID performance. The proposed phonetic and phonotactic LID (PHO-LID) method incorporates phonetic and phonotactic information hierarchically via a convolutional neural network-transformer encoder (CNN-Trans) without the use of phoneme annotations [23]. Inspired by the success of self-supervised phoneme segmentation in [106], the CNN module in the proposed model is shared by the LID and self-supervised phoneme segmentation tasks, through which it learns the language and positional phoneme information during training. To perform phonotactic LID, the proposed method mimics the phone n-gram modeling in a statistical manner. The language and phoneme-aware output features of the CNN module are first aggregated by a statistics pooling layer for each short-duration (hundreds of milliseconds) segment of the input speech signal. The segment-level phonotactic embeddings are then generated before the utterance-level LID, in which a statistics pooling layer computes the utterance-level statistics.

Apart from the high LID performance resulting from the complementary characteristics of acoustic-phonetic and phonotactic features, the proposed model possesses two desirable properties. Firstly, as opposed to existing phoneme or phonotactic-aware methods, the proposed approach does not require phoneme annotation or transcription due to the self-supervised training. Secondly, the proposed PHO-LID model combines different language cues in an end-to-end manner resulting in lower complexity compared to the fusion of several subsystems with respect to different language cues [1].

5.2 Related work

In [106], the author proposed a self-supervised approach for phoneme segmentation. Given the representations of a raw speech signal, a CNN model is optimized to identify the spectral changes. This is achieved by minimizing a noise-contrastive

estimation (NCE) loss [107, 108], in which the anchor, positive, and negative samples correspond to the current frame, its adjacent frames, and non-adjacent frames, respectively. Denoting $\mathrm{sim}(\cdot, \cdot)$ as cosine similarity between two vectors, the NCE loss for each frame $\mathbf{z}_i$ is computed via

$$\mathcal{L}(\mathbf{z}_i) = -\log \frac{e^{\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_{i+1})}}{\sum_{\mathbf{z}_j \in \{\mathbf{z}_{i+1}\} \cup D_M(\mathbf{z}_i)} e^{\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)}}, \tag{5.1}$$

where $D_M(\mathbf{z}_i)$ is a set of M negative samples to $\mathbf{z}_i$. These negative samples are randomly selected from all non-adjacent frames. During inference, the similarities between every two adjacent frames are computed. The phoneme boundaries are then determined as where the similarity is lower than a predetermined threshold.

5.3 Methodology

5.3.1 PHO-LID model

In an early fusion system (i.e., feature concatenation), phonotactic features are extracted using phoneme labels before being concatenated with acoustic features. The advantage of the method proposed in this chapter is to incorporate phonotactic information without the use of phoneme annotations. I wish to clarify that we use 25ms-level acoustic features as the input to generate 400ms-level phonotactic features within a hierarchical framework. The hierarchical framework is employed because these two types of features are of different granularities. Therefore, they have to be generated hierarchically in this work.

As shown in Fig. 5.1, the proposed PHO-LID model incorporates both acoustic-phonetic and phonotactic information hierarchically and comprises two branches. To illustrate these two branches separately, $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^{K \times F} | t = 1, ..., T)$ is defined as input of the proposed model. Here, $\mathbf{X}$ comprises features extracted from the input speech signal partitioned in segments, and T is the number of segments. Each segment $\mathbf{x}_t$ includes K frames $[\mathbf{f}_{t,1}, ..., \mathbf{f}_{t,K}]^\intercal$, where $\mathbf{f}_{t,k}$ is an F-dimensional feature vector of the k-th frame in segment t, and the original speech signal consists of $T \times K$ frames. It is useful to note that, in general, the average duration of common phonemes varies between 50 ms to 200 ms. The number of frames in each segment is thus defined to allow that each phonotactic unit in the proposed model can

cover minimum two phonemes. These partitioned features are fed into the CNN module, which is jointly optimized by the primary LID task and the self-supervised phoneme segmentation task over each segment $\mathbf{x}_t$.

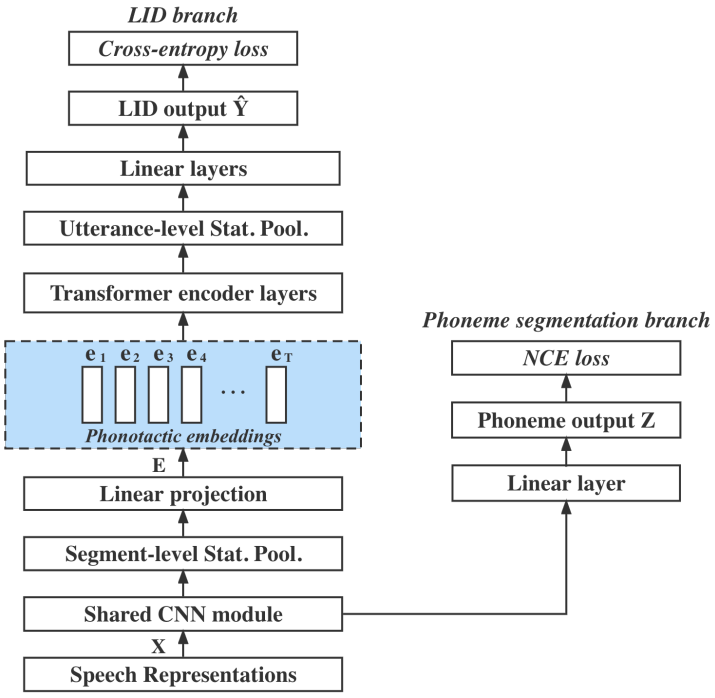


FIGURE 5.1: The PHO-LID model.

5.3.2 Self-supervised phoneme segmentation

To extract phonotactic information, the model should assimilate phoneme information before capturing the sequential patterns across several frames. Similar to the method originally proposed in [106], the self-supervised phoneme segmentation is then employed to supply the CNN module with phoneme information. Here, the anchor, positive, and negative samples correspond to the current frame at time t, its adjacent frames (i.e., $t-1$ or $t+1$), and non-adjacent frames. In the proposed PHO-LID model, the positive sample always locates at $t+1$ and the negative samples are randomly selected from all non-adjacent frames.

With reference to Fig. 5.1, the self-supervised phoneme segmentation branch consists of the shared CNN module and a linear layer. In [106], given an input raw audio, each output unit of their model corresponds to a 10 ms frame, which is of

a higher granularity than common phonemes. The premise that adjacents frames belong to the same phoneme is therefore reasonable. As opposed to [106], the input features of the proposed model are of 25 ms receptive field with 20 ms step size similar to those in conventional LID methods. With a minimum phoneme length being approximately 50 ms, convolutional layers with a kernel size of 1 is adopted in the CNN module to achieve a shorter receptive field of the output unit than the minimum phoneme length. The output of the linear projection layer corresponding to the input segment-level features $\mathbf{x}_t$ is given by $\mathbf{Z} = (\mathbf{z}_i \in \mathbb{R}^G | i = 1, ..., K)$, where G is the output dimension of the phoneme segmentation branch. The utterance-level NCE loss is then computed via

$$L^{NCE} = \frac{1}{KT} \sum_{t=1}^{T} \sum_{i=1}^{K} \mathcal{L}\left(\mathbf{z}_i\right), \tag{5.2}$$

where $\mathcal{L}\left(\mathbf{z}_i\right)$ denotes the frame-level NCE loss defined in (5.1).

As highlighted in Section 5.2, similarities between adjacent frames can reveal phoneme boundaries. Therefore, after optimization, positional phoneme information is intrinsically encoded in the phoneme segmentation branch. Since phonotactics characterizes the combinations of phonemes spanning a short proportion of speech, the learned phoneme information enables the proposed PHO-LID model to incorporate the phonotactic information in segment-level features.

5.3.3 Supervised language identification

As the CNN module learns the phoneme and language information for each segment $\mathbf{x}_t$, the rest of the LID branch aims to perform phonotactic modeling. Features in each segment $\mathbf{x}_t$ are then aggregated as mean and standard deviation vectors, which are concatenated into a representation vector. These representations are linearly projected to segment-level phonotactic embeddings $\mathbf{E} = (\mathbf{e}_t \in \mathbb{R}^D | t = 1, ..., T)$, with each D-dimensional vector $\mathbf{e}_t$ corresponding to a segment $\mathbf{x}_t$. This statistical process serves as the phone n-gram modeling of the input speech. The transformer encoder layers subsequently capture the global dependencies of these phonotatic embeddings [23], and the output is aggregated into an utterance-level language embedding. Consequently, a score vector $\widehat{\mathbf{Y}}$ corresponding to C target languages is generated by the linear layers.

During training, the model is first pre-trained on the self-supervised phoneme segmentation task for several epochs. Two strategies are employed and compared in the remaining updates. The first strategy solely updates the LID branch via a cross-entropy loss, while the second strategy optimizes the model by a multi-task objective function. These strategies are, respectively, given by

$$L^{LID} = \text{CrossEntropy}\left(\widehat{\mathbf{Y}}, \mathbf{Y}\right), \tag{5.3}$$

$$L^{MUL} = \alpha L^{LID} + (1-\alpha) L^{NCE}, \tag{5.4}$$

where $\mathbf{Y}$ denotes the true language label and α is a parameter associated with multi-task learning. The model optimized by these two strategies are denoted as PHO-LID and PHO-LID-MUL, respectively. During inference, the input speech signal will be classified to the language of the highest score in $\widehat{\mathbf{Y}}$.

5.4 Data and model configuration

5.4.1 Datasets

The proposed approaches were evaluated on the AP17-OLR and NIST LRE 2017 dataset. The AP17-OLR dataset contains training, development, and test sets [73]. The audios are of 16 kHz sampling rate and from ten oriental languages. Systems in this work were trained on the training set without THCHS30 and evaluated on the test set. The NIST LRE 2017 dataset, on the other hand, consists of the Fisher corpus [70], Switchboard corpus [71], a narrow-band telephony training set (TRN17) built from previous LRE data with over 2000 hours of audio data, a development set (DEV17), and an evaluation set (EVAL17) [72]. The DEV17 and EVAL17 comprise narrow-band MLS14 data and wide-band VAST data, while the MLS14 data consist of 3 s, 10 s, and 30 s duration and the VAST data comprise segments with speech duration ranging from 10 to 600 s. Fourteen languages exist in total within the data. All systems were then trained on TRN17 and DEV17 and tested on the MLS14 data in EVAL17 to compare their performance on different duration levels.

5.4.2 Data preprocessing and feature extraction

The recordings in TRN17 of the MLS14 data in NIST LRE 2017 are first upsampled from 8 kHz to 16 kHz before an energy-based voice activity detection (VAD) is applied. The upsampled TRN17 recording are then partitioned into maximum 30 s segments before the feature extraction. The recordings in the AP17-OLR training set are partitioned into maximum 20 s segments without VAD before the feature extraction.

In this work, input features to the systems are wav2vec features (W2V) extracted from the 16th context network block of the XLSR-53 cross-lingual wav2vec 2.0 model [24,56]. This model has been pre-trained on fifty-three languages and 56,000 hours of speech data from Multilingual LibriSpeech, CommonVoice, and Babel [99–101]. The input wav2vec features $\mathbf{X}$ are of dimension $F = 1024$ and partitioned into segments, each of which comprises $K = 20$ frames before being fed into the model. Therefore, phonotactic unit in this work is approximately 400 ms long to cover minimum two phonemes. In addition, two baseline models in the experiments with respect to the NIST LRE 2017 data are trained on 80-dimensional bottleneck features (BNFs). The bottleneck features are extracted from an ASR model pre-trained on the Fisher and Swithchboard corpora using the Kaldi toolkit [88].

5.4.3 Model configuration

Two baseline x-vector and XSA-LID models are provided for the experiments of NIST LRE 2017, with the configuration of x-vector approach following that of [5]. The x-vector is trained by modifying the SRE16 recipe in the Kaldi toolkit with a back-end logistic regression classifier. The XSA-LID model is trained following the same strategy as the CNN-Trans model.

In the proposed model, the shared CNN module comprises three convolutional layers with kernel sizes $(1, 1, 1)$ and output dimensions $(512, 512, 512)$. The output dimension of the phoneme segmentation branch is given by $G = 64$. The outputs of the segment-level statistics pooling layer are projected to D-dimensional phonotactic embeddings with $D = 64$. The transformer encoder module consists of a layer norm followed by two transformer encoder layers, each of which has 8 heads, $d_{\text{model}} = 512$, and $d_{\text{ff}} = 2048$ [23]. The following linear layers comprise 512, 512,

TABLE 5.1: Evaluation on AP17-OLR by employing Accuracy (%), EER (%) and C_{avg}

Method	**Feat.**	Acc.↑	EER↓	C_{avg}↓
i-vector	MFCCs	-	3.39	0.0352
PTN-LID	Fbanks	-	8.15	0.0689
CNN-Trans	W2V	96.74	1.10	0.0098
PHO-LID-3	W2V	97.57	0.80	0.0073
PHO-LID-10	W2V	97.67	0.82	0.0075
PHO-LID-MUL-3	W2V	**98.13**	**0.63**	**0.0058**
PHO-LID-MUL-10	W2V	97.50	0.86	0.0080

and C output nodes, respectively. The CNN-Trans model shares the same configuration as the PHO-LID model but excludes the phoneme segmentation branch.

The PHO-LID model is trained on the AP17-OLR data and NIST LRE 2017 data for 13 and 23 epochs, respectively. In the first 3 epochs, the CNN module is updated by the self-supervised phoneme segmentation task with a constant learning rate of 10^{-4}. In the remaining epochs, the model is updated with a learning rate that warms up from 0 to 1×10^{-4} in 3 epochs followed by the cosine annealing decay. The PHO-LID-MUL employ two α values 0.95 for AP17-OLR data and 0.97 for NIST LRE 2017 data. The Adam optimizer with batch size 128 is used. Systems were evaluated by employing accuracy (Acc.), equal error rate (EER) and the average cost (C_{avg}) according to Section 2.4 and [78]. The source code has been made available in GitHub. [1]

5.5 Experiment, results and analysis

5.5.1 Performance on AP17-OLR data

Evaluation results of the baseline and the proposed models on AP17-OLR data are presented in Table 5.1. Results of the i-vector and phonetic temporal neural (PTN-LID) models are provided by the AP17-OLR evaluation plan [4, 39, 73].

[1] https://github.com/Lhx94As/PHO-LID

The averaged results of the models across several trials are presented in Table 5.1. Here, the suffixes of PHO-LID and PHO-LID-MUL denote the number of negative samples M in (5.1). In Table 5.1, the PHO-LID-MUL-3 model achieves the highest LID performance compared to other methods and exhibits 40.82% relative improvement in terms of C_{avg} compared to the CNN-Trans model. This indicates that incorporating acoustic-phonetic and phonotactic information effectively enhances the LID performance. In addition, the PHO-LID-MUL-3 model achieves higher performance in terms of EER and accuracy compared to the PHO-LID-3. This suggests that the performance improvement is mainly attributed to the pre-training for self-supervised phoneme segmentation.

As M increases, the PHO-LID-MUL model suffers from significant performance degradation while model exhibits few variations of LID performance. Comparing the performance between the PHO-LID-MUL-3 and PHO-LID-3 models, these results imply that a small M with multi-task learning is preferred. The assumption made when training the phoneme segmentation branch is that adjacent frames belong to the same phoneme while non-adjacent frames more likely correspond to different phonemes. However, phoneme segmentation in this chapter is performed on 400 ms speech segments. Each speech segment comprises twenty frames. Since the average duration of common phonemes varies between 50 ms to 200 ms, some non-adjacent frames, although they are expected to be negative samples, belong to the same phoneme as the anchor frame, and are actually positive samples. When augmenting the number of negative samples, there is a higher probability of selecting a poor negative sample for the NCE loss leading to lower performance.

5.5.2 Performance on the MLS14 set of NIST LRE 2017

Table 5.2 presents the results on the MLS14 set of NIST LRE 2017 by the test speech durations. The proposed PHO-LID-MUL-3 model exhibits the highest overall performance compared to other systems in terms of C_{avg}, and both PHO-LID-3 and PHO-LID-MUL-3 models outperform the CNN-Trans model on overall performance in terms of C_{avg}. The above is consistent with the results shown in Table 5.1 and suggests the effectiveness of the proposed method. Moreover, compared to the CNN-Trans model, while the PHO-LID-MUL-3 achieves 1.97% relative improvement on 3 s test speech in terms of C_{avg}, it achieves 4.05% and 3.44% relative

TABLE 5.2: Evaluation on the MLS14 set of NIST LRE 2017 by employing Accuracy (%), EER (%) and C_{avg} across three test speech durations 3 s, 10 s, and 30 s

Method	**Feat.**	**3 s**			**10 s**			**30 s**		
		Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓	Acc.↑	EER↓	C_{avg}↓
x-vector	BN	-	11.79	0.1159	-	7.81	0.0748	-	6.84	0.0654
XSA-LID	BN	54.18	15.47	0.1685	74.90	7.39	0.0739	84.09	4.46	0.0406
	W2V	70.16	10.71	0.1004	84.32	4.89	0.0432	87.20	3.71	0.0329
CNN-Trans	W2V	**73.63**	**7.89**	0.0710	**86.36**	3.89	0.0346	89.93	2.91	0.0262
PHO-LID-3	W2V	72.61	8.22	0.0712	85.89	**3.84**	0.0344	**90.00**	**2.77**	**0.0242**
PHO-LID-MUL-3	W2V	72.52	8.10	**0.0696**	84.90	3.86	**0.0332**	89.81	2.87	0.0253

improvement on 10 s and 30 s speech, respectively. This is not surprising since phonotactic features are extracted from longer speech units as opposed to acoustic features.

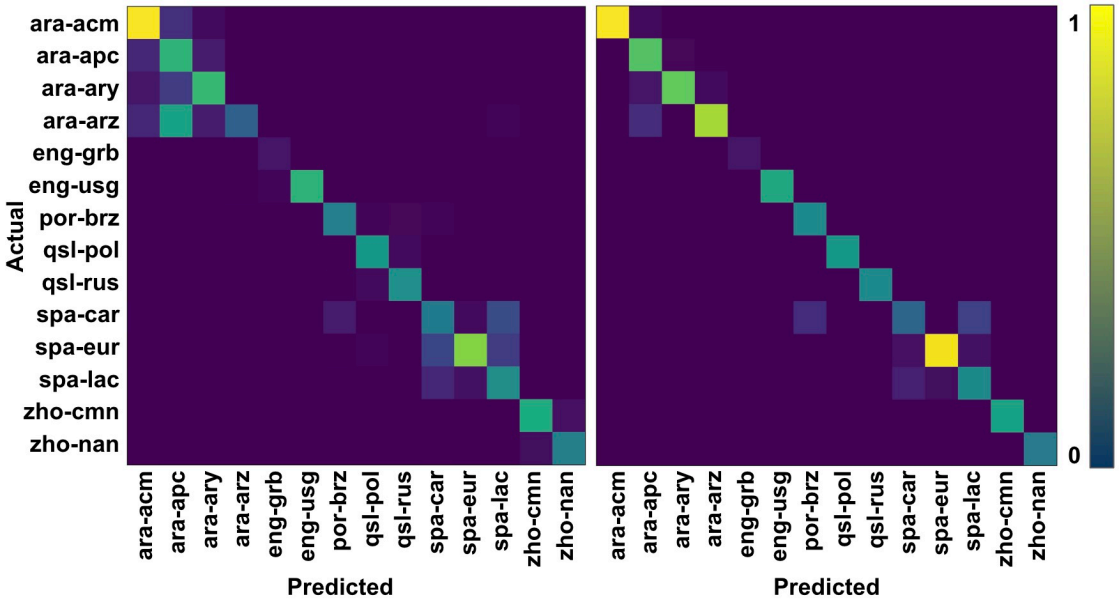


FIGURE 5.2: Performance confusion matrices of the CNN-Trans model (left) and the PHO-LID-MUL-3 (right) on 30 s test speech in NIST LRE 2017 data.

The confusion matrices of the LID performance of the CNN-Trans and PHO-LID-MUL-3 models on 30 s test speech are visualized in Fig. 5.2. Compared to the CNN-Trans model, the proposed PHO-LID-MUL-3 model with phonemic and phonotactic information effectively reduces the confusion resulting from languages belonging to the same cluster—Arabic and Iberian clusters, and thus achieves higher performance.

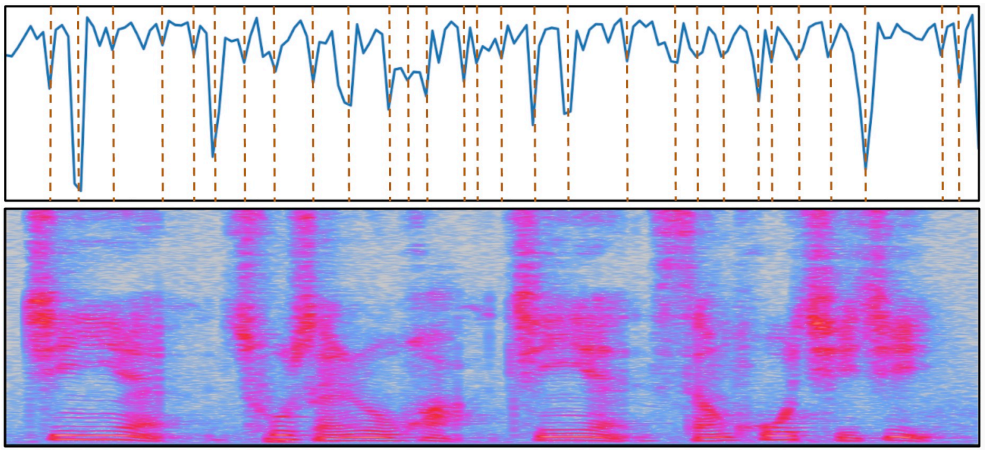


FIGURE 5.3: Comparison between the phoneme boundaries predicted by the proposed model (top) and the audio spectrogram (bottom).

In addition to the above, results presented in Tables 5.1 and 5.2 show that the W2V features introduce significant performance improvement compared to the use of conventional acoustic-phonetic features. This suggests that the W2V features are more discriminative and well-suited for the LID task. The comparison between these two tables also shows that the PHO-LID model trained on fewer data gains more improvement. This indicates that the phonotactic information, which intrinsically exists in W2V features, mitigates the lack of data.

5.6 Visulization and discussion

With reference to [106], phoneme boundaries occur where the similarity of two adjacent frames is lower than a predefined threshold. To demonstrate the existence of the positional phoneme information in the proposed model, similarities between adjacent frames in a speech sample are compared with the corresponding spectrogram. Since phoneme boundaries detection is not the target of this work during inference, I annotate regions with significantly low similarities are annotated to indicate the predicted phoneme boundaries using dashed lines.

As shown in Fig. 5.3, the spectral changes can be approximately detected by the proposed model. This indicates that the model output contains positional phonetic information. Since the proposed phonotactic embedding is the concatenation of the mean and standard deviation vectors computed over frames in a segment, the above

comparison highlights the feasibility of incorporating phonetic and phonotactic information.

In addition, it is useful to note that the adjacent frames contain more rich and complex information, such as the time dependencies, the acoustics similarity, and the speaker identity among others. However, the aim of the use of self-supervised phoneme segmentation is to capture positional phoneme information but not exact phoneme classes. Here, other information like time dependencies should also be learned during training since the proposed PHO-LID mimics phone n-gram modeling in a statistical manner. However, the primary information encoded into the model via the phoneme segmentation task should be the positional information. Since the visualization of the predicted phoneme boundaries shown in Fig. 5.3 also indicates that the spectral changes can be approximately detected by the proposed model, the above implies that performance improvement compared to the vanilla model mostly results from the positional phoneme information.

5.7 Summary

This chapter proposed the PHO-LID model that incorporates the phonetic and phonotactic information for LID without the need for phoneme annotations. Compared to the CNN-Trans model, the proposed model with multi-task optimization achieves higher performance on AP17-OLR and NIST LRE 2017 data, and the performance confusion matrices indicate that the proposed method can effectively distinguish languages of the same cluster in NIST LRE 2017. The predicted phoneme boundaries are visualized using the output units of the shared CNN module. The comparison between the predicted phoneme boundaries and the corresponding audio spectrogram shows the existence of phoneme information, which, in turn, highlights the feasibility of the proposed method.

Part III

End-to-End Language Diarization

Chapter 6

End-to-End Language Diarization for Binlingual Code-Switching Speech

This chapter proposes an end-to-end BLSTM and x-vector self-attention (XSA) neural configurations, named BLSTM-E2E and XSA-E2E, respectively, for language diarization on bilingual code-switching speech.

Section 6.1 describes existing speaker diarization, language identification, and language diarization approaches, and illustrates the importance to develop an end-to-end language diarization method. Section 6.2 introduces an BLSTM-E2E model which is adapted from a speaker diarization configuration and the proposed XSA-E2E model. The datasets, model configurations and detailed experiments are introduced in Section 6.3. The results are described in Section 6.4. The proposed XSA-E2E model is then analyzed by visualizing the attention matrices within self-attention heads as will be presented in Section 6.5. Conclusions derived from of this work is given in Section 6.6.

6.1 Introduction

Language diarization (LD) is a special case of language identification (LID), where the task is to identify the languages of each utterance in a natural multilingual recording. This work addresses language diarization in the context of code-switching speech (cf e.g. [20]) where the same speaker may use more than one language, often within the same utterance.

Traditional speaker diarization methods, which aim to segment the speech signal and group together segments belonging to the same speaker, are traditionally based on a speaker encoding front-end, such as x-vectors [5], and a clustering back-end model [65, 66, 109–112]. These are inherently unsupervised methods, in the sense that they neither require nor leverage examples of segmented-and-labeled multi-speaker recordings [67]. End-to-end (E2E) speaker diarization methods, by contrast, require (and learn from) labeled multi-speaker audio to jointly train an integrated neural module for speaker encoding and clustering [67, 68, 113, 114].

Traditional LID methods, which assume that each audio segment presented to the system contains speech in one language that must be identified [1], are similarly based on a two-stage process [4, 5, 115], where a language embedding such as an x-vector is first extracted from the speech using a front-end encoder, and a separately trained classifier is then employed as the back-end to identify the language using the embedding. Again, by contrast, recently proposed E2E neural LID methods [11, 17, 116, 117] integrate this two-stage process into a single neural module. They work in a similar manner as E2E speaker diarization —initial layers of a deep neural network (DNN) generate an embedding for the input speech, and subsequent DNN layers perform language classification.

Note that language diarization cannot be done using LID methods due to the one-language-per-audio-sample assumption. Instead, inspired by E2E speaker diarization [68, 113] and the x-vector language encoder [5], two methods are proposed for E2E neural language diarization. The first approach uses a bidirectional long short-term memory neural network [118] to build an end-to-end LD model (BLSTM-E2E), as originally proposed for speaker diarization in [68]. This model is jointly trained for language diarization by minimizing the cross-entropy (CE) loss and a deep-clustering (DC) loss [69]. The second employs an x-vector network followed by a self-attention transformer encoder [23] to build an end-to-end LD

model (XSA-E2E). The x-vector layers encode frame-level features into a sequence of segment-level embeddings, from which the self-attention transformer generates a sequence of segment-level language labels. For completeness, the performance of these two models is compared with an intermediate model that uses only a self-attention module on top of the BLSTM (SA-E2E).

These approaches have three desirable properties. Firstly, language diarization is a multi-label classification problem, for which these models appropriately generate a sequence of segment-level labels for each test recording, where a segment consists of several adjacent frames, allowing the model to identify different languages within an utterance, detect the language change point, and tag the silences in a unified manner. Secondly, as opposed to diarization approaches that require pre-processing for speech-activity detection (SAD), these proposed models perform SAD implicitly by defining silence as an output label. Finally, hierarchical processing is employed in the proposed XSA-E2E model with the multi-objective training. This hierarchical processing captures local language information in each segment before establishing global dependency between input and output, which benefits language diarization. The proposed approaches are evaluated on the code-switching data set from the Shared Task B of the First Workshop on Speech Technologies for Code-Switching in Multilingual Communities (WSTCSMC 2020) [77], and on simulated data derived from the SEAME data set [75].

6.2 Proposed Methods

6.2.1 BLSTM-based end-to-end model (BLSTM-E2E)

As shown in Fig. 6.1, the BLSTM-E2E model that was originally proposed in [68] for speaker diarization is applied to language diarization. Consider a sequence of segments, where I define $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^B | t = 1, ..., T)$ as features extracted from those segments with T being the number of segments and B the dimension of the segment-level feature vector. The ground-truth language label sequence is defined as $\mathbf{Y} = (\mathbf{y}_t | t = 1, ..., T)$, where the class label $\mathbf{y}_t \in \{0, 1, ..., C\}$ given that 0 is the label corresponding to a silent segment when SAD task is included and C is the total number of languages. In this architecture, the first N BLSTM layers generate

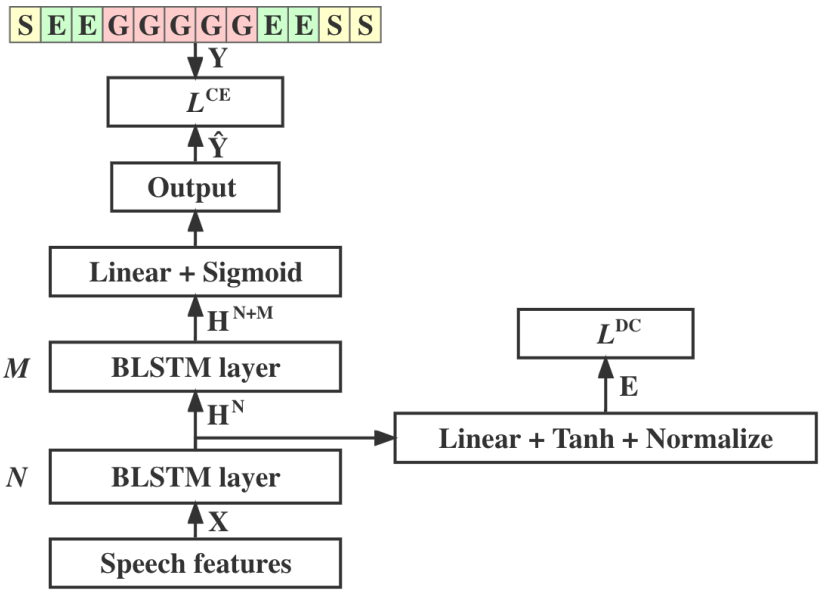


FIGURE 6.1: BLSTM end-to-end language diarization model with deep clustering loss.

language representations (embeddings) for each segment and the next M BLSTM layers estimate the label sequence for these embeddings.

To apply the DC loss [69], a D-dimensional embedding vector $\mathbf{e}_t$ is transformed from the hidden activations $\mathbf{h}_t^N$ of the N-th BLSTM layer. The permutation-free loss used in [68] was replaced with the cross-entropy loss and the multi-objective loss can be then computed via

$$L^{\mathrm{CE}} = \mathrm{CE}\left(\mathbf{Y}, \hat{\mathbf{Y}}\right), \tag{6.1}$$

$$L^{\mathrm{BLSTM}} = \alpha L^{\mathrm{CE}} + (1-\alpha)\, L^{\mathrm{DC}}. \tag{6.2}$$

Here, L denotes the loss, $\mathrm{CE}(\cdot, \cdot)$ denotes the cross-entropy loss function between the ground-truth label sequence $\mathbf{Y}$, the output $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}_1 \cdots \widehat{\mathbf{y}}_t]^\intercal$ of the BLSTM-E2E model and α is a scaling factor to facilitate multi-objective training.

6.2.2 Self-attention-based end-to-end model (SA-E2E)

The SA-E2E model is first explored as a preliminary method to understand the XSA-E2E model. This SA-E2E model employs the encoder module of the transformer in [23]. The SA-E2E model comprises the positional encoding, encoder

blocks, and a linear layer with sigmoid activation function. The input $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^B | t = 1, ..., T)$ of SA-E2E is the same as that of the BLSTM-E2E model and the output $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}_1 \cdots \widehat{\mathbf{y}}_t]^\intercal$ is computed via

$$\mathbf{Z} = \text{SelfAtten}\left(\mathbf{E}, \mathbf{Mask}\right), \tag{6.3}$$

$$\widehat{\mathbf{Y}} = \sigma\Big(\mathbf{W}_1\Big(\text{Concat}\big(\text{Mean}\left(\mathbf{Z}\right), \text{Std}\left(\mathbf{Z}\right)\big)\Big)\Big), \tag{6.4}$$

where $\text{SelfAtten}\left(\cdot, \cdot\right)$ denotes computations performed within the self-attention encoder module, $\mathbf{Z}$ is the output of the transformer encoder layers, and $\mathbf{Mask}$ is defined as the attention mask applied to the attention weight matrix. $\mathbf{W}_1$ denotes the linear layer after the final transformer encoder layer, σ is the sigmoid activation function. The architecture of the SA-E2E model is presented as the self-attention encoder module in Fig. 6.2.

6.2.3 X-vector self-attention end-to-end model (XSA-E2E)

The x-vector has shown to achieve high performance on short-utterance LID [5] by capturing local language information. The transformer, on the other hand, was proposed to draw global dependencies between input and output [23]. Considering the benefits of both techniques, the x-vectors is used to capture the local information of each 200 ms segment before generating an embedding. The transformers are then employed to take the temporal dynamics of the signal into account.

For a speech signal partitioned in segments, each $\mathbf{x}_t$ in the input sequence $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^{K \times F} | t = 1, ..., T)$ of the XSA-E2E model is a matrix $[\mathbf{f}_1, ..., \mathbf{f}_K]^\intercal$, where $\mathbf{f}_K$ is an F-dimensional frame-level feature vector of the K-th frame in segment t. The proposed XSA-E2E model operates in a hierarchical manner as shown in Fig. 6.2. It first processes frame-level into segment-level features before estimating the posterior for each segment.

Frame-level features of each $\mathbf{x}_t$ are encoded into a segment-level embedding through the x-vector embedding module. The use of x-vector embedding allows the algorithm to capture the language identity of each segment. Segment-level embeddings

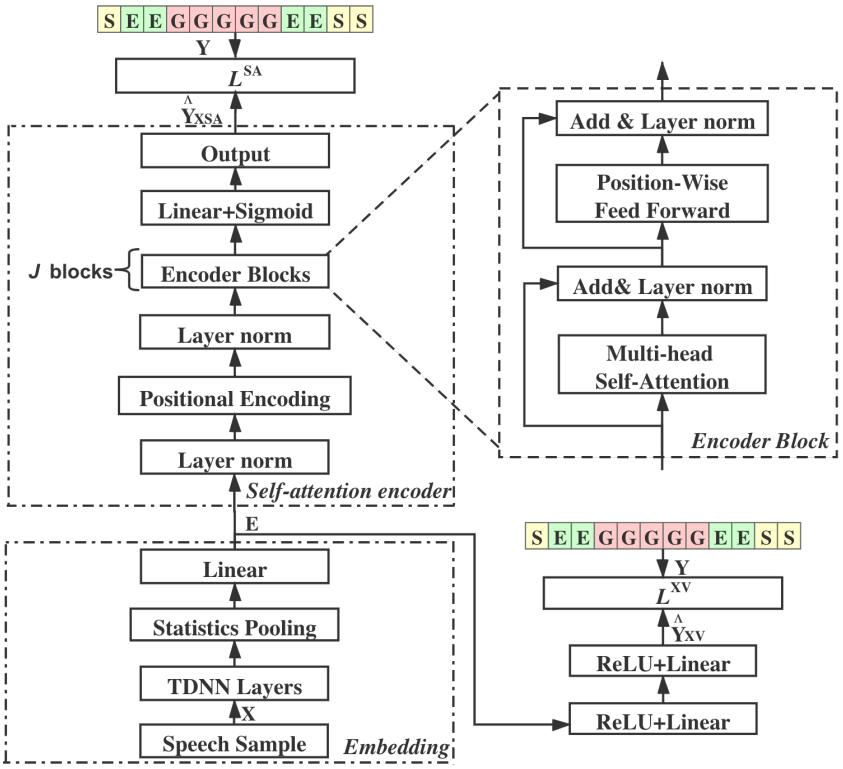


FIGURE 6.2: X-vector-Self-Attention end-to-end (XSA-E2E) language diarization model with multi-objective training.

are defined as $\mathbf{E} = (\mathbf{e}_t \in \mathbb{R}^D | t = 1, ..., T)$, and computed via

$$\mathbf{H} = \text{TDNN}(\mathbf{X}), \tag{6.5}$$

$$\mathbf{E} = \mathbf{W}_{Proj}\text{Concat}\big(\text{Mean}(\mathbf{H}), \text{Std}(\mathbf{H})\big), \tag{6.6}$$

where the above computations are similar to (3.1) and (3.2) corresponding to the embedding module in Fig. 6.2. The outputs of the x-vector model and the XSA-E2E model are then computed, respectively as

$$\widehat{\mathbf{Y}}_{\text{XSA}} = \text{Encoder}(\mathbf{E}), \tag{6.7}$$

$$\widehat{\mathbf{Y}}_{\text{XV}} = \mathbf{W}_3\delta\big(\mathbf{W}_2\delta(\mathbf{E})\big), \tag{6.8}$$

where $\text{Encoder}(\cdot)$ corresponds to (6.3) and (6.4). $\mathbf{W}_2$ and $\mathbf{W}_3$ denote computations within two linear layers for generating $\widehat{\mathbf{Y}}_{\text{XV}}$, δ is the ReLU activation function. XV and XSA denote the x-vector model and the XSA-E2E model in Fig. 6.2, respectively.

Similar to BLSTM-E2E, the proposed XSA-E2E employs a multi-objective training paradigm. Inspired by the cross-entropy loss function of the TDNN-based x-vector model in [5], the cross-entropy loss is adopted for the proposed XSA-E2E model. Consequently, the multi-objective loss function is computed as

$$L^{\mathrm{XV}} = \mathrm{CE}\left(\mathbf{Y}, \hat{\mathbf{Y}}_{\mathrm{XV}}\right), \tag{6.9}$$

$$L^{\mathrm{SA}} = \mathrm{CE}\left(\mathbf{Y}, \hat{\mathbf{Y}}_{\mathrm{XSA}}\right), \tag{6.10}$$

$$L^{\mathrm{XSA}} = \beta L^{\mathrm{XV}} + (1-\beta)\, L^{\mathrm{SA}}, \tag{6.11}$$

where $\mathbf{Y} = (\mathbf{y}_t | t = 1, ..., T)$ is the ground-truth language label sequence, β is the parameter for multi-objective training.

6.3 Experiments

6.3.1 Dataset

Experiments were conducted on two datasets. The first dataset is obtained from the shared task B in WSTCSMC 2020 comprising three code-switching language pairs: Gujarati-English (gu-en), Tamil-English (ta-en) and Telugu-English (te-en). The proposed model for each language was trained on its corresponding training set and evaluated on the development set. Detailed information of this dataset [77] is shown in Table 2.4. In addition, to verify the robustness and the extension capability to multiple classes, the proposed models were trained on all three language pairs to achieve a five-class language diarization task consisting of English, Gujarati, Tamil, Telugu, and silence. Test results are then reported on the development sets of these three code-switching pairs.

The algorithms were also validated on the SEAME dataset [75]. This SEAME dataset comprises 290 recordings consisting of 110,145 utterances out of which 24,438 are Mandarin, 28,655 are English, and 57,052 are code-switching utterances. Since there are no time-stamps corresponding to the language change for these Mandarin-English code-switching utterances, code-switching data was simulated by concatenating no more than five monolingual utterances from the same recording in chronological order. The maximum length of the final utterance was set to

TABLE 6.1: Utterances description of the simulated data using SEAME dataset

Simulated data	**Num.of classes in utterances**		
	one class	two classes	three classes
without silence	7556	4936	-
with silence	3271	8521	3921

50 s. A total of 18.7-hour of English, 18.1-hour of Mandarin and 5.7-hour of silence were used to build the simulated data. For the version without silence, 12,492 speech samples with an average duration of 10.6 s were used. For the simulated data with silence, silent segments are occasionally interleaved and 15,713 simulated speech samples are achieved with an average duration of 9.7 s. In contrast to the WSTCSMC 2020 dataset, in which each utterance comprises three classes, the simulated data utterances can contain one, two or three classes. The related information is shown in Table 6.1. In addition, since there is no partition on development or test, a ten-fold cross validation is conducted.

6.3.2 Experiment setup

6.3.2.1 Feature extraction

Since the ground-truth language labels of the data in WSTCSMC 2020 are assigned to 200 ms segments, the input speech sample is first partitioned into segments of the same duration. The remaining of the speech samples that cannot be divided exactly into 200ms segments was ignored. For each segment, 23-dimensional log-Mel-filterbank features are extracted with a 25 ms window and a 10 ms shift as [68] for all systems.

The 19-frame log-Mel-filterbank features are then processed into segment-level features in two different forms. The first directly concatenates features of all 19 frames into a 437-dimensional vector before feeding the vector into the proposed E2E models. The second employs a pruned x-vector model [5] as a feature extractor. A 256-dimensional embedding is extracted as the segment-level input of the encoding module.

6.3.2.2 Model configuration

The BLSTM-E2E language diarization model employs five BLSTM layers with 256 hidden units in each layer. The outputs of the second BLSTM layer are transformed into 256-dimensional vectors for the computation of DC loss and $\alpha = 0.5$ in (6.2). The Adam optimizer was used with an initial learning rate of 10^{-3} and cosine annealing learning rate decay [119]. The model was trained for 60 epochs with a batch size of 8.

For the XSA-E2E language diarization model, an x-vector model is applied followed by a self-attention encoder. The x-vector model is the same as that in [5] except that the third TDNN layer was removed to adapt to the length of segment and the dimension of the x-vector embedding $\mathbf{e}_t$ is given by $D = 256$. As shown in Fig. 6.2, the 256-dimensional x-vectors are fed into the self-attention encoder module. This module comprises four encoder blocks with four heads in each multi-head self-attention layer ($J = 4$) and a position-wise feed forward layer with 2048 hidden units in the inner-layer. The Adam optimizer with an initial learning rate of 10^{-4} was used with cosine annealing learning rate decay. The model was trained for 30 epochs with a batch size of 32.

The SA-E2E model which is equivalent to the self-attention encoder module of the proposed XSA-E2E model is also implemented as baseline. The Adam optimizer was applied to train the SA-E2E model for 60 epochs in total with an initial learning rate of 10^{-4} which decays after 10 warm-up epochs. The source code for this research is made publicly available in GitHub. [1]

All systems were evaluated using accuracy and equal error rate (EER) in accordance with Section 2.4 to compare with the models in WSTCSMC 2020 [77].

6.4 Results

6.4.1 Evaluation on WSTCSMC 2020 shared task B

Results for the shared task B in WSTCSMC 2020 are shown in Table 6.2. The baseline models include DeepSpeech2 [77, 120] and Vocapia-LIMSI system [121], where

[1] https://github.com/Lhx94As/E2E-langauge-diarization

TABLE 6.2: Comparison of the proposed approaches with DeepSpeech2 system and Vocapia-LIMSI system on the dev set of the shared task B in WSTCSMC 2020 by employing EER (%) and Accuracy (%)

Method	**gu-en**		**ta-en**		**te-en**		**Average**	
	EER↓	Acc.↑	EER↓	Acc.↑	EER↓	Acc.↑	EER↓	Acc.↑
DeepSpeech2	6.7	76.7	6.5	77.6	6.7	76.5	6.6	76.9
Vocapia-LIMSI	-	80.5	-	81.2	-	81.8	-	81.2
BLSTM-E2E	6.1	81.8	5.9	82.4	**5.7**	**82.8**	5.9	82.3
SA-E2E	6.4	80.9	6.1	81.6	6.0	81.9	6.2	81.5
XSA-E2E	**5.8**	**82.7**	**5.9**	**82.4**	5.8	82.6	**5.8**	**82.6**

TABLE 6.3: Comparison of the proposed approaches on 3-language-pair code-switching data in WSTCSMC 2020 by employing EER (%) of each language and Accuracy (%)

Method	en	gu	ta	te	silence	Acc.↑
BLSTM-E2E	6.27	3.94	3.55	3.52	**2.97**	80.15
SA-E2E	6.33	3.59	3.73	3.65	3.49	79.21
XSA-E2E	**5.99**	**2.98**	**3.21**	**3.05**	3.56	**81.20**

the latter is a fusion system of an unsupervised GMM-based i-vector model [4] and a phonotactic model. As shown in Table 6.2, both BLSTM-E2E and XSA-E2E models outperformed baseline algorithms on this dataset; the proposed BLSTM-E2E system achieved the best performance on Telugu-English code-switching data while the XSA-E2E system achieved the best performance on Gujarati-English and Tamil-English code-switching data.

To validate the performance of the proposed models under a more challenging condition, code-switching data of all three language pairs were pooled together as the training data. The results presented in Table 6.3 show that the XSA-E2E model achieves the highest overall accuracy among the three proposed models and lowest EER on all classes except silence. It is worth noting that although the SA-E2E model suffers from the worst overall performance, it achieves similar performance to that of BLSTM-E2E on four language classes. It also achieves worse performance on silence segments than BLSTM-E2E. These results imply that the self-attention mechanism may not perform as well as the BLSTM model for data with silence.

TABLE 6.4: 10-fold cross validation results of the proposed approaches on simulated code-switching data using SEAME dataset by employing EER (%) and Accuracy (%). EER for the simulated data with silence is composed of EER for English (Eng), EER for Mandarin (Man), and EER for Silence (Sil)

Method	**Simulated data using SEAME**					
	no silence		with silence			
	EER↓	Acc.↑	Eng	Man	Sil	Acc.↑
BLSTM-E2E	7.20	85.60	6.47	6.37	**0.93**	86.23
SA-E2E	7.15	85.71	6.47	6.25	1.67	85.60
XSA-E2E	**5.08**	**89.84**	**5.06**	**4.91**	2.38	**87.66**

The performance of XSA-E2E is reduced further compared to SA-E2E by the silent segments. This is not surprising given that about 20% of the data from WSTCSMC 2020 is silence [77]. Moreover, English only accounts for 12% of this dataset, leading to high EER on English for all models in this experiment.

6.4.2 Evaluation on simulated code-switching data using SEAME dataset

The approaches were evaluated on two types of simulated data. The results presented in Table 6.4 show that the proposed XSA-E2E system achieves the best performance on both types of simulated data in this evaluation. In addition, both XSA-E2E and SA-E2E models which employ the self-attention mechanism perform worse on silence segments than the BLSTM-E2E model. After including silences in the simulated data, XSA-E2E and SA-E2E exhibit degradation of accuracy when compared with the BLSTM-E2E model. The XSA-E2E model suffers from the highest EER on the silence data. These observations are consistent with results presented in Table 6.2 and Table 6.3.

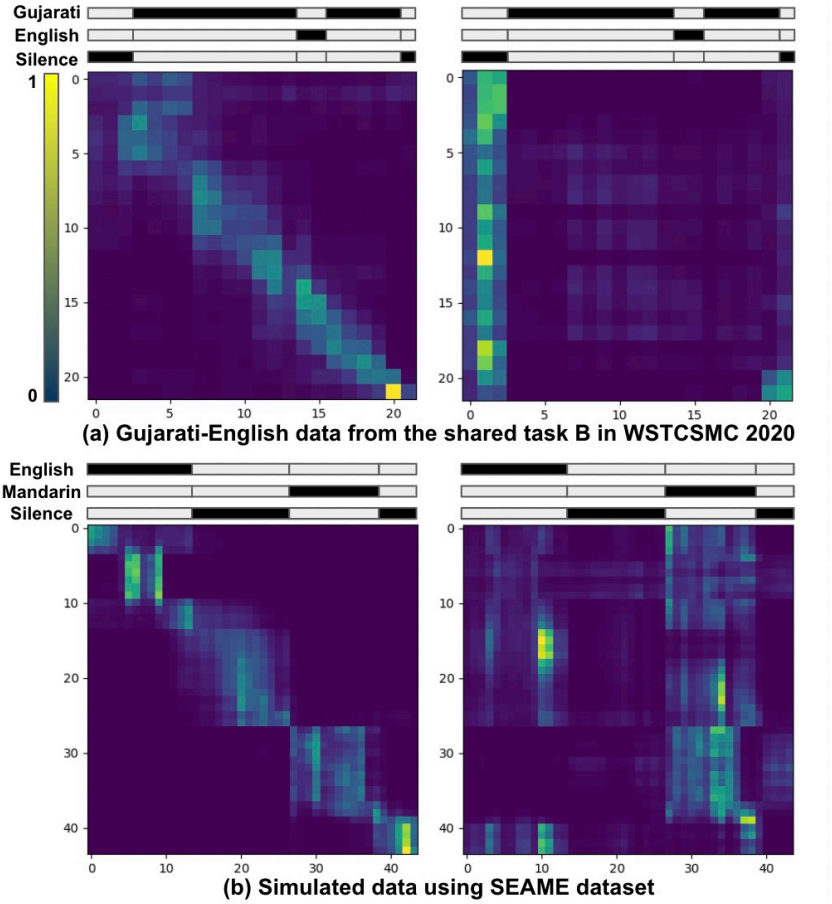


FIGURE 6.3: Attention weights at the second encoder block of the XSA-E2E model trained on (a) Gujaratu-English code-switching data and (b) simulated data using SEAME dataset with silence.

6.5 Analysis and visualization of self-attention heads

To investigate the operation of the self-attention mechanism for language diarization task, attention weight matrices of two attention heads of the second encoder block in the XSA-E2E model are analyzed as shown in Fig. 6.3. The attention weights in the left head lead to a linear transformation, while the right head horizontally exhibits different color depths for different classes. This implies that the self-attention mechanism in the proposed XSA-E2E model is able to capture the language identity for language diarization and speech activity detection.

In addition, Fig. 6.3 shows how data composition influences the proposed XSA-E2E model. The XSA-E2E model trained on the simulated version of the SEAME dataset shows clearer boundaries of different classes in the right head than trained

on the Gurajarati-English data. This is due to the simulated data being more balanced than the the WSTCSMC 2020 dataset resulting in higher accuracy and lower EER.

6.6 Summary

This chapter proposed the BLSTM-E2E and the XSA-E2E models for the language diarization task on bilingual code-switching speech. The proposed XSA-E2E model improves the state-of-the-art performance on the development set of the shared task B in WSTCSMC 2020 and achieves the best performance on simulated data derived from the SEAME dataset. Compared to SA-E2E model, the proposed models that employ embedding-related loss for joint training achieve higher performance in most experiments. Compared to the SA-E2E model, the XSA-E2E also achieves higher performance with a hierarchical processing. These underpin the importance of the local information in each segment for the language diarization task. Results also highlight that both x-vector and self-attention mechanisms can perform higher on data with less silence. This chapter also shows how self-attention captures the language characteristics through the attention weights. The proposed model may be employed as a preprocessing module of a multilingual speech recognition system. In addition, the research into improving the performance of language diarization systems on silence frames can be interesting as future work.

Part IV

Conclusion

Chapter 7

Conclusions and Future Work

In this thesis, three approaches were first proposed to improve LID from configuration and language cues perspectives, respectively. Two end-to-end models were next proposed to extend LID to a code-switching scenario, language diarzation (LD).

The first approach improves the duration robustness of the proposed XSA-LID model by improving its model configuration, where a dual-mode framework and knowledge distillation (KD) were utilized. The model with the dual-mode framework achieves higher performance on long speech with modest performance improvement on short speech. The LID performance on short speech is enhanced by further applying KD to the two modes. Compared to the vanilla LID model, the proposed dual-mode XSA-LID model achieves significant performance improvement on both long and short speech in the MLS14 data of NIST LRE 2017. However, the performance on short speech is not comparable to that of the x-vector method, which should be further improved. The models were evaluated with different Boolean masks used to mimic short speech clips. The results suggest that while using random-location masks leads to higher performance compared to using fixed-location masks on short speech, the model with a fixed-location mask exhibits higher performance on long speech. This is because random-location masks introduce clip-wise linguistic variability and fixed-location masks maintain the clip-wise lexical integrity during training. The effect of different Boolean masks regarding locations and lengths could be further explored to adopt the proposed method to diverse scenarios.

The second and third proposed approaches aim to improve language cues which are then used to develop LID models. The second approach introduces two methods to compute efficient self-supervised learning speech representations and reduce the irrelevant information of these representations in LID. The first is the attentive squeeze-and-excitation (SE) block for dimension-wise scaling and the second is the linear bottleneck (LBN) block that leads to irrelevant information loss through nonlinear dimension reduction. The proposed approaches are evaluated in conjunction with the XSA-LID model on the AP19-OLR and NIST LRE 2017 data. The LBN-XSA model exhibits higher LID performance than other approaches and outperforms the XSA-LID model trained on the fine-tuned features. As opposed to the fine-tuning process which should be employed prior to the feature extraction and model training, the proposed LBN approach is trained with the model jointly, and thus, is of less computational complexity and need fewer computing resources

As opposed to the second approach, the third proposed method incorporates two language cues, acoustic and phonotactic information, in a unified PHO-LID model without using phoneme annotations for training. This model comprises LID and phoneme segmentation branches that share the same encoder module. While the LID branch performs fully supervised training, the phoneme segmentation branch is optimized in a self-supervised manner. The proposed model is evaluated on AP17-OLR and NIST LRE 2017 data. Compared to other approaches which only utilize acoustic features during training, the proposed method exhibits higher performance with significantly higher performance on long speech due to incorporating phonotactic information within the model. However, phoneme combinations, as a phonotactic pattern, have shown to be effective for language identification. Therefore, it is useful to investigate unsupervised learning for phoneme classification and employ it to enhance language identification.

Language identification is entended to diarization in the fourth proposed approach. This work investigated two end-to-end language diarization configurations for LD. The first is the BLSTM-E2E model adapted from speaker diarization and the second is the XSA-E2E model built upon the XSA-LID model which was originally proposed for LID. These two models encode language information of each speech unit in a high-dimensional vector via an encoder module before feeding these vectors into the back-end classification module to generate language decisions. Experiment results show the effectiveness of the proposed models, and the XSA-E2E

model exhibits the highest diarization performance among all models under consideration. Nevertheless, the training process requires language annotations in a high granularity (i.e., 200ms), which is not desirable due to the limitation of data resources.

Although the proposed approaches in this thesis have shown to be effective in improving duration robustness and language cues, other potential problems regarding LID are still challenging. Considering that practical LID systems are expected to be robust in a noisy scenario, it is helpful to explore an effective method for developing a noise-robust LID system for noisy speech.

In addition, language information has shown to be crucial in existing code-switching ASR works [122, 123]. We note that inter-sentence language switches, also referred to as inter-sentence code switches, can be identified by performing LID. However, it is challenging to detect intra-sentence language switches. This is because LID performance degrades with the test speech becoming shorter while intra-sentence language switches should be detected in the frame or token level to avoid degrading CS-ASR performance. Since the proposed LD approach in this thesis performs LID in 200 ms speech units and exhibits high performance, it is possible to investigate incorporating this LD process in an end-to-end CS-ASR to improve its performance [124, 125]. Apart from adopting LD as a front-end of the CS-ASR model, it is also valuable for investigating optimizing CS-ASR and LID jointly to encode language information into a CS-ASR model.

List of Author's Awards, Patents, and Publications

Journal Articles

- **Hexin Liu**, Leibny Paola Garcia Perera, Andy W. H. Khong, Eng Siong Chng, Suzy J. Styles, Sanjeev Khudanpur, "Efficient Self-supervised Learning Representations for Spoken Language Identification" *IEEE Journal of Selected Topics in Signal Processing, vol. 16, no. 6, pp. 1296–1307, 2022.*

Conference Proceedings

- **Hexin Liu**, Leibny Paola Garcia Perera, Xinyi Zhang, Justin Dauwels, Andy W. H. Khong, Sanjeev Khudanpur, Suzy J. Styles, "End-to-End Language Diarization for Bilingual Code-Switching Speech," in *Proc. Interspeech, 2021.*

- **Hexin Liu**, Leibny Paola Garcia Perera, Andy W. H. Khong, Justin Dauwels, Suzy J. Styles, Sanjeev Khudanpur, "Enhancing Language Identification using Dual-mode Model with Knowledge Distillation," in *Proc. The Speaker and Language Recognition Workshop (Odyssey), 2022.*

- **Hexin Liu**, Leibny Paola Garcia Perera, Andy W. H. Khong, Suzy J. Styles, Sanjeev Khudanpur, "PHO-LID: A Unified Model Incorporating Acoustic-Phonetic and Phonotactic Information for Language Identification," in *Proc. Interspeech, 2022.*

- **Hexin Liu**, Haihua Xu, Leibny Paola Garcia Perera, Andy W. H. Khong, Yi He, Sanjeev Khudanpur, "Reducing Language Confusion for Code-switching Speech Recognition with Token-level Language Diarization," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, 2023.*

- Shuyue Stella Li, Xiangyu Zhang, Shu Zhou, Hongchao Shu, Ruixing Liang, **Hexin Liu**, Leibny Paola Garcia Perera, "PQLM - Multilingual Decentralized Portable Quantum Language Model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, 2023.*

- Yi Han Victoria Chua*, **Hexin Liu***, Leibny Paola Garcia Perera, Fei Ting Wong, Jinyi Wong, Xiangyu Zhang, Sanjeev Khudanpur, Andy W. H. Khong, Justin Dauwels, Suzy J. Styles, " MERLIon CCS Challenge: A English-Mandarin code-switching child-directed speech corpus for language identification and diarization," in *Proc. Interspeech, 2023.*

- Suzy J. Styles, Yi Han Victoria Chua, Fei Ting Wong, **Hexin Liu**, Leibny Paola Garcia Perera, Sanjeev Khudanpur, Andy W. H. Khong, Justin Dauwels, "Investigating model performance in language identification: beyond simple error statistics," in *Proc. Interspeech, 2023.*

- Rui Li, Zhiwei Xie, Haihua Xu, Yizhou Peng, **Hexin Liu**, Hao Huang, Eng Siong Chng, "Self-supervised Learning Representation based Accent Recognition with Persistent Accent Memory," in *Proc. Interspeech, 2023.*

Bibliography

[1] H. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," *Proc. IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013. 1, 2, 5, 15, 17, 78, 92

[2] A. Waters, N. Gaur, P. Haghani, P. Moreno, and Z. Qu, "Leveraging Language ID in Multilingual End-to-End Speech Recognition," in *Proc. IEEE Workshop Autom. Speech Recognit. and Understanding*, 2019, pp. 928–935. 1

[3] R. Lahiri, K. Kumatani, E. Sun, and Y. Qian, "Multilingual speech recognition using knowledge transfer across learning processes," *arXiv preprint arXiv:2110.07909*, 2021. 1

[4] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proc. Interspeech*, 2011, pp. 857–860. 1, 2, 8, 51, 84, 92, 100

[5] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors," in *Proc. Odyssey*, 2018, pp. 105–111. 1, 2, 10, 11, 19, 42, 52, 61, 83, 92, 95, 97, 98, 99

[6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, 2011. 1, 8

[7] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Proc. Interspeech*, 2017, pp. 999–1003. 1

[8] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5329–5333. 2, 10, 21

[9] S. O. Sadjadi, T. Kheyrkhah, C. S. Greenberg, E. Singer, D. A. Reynolds, L. P. Mason, and J. Hernandez-Cordero, "Performance analysis of the 2017 NIST language recognition evaluation," in *Proc. Interspeech*, 2018, pp. 1798–1802. 2, 23, 60

[10] F. Richardson, P. Torres-Carrasquillo, J. Borgstrom, D. Sturim, Y. Gwon, J. Villalba, J. Trmal, N. Chen, R. Dehak, and N. Dehak, "The MIT Lincoln Laboratory / JHU / EPITA-LSE LRE17 System ," in *Proc. Odyssey*, 2018, pp. 54–59. 2, 11

[11] X. Miao, I. McLoughlin, and Y. Yan, "A new time-frequency attention mechanism for TDNN and CNN-LSTM-TDNN, with application to language identification," in *Proc. Interspeech*, 2019, pp. 4080–4084. 2, 32, 92

[12] J. Gonzalez-Dominguez, I. Lopez-Moreno, P. J. Moreno, and J. Gonzalez-Rodriguez, "Frame by frame language identification in short utterances using deep neural networks," *Neural Networks Special Issue: Neural Network Learning in Big Data*, 2014. 2, 32

[13] J. Zhao, H. Shu, L. Zhang, X. Wang, Q. Gong, and P. Li, "Cortical competition during language discrimination," *NeuroImage*, vol. 43, no. 3, pp. 624–633, 2008. 2

[14] Y. Muthusamy, E. Barnard, and R. Cole, "Reviewing automatic language identification," *IEEE Signal Process. Mag.*, vol. 11, no. 4, pp. 33–41, 1994. 2, 51, 78

[15] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech Audio Process.*, vol. 4, no. 1, pp. 31–44, 1996. 2, 3, 15, 51, 78

[16] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, no. 4, pp. 357–366, 1980. 2

[17] W. Cai, D. Cai, S. Huang, and M. Li, "Utterance-level end-to-end language identification using attention-based CNN-BLSTM," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5991–5995. 2, 8, 13, 51, 92

[18] R. Tong, B. Ma, H. Li, E. S. Chng, and K.-A. Lee, "Target-aware language models for spoken language recognition," in *Proc. Interspeech*, 2009, pp. 200–203. 3, 16

[19] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "Wav2vec: Unsupervised pre-training for speech recognition," in *Proc. Interspeech*, 2019, pp. 3465–3469. 3, 19, 52

[20] D. Lyu, C. E. Siong, and H. Li, "Language diarization for code-switch conversational speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 7314–7318. 3, 92

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *arXivpreprint. arXiv 1503.02531*, 2015. 4, 32, 39

[22] H. Liu, L. P. G. Perera, X. Zhang, J. Dauwels, A. W. H. Khong, S. Khudanpur, and S. J. Styles, "End-to-end language diarization for bilingual code-switching speech," in *Proc. Interspeech*, 2021, pp. 1489–1493. 4

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008. 4, 35, 36, 42, 54, 78, 81, 83, 92, 94, 95

[24] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. NeurIPS*, vol. 33, 2020, pp. 12 449–12 460. 4, 19, 20, 52, 53, 83

[25] 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, pp. 74–81. 8

[26] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011, pp. 249–252. 9

[27] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 1695–1699. 9

[28] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Process. Letters*, vol. 22, no. 10, pp. 1671–1675, 2015. 9

[29] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. on Computer Vision Pattern Recognit.*, 2017, pp. 4700–4708. 11

[30] 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. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5796–5800. 11

[31] D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, and S. Khudanpur, "The JHU speaker recognition system for the VOiCES 2019 challenge," in *Proc. Interspeech*, 2019, pp. 2468–2472. 11

[32] M. K. Nandwana, J. van Hout, C. Richey, M. McLaren, M. A. Barrios, and A. Lawson, "The VOiCES from a distance challenge 2019," in *Proc. Interspeech*, 2019, pp. 2438–2442. 11

[33] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. on Computer Vision Pattern Recognit.*, 2018, pp. 7132–7141. 11, 54, 56

[34] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834. 11, 56, 59

[35] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003. 12

[36] A. Lozano-Diez, R. Zazo-Candil, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "An end-to-end approach to language identification in short utterances using convolutional neural networks," in *Proc. Interspeech*, 2015, pp. 403–407. 12

[37] W. Geng, W. Wang, Y. Zhao, X. Cai, and B. Xu, "End-to-end language identification using attention-based recurrent neural networks," in *Proc. Interspeech*, 2016, pp. 2944–2948. 12

[38] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5189–5193. 13

[39] Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic temporal neural model for language identification," *IEEE/ACM Trans. Audio Speech Language Process.*, vol. 26, no. 1, pp. 134–144, 2017. 14, 84

[40] M. Zhao, R. Li, S. Yan, Z. Li, H. Lu, S. Xia, Q. Hong, and L. Li, "Phone-aware multi-task learning and length expanding for short-duration language recognition," in *Proc. APSIPA ASC*, 2019, pp. 433–437. 14

[41] Z. Li, M. Zhao, J. Li, Y. Zhi, L. Li, and Q. Hong, "The XMUSPEECH system for the AP19-OLR challenge," in *Proc. Interspeech*, 2020, pp. 452–456. 14

[42] P. Matejka, P. Schwarz, J. Cernockỳ, and P. Chytil, "Phonotactic language identification using high quality phoneme recognition." in *Proc. Interspeech*, 2005, pp. 2237–2240. 15

[43] R. Tong, B. Ma, H. Li, and C. E. Siong, "A target-oriented phonotactic front-end for spoken language recognition," *IEEE/ACM Trans. Audio Speech Language Process.*, vol. 17, pp. 1335–1347, 2009. 15

[44] C. Salamea, L. F. D'Haro, R. Cordoba, and R. San-Segundo, "On the use of phone-gram units in recurrent neural networks for language identification," in *Proc. Odyssey*, 2016, pp. 117–123. 16

[45] D. Romero, L. F. D'Haro, M. Estecha-Garitagoitia, and C. Salamea, "Phonotactic language recognition using a universal phoneme recognizer and a transformer architecture," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 6872–6876. 16

[46] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016. 16

[47] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. ACL*, Aug. 2016, pp. 1715–1725. 16

[48] M. Ashby and J. Maidment, *Introducing phonetic science*. Cambridge University Press, 2005. 17

[49] R. Tong, B. Ma, D. Zhu, H. Li, and E. S. Chng, "Integrating acoustic, prosodic and phonotactic features for spoken language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2006, pp. 205–208. 17

[50] R. W. M. Ng, C.-C. Leung, T. Lee, B. Ma, and H. Li, "Prosodic attribute model for spoken language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2010, pp. 5022–5025. 17

[51] R. W. M. Ng, C.-C. Leung, V. Hautamäki, T. Lee, B. Ma, and H. Li, "Towards long-range prosodic attribute modeling for language recognition," in *Proc. Interspeech*, 2010, pp. 1792–1795. 17

[52] T. Dunning, *Statistical identification of language*. Computing Research Laboratory, New Mexico State University Las Cruces, 1994. 17

[53] S. Mendoza, L. Gillick, Y. Ito, S. Lowe, and M. Newman, "Automatic language identification using large vocabulary continuous speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, 1996, pp. 785–788. 18

[54] T. Schultz, I. Rogina, and A. Waibel, "Lvcsr-based language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, 1996, pp. 781–784. 18

[55] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," in *Proc. Interspeech*, 2021, pp. 1509–1513. 18, 52, 65

[56] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," in *Proc. Interspeech*, 2021, pp. 2426–2430. 18, 19, 52, 54, 65, 83

[57] A. Tjandra, D. G. Choudhury, F. Zhang, K. Singh, A. Conneau, A. Baevski, A. Sela, Y. Saraf, and M. Auli, "Improved language identification through cross-lingual self-supervised learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 6877–6881. 18, 20

[58] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "i-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013. 19

[59] P. Matejka, L. Zhang, T. Ng, O. Glembek, J. Z. Ma, B. Zhang, and S. H. Mallidi, "Neural network bottleneck features for language identification." in *Proc. Odyssey*, 2014, pp. 299–304. 19

[60] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Trans. Audio Speech Language Process.*, vol. 29, pp. 3451–3460, 2021. 19

[61] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," in *Proc. Interspeech*, 2021, pp. 1509–1513. 20

[62] D.-C. Lyu and R.-Y. Lyu, "Language identification on code-switching utterances using multiple cues," in *Proc. Interspeech 2008*, 2008, pp. 711–714. 21

[63] D.-C. Lyu, E.-S. Chng, and H. Li, "Language diarization for code-switch conversational speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 7314–7318. 21

[64] S. V, V. Thenkanidiyoor, and D. A. D, "SVM based language diarization for code-switched bilingual indian speech using bottleneck features," in *Proc. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*, 2018, pp. 132–136. 21

[65] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 4930–4934. 21, 92

[66] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with LSTM," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5239–5243. 21, 92

[67] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6301–6305. 22, 92

[68] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proc. Interspeech*, 2019, pp. 4300–4304. 22, 92, 93, 94, 98

[69] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 31–35. 22, 92, 94

[70] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: A resource for the next generations of speech-to-text," in *LREC*, vol. 4, 2004, pp. 69–71. 23, 41, 60, 82

[71] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 1, 1992, pp. 517–520. 23, 41, 60, 82

[72] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. Greenberg, D. Reynolds, E. Singer, L. Mason, and J. Hernandez-Cordero, "The 2017 NIST language recognition evaluation," in *Proc. Odyssey*, 2018, pp. 82–89. 23, 82

[73] Z. Tang, D. Wang, Y. Chen, and Q. Chen, "AP17-OLR challenge: Data, plan, and baseline," in *Proc. APSIPA ASC*, 2017, pp. 749–753. 24, 82, 84

[74] Z. Tang, D. Wang, and L. Song, "AP19-OLR challenge: Three tasks and their baselines," in *Proc. APSIPA ASC*, 2019, pp. 1917–1921. 24, 60

[75] D.-C. Lyu, T. P. Tan, C. E. Siong, and H. Li, "SEAME: A Mandarin-English code-switching speech corpus in south-east asia," in *Proc. Interspeech*, 2010, pp. 1986–1989. 25, 93, 97

[76] Z. Zeng, Y. Khassanov, V. T. Pham, H. Xu, E. S. Chng, and H. Li, "On the End-to-End Solution to Mandarin-English Code-Switching Speech Recognition," in *Proc. Interspeech*, 2019, pp. 2165–2169. 26

[77] S. Shah, S. Sitaram, and R. Mehta, "First workshop on speech processing for code-switching in multilingual communities: Shared task on code-switched spoken language identification," *WSTCSMC 2020*, p. 24, 2020. 26, 28, 93, 97, 99, 101

[78] A. F. Martin and A. N. Le, "NIST 2007 language recognition evaluation," in *Proc. Odyssey*, 2008, p. paper 16. 27, 84

[79] A. Lozano-Diez, R. Zazo-Candil, J. Gonzalez-Dominguez, D. Toledano, and J. González-Rodríguez, "An end-to-end approach to language identification in short utterances using convolutional neural networks," in *Proc. Interspeech*, 2015. 32

[80] S. Fernando, V. Sethu, E. Ambikairajah, and J. Epps, "Bidirectional modelling for short duration language identification." in *Proc. Interspeech*, 2017. 32

[81] P. Shen, X. Lu, S. Li, and H. Kawai, "Feature representation of short utterances based on knowledge distillation for spoken language identification." in *Proc. Interspeech*, 2018, pp. 1813–1817. 32

[82] ——, "Interactive learning of teacher-student model for short utterance spoken language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5981–5985. 32, 33

[83] P. Shen, X. Lu, K. Sugiura, S. Li, and H. Kawai, "Compensation on x-vector for short utterance spoken language identification," in *Proc. Odyssey*, 2020, pp. 47–52. 32

[84] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-y. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6381–6385. 32

[85] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode ASR: Unify and improve streaming ASR with full-context modeling," in *Proc. Int. Conf. Learn. Represent.*, 2021. 32, 33

[86] M. Rouvier, P.-M. Bousquet, and J. Duret, "Study on the temporal pooling used in deep neural networks for speaker verification," in *Proc. EUSIPCO*, 2021, pp. 501–505. 34

[87] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016. 35, 59, 72

[88] 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 Workshop Autom. Speech Recognit. and Understanding*, 2011. 42, 61, 83

[89] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020. 42

[90] P. Matejka, L. Zhang, T. Ng, O. Glembek, J. Z. Ma, B. Zhang, and S. H. Mallidi, "Neural network bottleneck features for language identification," in *Proc. Odyssey*, 2014. 52

[91] G. Ramesh, C. S. Kumar, and K. S. R. Murty, "Self-supervised phonotactic representations for language identification," in *Proc. Interspeech*, 2021, pp. 1514–1518. 52

[92] S.-W. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K.-T. Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "SUPERB: Speech processing universal performance benchmark," in *Proc. Interspeech*, 2021, pp. 1194–1198. 52, 61

[93] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition," *arXiv preprint arXiv:2105.11084*, 2021. 52, 65, 73

[94] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit.*, 2016, pp. 770–778. 53, 54, 59

[95] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit.*, 2018, pp. 4510–4520. 53, 54, 58

[96] A. Tjandra, D. G. Choudhury, F. Zhang, K. Singh, A. Conneau, A. Baevski, A. Sela, Y. Saraf, and M. Auli, "Improved language identification through cross-lingual self-supervised learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 6877–6881. 54

[97] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino *et al.*, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," *arXiv preprint arXiv:2111.09296*, 2021. 54

[98] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. Interspeech*, 2018, pp. 2252–2256. 56

[99] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A large-scale multilingual dataset for speech research," in *Proc. Interspeech*, 2020, pp. 2757–2761. 61, 83

[100] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019. 61, 83

[101] M. J. Gales, K. M. Knill, A. Ragni, and S. P. Rath, "Speech recognition and keyword spotting for low-resource languages: BABEL project research at CUED," in *Proc. Int. Workshop Spoken Language Tech. for Under-resourced Languages*, 2014, pp. 16–23. 61, 83

[102] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. Python in Science Conf.*, vol. 8, 2015, pp. 18–25. 61

[103] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods in Natural Language Process.: System Demonstrations*, 2020, pp. 38–45. 64

[104] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2015, pp. 448–456. 72

[105] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *arXiv preprint arXiv:2110.13900*, 2021. 73

[106] F. Kreuk, J. Keshet, and Y. Adi, "Self-supervised contrastive learning for unsupervised phoneme segmentation," in *Proc. Interspeech*, 2020, pp. 3700–3704. 78, 80, 81, 87

[107] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304. 79

[108] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018. 79

[109] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 10, pp. 2015–2028, 2013. 92

[110] G. Sell and D. Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *Proc. IEEE Spoken Language Technology Workshop*, 2014, pp. 413–417. 92

[111] H. Ning, M. Liu, H. Tang, and T. S. Huang, "A spectral clustering approach to speaker diarization," in *Ninth International Conference on Spoken Language Processing*, 2006. 92

[112] M. Diez, F. Landini, L. Burget, J. Rohdin, A. Silnova, K. Žmolíková, O. Novotný, K. Veselý, O. Glembek, O. Plchot, L. Mošner, and P. Matějka, "BUT system for DIHARD speech diarization challenge 2018," in *Proc. Interspeech*, 2018, pp. 2798–2802. 92

[113] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *Proc. IEEE Workshop Autom. Speech Recognit. and Understanding*, 2019, pp. 296–303. 92

[114] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "Generative adversarial speaker embedding networks for domain robust end-to-end speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6226–6230. 92

[115] V. Mingote, D. Castan, M. McLaren, M. K. Nandwana, A. O. Giménez, E. Lleida, and A. Miguel, "Language recognition using triplet neural networks." in *Proc. Interspeech*, 2019, pp. 4025–4029. 92

[116] B. Padi, A. Mohan, and S. Ganapathy, "Attention based hybrid i-vector BLSTM model for language recognition." in *Proc. Interspeech*, 2019, pp. 1263–1267. 92

[117] L. Wan, P. Sridhar, Y. Yu, Q. Wang, and I. L. Moreno, "Tuplemax loss for language identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5976–5980. 92

[118] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005. 92

[119] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015. 99

[120] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 173–182. 99

[121] B. Claude, L. Viet-Bac, and G. Jean-Luc, "Vocapia-limsi system for 2020 shared task on code-switched spoken language identification," in *The First Workshop on Speech Technologies for CodeSwitching in Multilingual Communities*, 2020. 99

[122] Z. Zeng, Y. Khassanov, V. T. Pham, H. Xu, E. S. Chng, and H. Li, "On the end-to-end solution to Mandarin-English code-switching speech recognition," in *Proc. Interspeech*, 2019, pp. 2165–2169. 109

[123] X. Zhou, E. Yılmaz, Y. Long, Y. Li, and H. Li, "Multi-Encoder-Decoder Transformer for Code-Switching Speech Recognition," in *Proc. Interspeech*, 2020, pp. 1042–1046. 109

[124] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012. 109

[125] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1240–1253, 2017. 109