Aalto University
MASTER'S THESIS 2020

Deep learning for spoken language identification

Matias Erik Lindgren

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.
Espoo, on 26th April 2020.

Supervisor: Professor Mikko Kurimo
Advisor: PhD Tommi Jauhiainen

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Author
Matias Erik Lindgren

Title
Deep learning for spoken language identification

School School of Science

Master's programme Computer, Communication and Information Sciences

Major Computer Science **Code** SCI3042

Supervisor Professor Mikko Kurimo

Advisor PhD Tommi Jauhiainen

Level Master's thesis **Date** 26th April 2020 **Pages** 10+91 **Language** English

Abstract

This thesis applies deep learning based classification techniques to identify natural languages from speech. The primary motivation behind this thesis is to implement accurate techniques for segmenting multimedia materials by the languages spoken in them.

Several existing state-of-the-art, deep learning based approaches are discussed and a subset of the discussed approaches are selected for quantitative experimentation. The selected model architectures are trained on several well-known spoken language identification datasets containing several different languages. Segmentation granularity varies between models, some supporting input audio lengths of 0.2 seconds, while others require 10 second long input to make a language decision.

Results from the thesis experiments show that an unsupervised representation of acoustic units, produced by a deep sequence-to-sequence autoencoder, cannot reach the language identification performance of a supervised representation, produced by a multilingual phoneme recognizer. Contrary to most existing results, in this thesis, acoustic-phonetic language classifiers trained on labeled spectral representations outperform phonotactic classifiers trained on bottleneck features of a multilingual phoneme recognizer. More work is required, using transcribed datasets and automatic speech recognition techniques, to investigate why phoneme embeddings did not outperform simple, labeled spectral features.

While an accurate online language segmentation tool for multimedia materials could not be constructed, the work completed in this thesis provides several insights for building feasible, modern spoken language identification systems. As a side-product of the experiments performed during this thesis, a free open source spoken language identification software library called "lidbox" was developed, allowing future experiments to begin where the experiments of this thesis end.

Keywords language identification, machine learning, deep neural networks, speech analysis

Tekijä
Matias Erik Lindgren

Työn nimi
Syväoppiminen puhutun kielen tunnistamisessa

Korkeakoulu Perustieteiden korkeakoulu

Maisteriohjelma Computer, Communication and Information Sciences

Pääaine Computer Science **Koodi** SCI3042

Valvoja Professori Mikko Kurimo

Ohjaaja FT Tommi Jauhiainen

Työn laji Diplomityö **Päiväys** 26.4.2020 **Sivuja** 10+91 **Kieli** englanti

Tiivistelmä

Tämä diplomityö keskittyy soveltamaan syviä neuroverkkomalleja luonnollisten kielien automaattiseen tunnistamiseen puheesta. Tämän työn ensisijainen tavoite on toteuttaa tarkka menetelmä multimediamateriaalien ositteluun niissä esiintyvien puhuttujen kielien perusteella.

Työssä tarkastellaan useampaa jo olemassaolevaa neuroverkkoihin perustuvaa lähestymistapaa, joista valitaan alijoukko tarkempaan tarkasteluun, kvantitatiivisten kokeiden suorittamiseksi. Valitut malliarkkitehtuurit koulutetaan käyttäen eri puhetietokantoja, sisältäen useampia eri kieliä. Kieliosittelun hienojakoisuus vaihtelee käytettyjen mallien mukaan, 0,2 sekunnista 10 sekuntiin, riippuen kuinka pitkän aikaikkunan perusteella malli pystyy tuottamaan kieliennusteen.

Diplomityön aikana suoritetut kokeet osoittavat, että sekvenssiautoenkoodaajalla ohjaamattomasti löydetty puheen diskreetti akustinen esitysmuoto ei ole riittävä kielen tunnistamista varten, verrattuna foneemitunnistimen tuottamaan, ohjatusti opetettuun foneemiesitysmuotoon. Tässä työssä havaittiin, että akustisfoneettiset kielentunnistusmallit saavuttavat korkeamman kielentunnistustarkkuuden kuin foneemiesitysmuotoa käyttävät kielentunnistusmallit, mikä eroaa monista kirjallisuudessa esitetyistä tuloksista. Diplomityön tutkimuksia on jatkettava, esimerkiksi litteroituja puhetietokantoja ja puheentunnistusmenetelmiä käyttäen, jotta pystyttäisiin selittämään miksi foneemimallin tuottamalla esitysmuodolla ei saatu parempia tuloksia kuin yksinkertaisemmalla, taajuusspektrin esitysmuodolla.

Tämän työn aikana puhutun kielen tunnistaminen osoittautui huomattavasti haastellisemmaksi kuin mitä työn alussa oli arvioitu, eikä työn aikana onnistuttu toteuttamaan tarpeeksi tarkkaa multimediamateriaalien kielienosittelumenetelmää. Tästä huolimatta, työssä esitetyt lähestymistavat tarjoavat toimivia käytännön menetelmiä puhutun kielen tunnistamiseen tarkoitettujen, modernien järjestelmien rakentamiseksi. Tämän diplomityön sivutuotteena syntyi myös puhutun kielen tunnistamiseen tarkoitettu avoimen lähdekoodin kirjasto nimeltä “lidbox”, jonka ansiosta tämän työn kvantitatiivisia kokeita voi jatkaa siitä, mihin ne tämän työn päätteeksi jäivät.

Avainsanat kielen tunnistaminen, koneoppiminen, syvät neuroverkot, puheanalyysi

Preface

I want to thank Professor Mikko Kurimo for offering me a position at the speech recognition research group and for providing valuable advice on speech recognition techniques. I also want to thank PhD Tommi Jauhiainen for all the helpful comments to several revisions of this thesis and for running the text language identification experiments. Special thanks also to D.Sc. Reima Karhila for providing me with scripts for applying his multilingual phoneme recognizer in my experiments.

This thesis is funded by the European Union's Horizon 2020 research and innovation programme under the MeMAD project (grant agreement No 780069). The experiments completed during this thesis were performed using computer resources within the Aalto University School of Science "Science-IT" project.

Espoo, 26th April 2020 Matias Erik Lindgren

Contents

List of abbreviations and symbols

Adam	Adaptive moment estimation
AP19-OLR	Asia-Pacific Oriental Language Recognition 2019 (dataset)
BGRU	Bidirectional GRU
BLSTM	Bidirectional LSTM
BNF	Bottleneck features
C_{avg}	Average detection cost
CNN	Convolutional neural network
CTC	Connectionist temporal classification
DCT	Discrete cosine transform
DNN	Deep neural network
EER	Equal error rate
EER_{avg}	Average EER
EM	Expectation maximization
$f^{\mathbf{W}}$	Discriminant function with hidden weights $\mathbf{W}$
$F_{1\text{-avg}}$	Average F1-score
FC	Fully connected (layer)
FNR	False negative rate
FPR	False positive rate
GMM	Gaussian mixture model
GRU	Gated recurrent unit
LDC	Linguistic Data Consortium
LM	Language model
LRE	Language recognition evaluation
LSTM	Long short-term memory
LVCSR	Large vocabulary continuous speech recognition
$\mathbb{L}$	Target set of natural languages
MFCC	Mel-frequency cepstral coefficients
MGB-3	Multi-Genre Broadcast news (dataset)
MLA	Multi-level (time) attention
MLE	Maximum likelihood estimate
NB-TLI	Naive Bayes text language identification
NIST	National Institute of Standards and Technology

NN	Neural network
OGI-11L	Oregon Graduate Institute 11-language (dataset)
OOS	Out-of-set (language)
PPRLM	Parallel PRLM
PR	Phoneme recognizer
PRLM	Phone recognition followed by language modeling
PTN	Phonetic temporal neural model
RNN	Recurrent neural network
ReLU	Rectified linear unit
SBS	Slavic Broadcast Speech (dataset)
SDC	Shifted-delta-cepstral (features)
SGD	Stochastic gradient descent
SLI	Spoken language identification
SNR	Signal-to-noise ratio
SS	SparseSpeech (autoencoder)
STFT	Short-time Fourier transform
TLI	Text language identification
VAD	Voice activity detection
$\mathbf{X}_{\text{dB}}$	Decibel-scale spectrum matrix
$\mathbf{X}_{\text{mag}}$	Magnitude spectrum matrix
$\mathbf{X}_{\text{Mel}}$	Log-scale Mel-spectrum matrix
$\mathbf{X}_{\text{MFCC}}$	Mel-frequency cepstral coefficients matrix
$\mathbf{X}_{\text{BNF}}$	BNF matrix
YTN	YouTube News (dataset)
YTN-Aalto2019	YTN dataset instance collected during this thesis

1. Introduction

This thesis studies different deep learning based approaches for detecting natural languages from speech data. Several state-of-the-art spoken language identification (SLI) models are discussed and compared. Efficient SLI systems are useful as pre-processing or filtering tools in several applications, such as rapid selection of correct single-language speech applications or quickly connecting human callers to correct operators in multilingual emergency services (Muthusamy et al. 1994). While humans are capable of recognizing familiar languages with high accuracy, the task of distinguishing between several unfamiliar languages can be surprisingly challenging. Human listening experiments have shown (Zhao et al. 2008) that when adults attempt to recognize an unfamiliar language, brain activity is higher in areas responsible for recognizing low-level acoustical cues such as different sound patterns or intonations. On the other hand, when humans are listening to familiar languages, brain activity moves towards areas responsible for recognizing language semantics such as grammar. Automating the process of correctly detecting languages from speech signals is the central topic in SLI and this thesis.

Researchers have been experimenting with several approaches to SLI at least since the early 1990s (Muthusamy and Cole 1992). Recently, a multilingual speech-to-text system for ten languages solved the SLI problem as a useful side-effect from its internal, multilingual representation of phoneme sequences (Watanabe et al. 2017). However, much like it does not make sense for humans to learn new languages only for the purpose of recognizing them, automatic language identification systems must balance the trade-off between speech representation complexity and language identification difficulty (Li et al. 2013). This thesis shows that SLI is a simple and intuitive problem to formulate, but surprisingly complex and challenging problem to solve.

1.1 Research questions

This thesis investigates several different, state-of-the-art SLI architectures by aiming to reproduce their original results. Specifically, we choose three SLI models and reproduce the results reported by the original authors on three SLI datasets. In addition, a grid search over five voice activity detection (VAD) configurations and two normalization techniques is performed to compare the effect of these hyper-parameters when attempting to reproduce the results. Furthermore, we use two additional datasets, five in total, and train all chosen SLI models on all datasets. These results are then compared to see whether some of these architectures work well on all datasets or if they work only on the reference dataset. We also investigate if it is possible to perform SLI on text tokens discovered in an unsupervised manner or whether a phoneme representation is required for accurate results.

1.2 Contributions of this thesis

Several state-of-the-art SLI models were implemented in TensorFlow (Abadi et al. 2015), trained on their reference datasets to reproduce their corresponding original results, and then trained and evaluated on all other datasets. The sequence-to-sequence autoencoder proposed by Milde and Biemann (2019) was trained on several datasets to produce an unsupervised acoustic unit representation of speech. The feasibility of using this unsupervised representation to replace supervised phoneme representations in SLI was investigated by applying a text language identification (TLI) model by Jauhiainen et al. (2019b) on the acoustic units. In order to manage a large amount of different experiments with different data pre-processing requirements, an open source SLI toolbox[1] was developed during this thesis. In addition, a YouTube News (YTN) dataset instance called YTN-Aalto2019[2], containing 1200 hours of broadcast news speech, was collected using the data collection program proposed by Bartz et al. (2017). Finally, a brief code review was performed on the open source WebRTC voice activity detection[3] (VAD) implementation to gain insight on the theoretical foundations of its VAD approach.

[1] https://github.com/matiaslindgren/lidbox (visited on 2020-04-16)

[2] http://urn.fi/urn:nbn:fi:lb-2020041701 (visited on 2020-04-23)

[3] https://webrtc.googlesource.com/src/+/7a709c0e85eb938a052b74fb39ebcaf5981f84be/common_audio/vad (visited on 2020-01-27).

1.3 Thesis structure

Chapter 2 defines spoken language identification (SLI) and provides a brief discussion of differences between languages and language variants. Chapter 3 formalizes SLI as a machine learning based classification problem and compares different speech feature representations that enable deep learning based SLI approaches. Chapter 4 introduces all speech datasets that are used in the experiments performed during this thesis as well as some other interesting datasets. Chapter 5 provides a literature study on existing SLI approaches, many of which have directly influenced the work done during this thesis. Some of these existing results are reproduced and serve as comparable reference points. We also discuss models that were not originally proposed for SLI, but are used in this thesis as building blocks to provide new SLI models. All models used for the experiments in this thesis are discussed in Chapter 6. Even though many of these models are also discussed in Chapter 5, a separate, more detailed, chapter was dedicated for all models used in the thesis experiments to distinguish them from existing work that was not reproduced during this thesis. The results from all completed experiments are discussed in Chapter 7 and suggestions for additional experiments or other future work is discussed in Chapter 8. Finally, we conclude the thesis in Chapter 9 by discussing how the experiment results answered our research questions presented in Section 1.1.

2. Definition of spoken language identification

Spoken language identification[1] (SLI) can be viewed as a multiclass classification problem of assigning natural language labels onto audio signals of arbitrary length, based on the unique acoustic structure of each language (Muthusamy and Cole 1992). In other words, SLI is about automating the process of detecting natural languages from speech. While this formulation intuitively encapsulates SLI, a more rigorous statistical framework is required before quantitative analysis may be performed. This topic is discussed later in Chapter 3. Instead, let us begin by discussing the terminology of what language is and is not, since these definitions are not entirely obvious.

2.1 Languages, dialects and language variants

It is estimated that between 4000 to 8000 different languages exists in the world (Schultz and Kirchhoff 2006, p. 7; Kamusella 2016). Even though only 5% of these languages are spoken by as much as 94% of the world's population, there are still hundreds of widely spoken, mutually unintelligible languages (Schultz and Kirchhoff 2006, p. 8). However, the distinction between languages, dialects, and other language variants is not clearly defined. Schultz and Kirchhoff (2006, pp. 5–7) used a definition where dialects are regional variants of a language that have undergone lexical and grammatical modifications, while accents are regional variants that differ only by pronunciation. In other words, all language variants, e.g. dialects or accents, of a single language are considered mutually intelligible, whereas different languages are not. However, they also suggest that sometimes the terminology of dialects and language varieties might be ambiguous. Indeed, it is not difficult to find several examples of such ambiguities. For example, while regional Arabic dialects share a common phonetic inventory and characters, the dialects are mutually unintelligible (Ali et al.

[1] Some authors use the term language *recognition* instead of language identification, but in this thesis these terms are considered interchangeable and "identification" is preferred.

2016; Shon et al. 2018). In contrast, Croatian and Serbian are considered to be two different South Slavic languages, but are linguistically so close that even native speakers might have difficulties distinguishing between these languages (Mateju et al. 2018). Similarly, Moldovan is considered a mutually intelligible dialect of Romanian, but due to political reasons, Moldovan was declared as an independent language (Schultz and Kirchhoff 2006, pp. 6–7). Ren et al. (2019) wrote that some regional Chinese dialects contain several cognate words or words with similar pronunciations, making it difficult even for humans to distinguish these dialects.

Sometimes the terminology of dialect or language might be misleading, especially when two language variants are linguistically different. For example, even though Brazilian and European Portuguese have significant differences, they are considered two varieties of the same language (Zampieri and Gebre 2012). In contrast to Ren et al. (2019), Schultz and Kirchhoff (2006, p. 6) suggested that several mutually unintelligible regional Chinese varieties exist, which would make them languages, rather than dialects.

To alleviate these ambiguities, one might hope to define a language similarity measure, which could be used to compare language variants in a quantitative manner. However, Ringbom (2007, pp. 5–13) argued that while attempts have been made to define objective, cross-linguistic similarity measures, a generally accepted method for objectively measuring language similarity does not exist. Furthermore, he suggested that our native languages might influence how we perceive the similarity of a language pair, making it difficult to define a similarity measure in the first place. Since measuring objective similarity of different languages is non-trivial, such information will not be utilized in this thesis. In other words, if two samples have been labeled differently, then any prior knowledge about language groups, dialects or language variants deducible from the language label will not be used to influence the language identification result.

Finally, it is worth noting the difficulty of constructing a universal definition for language, even though we all might have a familiar understanding of what language means to us. For example, Räsänen (2013, pp. 1–3) wrote that it is not understood how infants learn that languages consist of combining meaningless units, such as phones, to form meaningful words. He also noted that researchers have struggled for decades to find methods for describing the complex mapping from abstract, linguistic messages to physical speech signals.

This thesis will not focus on the qualitative analysis of language, but rather how to automate the discovery of a mapping from pre-labeled audio signal representations to natural languages. In order to accomplish this, we will now discuss how language differences appear in speech signals.

2.2 Cues for discriminating languages

In order to distinguish between different language pairs in speech, we must be aware of the various cues that imply language difference. Li et al. (2013) used a two-fold taxonomy for language cues: pre-lexical information and lexical semantic knowledge. They suggested that humans are able to accurately identify languages from speech that they understand, since in this case they can utilize lexical knowledge such as words and syntax to identify the language. More intuitively, if one understands what is said, then it must be obvious what language is spoken. In contrast, when we are listening to unfamiliar languages, we must rely on pre-lexical cues such as sequences of sound or prosody to distinguish between languages.

Human SLI Zhao et al. (2008) performed an experiment where 18 native Mandarin Chinese speakers were tasked to identify two familiar languages (Mandarin and English) and two unfamiliar languages (Japanese and Italian). When speech samples of these four languages were played to the participants, their brain activations were measured to detect which areas of the brain are active when attempting to recognize each language. The study shows how areas in the human brain responsible for processing lower-level acoustic cues such as prosody, become less active when higher-level information is available. In other words, when a participant was listening to speech in a familiar language, their brain favored lexical knowledge of the language over simpler pre-lexical cues. Conversely, when they were listening to an unfamiliar language, the brain areas responsible for recognizing pre-lexical information were more active. On a similar note, Ramus and Mehler (1999) discussed that bilingual children who are still learning their native languages are able to somehow distinguish between languages, before they have learned to speak them. This suggests that language identification is possible without lexical semantic knowledge.

Lexical and pre-lexical knowledge Lexical semantic knowledge may significantly improve the accuracy of neural network (NN) based SLI. For example, Watanabe et al. (2017) proposed a multilingual speech-to-text system, capable of recognizing speech in 10 different languages. When the system produces correct text output for some input speech, it simultaneously detects also the correct language by its internal, lexical representation. However, it is worth noting that the SLI problem is solved as a useful side-effect since the primary goal of the system is multilingual speech recognition, Li et al. (2013) argued that in general, it is not cost effective to solve the large vocabulary continuous speech recognition (LVCSR) problem if it is desirable to only solve the SLI problem. From a computational cost perspective, it is significantly cheaper to only rely on pre-lexical cues to perform SLI. More intuitively, it might not be feasible for a person to learn how to speak a large set of different languages if the objective is only to

accurately distinguish between those languages.

Pre-lexical cues might include sequences or patterns of phonemes, i.e. phonotactics, or simply the presence of a specific set of phonemes, i.e. acoustic-phonetics, which can both be used to narrow down the set of candidate languages (Li et al. 2013). There are countless examples of pre-lexical cues for SLI, and only a few are listed here to give an intuitive idea of why they might work in practice. For example, Hawaiian has a significantly smaller inventory of unique vowels and consonants compared to German, which has more complex patterns of permissible phoneme sequences (Schultz and Kirchhoff 2006, p. 14). Phoneme /x/ does not occur in French and Italian but does occur in Dutch and German (Schultz and Kirchhoff 2006, p. 234). In Japanese, the liquid consonant /r/ cannot appear after stop consonants, while this is common in English and French (Ramus and Mehler 1999). Similarly, consonant clusters /fl/, /pr/, and /str/ are common in English, but do not appear in Mandarin Chinese (Li et al. 2013). On the other hand, the cluster /sr/ does not occur in English, but is common in Tamil (Muthusamy et al. 1994). The German word “spiel” begins with a consonant cluster /sh p/, which can occur in English only at the boundaries of two separate words, such as in the compound word “flashpoint” (Zissman 1996).

It is important to note that such pre-lexical information is rarely engineered manually into a SLI model. Rather, these examples suggest it is feasible to assume SLI models may be able to learn the correlations between different languages and phoneme patterns in an unsupervised manner, assuming such correlations and patterns are learnable from the chosen speech representation. Indeed, explicit utilization of pre-lexical cues, such as phonotactics, has been studied extensively in the past and is discussed in Section 5.1. We also return to the topic of different speech representations in Section 3.2.

SLI model taxonomy Lets briefly discuss three common SLI approaches that attempt to distinguish languages in different ways. Li et al. (2013) distinguished between two major approaches to SLI: **phonotactic** and **acoustic-phonetic**. Phonotactic SLI models usually use a phoneme recognizer (PR) to produce a phoneme sequence from the input speech, followed by a language classifier which aims to then detect unique patterns and structures in the sequence that might uniquely identify one or more target languages. This approach is called phoneme recognition followed by language modeling (PRLM), which is usually applied using several PR models of different languages and several LM models, in which case it is called parallel PRLM (PPRLM) (Zissman 1996). While phonotactic models tend to be highly accurate at recognizing languages, their performance is usually constrained by the output quality of the PR model, which could perform poorly on noisy or mismatching data (Fernando et al. 2018).

Acoustic-phonetic SLI models, on the other hand, aim to distinguish

each target language in the acoustic domain, without an intermediate, explicit step that decodes the audio into a sequence of phonemes or similar discrete tokens. This approach relies on the assumption that each language has a unique acoustic structure that can be detected from an utterance representation (Muthusamy and Cole 1992). Discriminative acoustic-phonetic models are probably the easiest way of building a SLI model, since they only require labeled audio files for training. However, the amount of training data required might be impractically large before satisfactory recognition accuracy can be achieved (Lopez-Moreno et al. 2014). While phonotactic models usually provide good SLI performance, they are also computationally demanding and difficult to utilize in real-time applications (Torres-Carrasquillo et al. 2002; Tang et al. 2018b).

In addition to these two different approaches, we highlight a third approach which is based on **language embeddings**. One intuitive example of this is the embedding of individual languages as low-dimensional vectors into some vector space, where the angular proximity of two vectors imply similarity of the languages that these "language-vectors" encode (Gelly and Gauvain 2017). We return to the topic of different SLI approaches in Chapter 5.

2.3 Languages in speech and text

While this thesis focuses mainly on language identification from speech data, we also make some use of text language identification (TLI). TLI is the problem of detecting natural languages from text documents, with research dating back to at least the 1960s (Jauhiainen et al. 2019a). Although related, TLI should be considered a separate problem from SLI, since the goal of SLI is to predict languages from audio input, rather than text input. Some SLI models utilize textual information during the training phase, e.g. transcriptions of the audio input, in order to learn a refined representation of the speech input, usually making the SLI problem easier (Tang et al. 2018b; Ren et al. 2019). However, a distinguishing feature between SLI and TLI is that SLI models are assumed not to have access to any text data during inference and must predict the language class only from audio signals. While SLI and TLI might have the same target set of languages, in this thesis we make the assumption that the sets of input data for SLI and TLI are always disjoint during inference. Therefore, SLI and TLI should be considered two separate problems. However, this does not constrain the use of SLI only on speech data and TLI only on text data. In fact, it is possible to construct highly accurate SLI models by combining the use of speech and text representations (Watanabe et al. 2017).

3. Machine learning and speech

This chapter focuses on formalizing some of the intuitive definitions of SLI discussed in Chapter 2. In addition, we review some commonly used feature representations of speech signals and discuss how these representations are used in machine learning based speech analysis applications. Section 3.1 formalizes the notion of speech representation analysis, while Section 3.2 provides a more intuitive overview of how those representations are obtained in practice. In Section 3.3 we discuss some common evaluation metrics used for architecture-independent comparison of SLI models. Section 3.4 briefly discusses some numerical optimization methods used in machine learning and we conclude by discussing voice activity detection (VAD) in Section 3.5.

3.1 Machine learning formulation

Since this thesis focuses on solving SLI problems using a machine learning based approach, a statistical formulation of SLI must be established. We begin by outlining the sample space, discuss some simplifying assumptions, and then formulate the relationship between observations and language classes. This relationship is simplified and encapsulated into a statistical model which is computationally feasible for the machine learning applications discussed later in this thesis. The resulting formulation serves as a mathematical framework, independent of the tools applied in practice.

Much of the terminology used in this section is based on the work by Gauvain and Lee (1994, p. 291), Bishop (2006, Section 1.5), Li et al. (2013, pp. 1139–1141), and Pohjalainen (2014, Section 3.1).

Sample space Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T \times F}$ denote an i.i.d. random variable of an acoustic feature representation of a single speech signal, where the feature representation contains $T > 0$ time frames and $F > 0$ channels. Let $y \in \mathbb{L}$ denote a random variable of a natural language from a finite and non-empty target set $\mathbb{L} := \{L_1, L_2, \ldots, L_N\}$. For simplicity, it is assumed only a single language can be present in any time frame

$\mathbf{x}_t \in \mathbf{X}$. In addition, it is assumed every time frame of all acoustic feature representations always contain speech in some language in $\mathbb{L}$. However, it is unlikely that this assumption can be guaranteed in practice. Furthermore, it would be meaningless to assign natural language labels to non-speech time frames. In this thesis, it is assumed that such frames can be detected with probability 1 by applying VAD (discussed in Section 3.5). As a consequence, all non-speech frames can be removed and our initial assumption holds. On the other hand, one might argue that this simplifying assumption leads to a non-trivial amount of errors if $\mathbf{X}$ represents noisy or poorly recorded audio signals. While current state-of-the-art VAD models may reach very low misclassification rates (Lin et al. 2019; Vafeiadis et al. 2019), the assumption that these models will classify all frames correctly is too strict. Nevertheless, considering that SLI from noisy input was featured in a recent, state-of-the-art SLI challenge by Tang et al. (2019, task 2), the topic of noisy SLI will be considered out of scope for this thesis.

Statistical model With these definitions and simplifying assumptions, SLI can be formulated as a multiclass classification problem of assigning correct language labels y to every time frame $\mathbf{x}_t$ of an acoustic feature representation $\mathbf{X}$. One could encapsulate the binary relation between all possible acoustic feature representations $\mathbf{X}$ and all possible natural languages y within the joint probability distribution $p(\mathbf{X}, y)$. Then, discovering $p(\mathbf{X}, y)$ would yield a perfect solution to all possible SLI problems. However, Bishop (2006, pp. 38–44) suggested that inferring a true joint distribution from training data is in general a very difficult problem. As an alternative, if the objective is only to make classification decisions, he noted that it might be sufficient and computationally less demanding to find the generative relationship of how observation $\mathbf{X}$ generates y by modeling the class posterior distribution $p(y \mid \mathbf{X})$. While such generative models have many desirable properties, inferring non-trivial class posterior distributions requires statistical methods which are beyond the scope of this thesis. In order to simplify our problem formulation even further, we apply Bayes' formula on the class posterior distribution (Pohjalainen 2014, Eq. 3.2), resulting in

$$p(y \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid y)p(y)}{p(\mathbf{X})}.$$

Assuming $p(\mathbf{X})$ does not depend on y, and assuming uniform prior distributions for all target languages in $\mathbb{L}$, we can use the maximum likelihood criterion (Gauvain and Lee 1994, Eq. 2) to formulate the maximum likelihood estimate (MLE) for a target language $\hat{y} \in \mathbb{L}$ as

$$\hat{y} = \arg\max_{y} p(\mathbf{X} \mid y).$$

Finding MLE solutions is a classical problem with many existing solutions, such as the expectation maximization (EM) algorithm (Bishop 2006, pp. 450–455). Indeed, a common approach used in speech related machine learning applications is to model the class specific probability density function $p(\mathbf{X} \mid y)$ using a Gaussian mixture model (GMM) of several Gaussian distributions (Pohjalainen 2014, pp. 44–46; Yu and Deng 2015b). Then, the expected means of each Gaussian component of the GMM can be discovered using the EM algorithm, which yields a MLE solution.

Discriminant model While the GMM based approach has been a popular approach in SLI (Li et al. 2013), this thesis will instead focus on deep learning. Now, assume any discriminative SLI model can be represented by a non-linear discriminant function $f^{\mathbf{W}} : \mathbb{R}^{T \times F} \rightarrow \mathbb{R}^N$, such that the mapping is entirely dependent on a finite set of real-valued weights[1] $\mathbf{W}$. The output of $f^{\mathbf{W}}$ is a vector of language scores $\mathbf{y} = [y_1, y_2, \ldots, y_N]$, where N is the amount of languages in the target set $\mathbb{L}$. It is assumed that each score $y_l \in \mathbb{R}$ encodes the certainty that language $L_l \in \mathbb{L}$ is present in the utterance representation $\mathbf{X} \in \mathbb{R}^{T \times F}$, i.e. higher values imply presence while lower values imply absence. Note that these scores might or might not be probabilities since probabilities usually have no meaning when using a discriminant function (Bishop 2006, p. 43). Given an utterance $\mathbf{X}$, let $f^{\mathbf{W}}(\mathbf{X} \mid y_l)$ denote the predicted score for the presence of language L_l in $\mathbf{X}$. Now, we want to discover values for $\mathbf{W}$ such that the predicted language classes

$$\hat{y} = \arg\max_{L_l} f^{\mathbf{W}}(\mathbf{X} \mid y_l) \in \mathbb{L}$$

always correspond to the true language class of $\mathbf{X}$. In order to discover $f^{\mathbf{W}}$, we can search for values of $\mathbf{W}$ through minimizing the multiclass cross-entropy error (Bishop 2006, Eq. 4.108), defined as

$$E(\mathbf{W}) = -\sum_{m=1}^{M} \sum_{l=1}^{N} y_l \ln f^{\mathbf{W}}(\mathbf{X}_m | y_l), \tag{3.1}$$

where each observation $\mathbf{X}_m$ is from a dataset $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_M\} \in \mathbb{R}^{M \times T \times F}$. Note that the word "dataset" is used here synonymously with a sample of size M, drawn from the population of all possible speech signal representations $\mathbb{R}^{T \times F}$. In Equation 3.1, it is assumed that $\forall l : y_l \in [0, 1]$ and $\forall l, m : f^{\mathbf{W}}(\mathbf{X}_m | y_l) \in (0, 1]$, which implies $E(\mathbf{W}) \geq 0$. Since the error function $E(\mathbf{W})$ is differentiable, we can use an optimization method to discover the values for $\mathbf{W}$ that minimizes the error function. From a more general point of view, Goodfellow et al. (2016b) formulates this approach as empirical risk minimization. Here, the goal is to discover the empiri-

[1] Commonly also called hidden weights.

cal distribution $\hat{p}(\mathbf{X}, y)$ (e.g. $f^{\mathbf{W}}$), instead of the true joint distribution, by minimizing some risk function (e.g. Equation 3.1) using a finite training dataset. However, they point out that discovering $\hat{p}$ using this approach does not guarantee that $\hat{p}$ is an accurate generalization of the actual, joint distribution.

Even though Equation 3.1 can be used for measuring the performance of $f^{\mathbf{W}}$ during training, it is rarely used for evaluating SLI model performance on a test set. Some commonly used performance metrics that can be used with any SLI model, not only deep learning based models, are discussed in Section 3.3.

Deep learning model Since this thesis focuses on deep learning based SLI models, we should briefly mention the concept of layers and how a deep learning model can be represented using the discriminant model framework. Let $f^{\mathbf{W}} : \mathbb{R}^{T \times F} \to \mathbb{R}^N$ denote some deep learning based model and assume it is constructed from L intermediate, or hidden, layers defined as

$$\begin{aligned} f_1^{\mathbf{W}_1} &: \mathbb{R}^{T \times F} \to A_1, \\ f_2^{\mathbf{W}_2} &: A_1 \to A_2, \\ f_3^{\mathbf{W}_3} &: A_2 \to A_3, \\ &\vdots \\ f_L^{\mathbf{W}_L} &: A_{L-1} \to \mathbb{R}^N. \end{aligned}$$

Each layer i has an independent set of hidden weights $\mathbf{W}_i$, an input space A_{i-1} and an output space A_i. In this thesis all input and output spaces are assumed to be finite-dimensional, real vector spaces. This particular example is a feed-forward NN, or DNN (Goodfellow et al. 2016a), although many different architectures are also used. We return to the topic of different deep learning based SLI models in Chapter 6.

Deep learning and the amount of data One crucial assumption for applying deep learning to SLI is that it is assumed each SLI problem can be solved with a discriminative model, without explicitly modeling any probability distribution. It is known that feed-forward NNs are highly capable of approximating continuous functions to arbitrary accuracy, as long as the networks have a sufficient amount of hidden units (Bishop 2006, p. 230; Yu and Deng 2015a, Section 4.2). However, it should be noted that our assumption is viable only if a sufficient amount of training data exists, since the performance of deep learning models is highly dependent on the amount of training data, regardless of model complexity or the novelty of its implementation (Goodfellow et al. 2016c, pp. 421–422). Furthermore, the exact threshold for "sufficient" amount of data is difficult to define in an objective manner, although it is well known that increasing the amount

of data improves SLI performance (Lozano-Diez et al. 2015; Snyder et al. 2018a; Shon et al. 2018).

Sometimes, it is possible to create new data samples by augmenting the existing dataset. Ko et al. (2015) experimented with different speech augmentation techniques for DNNs and evaluated the models on 5 different LVCSR tasks. They found out that simple resampling through modifying the speed of the speech signal to 90% and 110% of the original speed was sufficient to yield a noticeable improvement, even compared to the more complex augmentation techniques. Similarly, Shon et al. (2018) noticed that SLI performance is significantly increased after performing data augmentation by speed and volume modifications. Ma et al. (2019) compared several time scale modifications ratios and noticed that the largest SLI performance improvements can be achieved by modifying the speed of the signal to 80% and 120% of the original speed.

3.2 Speech representations

Speech feature extraction is a vast, active research topic and cannot be sufficiently summarized without greatly exceeding the scope of this thesis. Furthermore, choosing the best, discrete representation for speech is non-trivial and usually application specific. This section will provide only a brief overview of the most common speech features, mostly focusing on features used in the experiments discussed in Chapter 7.

Overview Audio signals are usually encoded as a waveform, a single-dimensional time-domain representation where each sample is a single real number. However, the waveform is rarely used directly in speech analysis systems (Yu and Deng 2015c, Section 3.6). Instead, the signal is converted into a two-dimensional, frequency spectrum by applying the short-time Fourier transform (STFT) (Pohjalainen 2014, Chap. 2). This spectral representation is usually transformed with a frequency-domain warping operation that emphasizes the frequencies important for human hearing. Yu and Deng (2015c, Section 3.6) argued that a spectral representation, containing both spatial (frequency) and temporal dimensions, enables more accurate analysis of correlations and variability that occur within one of the dimensions but not both. Sometimes, the discrete cosine transformation (DCT) is applied on the spectrum to create a cepstral representation, although this seems to be less common in contemporary SLI systems. This is discussed in more detail at the end of this section.

Feature extraction procedure We will now briefly walk through how to create a two-dimensional representation $\mathbf{X} \in \mathbb{R}^{T \times F}$ from a single-dimensional audio signal $\mathrm{s} \in \mathbb{R}^{S}$, encoded as a waveform, containing S samples. Before we begin, it should be noted that per-sample mean-normalization is

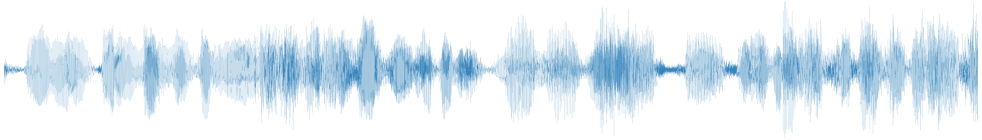
commonly applied on each $\mathbf{X}$ after feature extraction (Yu and Deng 2015a, pp. 65–67). In this thesis, both mean-normalization as well as standardization was experimented with, and it was discovered that normalizing the variance to 1 is in fact not beneficial. These results are discussed in more detail in Chapter 7, but for now, it is assumed all $\mathbf{X}$ have been mean-normalized such that each of its frequency channels have zero mean.

Consider three audio signals chosen uniformly at random from the YouTube News (YTN) dataset (discussed later in Section 4.4), each containing speech in Spanish, Mandarin Chinese, and German. Signals are shown in Figures 3.1a, 3.2a, and 3.3a. Each signal is a waveform representation of an approximately 5 second long signal, recorded at a sample rate of 16 kHz, which implies $S = 5\,\text{s} \cdot 16\,\text{kHz} \approx 80\,000$. The first spectral representation we will compute is the magnitude spectrum. First, a 512-point STFT with window length 25 ms and offset 10 ms is applied on each s. Then, we take the absolute value of each window to get a real-valued representation $\mathbf{X}_{\text{mag}} \in \mathbb{R}^{498\times 257}$, which is shown in Figures 3.1b, 3.2b, and 3.3b. It is evident that this representation is too sparse, due to the larger values of $\mathbf{X}_{\text{mag}}$ being disproportionately large compared to the smaller values. This makes it difficult to compare the energies in different frequency bands. In addition, auditory perception is more sensitive to changes on a logarithmic, rather than a linear scale (Pohjalainen 2014, p. 25). By mapping the squared values of $\mathbf{X}_{\text{mag}}$ onto the decibel-scale, we get a decibel-scale spectrum $\mathbf{X}_{\text{dB}} \in \mathbb{R}^{498\times 257}$, shown as spectrograms in Figures 3.1c, 3.2c, and 3.3c. While a log-scale representation allows for easier comparison of energy levels in different frequency-bands, we can see that most of the information of a speech signal is contained within the lower frequencies, leaving some sparsity in the higher frequency-bands. Furthermore, it is known that human hearing is most sensitive in the lower frequencies, between 1 kHz and 5 kHz (Pohjalainen 2014, p. 24). One solution is to warp the frequency-dimension to give more emphasis on lower frequencies. There are several well-known psychoacoustic scales that are modeled after human hearing, such as the Bark, ERB, and Mel scales (Pohjalainen 2014, p. 25). The Mel scale is popular choice, and will be used in this example. Taking the natural logarithm of the squared values of $\mathbf{X}_{\text{mag}}$ and warping the frequency dimension by multiplying it with the 64 Mel scale filter banks[1], we get the log-scale Mel spectrum $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{498\times 64}$, seen as spectrograms in Figures 3.1d, 3.2d, and 3.3d. Note the significant reduction in sparsity compared to the decibel-scale representation.

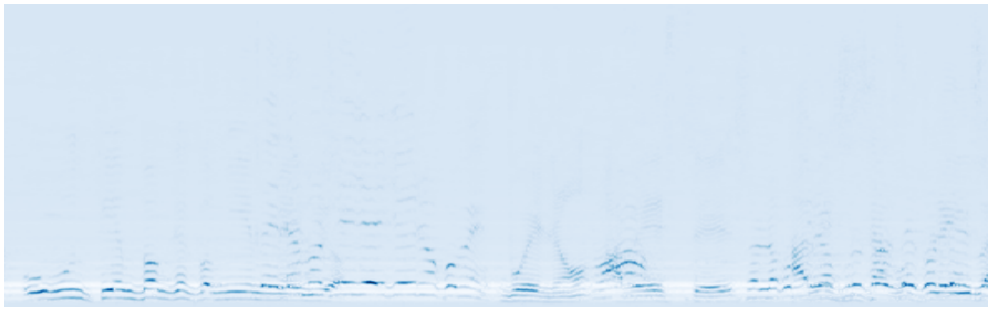
Another commonly used representation are the Mel-frequency cepstral coefficients (MFCC) $\mathbf{X}_{\text{MFCC}} \in \mathbb{R}^{498\times m}$, which are computed by applying

[1] It might be good to point out that the term “filter banks” is commonly misused in SLI literature. In the case of the Mel scale, $\mathbf{X}_{\text{Mel}}$ is obtained by multiplying the power spectrum with Mel filter banks, a constant matrix that encodes the frequency channel mapping from a linear scale to the Mel scale. However, $\mathbf{X}_{\text{Mel}}$ is not equal the Mel filter banks, even though this incorrect use of terminology is common.

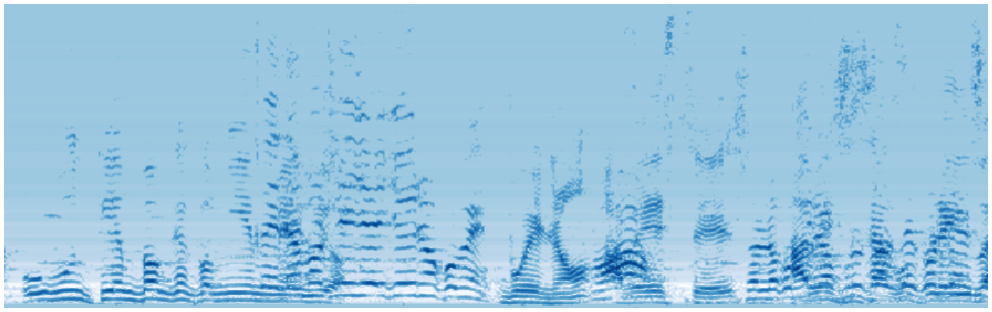
the DCT on $\mathbf{X}_{\text{Mel}}$. In this example, we select the $m = 20$ first coefficients, although smaller values such as $m = 13$ are also common in literature. This cepstral representation can be seen in Figures 3.1e, 3.2e, and 3.3e. Note how the zeroth coefficient (energy) at the bottom overshadows the distribution of values in coefficients c_1 to c_{19}. Some authors prefer to simply drop the c_0 coefficient for this reason. While the MFCC representation is very compact and a popular choice in speech related analysis, applying the DCT is a lossy operation and might sometimes lead to slightly worse performance compared to using a spectral representation, which we will discuss next.



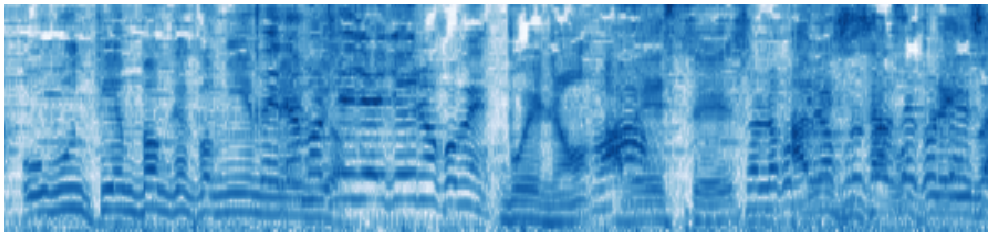
(a) 5 second, 16 kHz audio signal $\mathbf{s} \in \mathbb{R}^{80042}$. Starting point for the segment in the original source audio: `https://youtu.be/wtBZRZ0Bfu8?t=650` (visited on 2020-02-14).



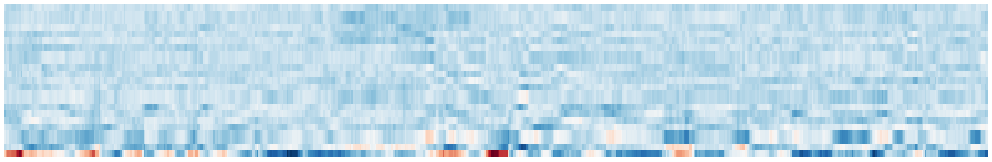
(b) Magnitude spectrogram $\mathbf{X}_{\text{mag}} \in \mathbb{R}^{498\times257}$ of the signal in (a), extracted with a 512-point STFT from windows of length 25 ms and offset of 10 ms.



(c) Decibel-scale spectrogram $\mathbf{X}_{\text{dB}} \in \mathbb{R}^{498\times257}$ with values ranging from -120 to 0 dB, with the upper reference point being the square of the maximum value of the magnitude spectrogram in (b).

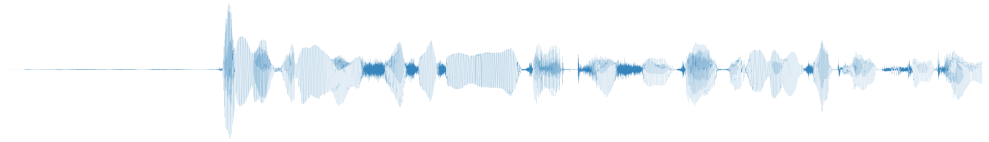


(d) Log-scale Mel-spectrogram $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{498\times64}$ extracted from the squared values of (b) by warping the frequency dimension into 64 Mel-frequency bins in the Mel-band $[20, 8000]$ Hz and applying the natural logarithm on the values.



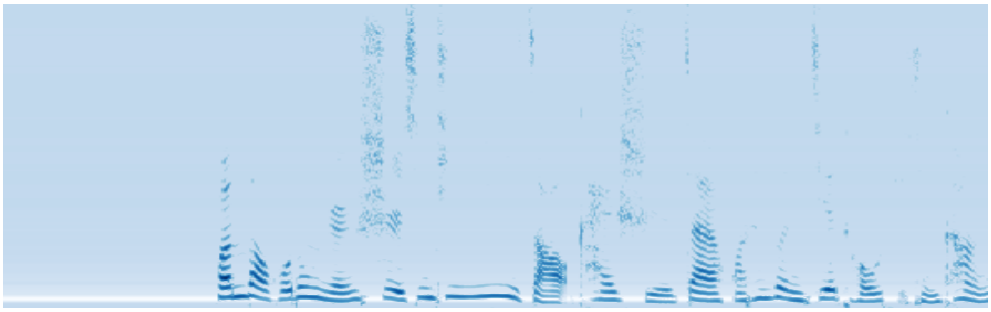
(e) 20 Mel-frequency cepstral coefficients $\mathbf{X}_{\text{MFCC}} \in \mathbb{R}^{498\times20}$ extracted from the log-scale Mel-spectrogram in (d).

Figure 3.1. Features extracted from a randomly chosen utterance from the YTN dataset, containing speech in Spanish. The utterance is shown as a waveform representation, 3 different spectral representations, and one cepstral representation (red < 0 and blue > 0). All rows of each two-dimensional representation have been mean-centered.

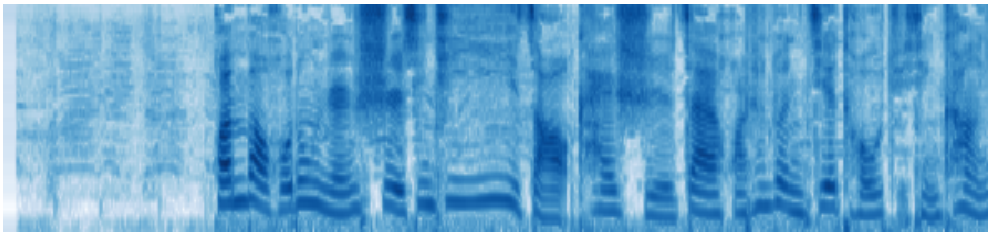


(a) 5 second, 16 kHz audio signal $\mathbf{s} \in \mathbb{R}^{80054}$. The source audio is the first 5 seconds of `https://www.youtube.com/watch?v=mmx29HS2Ejc` (visited on 2020-02-14).

(b) Magnitude spectrogram $\mathbf{X}_{\text{mag}} \in \mathbb{R}^{498 \times 257}$.



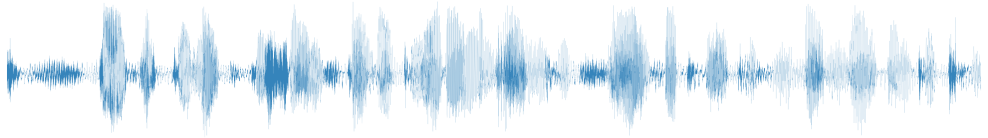
(c) Decibel-scale spectrogram $\mathbf{X}_{\text{dB}} \in \mathbb{R}^{498 \times 257}$.



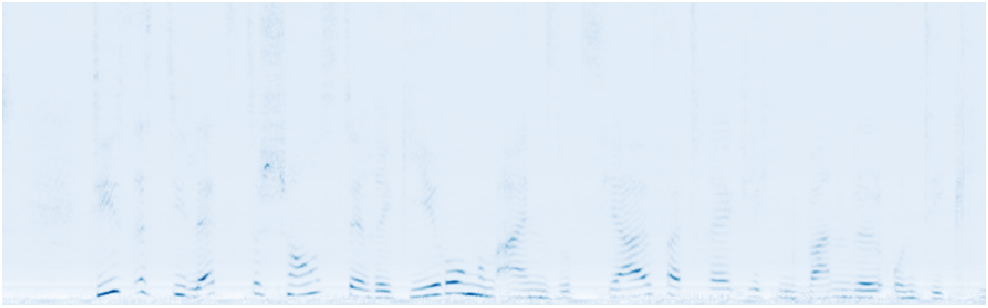
(d) Log-scale Mel-spectrogram $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{498 \times 64}$.

(e) 20 Mel-frequency cepstral coefficients $\mathbf{X}_{\text{MFCC}} \in \mathbb{R}^{498 \times 20}$.

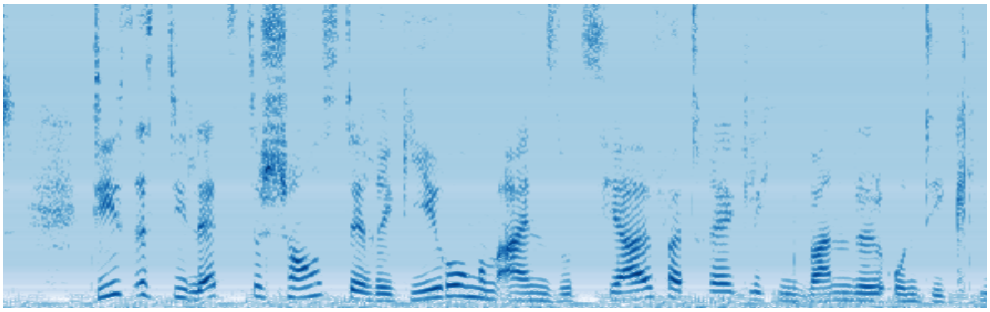
Figure 3.2. Features extracted from a randomly chosen utterance from the YTN dataset, containing speech in Mandarin Chinese. All feature extraction configurations are exactly as described in Figure 3.1



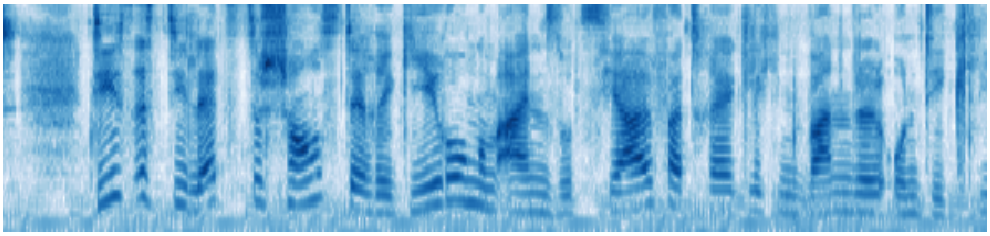
(a) 5 second, 16 kHz audio signal $\mathbf{s} \in \mathbb{R}^{80042}$. Starting point for the segment in the original source audio: `https://youtu.be/3uv9vIB4MRk?t=50` (visited on 2020-02-14).



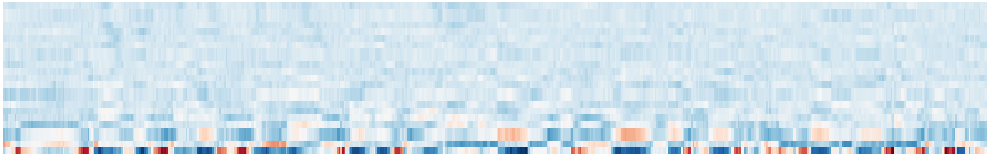
(b) Magnitude spectrogram $\mathbf{X}_{\text{mag}} \in \mathbb{R}^{498 \times 257}$.



(c) Decibel-scale spectrogram $\mathbf{X}_{\text{dB}} \in \mathbb{R}^{498 \times 257}$.



(d) Log-scale Mel-spectrogram $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{498 \times 64}$.



(e) 20 Mel-frequency cepstral coefficients $\mathbf{X}_{\text{MFCC}} \in \mathbb{R}^{498 \times 20}$.

Figure 3.3. Features extracted from a randomly chosen utterance from the YTN dataset, containing speech in German. All feature extraction configurations are exactly as described in Figure 3.1

Arguments for and against MFCCs Fayek (2016) argued that while MFCCs have been popular speech features in GMM based systems, decorrelating the filter bank coefficients by applying a DCT might be unnecessary, since DNN models are usually robust against correlated input. Furthermore, he suggested that applying any linear transformations such as DCT or

even the Fourier transform might discard some valuable, non-linear information from the audio signal. Nevertheless, he noted that the Fourier transform might be too difficult for NNs to learn, making training on waveforms infeasible.

Mohamed (2014, Chap. 4) argued that there is no a priori reason to assume MFCCs are a good, general feature type for all NN based architectures. He wrote that MFCCs have mainly evolved alongside GMM based models, which benefit from non-correlated features for several reasons, such as performance and easier inference, whereas using MFCCs in NN based approaches might even reduce performance.

Shon et al. (2018) noticed that MFCC features slightly outperform log-scale Mel-spectra on an unaugmented dataset, but after augmenting the training set by speed and volume modifications, the log-scale Mel-spectra slightly outperformed MFCCs. They argued that this implies it might be possible to perform SLI on raw waveforms if there is enough data, although it is unknown in general how much data this would require.

Xu (2018, p. 35) suggested that while MFCCs have shown to work well for phoneme recognition, the inherently lossy spectral representation of MFCCs might be detrimental for general audio classification, beyond the domain of speech.

Signal variability Since the main goal of SLI is to distinguish between languages, it is highly desirable to ensure a NN model is actually classifying languages and not some unwanted correlations caused by background noise or speaker variability. One notable, although not only, reason for speaker variability is due to physiological differences in our vocal tracts. This can be alleviated by applying a vocal tract length normalization (VTLN) transformation on the speech representation (Wakita 1977; Zhan and Waibel 1997), although the benefit of applying VTLN seems to diminish when NN models are used (Seide et al. 2011; Mohamed 2014, p. 105). Unwanted correlations that distort the language-separating cues are not limited to differences between speakers. Some other sources of unwanted variability might be caused by additive noise, channel variability, and source variability (Pohjalainen 2014, Chap. 4). In general, ensuring that speech analysis models learn only from the most salient features of a speech representation falls into the field of robust speech recognition, which is a non-trivial problem and an active research topic. While it might be possible to significantly improve the performance of a SLI model by careful feature engineering for increased robustness, this approach is considered out of scope for this thesis.

3.3 Evaluation metrics

Two commonly used metrics for evaluating SLI model performance within the SLI community are the average equal error rate (EER_{avg}) and average detection cost (C_{avg}), which are defined in this section. In addition, we discuss the average weighted F_1 score ($F_{1\text{-avg}}$) for multiclass classification, since it is used in Chapter 7 as an accuracy measure for reporting experiment results. The F_1 score is a commonly used metric also in TLI (Jauhiainen et al. 2019a, p. 722).

The intuition behind both EER_{avg} and C_{avg} is to find a threshold value for predicted language scores, such that the binary decisions made separately for each class minimize both the false positive rate[1] (FPR) and the false negative rate[2] (FNR). Furthermore, C_{avg} uses separate FNR values for all target-nontarget *pairs* to minimize skew from class imbalances, instead of using one FNR for each target language. We begin with EER_{avg} since it is the simpler one of these two metrics.

Equal error rate Pohjalainen (2014, p. 46) defines equal error rate (EER) as the point on a detection-error-tradeoff (DET) curve, parametrized by a decision threshold, where FPR is equal to FNR. Similarly, Cheng and Wang (2004) defines EER as the point where FPR equals FNR. On the other hand, Alphonsa et al. (2017, Eq. 5) defines EER as the arithmetic mean of FPR and FNR, which implies an assumption that the optimal decision threshold has already been found and the FPR and FNR values have been computed using this threshold. We'll discuss the issue of choosing the decision threshold later in this section.

Some authors omit the EER formula completely. For example, Gonzalez-Dominguez et al. (2014) describes EER as "the well-known metric", without providing a definition. Furthermore, since EER is a metric for binary classification, it is usually implied that EER values reported for multiclass classification of several language classes are averages of EER values computed separately for each class. Some authors use the notation EER_{avg} for average EER and this convention is followed also in this thesis.

Average detection cost One obvious problem with EER_{avg} arises if one uses a dataset containing significant class imbalances. In this case, placing equal weight for each EER value regardless of the amount samples per class will make EER_{avg} sensitive for small changes in the EER values for classes than contain only a few samples. One solution to this problem is to compute a separate FPR for each target class, and a separate FNR for each pair of target and nontarget classes. Then, each FPR is normalized by the total amount of target classes and FNR is normalized by the amount of target-nontarget pairs. One such metric is the C_{avg}, which has been

[1] Also known as false acceptance rate (FAR), or type I error.
[2] Also known as false rejection rate (FRR), or type II error.

extensively used in the SLI community. Li et al. (2013, Eq. 32) defines the average detection cost as

$$\begin{aligned} \mathrm{C}_{\text{avg}} = \mathrm{C}_{\text{miss}}\mathrm{P}_{\text{tar}}\frac{1}{N}\sum_{l=1}^{N}\mathrm{P}_{\text{miss}}(L_l) \\ + \mathrm{C}_{\text{fa}}(1-\mathrm{P}_{\text{tar}})\frac{1}{N}\sum_{l=1}^{N}\Big(\frac{1}{1-N}\sum_{m\neq l}\mathrm{P}_{\text{fa}}(L_l, L_m)\Big) \end{aligned} \tag{3.2}$$

where N is the amount of languages in the target set, $\mathrm{P}_{\text{miss}}(L_l)$ is FNR for classifying a target language L_l as some nontarget language, $\mathrm{P}_{\text{fa}}(L_l, L_m)$ is FPR for classifying some nontarget language L_m as the target language L_l, and $\{\mathrm{C}_{\text{miss}}, \mathrm{C}_{\text{fa}}, \mathrm{P}_{\text{tar}}\}$ are application specific parameters that have been set to $\{1, 1, 0.5\}$ during past NIST LRE[1] events. Setting $\mathrm{C}_{\text{miss}} = 1,\ \mathrm{C}_{\text{fa}} = 1,\ \mathrm{P}_{\text{tar}} = 0.5$ slightly simplifies Equation 3.2 to

$$\mathrm{C}_{\text{avg}} = \frac{1}{2N}\sum_{l=1}^{N}\mathrm{P}_{\text{miss}}(L_l) + \frac{1}{2N}\sum_{l=1}^{N}\frac{1}{1-N}\sum_{\substack{m=1\\m\neq l}}^{N}\mathrm{P}_{\text{fa}}(L_l, L_m). \tag{3.3}$$

For clarity, it is good to note that FPR and FNR are binary metrics, computed by comparing a given score to some fixed decision threshold. If the score is less (greater) than the threshold, we record a negative (positive). C_{avg} requires N different FNR metrics $\mathrm{P}_{\text{miss}}(L_l)$, one for each language l, and $N \cdot (N-1)$ different FPR metrics $\mathrm{P}_{\text{fa}}(L_l, L_m)$, one for each language pair $l \neq m$. The final result is then computed by averaging over all these metrics, multiplied by the application specific parameters. However, defining this fixed decision threshold is not trivial and depends on the application. This will be discussed next.

Choosing a score threshold for decisions Assume we have a SLI model which produces language scores $\mathbf{y} = [y_1, y_2, \ldots, y_N] \in \mathbb{R}^N$ for N different languages. Let θ_{min} denote the smallest possible language score, such that $\forall l : y_l \geq \theta_{\text{min}}$. Similarly, let θ_{max} denote the largest possible language score, such that $\forall l : y_l \leq \theta_{\text{max}}$. Let $\theta \in \mathbb{R}$ denote a language-independent, global score threshold used for deciding whether a given score indicates a positive or negative result. Then, setting $\theta < \theta_{\text{min}}$ would yield only positive results, minimizing all FNR values but maximizing all FPR values because $\forall l : y_l > \theta$. Conversely, setting $\theta > \theta_{\text{max}}$ would maximize all FNR values but minimize all FPR values because $\forall l : y_l < \theta$. Now, the challenge is to somehow choose the best value for $\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]$ such that we minimize FNR and FPR at the same time. One commonly used approach described by Li et al. (2013) is to select a fixed set of threshold values between θ_{min} and θ_{max}, then perform a grid search over all values in the set, computing

[1] National Institute of Standards and Technology (NIST) language recognition evaluation

the C_{avg} for each θ. Then, the smallest C_{avg} value is chosen as the final value. The same approach can also be applied to finding EER_{avg}, since it is also defined by FPR and FNR metrics.

Although one might achieve reasonable results for a specific SLI task by using this heuristic of probing the evaluation set using different values of θ until C_{avg} is minimized, more systematic approaches are available. Li et al. (2013) suggested that application-independent calibration might provide a better way to minimize C_{avg} indirectly, making fusion of language score predictions across different SLI models easier. Brümmer and Preez (2006) provided a thorough analysis on speech-related classification decision theory, although this analysis is out of scope for this thesis and will not be discussed here. Instead, this thesis uses the evaluation set probing heuristic (Li et al. 2013). First, we choose a finite set of candidate thresholds $\{\theta : \theta \in [\theta_{\min}, \theta_{\max}]\}$, evenly spaced between the smallest ($\theta_{\min}$) and largest ($\theta_{\max}$) language scores produced by each SLI model. Then, C_{avg} is computed from the predicted language scores for each θ, after which the smallest C_{avg} value is chosen as the final, reported C_{avg} value.

Average weighted F_1 score Even though most results in SLI literature are reported using error metrics such as EER_{avg} and C_{avg}, a common accuracy score used in other domains is the F_1 score. For completeness, this score will be used in thesis in addition to the error metrics. In order to allow the usage of a single metric for an arbitrary amount of languages, we will use the average weighted F_1 score ($F_{1\text{-avg}}$), which is the average over several F_1 scores, one for each language, weighted by the amount of samples for each language. It should be noted that the weighted F_1 score is a commonly used metric in many domains and is not in any way unique to this thesis.

The binary F_1 score for a class L_l is defined as the harmonic mean of precision $P(L_l)$ and recall $R(L_l)$ (Jauhiainen et al. 2019a), i.e.

$$\mathbf{F}_1(L_l) = 2\frac{P(L_l)R(L_l)}{P(L_l) + R(L_l)}.$$

In order to apply this binary metric on predictions with multiple classes, $\mathbf{F}_1(L_l)$ is computed for each class L_l and then averaged over all scores, with each score weighted by the support $S(L_l)$ of the corresponding class, i.e. the total number of true samples for that class in the evaluation set. Then, we define the average weighted F_1 score for N classes as

$$\mathbf{F}_{1\text{-avg}} = \frac{1}{N}\sum_{l=1}^{N} S(L_l)\mathbf{F}_1(L_l). \tag{3.4}$$

(LRE), which will be discussed in more detail in Section 4.6.

3.4 Optimization methods

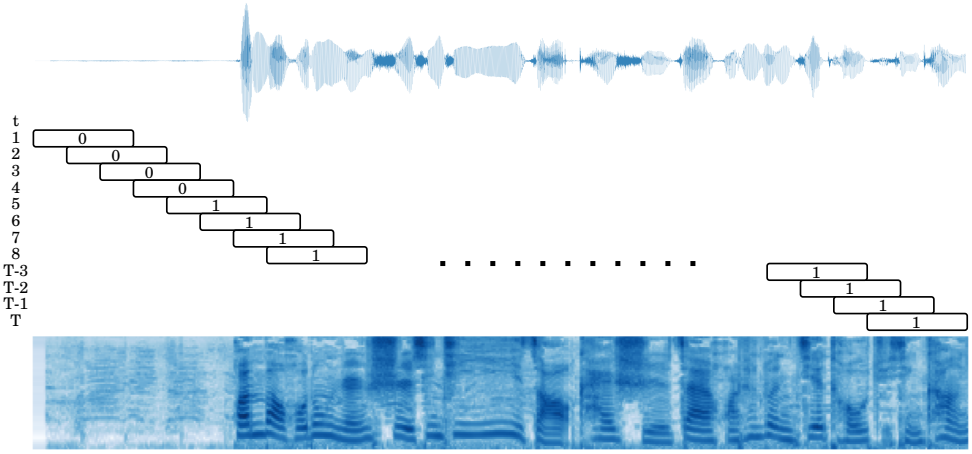
The driving force at the heart of most deep learning architectures arguably is the numerical optimization algorithm that updates the hidden weights of a model to minimize the empirical risk, in hope that this will also improve the prediction strength of the model (Goodfellow et al. 2016b). This section provides a brief overview of the optimization methods that are relevant to the experiments performed during this thesis.

Unless stated otherwise, the optimization algorithm used in all experiments in this thesis is stochastic gradient descent (SGD) with Nesterov accelerated gradients (Sutskever et al. 2013). Research into new optimization methods is a very active field, with increasingly complex optimization approaches emerging each year. For example, Kingma and Ba (2015) proposed the adaptive moment estimation (Adam), along with a proof of convergence and empirical evidence of superior performance over SGD. However, Reddi et al. (2018) later showed a simple counter-example to the initial proof of convergence, in which Adam fails to find the global optimum for a single-dimensional, convex problem when hyper-parameters are chosen in a specific way. They argue that while this particular instance of non-convergence can be solved by updating the hyper-parameters correctly, such manual tuning defeats the purpose of an *adaptive* optimization algorithm. As a solution, they propose an update to the Adam optimization algorithm, named AMSGrad. Later, Zou et al. (2019) showed how the performance of Adam and AMSGrad can be significantly improved using a "generic Adam" algorithm, which is a generalization of weighted AdaGrad with exponential moving average momentum. They also provide a comprehensive overview of known, common conditions that must be fulfilled for non-convex optimization problems to converge with the Adam algorithm.

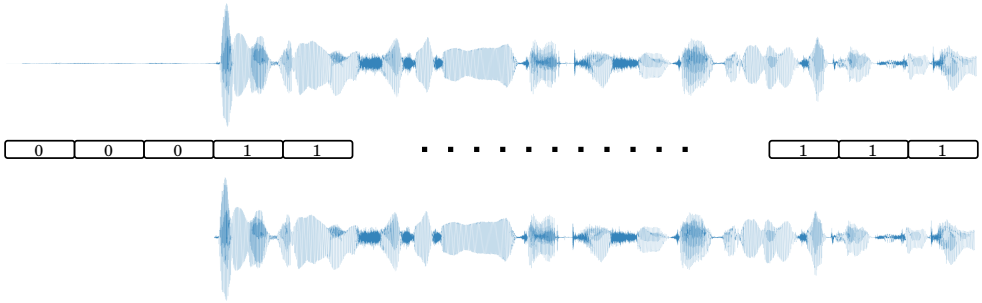
Choosing the best optimization method is a non-trivial task and like choosing the best speech feature representation, also depends on the application.

3.5 Voice activity detection

The last topic we discuss in this chapter is voice activity detection (VAD), which is a common pre-processing technique in speech analysis. The central part of VAD is deciding whether a given segment in an audio signal contains speech or not. VAD is a non-trivial task, with solutions ranging from simple energy based approaches to complex deep learning classifiers. Some authors also use the term speech activity detection (SAD) instead of VAD, but in this thesis it is assumed to be equivalent to VAD and will not be used.



(a) VAD on overlapping windows, matching the size of STFT windows. In this case, feature extraction can be performed on the full, unmodified signal and silence frames dropped after the features have been extracted. Unfiltered log-scale Mel-spectrogram is shown for comparison.



(b) VAD on non-overlapping windows and filtered signal.

Figure 3.4. Two different approaches for performing VAD on a 5 second utterance, with 0 denoting not speech and 1 denoting speech. The size of each window is approximately 20 times larger compared to the actual size, in order to make them visible in the figures.

Consider the audio signal shown in Figure 3.2a, and the log-scale Mel-spectrum $\mathbf{X}_{\text{Mel}}$ extracted from this signal, shown as a spectrogram in Figure 3.2d. Each feature frame of $\mathbf{X}_{\text{Mel}}$ will share the language label corresponding to the language that is spoken in the audio signal. However, at the beginning of the signal, there is approximately one second of silence. As discussed in Section 3.1, it is meaningless to add language labels on frames without speech, and we would prefer to remove such frames. One approach is to divide the audio signal into windows, such that their length and offset equal the STFT windows. This results in T overlapping windows, which map one-to-one to the feature frames of $\mathbf{X}_{\text{Mel}}$. By computing a binary VAD decision $d_t \in \{0, 1\}$ for each window t (see Figure 3.4a), we get T decisions $\mathbf{d} \in \{0, 1\}^T$. Using these decisions, we can choose to either drop or keep each frame of $\mathbf{X}_{\text{Mel}}$, without modifying the original signal.

Alternatively, an audio signal could be filtered directly to remove non-speech frames. An example is shown in Figure 3.4b.

WebRTC VAD The *WebRTC* (2020) project implements a GMM-based VAD algorithm, built on the assumption that the frequency domain of each input signal can be divided into bands such that the decibel-scale energy of each frequency band is normally distributed with different mean and variance depending on whether the source signal contains speech or noise. The open source code[1] of WebRTC VAD was analyzed during this thesis and a simplified overview of its theoretical background will now be discussed. Note that several details about numerical approximations and smoothing methods have been omitted in this analysis.

The system uses a pre-trained GMM to produce binary VAD decisions on short segments of the original signal, requiring all input segments to have a duration of 10, 20, or 30 ms. It is up to the user to decide how these decisions are combined and applied to the full signal. The system uses 6 weighted frequency bands ranging from 80 Hz to 4000 Hz, with the spectral weights and band boundaries listed in Table 3.1.

i	α_i	f_{low}	f_{high}
1	6	80	250
2	8	250	500
3	10	500	1000
4	12	1000	2000
5	14	2000	3000
6	16	3000	4000

Table 3.1. 6 frequency bands (Hz) and spectral weights α_i for each band, used by WebRTC VAD to output VAD decisions.

Let $x_i \in \mathbb{R}$ denote the decibel-scale energy of the ith frequency band for some input segment $\mathbf{x} \in \mathbb{R}^6$ of duration 10, 20, or 30 ms. For each band, the following speech and non-speech hypotheses are made:

$$\begin{aligned} H_1 &: x_i \sim \mathcal{N}(\mu_i^{\text{speech}}, (\sigma_i^{\text{speech}})^2), \quad \text{i.e. } x_i \text{ is speech}, \\ H_0 &: x_i \sim \mathcal{N}(\mu_i^{\text{noise}}, (\sigma_i^{\text{noise}})^2), \quad \text{i.e. } x_i \text{ is noise}. \end{aligned}$$

Now, assume we are given an input segment $\mathbf{x} = [x_1, x_2, \ldots, x_6]$ containing 6 frequency bands. The system performs a likelihood ratio test by computing, for all frequency bands, a numerical approximation of the log-likelihoods whether hypothesis H_1 holds or should be rejected, resulting in the log-likelihood ratio

$$\lambda = \sum_{x_i \in \mathbf{x}} \alpha_i \frac{\log \Pr(x_i \mid H_1)}{\log \Pr(x_i \mid H_0)}.$$

[1] `https://webrtc.googlesource.com/src/+/7a709c0e85eb938a052b74fb39ebcaf5981f84be/common_audio/vad` (visited on 2020-01-27).

The log-likelihood ratio λ is then used to perform the final decision

$$\begin{aligned} &\text{If } \lambda \geq c, \text{ do not reject } H_1 && \text{(x is speech)}, \\ &\text{If } \lambda < c, \text{ reject } H_1 && \text{(x is not speech)}, \end{aligned}$$

using a constant c depending on the input segment length.

Value	Mode
0	Quality
1	Low-bitrate
2	Aggressive
3	Very aggressive

Table 3.2. Effect of different aggressiveness values of WebRTC VAD.

In addition, the user may specify an "aggressiveness" level 0, 1, 2, or 3 (see Table 3.2), which affects the probability of rejecting hypothesis H_1. These four aggressiveness modes will be compared in the experiments described in Chapter 7.

4. Datasets

This chapter provides an overview of five different speech datasets that were used both for training and evaluating different SLI models during the experiments performed in this thesis. All datasets have different characteristics, which will be discussed in the sections below. These datasets were chosen for this thesis because all of them have reference SLI models, with existing results available in the literature. These results will be reproduced in this thesis and are discussed in Chapter 7. It is worth noting that these five datasets represent only a small subset of all the public SLI datasets that have comparable results in the literature. Some other important datasets will be discussed in Section 4.6.

The first five sections in this chapter are dedicated to specific datasets, ordered by time of release, with OGI-11L being the oldest dataset and AP19-OLR the most recent dataset. OGI-11L is also the smallest dataset with only 32 hours of data, while YTN-Aalto2019 is the largest with 1214 hours of data. The contents of each dataset will be summarized in a table and two figures. Each table contains a column of original labels proposed by the dataset authors, while the BCP-47 column contains closest valid language tags, unless the original labels are already valid IETF BCP-47[1] tags or if suitable tags are not available. Each pair of figures depict the cumulative distribution of utterance durations in seconds for utterances in the training and test sets. Each figure also contains quartile values $Q_1, Q_2,$ and Q_3 in seconds, e.g. Q_2 is the median utterance length.

[1] Internet Engineering Task Force Best Current Practices 47 (Phillips and Davis 2019). Tags retrieved from *IANA - Language subtag registry* (2019).

4.1 OGI-11L (1994)

Label	BCP-47	Language	Speech (hours)
ma	zh	Chinese	2.6
ge	de	German	2.9
en	en	English	5.2
fa	fa	Farsi	2.6
fr	fr	French	2.9
hi	hi	Hindi	2.6
ja	ja	Japanese	2.4
ko	ko	Korean	2.2
sp	es	Spanish	3.1
ta	ta	Tamil	2.8
vi	vi	Vietnamese	2.5
		All	31.7

Table 4.1. 11 languages of the OGI-11L dataset.

Overview The first Oregon Graduate Institute (OGI) multi-language telephone speech corpus consists of freely spoken monologues or read speech in 10 different languages by 100 different speakers, recorded at a sample rate of 8 kHz over the telephone (Muthusamy et al. 1992). The data was collected by an automated telephone system, which played pre-recorded questions for the participants, who then gave responses of varying length. Most of the answers are rather short, except for the "story-before"[1] (stb) utterances, which are between 45 and 50 seconds. The 10-language dataset was expanded to 11 different languages by Cole and Muthusamy (1994), and later to 22 different languages by Lander et al. (1995). This thesis uses the same abbreviation as Li et al. (2013) and refers to the 11-language dataset as OGI-11L, which is outlined in Table 4.1 and used in the experiments described in Chapter 7. According to Li et al. (2013), the OGI speech datasets can be considered the first large-scale data collection effort towards producing a standard SLI dataset. The OGI datasets have been used extensively in SLI experiments (Muthusamy and Cole 1992; Hazen and Zue 1993; Zissman 1996; Torres-Carrasquillo et al. 2002; Martínez et al. 2011; Alphonsa et al. 2017). The popularity and originality of OGI-11L is the main reason why this dataset was included in this thesis.

Availability License purchased and data downloaded from the Linguistic Data Consortium at the University of Pennsylvania[2] (LDC) on 2019-08-09.

[1] During the data collection of OGI, participants were asked to speak for 60 seconds about any topic of their choosing. Shortly before the time was up, the recording system played a short beep to indicate the recording is about to end. The "story-before" files contain the recording up to that point (Muthusamy et al. 1992).

[2] https://www.ldc.upenn.edu/LDC94S17 (visited on 2020-03-18)

Training-test split Since OGI-11L does not provide a fixed training-test split, the approach followed in this thesis is based on the description by Zissman (1996), where only the "story-before" (`stb`) utterances are used. First, the amount of test data is chosen to be one hour of speech. Then, test set speakers are drawn uniformly at random from all speakers, one at a time, for one language at a time. For each chosen test speaker, all `stb` recordings that contains speech by that test speaker are chosen. From these recordings, the first 30 seconds is partitioned into three non-overlapping 10 second test utterances, for each recording. All resulting 10 second utterances are then added to the test set, and the chosen test speaker is removed from the set of all speakers. When the total duration of all test utterances reaches one hour of speech, the training set is all `stb` recordings of the remaining speakers. No separate validation set is used. Note that Table 4.1 includes all utterances in OGI-11L, whereas the total duration of all `stb` recordings is only 15 hours, as seen in Figure 4.1.

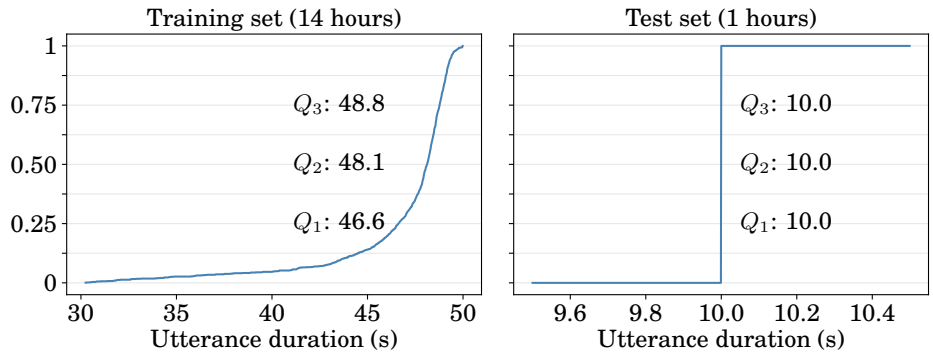


Figure 4.1. Cumulative distribution of utterance durations in OGI-11L and quartiles in seconds. Most training set utterances are between 45 and 50 seconds, while all test set utterances are exactly 10 seconds.

4.2 MGB-3 (2016)

Label	BCP-47	Language	Speech (hours)
`egy`	`arz`	Egyptian Arabic	16.4
`glf`	`afb`	Gulf Arabic	14.2
`lav`	`lav`	Levantine Arabic	14.3
`msa`	`arb`	Modern Standard Arabic	14.3
`nor`	`nor`	North African Arabic	14.6
		All	73.8

Table 4.2. 5 Arabic dialects/languages of the MGB-3 dataset. Levantine Arabic and North African Arabic do not have dedicated BCP-47 tags.

Overview The third Multi-Genre Broadcast speech recognition challenge (MGB-3) focuses on Arabic language speech recognition and Arabic dialect identification of modern standard Arabic and 4 regional, Arabic language groups (Ali et al. 2016). The speech data consists of 16 kHz recordings of broadcast news from the Al Jazeera news channel, and the data has been segmented into utterances of different lengths to avoid speaker overlap and to minimize the amount of non-speech frames within samples (Bahari et al. 2014). Due to significant similarities between these dialects, this dataset poses a challenging, state-of-the-art level classification task (Shon et al. 2018). MGB-3 was chosen for this thesis due to its state-of-the-art difficulty and easy availability.

Availability Public, downloaded from GitHub[1] on 2019-08-14. Regarding the naming convention of data directories, it is assumed `dev.vardial2017` contains the validation set of MGB-3, `train.vardial2017` contains the training set, and `test.MGB3` contains the test/evaluation set.

Training-test split MGB-3 provides a pre-defined training-test split, with an additional validation/development set. However, this thesis follows the approach of Shon et al. (2018), who chose randomly 90% of utterances from the validation set, created 4 new copies of each utterance, and included this 5-fold augmented validation set into the training set. Table 4.2 shows durations of the unmodified dataset, while Figure 4.2 includes the augmented training set and therefore appears to contain more data. The remaining 10% of utterances in the validation set are not used.

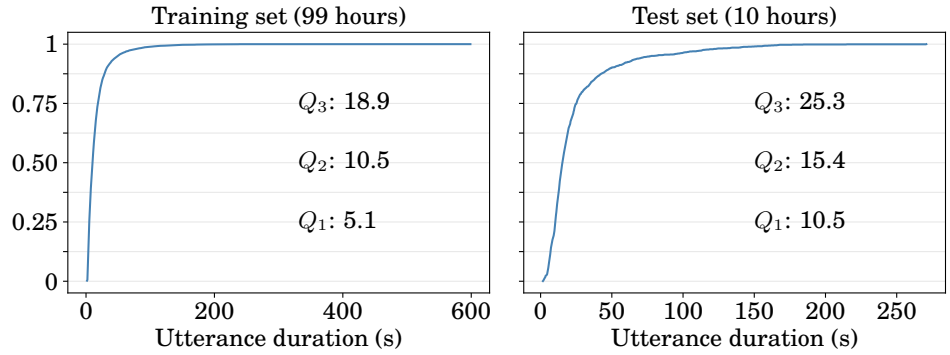


Figure 4.2. Cumulative distribution of utterance durations in MGB-3 and quartiles in seconds. Most utterances are between 10 and 20 seconds, although there are a few outliers with significantly longer duration.

[1] `https://github.com/qcri/dialectID/tree/ee6e7e7ca84098eda75cda61892a0ccbd3d8d0bd/data` (visited on 2020-02-10).

4.3 SBS (2018)

Label	BCP-47	Language	Speech (hours)
bulgarian_bg	bg	Bulgarian	20.8
belarusian_by	be	Belarusian	20.8
czech_cz	cs	Czech	20.8
croatian_hr	hr	Croatian	20.8
macedonian_mk	mk	Macedonian	20.7
polish_pl	pl	Polish	20.8
serbian_rs	sr	Serbian	20.7
russian_ru	ru	Russian	20.8
slovene_si	sl	Slovenian	20.7
slovak_sk	sk	Slovak	20.8
ukrainian_ua	uk	Ukrainian	22.8
		All	228.3

Table 4.3. 11 Slavic languages of the SBS dataset.

Overview The Slavic Broadcast Speech (SBS) dataset by Mateju et al. (2018) consists of broadcast news speech in 11 different Slavic languages (see Table 4.3), with an evaluation set consisting of 5 second utterances. All audio files have a sample rate of 16 kHz, although some files had corrupt file headers, which were fixed by performing a redundant resampling operation from 16 kHz to 16 kHz with *SoX - Sound eXchange* (2015). The automatic data collection methods that were used to create the original dataset are described in more detail by Nouza et al. (2016). SBS was chosen for this thesis due to its easy availability.

Availability Public, downloaded on 2019-11-14 from the URL[1] specified by Mateju et al. (2018).

Training-test split The training and test sets are pre-defined by the directory structure. No separate validation set is defined. We can see from Figure 4.3 that most test utterances are close to 5 seconds, while the training utterances are slightly shorter with a median length of 4 seconds.

[1] https://owncloud.cesnet.cz/index.php/s/gXHKFs9UDEqe34G (visited on 2019-11-14)

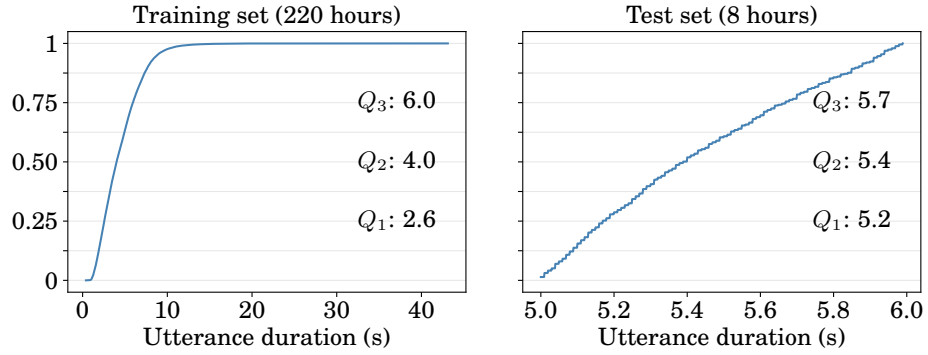


Figure 4.3. Cumulative distribution of all utterance durations in SBS. Most utterances have a duration of approx. 5 seconds.

4.4 YTN-Aalto2019 (2019)

Label	BCP-47	Language	Speech (hours)
`chinese`	`zh`	Chinese	202.7
`english`	`en`	English	200.1
`french`	`fr`	French	200.0
`german`	`de`	German	203.1
`russian`	`ru`	Russian	201.3
`spanish`	`es`	Spanish	207.0
		All	1214

Table 4.4. 6 languages of the YouTube News dataset and the amount of speech per language in the YTN-Aalto2019 instance used in this thesis.

Overview The YouTube News (YTN) dataset was collected by Bartz et al. (2017) and contains mostly broadcast news speech in 6 different languages. The dataset is collected automatically from news channel profiles on *YouTube* (2019) using the *youtube-dl: Download videos from YouTube* (2019) tool. After the waveform is extracted from a downloaded video, it is resampled to 16 kHz with a single channel and partitioned into non-overlapping 10 second utterances. Note that the acoustic data collected by Bartz et al. (2017) was not made public and could therefore not be used in this thesis. Instead, the original data collection program was applied during this thesis to collect a YTN dataset instance for the purposes of this thesis. We call this instance **YTN-Aalto2019**. Whenever acoustic data of YTN is mentioned, we are referring to the acoustic data from YTN-Aalto2019. During the data download procedure, it was noted that the default download parameters of 1200 videos per language resulted in significantly less speech in French compared to other languages. This

was compensated by setting the upper limit on the amount of downloaded French language videos to 2400, which yielded approximately 200 hours of French data. Furthermore, in order to balance the amount of data across all languages, 10 second utterances are picked uniformly at random from the downloaded, partitioned data until each language has approximately 200 hours of data. It is worth noting that compared to the careful segmentation of the MGB-3 dataset, the naive segmentation of YTN is likely to create samples with a significant amount of noise, or samples that do not even contain speech.

Availability The original YTN dataset used by Bartz et al. (2017) is not available. The instance used in this thesis, YTN-Aalto2019, was downloaded on 2019-11-25 from news channels on YouTube[1] using the original script[2] by Bartz et al. (2017). The metadata of YTN-Aalto2019 is available online[3], but access to the dataset is limited as of writing this thesis.

Training-test split YTN-Aalto2019 contains a training, validation, and test set. These were created randomly from the 10 second utterances, partitioned from the audio of all videos. First, the amount of test data and validation data is chosen to be ten hours of speech per language. Then, validation and test set videos are drawn uniformly at random from all videos, one at a time. For each chosen test (validation) video, all its 10 second utterances are moved into the test (validation) set. When the test set and validation set both contain ten hours of speech per language each[4], all remaining 10 second utterances will form the training set, which contains approximately 1081 hours of speech.

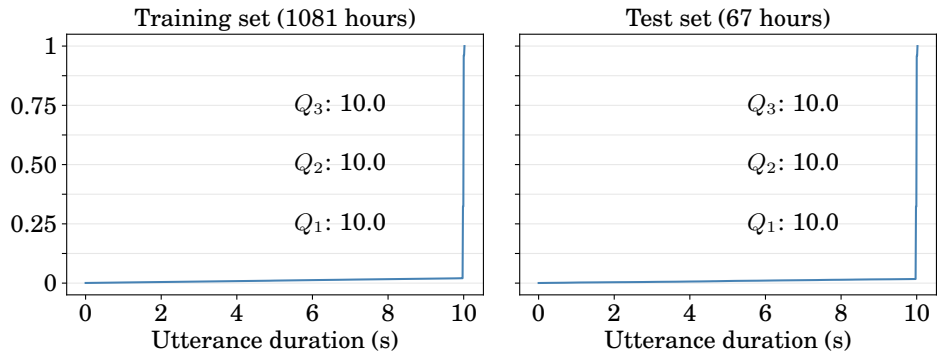


Figure 4.4. Almost all utterances in YTN-Aalto2019 are exactly 10 seconds. The validation set has 66 hours of speech but has otherwise same statistics as the test set.

[1] List of news channels used: `https://github.com/HPI-DeepLearning/crnn-lid/blob/00bb15c391f692a8fc65d073cb230254ca7a7cce/data/sources.yml` (visited on 2020-02-07)

[2] The script used for downloading the data: `https://github.com/HPI-DeepLearning/crnn-lid/blob/00bb15c391f692a8fc65d073cb230254ca7a7cce/data/download_youtube.py` (visited on 2020-02-07)

[3] `http://urn.fi/urn:nbn:fi:lb-2020041701` (visited on 2020-04-23)

[4] Approximately 66 hours of speech in total per set.

4.5 AP19-OLR (2019)

Label	BCP-47	Language	Speech (hours)
ct-CN	yue-CN	Cantonese	25.7
id-ID	id-ID	Indonesian	23.9
ja-JP	ja-JP	Japanese	18.5
ka-CN	kk-CN	Kazakh	22.2
ko-KR	ko-KR	Korean	20.9
ru-RU	ru-RU	Russian	23.2
ti-CN	bo-CN	Tibetan	20.4
uy-CN	ug-CN	Uyghur	27.6
vi-VN	vi-VN	Vietnamese	24.7
zh-CN	zh-CN	Chinese	25.6
unknown	N/A	Mixed OOS languages	33.6
		All	269.4

Table 4.5. 10 languages of AP19-OLR. In addition, the training set contains a group of various out-of-set (OOS) languages.

Overview The Asia-Pacific Oriental Language Recognition challenge 2019 (AP19-OLR) consists of three SLI tasks for 5 languages spoken in China and 5 additional languages that are spoken across eastern Asia (Tang et al. 2019). The training data used in AP19-OLR consists of all data from the three previous OLR challenges: AP16 (Wang et al. 2016; KingLine Data Center 2016), AP17 (Tang et al. 2017), and AP18 (Tang et al. 2018a). In addition, participants of AP19-OLR were allowed to utilize the free Chinese speech corpus (Wang et al. 2015) in their experiments. However, this dataset was not used in this thesis since including a large amount of Chinese language speech would have created a significant class imbalance, since the AP19-OLR dataset is already quite balanced (see Table 4.5). The AP19-OLR challenge included three different SLI tasks: short utterance, cross channel, and zero-resource. This thesis focuses only on the short utterance task, where the duration of all test set utterances is exactly one second.

Availability Restricted, access granted by the AP19-OLR challenge organizers[1].

Training-test split Pre-defined splits with one training set, three validation sets and three test sets. The three evaluation sets are for the three different tasks of AP19-OLR. This thesis will only use task one, which is the short utterance task, and whenever we refer to the test set of AP19-OLR, it is the short utterance task evaluation set.

[1] http://cslt.riit.tsinghua.edu.cn/mediawiki/index.php/OLR_Challenge_2019 (visited on 2020-

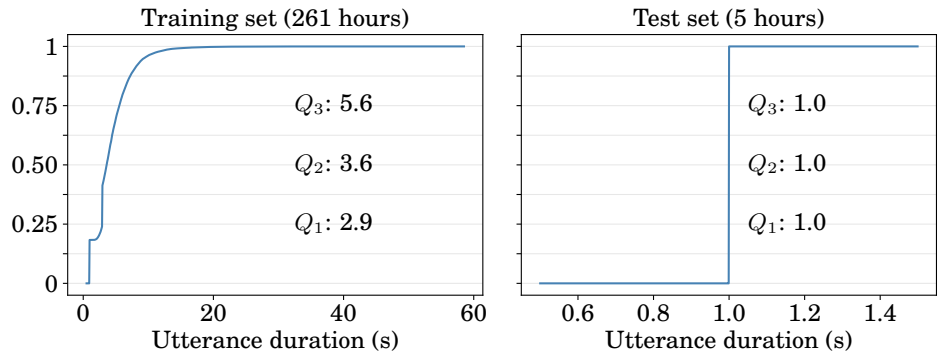


Figure 4.5. Cumulative distribution of all utterance durations in AP19-OLR. All test set utterances are one second. Note that the training set contains also the test sets of OLR challenges from previous years, which explains the notable amount of 1 second training utterances.

4.6 Other notable datasets

In addition to the datasets utilized during this thesis, several other important datasets exist and should be mentioned either due to their popularity or relevance to this thesis.

NIST LRE National Institute of Standards and Technology (NIST) language recognition evaluation[1] (LRE) is a series of popular SLI challenges that have been organized since 1996. The most recent event took place in 2017 (Sadjadi et al. 2018), and the datasets from past events have been used extensively in SLI experiments (Singer et al. 2003; Campbell et al. 2004; Tong et al. 2006; Navratil 2006; Glembek et al. 2008; Ng et al. 2010; Martínez et al. 2011; Lopez-Moreno et al. 2014; Richardson et al. 2015; Frederiksen et al. 2018; Snyder et al. 2018a; Padi et al. 2019). It is worth noting that in 2015 there was a large mismatch between the validation and test sets, as noted by Zazo et al. (2016a), Gelly et al. (2016), and He et al. (2016b). As of writing this thesis, all NIST LRE datasets require licenses to be purchased from the Linguistic Data Consortium at the University of Pennsylvania (LDC)[2], before the speech data can be accessed. Despite this limitation, the NIST LRE series have been a central part of the SLI community for several years, and many important contributions can be seen as directly or indirectly related to NIST LRE (Li et al. 2013). One such, direct contribution is the C_{avg} evaluation metric, which is discussed in Section 3.3.

03-20)

[1] `https://www.nist.gov/itl/iad/mig/language-recognition` (visited on 2020-02-10)

[2] `https://www.ldc.upenn.edu/` (visited on 2020-02-10)

Common Voice *Mozilla Common Voice* (2020) is an open-source, free collection of crowd-sourced datasets containing read speech in various languages with highly varying channel conditions. All files are distributed in MP3 format, accompanied with metadata for each dataset. Each dataset has been validated by volunteers, who judge the correctness of utterances by comparing the audio content to the textual prompt that was originally given to the speaker and then apply a binary good/bad label for each utterance. The sums of all binary labels for each utterance are included in the metadata of all datasets. However, the amount of validated data[1] varies greatly between languages. For example, English has 1118 hours of validated data while Japanese has only 3 hours. Furthermore, most datasets have considerable gender imbalances, which might contribute to speaker variability, as discussed in Section 3.2.

ADI17 The MGB-3 challenge was later followed by MGB-5, which included a dialect identification task with a significantly larger dataset than was used in MGB-3. This dataset, Arabic Dialect Identification for 17 countries (ADI17), contains 3000 hours of training data and 57 hours of validation and test data (Ali et al. 2019). The test set is divided into 3 subtasks, each with varying utterance lengths. ADI17 was collected from *YouTube* (2019) using publicly available scripts[2], which allows anyone to download the dataset. However, considering the dynamic nature of the internet, it is unlikely that a downloaded dataset will contain exactly the same utterances as in the dataset used by the MGB-5 participants, which is precisely the same problem as with YTN. While MGB-5 was not used during this thesis, a natural extension to the experiments of this thesis would be to compare how each model performs on MGB-3 and MGB-5.

[1] `https://voice.mozilla.org/en/datasets` (visited on 2020-02-10)

[2] `https://github.com/swshon/arabic-dialect-identification` (visited on 2019-01-28)

5. Existing work

In Section 2.2 we briefly discussed the taxonomy of phonotactic, acoustic-phonetic, and language embedding approaches to SLI. This chapter studies some existing work, with examples of SLI models that fall into at least one these three categories. Some of the discussed SLI models were implemented during this thesis and will be discussed in Chapter 6. The implemented approaches will be applied to datasets discussed in Chapter 4 and these experiments are discussed in Chapter 7.

5.1 Phonotactic approaches

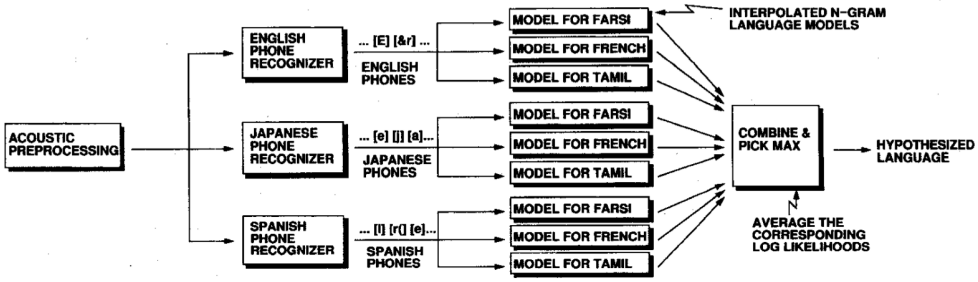


Figure 5.1. An example of a PPRLM approach (Zissman 1996, Fig. 3, cropped). Note how none of the PR languages match those of the LMs.

Muthusamy and Cole (1992) trained several NN models for automatically segmenting utterances into 3 ms time frames, such that each frame has been assigned one of 7 language independent, phonetic categories. All models were trained using speech data from the 10-language OGI dataset (not OGI-11L), discussed in Section 4.1. They experimented with several NN approaches, such as identifying all 10 languages using the same NN and identifying one target language at a time from all languages. Their 10-language NN achieves an identification accuracy of 47.7%, while the highest accuracy for identifying one language is for Tamil at 86.0% and the lowest for English and French, both at 69.5%. In this thesis, several exper-

iments were performed to identify all languages of the OGI-11L dataset with a single network, using only spectral features as input. These results suggest that it is possible to significantly outperform the 10-language NN.

Zissman (1996) compared the SLI performance of 4 different algorithms: GMM, PRLM, PPRLM, and parallel phone recognition, which were all applied to the OGI-11L dataset. PPRLM (see Figure 5.1) with additional enhancements is reported to achieve an error rate of 21% on the OGI-11L test set consisting of 10 second utterances. It is worth noting that error rates for the PR models were measured as a normalized edit distance, while SLI performance was measured using "March 1994 NIST guidelines", but a written definition of these guidelines could not be found at the time this thesis was written. Therefore, an absolute comparison of these results and the error rates reported in this thesis cannot be made, although the confusion matrix depicting classification results on the 10-second test set may give a reasonable approximation. The training-test split adopted in this thesis for OGI-11L follows the approach described by Zissman (1996), where the test set consists of only 10-second utterances. This is defined in more detail in Section 4.1. As an interesting side note, Zissman (1996) suggested that even though individual PR models may exhibit high phoneme recognition error rates, it does not necessarily imply high SLI error rates since a combination of several PR models can still provide good SLI performance. This suggestion will be later investigated using a multilingual PRLM model with varying degrees of mismatching languages between the training data of the PRLM model and the SLI datasets. We return to this topic in Section 6.1.

Torres-Carrasquillo et al. (2002) introduced a GMM-based SLI model, using MFCCs with shifted-delta-cepstral (SDC) features for capturing temporal information of adjacent cepstral coefficients, and compared it to a PPRLM-based approach. They reported that replacing conventional cepstral features with SDCs, as well as increasing the GMM order, both improve SLI performance noticeably. By training a GMM of order 1024 with SDC features, they managed to match the performance of a PPRLM-based approach, using the evaluation set of the CALLFRIEND speech corpus (Canavan and Zipperlen 1996), containing 12 languages. The models were also evaluated on the OGI-11L test set with 10-second utterances, and the reported $\mathrm{EER}_{\mathrm{avg}}$ values are approximately 29% for GMM and 21% for PPRLM (Torres-Carrasquillo et al. 2002, deduced from Figure 7). The NN models experimented with during this thesis have been evaluated on the same OGI-11L test set, which makes these results directly comparable with each other. It is also worth noting that the feature set resulting from concatenating MFCCs and SDCs (MFCC-SDC) have since been used extensively in several SLI experiments (Li et al. 2013; Lozano-Diez et al. 2015; Khurana et al. 2017; Zhang et al. 2019).

Tang et al. (2018b) proposed to utilize the frame-level bottleneck features

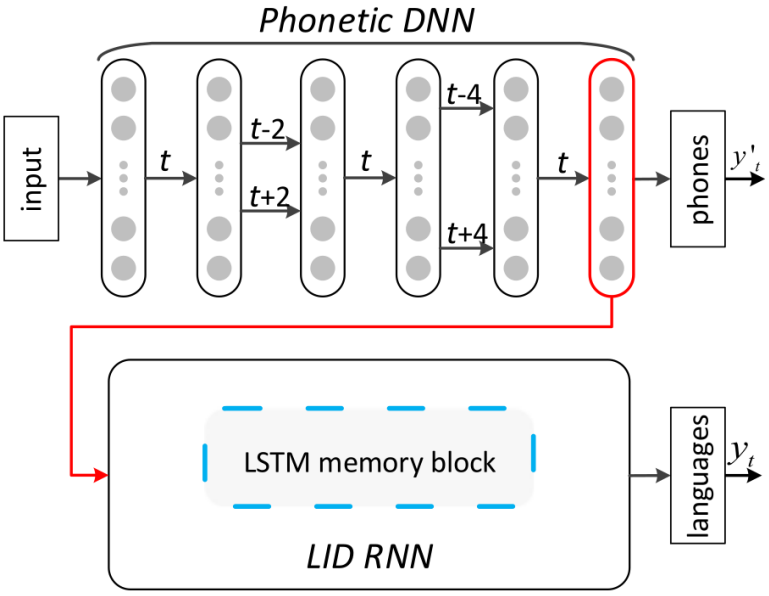


Figure 5.2. PTN approach (Tang et al. 2018b, Fig. 3b). Note how the input of the SLI classifier is an intermediate representation of the phones from a hidden layer of the PR model, rather than the phone labels.

from a phone-discriminative model as input to an recurrent NN (RNN) based SLI model (see Figure 5.2). They call this approach the phonetic temporal neural (PTN) model. This is in contrast with traditional PRLM models that use the token-level output (e.g. phone sequence from the PR) as input to the SLI model. They argued that while i-vector (discussed in Section 5.2) based SLI models have proven to be accurate, i-vectors require long input utterances to accumulate enough distributional properties to make a classification decision, making them infeasible for practical applications. PTN was applied in the experiments of this thesis and is discussed in Section 6.1.

Ren et al. (2019) applied a two-stage training approach for solving a dialect identification task of 10 Chinese regional dialects using a phonotactic approach. Their model consists of a convolutional NN (CNN) for feature extraction, based on ResNet-18 (He et al. 2016a), and two bidirectional long short-term memory (BLSTM) layers for sequence classification. They used a two-staged training procedure, where the model is first trained two solve a phoneme sequence annotation task using connectionist temporal classification (CTC) (Graves et al. 2006), based on the transcribed training data. In the second stage, weights of the CNN are frozen, and the BLSTM is retrained to solve the dialect identification task using cross-entropy loss, based on the language classes. They reported an accuracy of 87.7% on test utterances less than 3 seconds, and 90.0% on test utterances longer than 3 seconds. Although this approach was not used in this thesis due to time constraints, the end-to-end architecture proposed by Ren et al. (2019) contains several similarities to the NN approaches used in this thesis, making

it a natural continuation if the experiments made in this these is to be expanded.

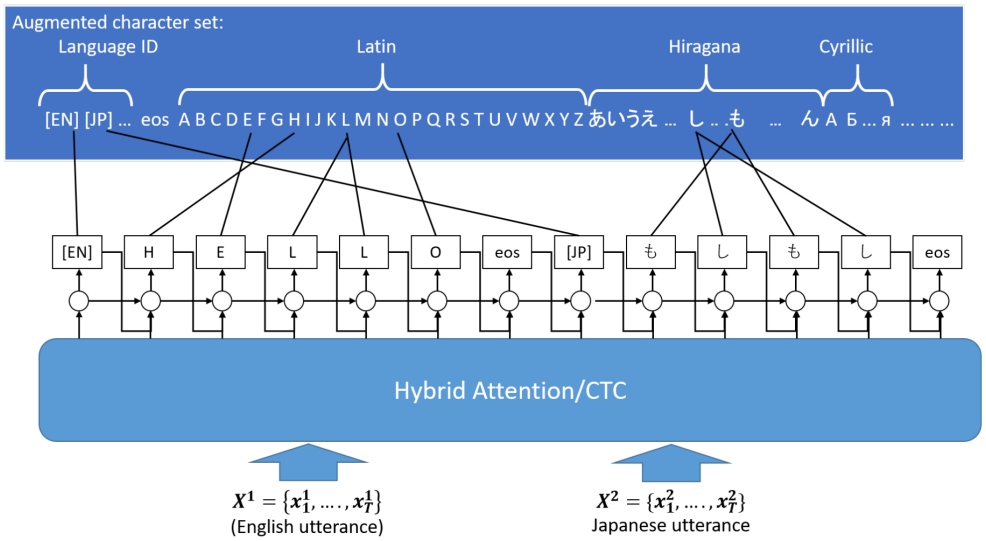


Figure 5.3. Speech-to-text system with a multilingual character set, capable of recognizing speech in 10 different languages (Watanabe et al. 2017, Fig. 1).

Watanabe et al. (2017) trained a multilingual speech-to-text system for ten languages using an hybrid attention-CTC based approach. The set of characters was chosen as the union of all character sets from each of the ten languages (see Figure 5.3), allowing the model to jointly not only map speech to text but also recognize the language. They noted that the system is capable of reaching a SLI error rate of close to 0%, recognizing the correct language from almost all input utterances. While this system was not applied in this thesis in any way, the impressive performance of both this system and the model by Ren et al. (2019) suggest that a CTC based approach using transcribed speech data can provide superior performance compared to other SLI models. This could prove a valuable fact to consider when designing a practical SLI system.

5.2 Language embeddings

An embedding (or imbedding) can be viewed as an injective function $f : A \to B$, where some property of interest is retained in $f(x) \in B$ for all $x \in A$ (Bishop and Goldberg 1968, pp. 40–41). From the point of view of SLI, we would like to discover a model f such that it maps utterances x into some low-dimensional space B, while retaining all language cues of x. We will discuss two approaches commonly applied in SLI and begin with the *i-vector* model.

i-vectors Although originally proposed for speaker verification, the i-vector (short for “identity-vector”) utterance representation by Dehak et al.

(2011a) has been extensively applied also to SLI. An i-vector representation is defined by the language-dependent supervector $M = m + Tw$, where m is a language-independent supervector (background model), T is a low-dimensional total variability matrix that contains speaker and channel variabilities, and w is a latent variable assumed to be normally distributed with zero mean and unit variance (Dehak et al. 2011b). The maximum a posteriori mean estimate of w is called an i-vector. Dehak et al. (2011a) also proposed a cosine kernel for comparing the similarity of two i-vectors by angle rather than magnitude, in order to improve system robustness against variability and to simplify the scoring process. Martínez et al. (2011) trained a GMM with 2048 components for i-vector extraction, using MFCC-SDC features as input, extracted from a dataset containing 54 languages. They reported that the i-vector based model outperforms a state-of-the-art model with a 7% lower C_{avg}. They also argued that an i-vector based approach is beneficial from a computational cost perspective, as the i-vector extractor can be trained independently from the SLI task, which in turn can then be solved in the i-vector space produced by the i-vector extractor. Tong et al. (2016) argued that using a low-dimensional vector representation for utterances, instead of the original signals, eliminates the need for separate speech pre-processing steps, therefore simplifying the SLI task and allowing wider collaboration in SLI competitions.

Representation learning Another way to discover low-dimensional embedding spaces in an unsupervised or semi-supervised manner is through representation learning (Weston et al. 2008; Goodfellow et al. 2016d). In this approach, a DNN model is trained to solve e.g. a classification task, which results in the useful side-effect of learning different feature representations in each layer of the model (see Section 3.1 for layer definition). Then, after training, we could choose f to be the concatenation of all hidden layers up to an arbitrary, intermediate layer. In other words, the output space of this intermediate layer will be the embedding space B. This has been successfully utilized in many problem domains, such as speaker identification and SLI. The fundamental assumption behind these DNN based approaches is that the classification performance of a DNN classifier is assumed to imply how well the properties of each class has been embedded into the hidden space of the DNN (Wang et al. 2017).

Gelly and Gauvain (2017) trained a BLSTM based language-vector extractor, which embeds perceptual linear predictive coefficients onto a hypersphere, guided by a cosine similarity based angular proximity loss function. This model was compared to a phonotactic and an i-vector model, and all models were trained on the NIST LRE 2007 and 2015 datasets. They reported that the language-vector model outperforms the baseline i-vector model on both evaluation sets, and outperforms even the phonotactic model on the LRE 2015 evaluation set. Furthermore, a fusion of the phonotactic model and the language-vector model leads to a significant re-

duction in the error rates on both evaluation sets. They also point out that the language-vector model requires significantly less trainable parameters compared to the baseline models.

Snyder et al. (2018a) proposed a TDNN based approach, where the NN is trained to solve a SLI task and then language embeddings are extracted from the outputs of a fully-connected layer inside the network. They call these embeddings *x-vectors*, which are designed to be used in a similar way as i-vectors, for example using linear discriminant analysis or GMM backends for classification. They reported significant SLI performance improvements over the baseline i-vector system when evaluated on the NIST LRE 2017 (Sadjadi et al. 2018) evaluation set. The baseline SLI model for the AP19-OLR competition by Tang et al. (2019) is based on x-vectors and is applied in this thesis. In addition, an x-vector implementation was created during this thesis (see Section 6.3) and was used in thesis experiments discussed in Chapter 7. However, note that we do not make use of the embedding capabilities of the x-vector model, but rather apply it as a discriminant model, leaving the embedded vectors untouched. Nevertheless, training the x-vector architecture on a vast amount of data, containing several different languages might provide a valuable, low-dimensional language-vector representation. This might enable more accurate SLI using some sophisticated statistical methods from outside the scope of this thesis.

5.3 Acoustic-phonetic approaches

In this section, we study existing acoustic-phonetic SLI models, which are trained using spectral or cepstral speech representations labeled by language. These models rely on the assumption that each language has a unique acoustic structure, which can be learned using a NN classifier (Muthusamy and Cole 1992). Note that most authors performed several experiments with both acoustic-phonetic and phonotactic approaches, which makes it difficult to make a clear distinction between work done using one of the approaches but not both. Some of the work discussed here has significant overlap also with the previous sections of this chapter.

Lopez-Moreno et al. (2014) compared a baseline i-vector model to an 8-layer DNN model, both trained on Google 5M SLI and NIST LRE 2009 datasets. The Google 5M SLI dataset was collected from voice search queries and consists of 5 million utterances with average duration of 2.1 seconds, 87.5 hours per language, and 2975 hours in total. They conclude that the DNN model outperforms the baseline model with an 70% relative improvement measured by C_{avg} when trained on the Google 5M SLI dataset, and 43% relative improvement measured by EER_{avg} when trained on 8 languages chosen from the NIST LRE 2009 dataset.

Furthermore, they trained and evaluated their models on the NIST LRE 2009 dataset with increasing amounts of training data, and noted that the i-vector model outperforms the 8-layer DNN until there is approximately 20 hours of training data available per language, after which the DNN model begins to perform better as the amount of data is increased.

Lozano-Diez et al. (2015) compared several CNN architectures to a baseline i-vector model trained on the NIST LRE 2009 dataset and noted that the i-vector model outperforms all CNN architectures. However, they point out that all CNN models require at least 100 times less trainable parameters compared to the i-vector model. Also, performing a fusion of the i-vector model with the CNN models performs better compared to evaluating each model standalone, which suggests that the CNN models are capable of learning language information not captured by the i-vector model.

Zazo et al. (2016b) compared several LSTM architectures to a baseline i-vector model trained on the NIST LRE 2009 dataset using a similar subset of 8 languages as used by Lopez-Moreno et al. (2014). They reported that 4 out of 5 LSTM architectures outperform the baseline i-vector model when measured on EER_{avg}. Furthermore, they noted that all their LSTM architectures have 5 to 21 times less trainable parameters compared to the i-vector baseline.

Bartz et al. (2017) collected 1500 hours of speech data from online broadcast news videos in 6 different languages and partitioned each video into non-overlapping 10 second utterances. 10% of the dataset is chosen randomly for a held-out test set. They then extracted decibel-scale spectrogram images for each utterance and trained a SLI model by classifying the spectrogram images using CNNs and BLSTMs. They reported that the CNN model achieves an F_1 score of 91% on the test set and adding a BLSTM layer (CNN + BLSTM = CRNN) did not yield significant improvements. However, when replacing the simple CNN model with a larger, Inception-v3 based architecture by Szegedy et al. (2016), they achieve an F_1 score of 95%, and adding a BLSTM layer improves the score to 96%. The CRNN model was used during the thesis experiments, and is discussed in more detail in Section 6.4.

Shon et al. (2018) trained a CNN based model on log-scale Mel-spectra extracted from the MGB-3 dataset and achieved an C_{avg} of 17.6%. They noted that using log-scale Mel-spectra yields slightly better performance than using MFCCs. Also, performing data augmentation by decreasing and increasing the speed of audio signals, as well as decreasing and increasing the signal amplitudes, is reported to improve the results. When combining the CNN output with phonotactic information provided by a Hungarian-language PR, the resulting fusion system achieved an C_{avg} of 12.5%. The CNN model was used during the thesis experiments as a discriminant model, without the PR component, and is discussed in more detail in

Section 6.3.

Mateju et al. (2018) collected 230 hours of broadcast news speech in 11 different Slavic languages from the internet. In addition to the baseline i-vector model, they trained three different NN models on the collected dataset. Model input consisted of MFCCs, filter bank coefficients, as well as bottleneck features produced by a DNN based phoneme classifier, trained to recognize Czech language triphones. They reported that MFCCs are slightly more accurate compared to filter banks, while bottleneck features clearly outperform MFCCs in all SLI models, of which a bidirectional gated recurrent unit (BGRU) based model was best. The BGRU model was used during the thesis experiments, and is discussed in more detail in Section 6.2.

Ma et al. (2019) compared i-vector and LSTM based SLI models, utilizing bottleneck features from a DNN based phoneme classifier, trained to recognize Mandarin Chinese language triphones. They evaluated the models on the short-utterance task of the AP17-OLR dataset (Tang et al. 2017), and reported that the LSTM based model trained on bottleneck features outperforms the baseline i-vector model. Furthermore, the LSTM based model performs even better when short test utterances are augmented by time-scale modifications (80% and 120%) using a phase vocoder, which does not degrade pitch or prosody information.

Miao et al. (2019) extended the x-vector TDNN proposed by Snyder et al. (2018a) by adding a two-dimensional CNN to extract features from frame level acoustic features, followed by an LSTM-layer for capturing temporal correlations, as well as a time-frequency attention mechanism before the pooling layer. They trained their system on the NIST LRE 2007 dataset and reported significant C_{avg} reductions compared to all baselines.

Padi et al. (2019) argued that the language discerning information of an utterance might sometimes be contained in a relatively short section of the audio signal, which makes the usage of i-vectors infeasible, since this short-term information might be lost if the whole utterance is represented as a single i-vector. As a solution, they suggest to use an attention based model which emphasizes the important sections of an audio signal. They reported that their model is robust against noise and outperforms the baseline models at all tested signal-to-noise ratio (SNR) levels.

5.4 Large-scale SLI

Since most NN approaches benefit from using more data (see Section 3.1), it is usually desirable to try training with as much data as possible. For example, Lopez-Moreno et al. (2014) noted that SLI based on large DNN models becomes more accurate as the amount of data grows. However, many simple approaches designed for small datasets become infeasible as

the amount of data grows. While this big data aspect is not the main topic of this thesis, a few interesting studies related to large-scale SLI should be mentioned since training with a large amount of speech data often requires alternative approaches.

Sak et al. (2014) compared large-scale deep LSTM architectures to DNN models on an LVCSR task consisting of 1900 hours of speech and noted how their 2-layer LSTM model achieves a word error rate (WER) of 10.9% within 48 hours of training, while the best DNN model achieves a WER of 11.3% in a few weeks of training.

Mazzawi et al. (2019) studied ways to automate the search process of SLI architectures and reported that an automatically discovered SLI model can outperform a human designed SLI model with an accuracy increase of 4% on a dataset containing 79 different languages. They suggest that human designed, LSTM-based SLI models are usually biased towards deep architectures with a large amount of parameters, while their NN architecture search algorithm prefers shallow LSTM models with fewer, but highly tuned hyperparameters. However, it is worth pointing out that the architecture search process is computationally expensive, and is reported to take one week using 15 parallel workers.

Wan et al. (2019) studied how to increase SLI accuracy of a large model with a target set containing 79 languages by reducing the amount of languages the model needs needs to consider during inference. As a motivation for their study, they noted that in a typical SLI system for North American users, approximately 95% users speak no more than 2 languages. They propose a generalization of the softmax loss function, called tuplemax loss, which considers only a subset of labels during inference, as opposed to softmax which always produces scores for all labels. They reported that tuplemax loss leads a lower mean error rate at 2.33%, while softmax loss leads to an error rate of 3.85%.

6. Models and building blocks

This chapter provides an overview of different models that are either used directly as SLI models, or combined to form SLI models. These SLI models are used during the experiments described in Chapter 7. Some of the SLI models have a one-to-one mapping to the datasets discussed in Chapter 4, since these models were each originally designed for specific datasets. In addition to training these models on the datasets they were designed for, they are also trained on all other datasets in order to gain comparable results. On the other hand, some of the models discussed in this chapter were not introduced in Chapter 5 since these models were not originally proposed and applied for SLI, but for other speech analysis tasks. All results are discussed in more detail in Chapter 7.

Overview We begin with an overview of all SLI models and model components which are used to build new SLI models in this thesis. Table 6.1 lists all models that are discussed in this chapter. Some of these models were implemented during this thesis using TensorFlow (Abadi et al. 2015), while following the architecture details as specified by the original authors as closely as possible. The implemented models are BGRU, CNN, CRNN, x-vector, LSTM for PTN, and multi-level time attention (MLA). Note that by "reimplementing" it is meant here that these models were constructed by merely combining existing TensorFlow layer implementations.

Model	Proposed by	Reference dataset
BGRU	Mateju et al. (2018)	SBS
CNN	Shon et al. (2018)	MGB-3
CRNN	Bartz et al. (2017)	YTN
x-vector	Snyder et al. (2018a)	AP19-OLR
NB-TLI	Jauhiainen et al. (2019b)	
CTC-PR	Karhila et al. (2019)	
LSTM	Hochreiter and Schmidhuber (1997)	
MLA	Yu et al. (2018)	
PTN	Tang et al. (2018b)	
SS	Milde and Biemann (2019)	

Table 6.1. Original SLI models and building blocks. Note that Snyder et al. (2018a) proposed the SLI x-vector model but did not report any results on the AP19-OLR dataset. This was done later by Tang et al. (2019).

Some of these models have no reference dataset from Chapter 4 or might not even be SLI models. Instead, they are used as building blocks to construct SLI models, which are listed in Table 6.2.

Model	Components
PRLM	CTC-PR, NB-TLI
SSLM	SS, NB-TLI
PTN	CTC-PR, LSTM
SS-PTN	SS, LSTM
x-vector (SS)	SS, x-vector
x-vector (CTC-PR)	CTC-PR, x-vector
MLA x-vector	MLA, x-vector

Table 6.2. SLI models constructed by combining models listed in Table 6.1.

Neural network architecture format Each NN architecture is listed in a table of layers (e.g. 6.3), where the shapes of input and output tensors are specified for each layer (see Section 3.1 for the definition of layers). Regarding the notation of layer input and output shapes, the shorthand $a/b := \lfloor \frac{a}{b} \rfloor$ is used for division of integers a and b, assuming $a > b > 0$. The input to the first layer of each NN model is assumed to be an utterance representation $\mathbf{X} \in \mathbb{R}^{T \times F}$, with input shape $T \times F$, and the output is assumed to be a vector of language scores $\mathbf{y} \in \mathbb{R}^{N}$, with output shape N, as discussed in Section 3.1.

If activation functions and batch normalization are shown after a layer, they also denote the order of computation. E.g. "FC 512 + ReLU + BN" denotes a 512-unit fully-connected (FC) layer with rectified linear unit (ReLU) activation, followed by batch normalization (BN). Configuration of convolutional layers is specified as "kernel×width×stride". In case of a 2-dimensional square kernel ("Conv2D"), the kernel size implies both

width and height.

6.1 SLI from tokenized speech

The models discussed in this section either tokenize speech by producing token sequences from speech representations or use token sequences as input for classifying languages. Within the scope of this thesis, token sequences consist of either acoustic unit sequences, discovered in an unsupervised manner, or International Phonetic Alphabet (IPA) based phoneme sequences. Some of these models operate directly on tokenized speech, which makes them TLI models, or indirectly by using the internal representation of a speech tokenizer, such as the outputs of a hidden layer in a PR model.

Multilingual phoneme recognition (CTC-PR) Karhila et al. (2019) proposed a two-component system for scoring pronunciations of language learners. The system consists of a multilingual, CTC-based phoneme recognizer (CTC-PR) and a regression model for pronunciation scoring. This thesis uses only the CTC-PR component (see Figure 6.1) for tokenizing speech into IPA based token sequences. Note that some IPA phone combinations that frequently occur together, such as diphthongs, are represented by a single token in the CTC-PR output. Input to CTC-PR consists of log-scale Mel-spectra with 60 Mel-frequency bins, $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T \times 60}$, extracted with a 512-point STFT from 25 ms wide windows with an offset of 10 ms. The CTC-PR instance used in this thesis is an expanded version, trained on three additional, Common Voice datasets: German, Spanish, and Italian. In total, all languages that were used for training CTC-PR are Finnish, Swedish, English in GB and US, German, Spanish, and Italian. It is worth noting that Common Voice datasets do not distinguish between dialects and language variants. Also, the CTC-PR instance used in this thesis was trained by its original authors, not the thesis author. In this thesis, CTC-PR is only used for extracting phoneme embeddings and to tokenize speech.

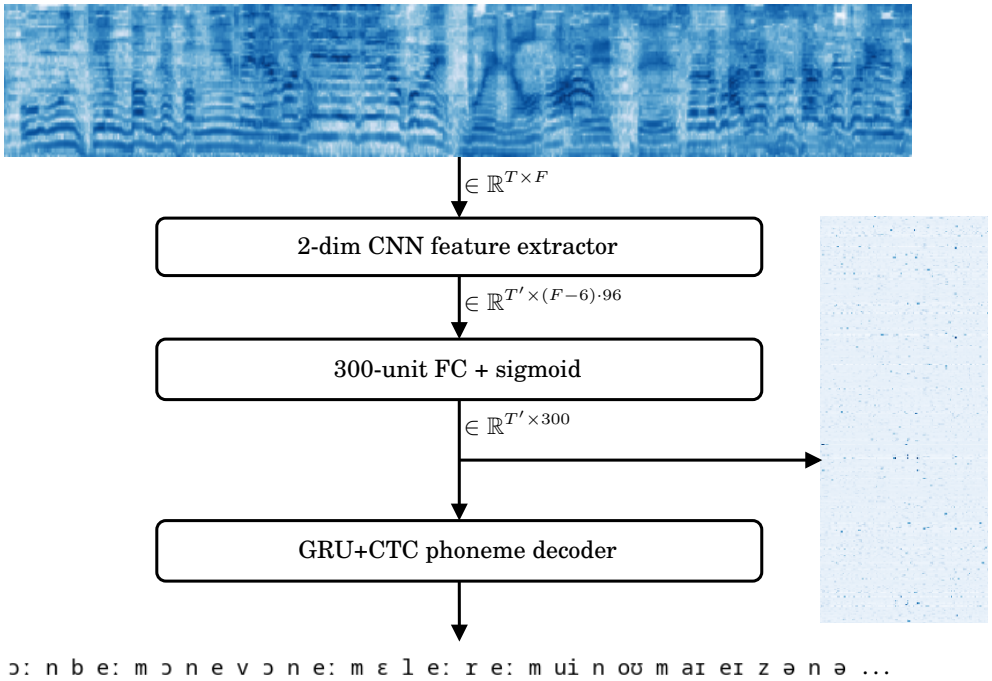


Figure 6.1. Simplified architecture overview of the CTC-PR model. In addition to mapping input speech representations to IPA phoneme sequences, we extract phoneme embeddings as bottleneck features from the FC sigmoid layer between the convolutional feature extractor and GRU layers.

Cuneiform TLI with naive Bayes (NB-TLI) Jauhiainen et al. (2019b) proposed to apply naive Bayes and n-gram based TLI methods (NB-TLI) on cuneiform texts in Sumerian and several dialects of Akkadian. The proposed model uses character n-grams of different lengths as input, trains language models for each language class, and predicts language scores for unseen input by comparing the n-gram features of the input to each trained language model. Whenever TLI is performed in the experiments, the NB-TLI model is used. All input token sequences for the NB-TLI model are space-separated strings of single characters.

PRLM The phone recognition followed by language modeling (PRLM) implementation used in this thesis is similar to the traditional PRLM approach of using a single language PR (Zissman 1996), except that the PR model is replaced by the multilingual CTC-PR model. For the LM model, NB-TLI is used. One important aspect to note is that the CTC-PR model is pre-trained with a fixed dataset and is only used as a feature extractor for producing text input for NB-TLI. This is in contrast with the conventional approach of training PR models using transcribed data in a language which is related to as many languages in the target set as possible. For example, Shon et al. (2018) trained an Arabic language PR model for MGB-3, Mateju et al. (2018) trained an Czech language PR model for SBS, and Miao et al. (2019) trained an English language PR for NIST LRE 2007.

SparseSpeech (SS) Milde and Biemann (2019) proposed a sequence-to-sequence autoencoder called SparseSpeech (SS) for unsupervised acoustic unit discovery (see Figure 6.2). The encoder and decoder both consist of 4 stacked 512-unit BLSTM layers, i.e. 8-layers in total. Sequence dropout is applied on the encoder output, dropping frames randomly with probability 0.67, to ensure the decoder learns to replace missing frames based on the time context. In addition, a memory subnetwork is placed after the encoder as a sparsity and diversity enforcing bottleneck, before the output is passed to the decoder. The memory subnetwork consists of a weight matrix $\mathbf{W} \in \mathbb{R}^{32\times32}$, bias vector $\mathbf{b} \in \mathbb{R}^{32}$, and memory values $\mathbf{V} \in \mathbb{R}^{32\times32}$. Its functionality is similar to a dot-product based attention model (Vaswani et al. 2017), such that the encoder output $\mathbf{X} \in \mathbb{R}^{T\times32}$ is used to form a query $\mathbf{Q} = \text{softmax}(\mathbf{XW} + \mathbf{b}) \in \mathbb{R}^{T\times32}$. This is then multiplied with the memory values to produce $\mathbf{QV}$, which is given as input to the decoder. When SS is trained while applying sparsity and diversity enforcing loss functions on $\mathbf{Q}$, the representation learned for $\mathbf{Q}$ is a one-hot encoded sequence. This in turn can easily be mapped to a sequence of integers $\mathbf{a} = [a_1, a_2, \ldots, a_T] \in \{1, 2, \ldots, 32\}^T$ by choosing the index $i \in \{1, 2, \ldots, 32\}$ of the greatest value of $\mathbf{Q}(t) \in \mathbb{R}^{32}$ at every time step $t \in \{1, 2, \ldots, T\}$. The elements of $\mathbf{a}$ are called acoustic units.

Note that the amount of unique acoustic units is 32 because this hyper-parameter value was chosen for the thesis experiments based on the results reported by Milde and Biemann (2019). Choosing a larger acoustic unit space is possible by increasing this hyper-parameter value, but this would create a larger model and require more computational resources.

All experiments that apply SS in this thesis use log-scaled Mel-spectra with 40 Mel-frequency bins as input, $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T\times40}$. Being an autoencoder, the output of SS is a reconstruction of its input and is naturally of little value in itself. However, Milde and Biemann (2019) noticed that the acoustic units might in some cases provide better ABX discriminability than MFCCs. Whether or not the unsupervised discovery of these acoustic units can produce a representation that correlates with actual phones is unclear. Therefore, this thesis explores the feasibility of using these acoustic units in SLI as a replacement of phoneme sequences produced by a PR model.

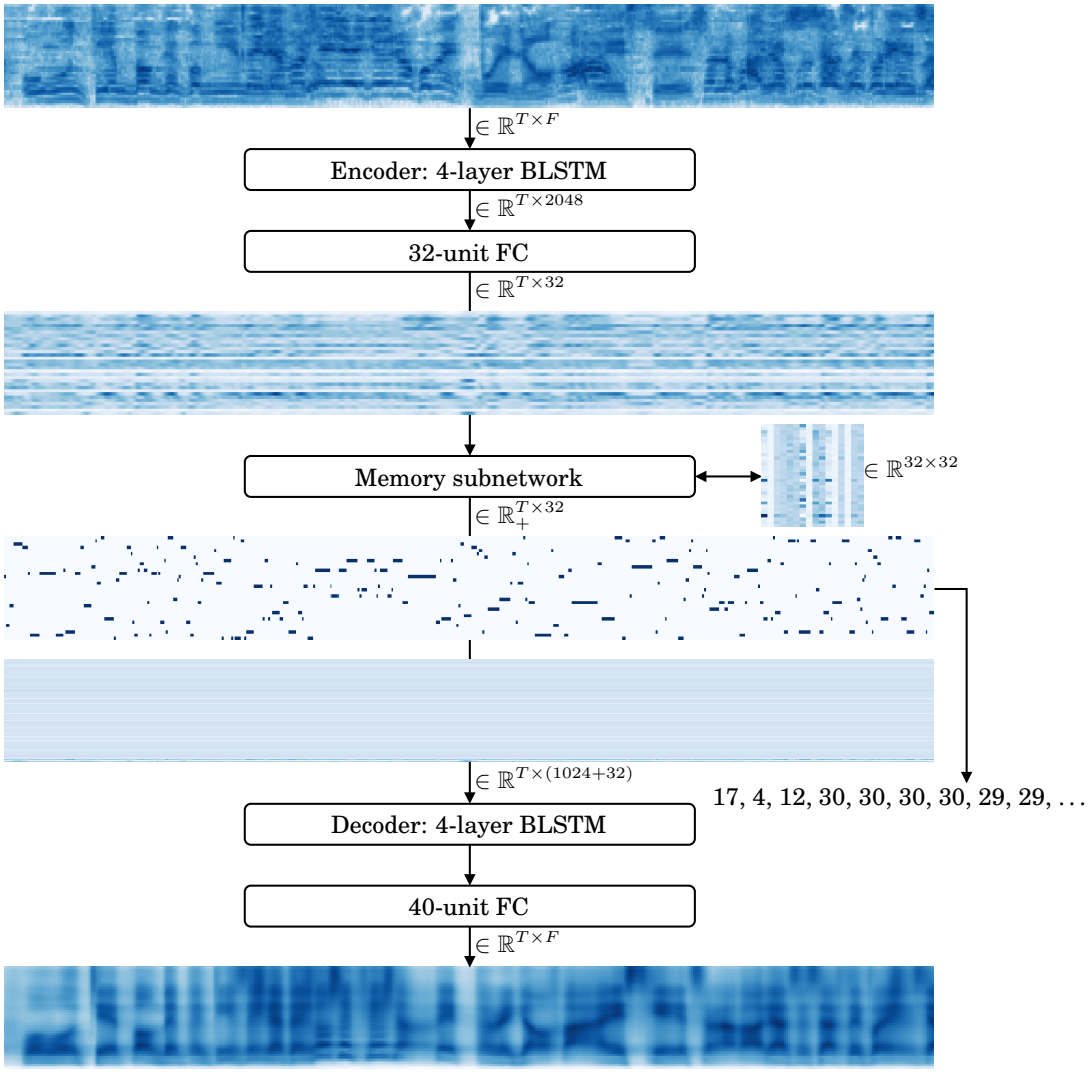
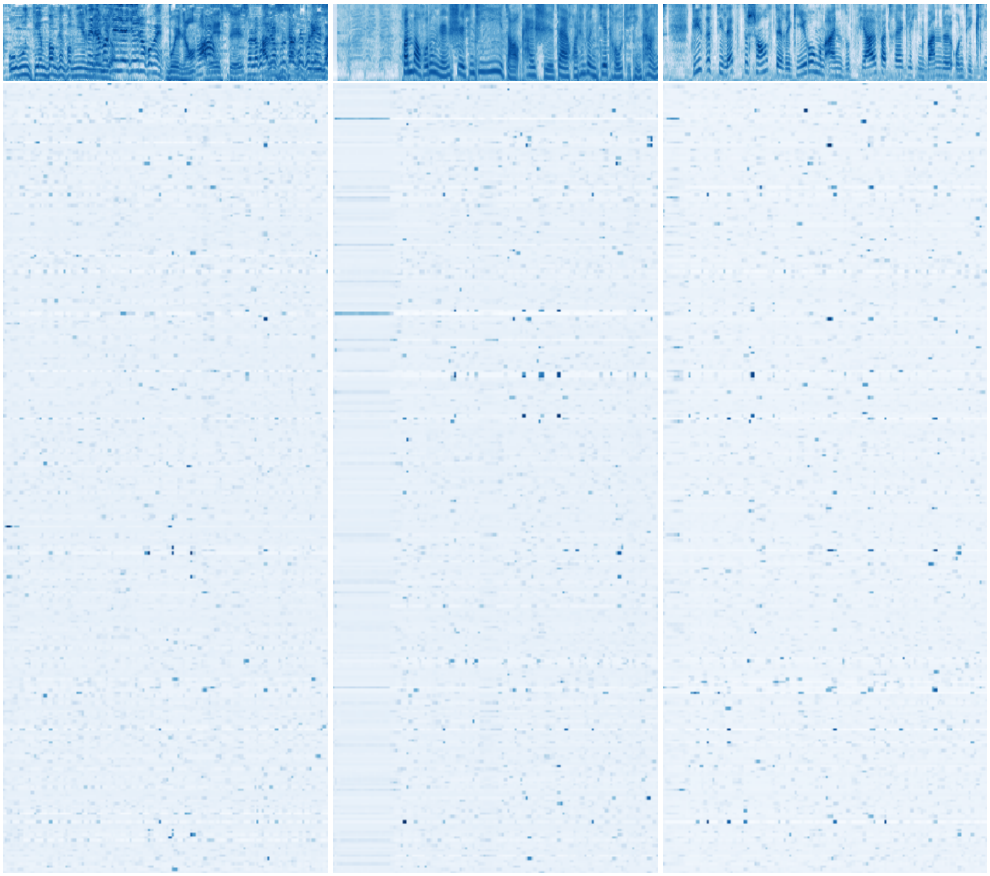


Figure 6.2. Simplified architecture overview of the SparseSpeech sequence-to-sequence autoencoder, given a Mel-spectrum input $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T \times F}$. This instance was warmed up for 30 epochs and trained for 6 epochs on the YTN training set (see Section 4.4). Temporal dropout is applied on the decoder input with probability 0.67 but it is omitted here for simplicity.

SSLM This model is similar to PRLM described earlier, but CTC-PR is replaced by SS. In other words, instead of using phone sequences as input to NB-TLI, the input consists of space-separated strings of acoustic units, represented by integers (see the integer sequence to the bottom-right in Figure 6.2). In addition, VAD is performed by first computing which acoustic unit most frequently is classified as non-speech and then dropping all those acoustic units. The acoustic unit sequence produced by SS is of length T and each acoustic unit maps one-to-one to each frame of the input $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T \times F}$. Therefore, if we compute VAD decisions $\mathbf{d} \in \{0, 1\}^T$ as seen in Figure 3.4a, those VAD decisions also map one-to-one to each acoustic unit. For example, if we label all VAD decisions $\mathbf{d}$ for every utterance by the corresponding acoustic unit produced by SS, then we can

compute the frequency of non-speech decisions $d_t = 0$ over all utterances for every acoustic unit. The acoustic units with largest amount of non-speech decisions is assumed to be a non-speech acoustic unit and all such units can be dropped from the acoustic unit decoding.



(a) **(b)** **(c)**

Figure 6.3. Bottleneck features (BNF) $\mathbf{X}_{\text{BNF}} \in \mathbb{R}_{+}^{123\times 300}$ extracted from CTC-PR. The BNFs are extracted from the last FC layer between the CNN front-end and GRU layers, with non-negative values due to the sigmoid activation function. Input consists of $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T\times 60}$ $(T > 123)$, shown above each BNF. All three signals are the same as in the example discussed in Section 3.2, i.e. (a) Figure 3.1a, (b) Figure 3.2a, (c) Figure 3.3a.

PTN with CTC-PR embeddings As discussed in Section 5.1, Tang et al. (2018b) proposed to use a hidden layer of a PR model to extract frame-level input for a LSTM-based SLI model. In this thesis, we implement PTN by using the output of a FC layer of CTC-PR as phoneme embeddings (see Figures 6.1 and 6.3), which are then used as input to a single-layer 1024-unit LSTM. The bottleneck layer from which features are extracted is the 300-unit, sigmoid-activated FC layer in CTC-PR between the CNN feature-extractor and the CTC-layers. The 300-dimensional bottleneck features (BNF) are partitioned into non-overlapping windows of 20 frames, e.g. $\mathbf{X} \in \mathbb{R}^{20\times 300}$, which are given as input to the LSTM model. Note that

the BNFs are phoneme embeddings, not language embeddings, since CTC-PR is a phoneme classifier, not a language classifier.

In addition to using a LSTM model on fixed length windows, another experiment is performed where the x-vector architecture (see Table 6.6) is used on the full BNF sequence. This model will be called **x-vector (CTC-PR)**.

PTN with SS embeddings (SS-PTN) The SS-PTN model is similar to PTN, except that CTC-PR is replaced by SS and phoneme BNFs are replaced with the SS memory context vector $\mathbf{X} \in \mathbb{R}^{T \times 32}$ (see Figure 6.2, bottom 32 rows of the decoder input). In other words, each utterance is encoded with SS and the output of the memory subnetwork, i.e. the context vector appended to the encoder output, is used as BNFs.

In addition to using a LSTM model, we perform another experiment with the x-vector architecture, similar to x-vector (CTC-PR), but with SS embeddings. This model will be called **x-vector (SS)**.

6.2 RNN based SLI

As discussed in Chapter 5, LSTMs and other RNN based models such as GRUs have been popular choices for SLI models. When applied to SLI, most RNN based models predict language scores on short time contexts, e.g. consisting of 10–30 spectral feature frames. The final language decision can be produced for example by reducing all window scores by averaging, as seen in Figure 6.4.

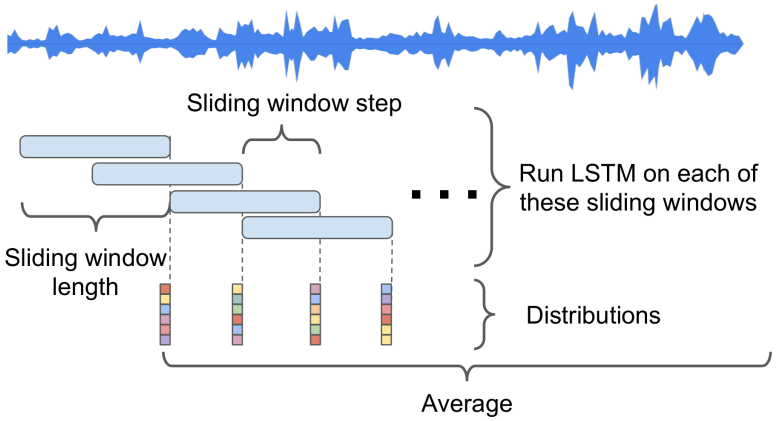


Figure 6.4. Partitioning audio data of arbitrary length into fixed length utterances with a sliding window (Wan et al. 2019, Fig. 2). The final language decision is based on the average language likelihood or score of all windows.

BGRU Two bidirectional GRU (BGRU) layers and two FC layers. Proposed by Mateju et al. (2018) for the SBS dataset, discussed in Section 4.3. Input consists of 30 time frames of log-scale Mel-spectra with 40 Mel bins,

e.g. $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{30\times 40}$, extracted with a Hann window of length 25 ms and step size of 10 ms. First, $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T\times 40}$ is extracted for each utterance. Then, each $\mathbf{X}_{\text{Mel}}$ is partitioned into non-overlapping windows of 30 frames each, such that the last window is dropped if it is shorter than 30 frames. Finally, the model is trained on all windows. When predicting results for an utterance of arbitrary length, the language scores of all its windows are averaged to form the final language score vector.

	Layer	Input shape	Output shape
1	Bidirectional GRU 512 units (concat)	T×F	T×1024
2	Bidirectional GRU 512 units (concat)	T×1024	1024
3	FC 1024 + ReLU	1024	1024
4	FC N + softmax	1024	N

Table 6.3. BGRU architecture for N languages, where the inputs are log-scale Mel-spectra of length T, containing F frequency bins. Note that the first BGRU layer returns full sequence, and the second layer returns only the last output of the sequence.

LSTM for PTN The PTN implementation in this thesis uses a single 1024-unit LSTM layer for classifying languages from BNF input. Its layers are listed in Table 6.4. Input consists of 20 frames of BNFs $\mathbf{X}_{\text{BNF}} \in \mathbb{R}^{20\times F}$. Final language scores for arbitrary length utterances are produced by averaging over the whole utterance, as in the BGRU model, except that PTN uses $T = 20$ instead of $T = 30$.

	Layer	Input shape	Output shape
1	LSTM 1024 units	T×F	T×1024
2	FC N + softmax	1024	N

Table 6.4. LSTM for PTN.

6.3 CNNs for variable length input

In this section we take a look at NN models that use spectral utterance representations of variable length as input to a CNN feature-extractor frontend. The output of the CNN is then reduced to a fixed length representation by performing a reduction over the time dimension.

CNN Four temporal convolution layers, followed by global average pooling and three FC layers. Proposed by Shon et al. (2018) for the MGB-3 dataset, discussed in Section 4.2. All layers use the ReLU activation function, except for the last FC layer, which uses softmax. Input consists of log-scaled Mel-spectra with 40 Mel-frequency bins, extracted with a Hann window of length 25 ms and offset 10 ms. All input are of arbitrary length,

and the result after the convolutions is pooled to a fixed size, before it is passed through the FC layers. This thesis refers to the model shown in Table 6.5 simply as CNN, although it should be noted that the term convolutional neural network is in general a very broad definition and does not uniquely identify this particular model outside the scope of this thesis.

	Layer	Input shape	Output shape
1	Conv1D 500×5×1 + ReLU	T×F	T×500
2	Conv1D 500×7×2 + ReLU	T×500	T/2×500
3	Conv1D 500×1×1 + ReLU	T/2×500	T/2×500
4	Conv1D 3000×1×1 + ReLU	T/2×500	T/2×3000
5	Reduce mean over time axis	T/2×3000	3000
6	FC 1500 + ReLU	3000	1500
7	FC 600 + ReLU	1500	600
8	FC N + softmax	600	N

Table 6.5. CNN architecture for N languages, where the inputs are log-scaled Mel-spectra of length T, containing F frequency bins.

x-vector Snyder et al. (2018b) proposed a TDNN based architecture for speaker recognition, which was adapted for SLI by Snyder et al. (2018a). Two different implementations are used in this thesis. The first one is the AP19-OLR competition baseline model by Tang et al. (2019), which they based on the original x-vector architecture implemented using the Kaldi toolkit by Povey et al. (2011). This model is called **Kaldi x-vector**. The second implementation was completed during this thesis (see Table 6.6) using temporal convolution layers instead of TDNNs and is called **x-vector**. Experiments are performed using both implementations.

	Layer	Input shape	Output shape
1	Conv1D 512×5×1 + BN + ReLU	T×F	T×512
2	Conv1D 512×3×2 + BN + ReLU	T×512	T/2×512
3	Conv1D 512×3×3 + BN + ReLU	T/2×512	T/6×512
4	Conv1D 512×1×1 + BN + ReLU	T/6×512	T/6×512
5	Conv1D 1500×1×1 + BN + ReLU	T/6×512	T/6×1500
6	Reduce mean and stddev. over time axis and concatenate	T/6×1500	3000
7	FC 512 + BN + ReLU	3000	512
8	FC 512 + BN + ReLU	512	512
9	FC N + softmax	512	N

Table 6.6. x-vector architecture for N languages, where the inputs are log-scale Mel-spectra of length T, containing F frequency bins. The 512-dimensional output of FC layer 7, before batch normalization and ReLU activation, are called "x-vectors" by Snyder et al. (2018a).

6.4 CNNs with RNNs or time-attention

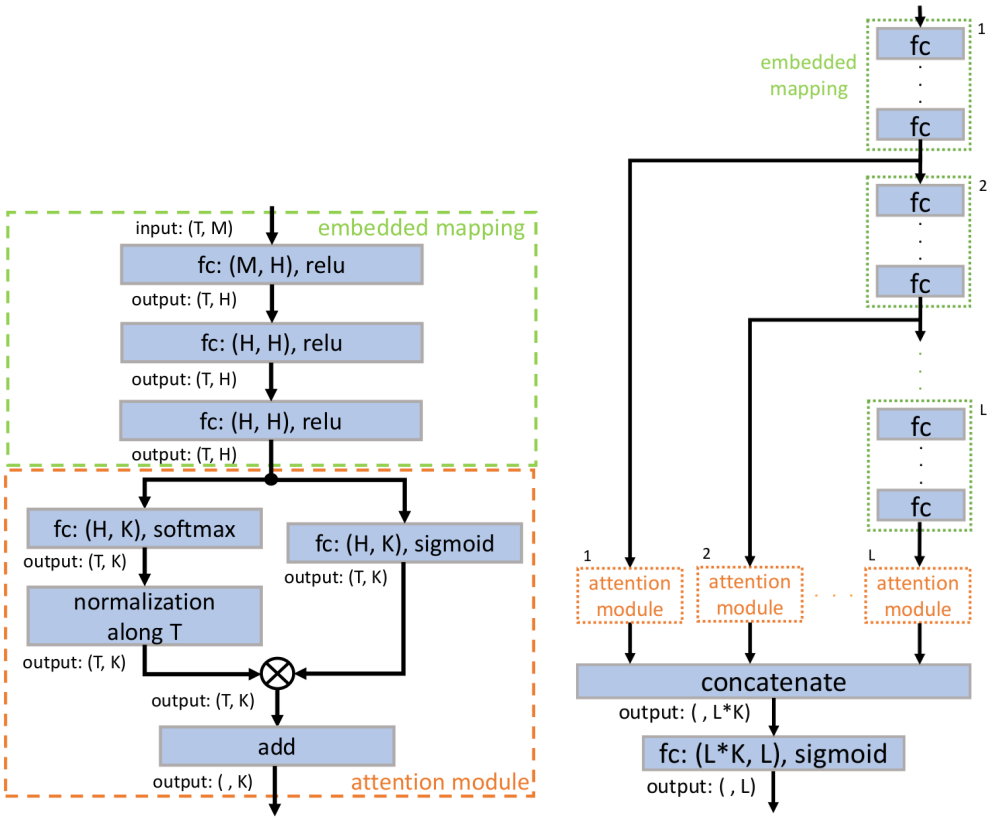
In this final section we discuss the CRNN model, which combines a CNN front-end and uses a BLSTM layer for sequence classification, and a model which combines the x-vector model with multi-level time attention.

CRNN Five 2-dimensional convolutional layers with batch normalization and max pooling, followed by one BLSTM layer and a FC layer (see Table 6.7). Proposed by Bartz et al. (2017) for the YTN dataset, discussed in Section 4.4. The outputs from each unidirectional LSTM in the BLSTM are concatenated, the convolutional layers use the ReLU activation function, and the last FC layer uses softmax. Input consists of monochrome decibel-scale spectrogram images $\mathbf{X}_{\text{dB}} \in \mathbb{R}^{T \times 129}$ with 129 frequency bins of frequencies up to 5 kHz, extracted with a Hann window at a resolution of 50 pixels per second, corresponding to a window length of 25 ms and step size of 20 ms. E.g. using 10 second utterances at sample rate 16 kHz, $T = 500$ and $F = 129$, with values mapped onto the dynamic dB range from -120 up to 0, with the maximum value of the power spectrogram as the upper reference point. As an interesting side note, Bartz et al. (2017) noticed that the Inception-v3 CNN architecture (Szegedy et al. 2016) performs significantly better than their CRNN when trained and evaluated on highly noisy data, and argued that the deeper structure of this CNN is beneficial to learn languages, as opposed to noise. However, the Inception architecture is not used in this thesis and instead we rely only the simpler, CRNN model.

	Layer	Input shape	Output shape
1	Expand image channel	T×F	T×F×1
2	Time dim. to image cols	T×F×1	F×T×1
3	Conv2D 16×7×1 + ReLU + BN	F×T×1	F×T×16
4	Max pooling 2×2	F×T×16	F/2×T/2×16
5	Conv2D 32×5×1 + ReLU + BN	F/2×T/2×16	F/2×T/2×32
6	Max pooling 2×2	F/2×T/2×32	F/4×T/4×32
7	Conv2D 64×3×1 + ReLU + BN	F/4×T/4×32	F/4×T/4×64
8	Max pooling 2×2	F/4×T/4×64	F/8×T/8×64
9	Conv2D 128×3×1 + ReLU + BN	F/8×T/8×64	F/8×T/8×128
10	Max pooling 2×2	F/8×T/8×128	F/16×T/16×128
11	Conv2D 256×3×1 + ReLU + BN	F/16×T/16×128	F/16×T/16×256
12	Max pooling 2×2	F/16×T/16×256	F/32×T/32×256
13	Permute timesteps for RNN	F/32×T/32×256	T/32×F/32×256
14	Flatten image channels	T/32×F/32×256	T/32×(256·F/32)
15	BLSTM 256 units (concat)	T/32×(256·F/32)	512
16	FC N + softmax	512	N

Table 6.7. CRNN architecture. L2 normalization is applied to the weight matrices of the convolutional layers with a decay rate of 0.001. Note that the input spectrum is converted to a monochrome image inside the model, eliminating the need to perform explicit image conversions before training.

x-vector with multi-level time attention Yu et al. (2018) proposed a multi-level time-attention (MLA) mechanism for general audio classification of deep embeddings (see Figure 6.5). In this thesis, the x-vector architecture seen in Table 6.6 is augmented by interleaving time-attention modules in between the segmentation layers of the x-vector TDNN. In other words, layers 7 to 9 in Table 6.6 are replaced by the MLA model from Figure 6.5b such that $T = 1$. In addition, experiments are performed where the x-vector stats pooling layer is removed, and time-pooling is performed within the time-attention mechanism. In other words, layers 6 to 9 in Table 6.6 is replaced by the MLA model. In both cases, each "embedded mapping" block consist of a single FC layer, equal to layer 8 in Table 6.6.



(a) Yu et al. (2018, Fig. 1). **(b)** Yu et al. (2018, Fig. 2).

Figure 6.5. Multilevel-attention model for classifying input $\mathbf{X} \in \mathbb{R}^{T \times M}$ into K classes using L attention modules. (a) Single-level attention, consisting of a block of FC layers (embedded mapping) and a time-attention module. (b) Multi-level attention with time-attention modules interleaved between FC blocks. The output of each attention module are concatenated.

7. Experiments and results

This chapter contains results for all SLI experiments performed during this thesis. All models discussed in Chapter 6 were trained on all five datasets discussed in Chapter 4. We begin by summarizing all experiment settings shared by all models, such as common hyper-parameters. Then, we discuss the values of evaluation metrics computed on each test set with all trained models. Finally, we compare the performance of different models by discussing all results.

7.1 Experiment settings and preprocessing

Validation and early stopping All models were trained until stopping on the condition that the value of multiclass cross-entropy (Equation 3.1), evaluated on the test set, had not improved within the past 10 epochs. Instead of using a held-out validation set, partitioned from the training set, the test set from each dataset was used for model validation. This is an important factor that should be considered when analyzing the results. Since there was access to the ground truth labels for each test set, it was possible to perform early stopping depending on the performance on the test set, instead of a separate development set. Therefore, when comparing the results discussed in this chapter to closed task SLI competition results such as AP19-OLR, one might expect that results in this thesis would have been slightly worse compared to other participants, since ground truth labels are revealed to all participants only after every participant has submitted their predictions on the test set. On the other hand, Shon et al. (2018) used 90% of the development set of MGB-3 for training in order to learn the channel conditions of the test set, greatly improving model performance on the test set. Similarly, the training set of AP19-OLR contains development and test data of previous AP-OLR challenges, which might contain channel information of the evaluation set of AP19-OLR. It is also worth noting that only minimal hyper-parameter tuning was performed, e.g. only to ensure convergence of the optimization methods. In

other words, after a specific configuration of hyper-parameter values that allows each model to converge to some minimum, no additional tuning based on the test set performance was performed.

Voice activity detection Whenever VAD is mentioned, it refers to the *WebRTC* (2020) VAD implementation discussed in Section 3.5. Each VAD value between 0 and 3 reported in this chapter equals the WebRTC VAD aggressiveness values, as defined in Table 3.2. Whenever "None" is specified, it denotes that VAD was not applied in any form. When VAD is applied to some input $\mathbf{X} \in \mathbb{R}^{T \times F}$, the VAD decisions $\mathbf{d} \in \{0, 1\}^T$ are first computed window-wise on the original signal $\mathbf{s} \in \mathbb{R}^S$, using the same window length and offset as in the STFT computation that produces $\mathbf{X}$ (see Figure 3.4a). After $\mathbf{X}$ has been computed from $\mathbf{s}$, every frame x_t that WebRTC VAD marked as not speech ($d_t = 0$) will be deleted, regardless of the amount of consecutive speech or non-speech frames.

Feature normalization Before training, all features are normalized in various ways. Spectral features are normalized within a 1 to 3 second window by applying channel-normalization on each channel, using two different approaches. The first one is mean normalization by centering all channels within the normalization window to zero mean. This is denoted by "m" (mean) in the tables. The second normalization approach is mean-variance normalization, where all channels within the window are normalized to have zero mean and unit variance. This is denoted by "mv" (mean-variance) in the tables. Whenever an utterance is shorter than the normalization window, the normalization is applied over the utterance. Whenever the normalization window overshoots an utterance at the beginning or the end, it is padded by reflecting the values from the non-overshooting part of the window. For example, assume we are given an utterance representation $\mathbf{X} = [x_1, x_2, x_3, x_4, \ldots, x_T] \in \mathbb{R}^{T \times F}$ and choose a window length of 5. Let $\mathbf{X}' \in \mathbb{R}^{T \times F}$ denote $\mathbf{X}$ after normalization. Then, each element of $\mathbf{X}'$ are computed from each 5-element window into $\mathbf{X}$ as follows:

$$
\begin{aligned}
x'_1 &= \text{normalize}([x_3, x_2, x_1, x_2, x_3]) \\
x'_2 &= \text{normalize}([x_2, x_1, x_2, x_3, x_4]) \\
x'_3 &= \text{normalize}([x_1, x_2, x_3, x_4, x_5]) \\
x'_4 &= \text{normalize}([x_2, x_3, x_4, x_5, x_6]) \\
&\vdots \\
x'_{T-1} &= \text{normalize}([x_{T-3}, x_{T-2}, x_{T-1}, x_T, x_{T-1}]) \\
x'_T &= \text{normalize}([x_{T-2}, x_{T-1}, x_T, x_{T-1}, x_{T-2}])
\end{aligned}
$$

In contrast to how we normalize spectral features, BNFs are only normalized by centering the means to zero, i.e. standardization is not performed. However, three different normalization axes are experimented with and

these are defined in Table 7.1.

Dimension	Mean centering
Time	Each channel over the time axis T
Features	Each time frame over the channel axis F
Both	All values over the whole sample

Table 7.1. Three different normalization approaches for normalizing BNFs.

Score evaluation After the best model weights are discovered by early stopping, each model will use these weights to compute N language scores for every utterance in the test set. Then, independent from the model training pipeline, these language scores are used to compute the C_{avg} (Equation 3.3) and $F_{1\text{-avg}}$ (Equation 3.4) score metrics. C_{avg} values are the minimum C_{avg} value from 20 different C_{avg} values, computed using 20 different decision thresholds, generated evenly spaced from the smallest ($\theta_{\min}$) to the largest ($\theta_{\max}$) language scores produced by the model (see Section 3.3). $F_{1\text{-avg}}$ values are computed by choosing the predicted class for a test utterance according to the maximum predicted language score.

7.2 Reproducing reference results

This section contains all results for models that have existing results in the literature on a reference dataset from Chapter 4. CRNN on YTN is not listed here since due to the size of the dataset it was decided not to perform a grid search over all VAD and normalization configurations. Based on the results from all other experiments, mean-normalization was chosen and no VAD was performed. Its results is listed in Section 7.6.

CNN on MGB-3 The CNN model (Section 6.3) was trained on the MGB-3 dataset (Section 4.2) using 5 different VAD configurations and 2 different normalization configurations. The best model was used to predict language scores on the MGB-3 test set and evaluation metrics were computed from the predictions. C_{avg} values are reported in Table 7.2a and $F_{1\text{-avg}}$ scores in Table 7.2b.

VAD	m	mv
None	0.205	**0.194**
0	0.204	0.217
1	**0.197**	0.210
2	0.205	0.222
3	0.223	0.223
Avg	**0.207**	0.213

(a) C_{avg}, less is better.

VAD	m	mv
None	0.630	**0.631**
0	0.631	0.589
1	**0.643**	0.597
2	0.631	0.588
3	0.588	0.595
Avg	**0.624**	0.600

(b) $F_{1\text{-avg}}$, more is better.

Table 7.2. Evaluation results with CNN on the MGB-3 test set. VAD was either not used or it was applied using 4 different aggressiveness settings. Normalization of spectral bins over time axis: zero mean (m), zero mean and unit variance (mv).

We see that the best configuration measured by $C_{avg} = 0.194$ is mean-variance normalized features (mv) without VAD. On the other hand, measured by $F_{1\text{-avg}} = 0.643$, the best configuration is mean-normalized features (m) with VAD level 1. However, when comparing results averaged over all VAD configurations, mean normalized features (m) are better on average at $C_{avg} = 0.207$ and $F_{1\text{-avg}} = 0.624$, compared to mean-variance normalized features. Compared to the $C_{avg} = 0.199$ reported by Shon et al. (2018) before performing speed and volume augmentation, our best result is 0.5 percentage points lower (better) than the reference result. They managed to reduce their result to $C_{avg} = 0.176$ by augmenting the dataset, which was not performed in this thesis.

x-vector on AP19-OLR The x-vector model (Section 6.3) was trained on the AP19-OLR dataset (Section 4.5) using 5 different VAD configurations and 2 different normalization configurations. The best model was used to predict language scores on the AP19-OLR short utterance test set (task 1) and evaluation metrics were computed from the predictions. C_{avg} values are reported in Table 7.3a and $F_{1\text{-avg}}$ scores in Table 7.3b.

VAD	m	mv
None	**0.142**	0.166
0	0.162	**0.152**
1	0.151	0.163
2	0.153	0.175
3	0.175	0.186
Avg	**0.157**	0.168

(a) C_{avg}, less is better.

VAD	m	mv
None	0.617	0.579
0	0.599	**0.615**
1	**0.624**	0.591
2	0.621	0.588
3	0.595	0.577
Avg	**0.611**	0.590

(b) $F_{1\text{-avg}}$, more is better.

Table 7.3. Evaluation results with x-vector on the AP19-OLR short utterance test set. VAD was either not used or it was applied using 4 different aggressiveness settings. Normalization of spectral bins over time axis: zero mean (m), zero mean and unit variance (mv).

The best configuration measured by $C_{avg} = 0.142$ is achieved with mean-

normalized features without VAD, while the best $F_{1\text{-avg}} = 0.624$ is achieved with mean-normalized features with VAD level 1, i.e. low-bitrate setting. However, on average, mean normalized features give better results. Compared to the baseline $C_{avg} = 0.126$ reported by Tang et al. (2019), our best result is 1.6 percentage points higher (worse) than the reference result. Note that the best result for the AP19-OLR short utterance task is reported to be $C_{avg} = 0.046$, which is 9.6 percentage points lower (better) than our best result.

BGRU on SBS The BGRU model (Section 6.2) was trained on the SBS dataset (Section 4.3) using 5 different VAD configurations and 2 different normalization configurations. The best model was used to predict language scores on the SBS test set and evaluation metrics were computed from the predictions. C_{avg} values are reported in Table 7.4a and $F_{1\text{-avg}}$ scores in Table 7.4b.

VAD	m	mv
None	**0.028**	**0.032**
0	0.035	0.189
1	0.035	0.033
2	0.033	0.034
3	0.039	0.035
Avg	**0.034**	0.065

(a) C_{avg}, less is better.

VAD	m	mv
None	**0.939**	**0.935**
0	0.927	0.558
1	0.923	0.930
2	0.931	0.926
3	0.910	0.919
Avg	**0.926**	0.854

(b) $F_{1\text{-avg}}$, more is better.

Table 7.4. Evaluation results with BGRU on the SBS test set. VAD was either not used or it was applied using 4 different aggressiveness settings. Normalization of spectral bins over time axis: zero mean (m), zero mean and unit variance (mv).

We can see that VAD is not beneficial in this experiment, since both C_{avg} and $F_{1\text{-avg}}$ reach their best values when VAD is not used. The best overall configuration is mean-normalization without VAD, with $C_{avg} = 0.028$ and $F_{1\text{-avg}} = 0.939$. On average, mean normalization is better than mean-variance normalization. Compared to the $C_{avg} = 0.013$ reported by Mateju et al. (2018), our best result is 1.5 percentage points higher (worse) than the reference result.

7.3 TLI based experiments

This section contains results from PRLM and SSLM, which decode input utterances into textual representations from which the language is detected using NB-TLI.

PRLM The PRLM model (Section 6.1) was trained on all reference datasets discussed in Chapter 4. All utterances from both the training and test

sets of all datasets were decoded into phoneme sequences with CTC-PR, outputting space separated strings of individual phoneme tokens, one line of phoneme tokens for each utterance. Then, NB-TLI was trained separately for each dataset, on all phoneme sequences in the training set of each dataset. Finally, NB-TLI was used to predict language scores for all phoneme sequences in the test set of each dataset. C_{avg} values are reported in Table 7.5a and $F_{1\text{-}avg}$ scores in Table 7.5b.

Dataset	C_{avg}
OGI-11L	0.385
SBS	0.387
MGB-3	0.455
AP19-OLR	0.406
YTN-Aalto2019	0.362

(a) Less is better.

Dataset	$F_{1\text{-}avg}$
OGI-11L	0.515
SBS	0.472
MGB-3	0.402
AP19-OLR	0.245
YTN-Aalto2019	0.660

(b) More is better.

Table 7.5. Evaluation results with PRLM on all test sets.

We can see that results are overall poor, with C_{avg} values being higher than the $F_{1\text{-}avg}$ scores for predictions on the AP19-OLR and MGB-3 test sets. Compared to the results of PTN, which used CTC-PR phoneme embeddings as input, these results might suggest that a significant amount of language discriminability is lost when the phoneme embeddings are converted into phoneme tokens by CTC decoding in the CTC-PR model.

SSLM The SSLM model (Section 6.1) was trained on all reference datasets discussed in Chapter 4. However, the SS autoencoder was unable to learn a reasonable acoustic unit representation for OGI-11L and instead generated constant output of a single acoustic unit, “6”. Therefore, no results are reported on OGI-11L with SSLM. For all remaining 4 datasets, utterances were decoded to acoustic unit sequences with SS and classified with NB-TLI. All utterances from both the training and test sets of all datasets were decoded into acoustic unit sequences with SS, outputting space separated strings of individual acoustic units (integers from 1 to 32), one line for each utterance. Then, NB-TLI was trained separately for each dataset, on all acoustic unit sequences in the training set of each dataset. In addition, VAD was performed on the acoustic unit sequences by dropping all acoustic units that most frequently were classified as non-speech by *WebRTC* (2020) VAD. In this experiment, VAD levels denote how many most likely non-speech acoustic units were dropped (see Section 6.1 for a description on the VAD approach for acoustic units). Finally, NB-TLI was used to predict language scores for all acoustic unit sequences in the test set of each dataset. C_{avg} values are reported in Table 7.6a and $F_{1\text{-}avg}$ scores in Table 7.6b.

VAD	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	**0.395**	0.482	0.399	0.449
1	0.406	0.483	0.397	0.445
2	0.420	**0.481**	**0.388**	0.481
3	0.429	0.482	0.396	**0.443**

(a) C_{avg}, less is better.

VAD	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	**0.313**	0.346	0.569	0.516
1	0.288	0.354	**0.570**	**0.529**
2	0.250	0.351	0.548	0.485
3	0.239	**0.366**	0.530	0.519

(b) $F_{1\text{-avg}}$, more is better.

Table 7.6. Evaluation results on all test sets with SSLM by dropping different amount of most likely non-speech acoustic units. I.e. VAD = 1 denotes the most likely non-speech unit was dropped, 2 denotes two most likely and 3 denotes 3 most likely non-speech units were dropped.

We can see that all results are worse than with PRLM. This is similar to the x-vector BNF results, where we noted that using SS BNFs leads to worse results compared to using CTC-PR BNFs.

7.4 BNF based experiments

This section contains results from all models which use BNFs, i.e. phoneme embeddings from CTC-PR and memory context vectors from SS.

x-vector with BNFs In order to compare x-vector performance using different features in addition to spectra, BNFs ($\mathbf{X}_{\text{BNF}} \in \mathbb{R}^{T \times F}$) were extracted from CTC-PR (Section 6.1, PTN) and SS (Section 6.1, SS-PTN) for all datasets described in Chapter 4. All utterances from both the training and test sets of all datasets were decoded into BNF vectors with CTC-PR and SS, one CTC-PR BNF vector and one SS BNF vector for each utterance. Different normalization approaches were compared by mean-normalizing over the time axis T and channel axis F, as described in Table 7.1. Then, x-vector models were trained with all BNF vectors in the training set. Using the best models, language scores were predicted from BNF vectors in the test set. C_{avg} values for x-vector with CTC-PR BNFs are reported in Table 7.7a and $F_{1\text{-avg}}$ scores in Table 7.7b. C_{avg} values for x-vector with SS BNFs are reported in Table 7.8a and $F_{1\text{-avg}}$ scores in Table 7.8b. Note that the SS autoencoder was unable to learn a usable acoustic unit representation for OGI-11L and no results are therefore available for x-vector with SS BNFs.

Norm.	OGI-11L	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	0.200	0.246	0.269	**0.040**	0.089
Time	0.243	0.249	0.282	0.045	0.068
Features	**0.187**	**0.228**	**0.261**	0.044	0.076
Both	0.202	0.230	0.274	0.041	**0.050**
Avg	0.208	0.238	0.271	0.043	0.071

(a) C_{avg}, less is better.

Norm.	OGI-11L	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	0.561	0.382	0.485	0.879	0.829
Time	0.458	0.384	0.473	0.865	0.841
Features	**0.580**	**0.421**	0.522	0.875	0.819
Both	0.513	0.416	**0.528**	**0.885**	**0.919**
Avg	0.528	0.401	0.502	0.876	0.852

(b) $F_{1\text{-avg}}$, more is better.

Table 7.7. Evaluation results with x-vector, using CTC-PR phoneme embeddings, on all test sets, mean-normalized over different axes.

We can see that the best results for OGI-11L ($C_{avg} = 0.187$), AP19-OLR ($C_{avg} = 0.228$), and MGB-3 ($C_{avg} = 0.261$) are achieved by normalizing each time step over the channels of the CTC-PR phoneme embedding features. Best results for SBS ($C_{avg} = 0.04$) is achieved by performing no normalization, while best results for YTN-Aalto2019 ($C_{avg} = 0.05$) is achieved by normalizing over the whole sample. We can see that none of these results outperform the reference models discussed in Section 7.2.

Interestingly, it appears that phoneme BNFs extracted from CTC-PR still provide reasonable language discriminability, even though CTC-PR was not trained (see Section 6.1) on any of the languages being identified, e.g. AP19-OLR, MGB-3, and SBS. For example, $C_{avg} = 0.228$ for AP19-OLR is only 10 percentage points higher (worse) than the baseline, $C_{avg} = 0.261$ for MGB-3 is only 7 percentage points higher (worse) than the baseline, and $C_{avg} = 0.04$ for SBS is only 3 percentage points higher (worse) than the baseline.

Norm.	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	**0.199**	0.307	0.088	0.196
Time	0.226	0.314	0.084	**0.155**
Features	0.202	0.312	0.086	0.188
Both	0.205	**0.304**	**0.081**	0.164
Avg	0.208	0.309	0.085	0.176

(a) C_{avg}, less is better.

Norm.	AP19-OLR	MGB-3	SBS	YTN-Aalto2019
None	**0.478**	**0.463**	0.765	0.675
Time	0.403	0.445	0.773	0.687
Features	0.473	0.437	0.773	**0.698**
Both	0.478	0.443	**0.778**	0.660
Avg	0.458	0.447	0.772	0.680

(b) $F_{1\text{-avg}}$, more is better.

Table 7.8. Evaluation results with x-vector, using SS memory context vector BNFs, on all test sets, mean-normalized over different axes. The SS autoencoder was unable to learn a usable acoustic unit representation for OGI-11L.

We can see that using the SS memory context vectors as phoneme embedding replacements do not provide better language discriminability compared to CTC-PR phoneme embeddings. Most results, both by C_{avg} and $F_{1\text{-avg}}$, with SS BNFs are worse than the best results with CTC-PR BNFs, regardless of which normalization configuration is used. However, for AP19-OLR, C_{avg} is on average 3 percentage points better and $F_{1\text{-avg}}$ is on average 6 percentage points better than with CTC-PR BNFs.

PTN The PTN model (Section 6.1) was trained on all reference datasets discussed in Chapter 4. All utterances from both the training and test sets of all datasets were decoded into BNFs with CTC-PR, one $\mathbf{X}_{\text{BNF}} \in \mathbb{R}^{T\times 300}$ for each utterance. All vectors were then partitioned over the T axis into non-overlapping sequences of 20 frames each and a single, 1024-unit LSTM model was trained on all 20-frame sequences separately for each training set of each dataset. The best performing model was used to predict language scores for all 20-frame sequences of each test set by averaging over all 20-frame sequences within each test set utterance. C_{avg} values are reported in Table 7.9a and $F_{1\text{-avg}}$ scores in Table 7.9b.

Dataset	C_{avg}
OGI-11L	0.305
MGB3	0.307
SBS	0.065
AP19-OLR	0.203
YTN-Aalto2019	0.101

(a) C_{avg}, less is better.

Dataset	$F_{1\text{-avg}}$
OGI-11L	0.298
MGB3	0.428
SBS	0.843
AP19-OLR	0.490
YTN-Aalto2019	0.812

(b) $F_{1\text{-avg}}$, more is better.

Table 7.9. Evaluation results with PTN on the all test sets.

SS-PTN The SS-PTN model (Section 6.1) was trained on all reference datasets discussed in Chapter 4. However, as already discussed, the SS autoencoder was unable to learn a reasonable acoustic unit representation for OGI-11L and no results are reported on OGI-11L. For all 4 remaining datasets, all utterances from both the training and test sets were decoded into SS BNFs, one $\mathbf{X}_{\text{BNF}} \in \mathbb{R}^{T\times 32}$ for each utterance. The training and evaluation using these SS BNFs was performed exactly as with PTN, described above. C_{avg} values are reported in Table 7.10a and $F_{1\text{-avg}}$ scores in Table 7.10b.

Dataset	C_{avg}
MGB3	0.419
SBS	0.164
AP19-OLR	0.235
YTN-Aalto2019	0.189

(a) C_{avg}, less is better.

Dataset	$F_{1\text{-avg}}$
MGB3	0.198
SBS	0.613
AP19-OLR	0.371
YTN-Aalto2019	0.674

(b) $F_{1\text{-avg}}$, more is better.

Table 7.10. Evaluation results with SS-PTN on all test sets.

We can see that the results or worse than with PTN. This is the third example of CTC-PR based features outperforming SS based features, both as BNFs and as a textual representation.

7.5 Other approaches

MLA x-vector on AP19-OLR The MLA x-vector model (Section 6.4) was trained on the AP19-OLR dataset (Section 4.5) with 2 to 5 levels (L) of added attention modules added. In addition, the attention modules were integrated to the x-vector model using 2 different configurations, as well as using a reference configuration by increasing the size of the x-vector model without using attention. The configurations are made as follows:

“With attention”: the global statistics pooling layer (x-vector layer 6) is

used normally for reducing the time axis, with attention modules being added after it as additional FC layers.

"Pooled attention": the global statistics pooling layer is replaced with time attention, such that the time axis is reduced by time attention as proposed by Yu et al. (2018).

"Without attention": MLA is not used, but instead L additional FC layers (x-vector layer 8) are added after the global statistics pooling layer to the standard x-vector architecture for comparing whether performance improvements are from MLA or simply increased x-vector model size.

C_{avg} values are reported in Table 7.11a and $F_{1\text{-avg}}$ scores in Table 7.11b.

L	with att.	pooled att.	without att.
2	0.152	0.213	0.145
3	0.152	0.205	**0.137**
4	**0.139**	0.229	0.159
5	0.143	**0.177**	0.139
Avg	0.146	0.206	**0.145**

(a) C_{avg}, less is better.

L	with att.	pooled att.	without att.
2	0.631	0.477	0.618
3	0.627	0.492	**0.635**
4	**0.651**	0.447	0.578
5	0.628	**0.551**	0.625
Avg	**0.634**	0.492	0.614

(b) $F_{1\text{-avg}}$, more is better.

Table 7.11. Evaluation results with MLA x-vector on the AP19-OLR short utterance test set. 4 different attention levels (L) were used, with 3 different ways of introducing the levels into the x-vector architecture.

Using 4 attention levels ($L = 4$) yields $C_{avg} = 0.139$, which is slightly better than $C_{avg} = 0.142$ achieved without attention (Table 7.3a). However, by adding 3 regular FC layers to the original x-vector model, without attention modules, we can see this extension alone reduces C_{avg} to 0.137 (column "without att."). We can also conclude that replacing the statistics pooling layer of the x-vector model with attention based pooling, as described in Section 6.4, is not beneficial (column "pooled att."). However, using 5 attention levels is clearly better than using less levels.

7.6 Comparing all results

The C_{avg} values of all evaluations mentioned above have been collected into Table 7.12a. Similarly, all $F_{1\text{-avg}}$ values have been collected into Table 7.12b. There are several interesting details that we will now analyze.

Model	AP19-OLR	MGB-3	OGI-11L	SBS	YTN-Aalto2019
Kaldi x-vector	**0.120**	0.350	**0.150**	0.030	0.060
x-vector	0.142	0.212	0.196	0.026	**0.037**
x-vector (CTC-PR)	0.228	0.261	0.187	0.040	0.050
x-vector (SS)	0.199	0.304		0.081	0.155
CNN	0.164	**0.197**	0.218	**0.020**	0.086
PRLM	0.406	0.455	0.385	0.387	0.362
SSLM	0.395	0.482		0.397	0.445
PTN	0.203	0.307	0.305	0.065	0.101
SS-PTN	0.235	0.419		0.164	0.189
BGRU	0.161	0.363	0.368	0.028	0.087
CRNN	0.151	0.342	0.339	0.056	0.137

(a) C_{avg}, less is better.

Model	AP19-OLR	MGB-3	OGI-11L	SBS	YTN-Aalto2019
x-vector	**0.617**	0.616	0.556	0.942	**0.948**
x-vector (CTC-PR)	0.421	0.528	**0.580**	0.885	0.919
x-vector (SS)	0.478	0.463		0.778	0.698
CNN	0.579	**0.643**	0.491	**0.946**	0.827
PRLM	0.245	0.402	0.515	0.472	0.660
SSLM	0.313	0.366		0.570	0.529
PTN	0.490	0.428	0.298	0.843	0.812
SS-PTN	0.371	0.198		0.613	0.674
BGRU	0.592	0.332	0.098	0.939	0.844
CRNN	0.582	0.418	0.236	0.857	0.719

(b) $F_{1\text{-avg}}$, more is better. The Kaldi x-vector baseline script did not implement evaluation of $F_{1\text{-avg}}$ scores.

Table 7.12. Evaluation results with all models, using best configurations, on all datasets. The SS autoencoder failed to learn an acoustic unit representation for OGI-11L and results for all models that use SS are therefore missing for OGI-11L.

CNN-based models We can clearly see that CNN-based classification of log-scale Mel-spectra performs best compared to all other approaches. The set of models which perform best on at least one dataset consists of CNN, x-vector, and Kaldi x-vector. The only exception is the $F_{1\text{-avg}}$ score from x-vector (CTC-PR), which uses phoneme embeddings from CTC-PR instead of a spectral representation as input. On the other hand, Kaldi x-vector did not provide $F_{1\text{-avg}}$ scores, which means we cannot say for certain if it would outperform x-vector (CTC-PR) also by $F_{1\text{-avg}}$.

CRNN on YTN We were unable to replicate the results reported by Bartz et al. (2017) with the CRNN model on the YTN-Aalto2019 dataset. They reported an average F_1 score of 0.91 on their YTN instance, with the lowest individual F_1 score for English at 0.88. We can see from Table 7.12b that our $F_{1\text{-avg}}$ score with CRNN on the YTN-Aalto2019 dataset is only 0.72. However, since our YTN instance does not contain the same acoustic data as used by Bartz et al. (2017), we cannot be sure if these results are comparable.

CTC-PR outperforms SS In Chapter 1, we asked if it is possible to replace supervised phonemes of the CTC-PR model by unsupervised acoustic units of the SS autoencoder. By looking at the C_{avg} results in Table 7.12a, we can see that on average, SS based features performed worse compared to CTC-PR based features. For example, PRLM is better than SSLM on all datasets, except for AP19-OLR, where SSLM is 1 percentage point better. Similarly, using CTC-PR BNFs as input to x-vector is better than using SS BNFs, on all datasets except for AP19-OLR, where SS BNFs are 3 percentage points better. Finally, PTN is better than SS-PTN on all datasets.

One could assume that CTC-PR is worse than SS on AP19-OLR because CTC-PR has not been trained on Asian languages. However, this does not explain why CTC-PR is better than SS on MGB-3, which contains only Arabic dialects, none of which are included in the training data of CTC-PR. Furthermore, CTC-PR and SS have been trained using different datasets, which means these results do not have satisfactory comparability. Further experiments are required, where CTC-PR and SS are trained with the same data, in order to conclude which model provides a representation with the best language discriminability.

Note on OGI-11L While this is not shown in the results, we also experimented with extracting spectral features for OGI-11L only from the narrowband between 300–3400 Hz, as suggested by Reynolds (1995). However, all results from every model that used narrowband features for OGI-11L were worse than any of the results which used spectral features extracted from a wider band between 20–8000 Hz. Therefore, all results from the narrowband experiments were omitted from the reports.

7.7 Online SLI

This section provides a brief overview of an real-time SLI model based on the x-vector architecture (Section 6.3). The model was trained on the YTN-Aalto2019 dataset (Section 4.4) and converted with the TensorFlow.js[1] tool into a self-contained format, which can be executed in a web browser. Audio input is recorded in real-time from the user's microphone with

[1] `https://www.tensorflow.org/js` (visited on 2020-01-31)

the JavaScript AnalyserNode API[1], which also implements a real-time STFT. The spectral data is extracted with a 2048-point STFT from non-overlapping 30 ms windows at 30 ms intervals, converted to Mel-scale using 60 Mel-bins and stored in a buffer of spectral frames $\mathbf{X}_{\text{Mel}} \in \mathbb{R}^{T \times 60}$. Whenever the buffer contains $T = 100$ frames, these are converted into model input $\mathbf{X}'_{\text{Mel}} \in \mathbb{R}^{100 \times 60}$ and each channel is mean-normalized (reduced over axis T). Then, $\mathbf{X}'_{\text{Mel}}$ is used as input to the JavaScript x-vector model, which outputs prediction scores $\mathbf{y} \in \mathbb{R}^6$ for the 6 languages of the YTN dataset. For every prediction event, the output scores are sorted in descending order and the three most likely language classes are displayed. A screenshot of the system in operation can be seen in Figure 7.1.

Figure 7.1. Online SLI system running in a web browser. The real-time spectrogram plot seen in the screenshot depicts a decibel-scale spectrogram of the STFT output provided by the JavaScript AnalyserNode, which is converted to model input. Here, the source audio contains speech in French, and was chosen uniformly at random from the YTN-Aalto2019 dataset.

It is worth noting that the system is unstable, occasionally predicting incorrect labels seemingly at random, which makes it unusable in a practical application. However, no quantitative evaluations were performed using the YTN-Aalto2019 test set and it is therefore not possible to say in

[1] `https://webaudio.github.io/web-audio-api/#analysernode` (visited on 2020-01-31)

an objective manner how well the model performs. More work is required to stabilize the system, for example by smoothening label transitions and adding a real-time VAD module to filter out frames with low SNR. For a stable and valuable application that could be used in practice, replacing the model with a phonotactic approach might also be a good alternative, assuming transcribed data is available.

8. Future work

This chapter contains a brief overview of topics which could be considered natural continuations to the work completed during this thesis.

8.1 Extending the experiments

As discussed in Section 3.4, the optimization algorithm used in all experiments of this thesis is SGD, even though several other options were available. The main argument for choosing the, arguably more traditional, SGD algorithm was the convergence issues of the newer algorithms, especially when training LSTM models with BNF input using Adam. However, no exhaustive quantitative comparisons were actually performed and SGD was chosen due to its stability during training. Therefore, replacing the SGD algorithm with e.g. Adam would provide a straightforward experiment setting for future work. One effect to investigate would be whether some other optimization methods, such as Adam, converges faster to a global optimum compared to SGD, while retaining the same SLI performance. This would be beneficial since it would allow completing more experiments per compute hour compared to SGD.

Another experiment setting to consider is using a different VAD approach. As we saw in Chapter 7, most experiments either did not benefit from VAD or the improvements were negligible. One possible reason for this could be that the VAD windows (see Figure 3.4a) were too short. As discussed in Section 7.1, VAD decisions were applied on each feature time frame, corresponding to 25 ms of the original signal, dropping all non-speech frames regardless of the time context. This could be detrimental for SLI performance, since even short pauses in speech might convey syntactic meaning and deleting such pauses could distort the language cues. Therefore, one could experiment with setting a minimum threshold of consecutive feature frames that must contain non-speech before the frames will be considered for removal. This might help retain the most meaningful information, while still removing significant sections of less

important, non-speech information.

Readers with more experience in deep learning might have noticed that the experiments in this thesis focused mostly on reproducing existing results or comparing several different models, while little to no attention was placed on improving the achieved results further by e.g. hyper-parameter tuning or regularization. One interesting thing to try is channel dropout on the frequency dimension, which might significantly increase the generalization strength of the model (Kovács et al. 2017). In addition, data augmentation was not performed in any way and it could also help, as discussed in Section 3.1.

It's also worth noting that the acoustic-phonetic models trained during this thesis used different data than CTC-PR, since we used a pre-trained instance of CTC-PR. Therefore, the results from experiments made with CTC-PR, e.g. PRLM and PTN, cannot be used to compare the relative performance of acoustic-phonetic and phonotactic models. For an accurate comparison, CTC-PR should be trained using the same acoustic data as the acoustic-phonetic models.

Milde and Biemann (2019) suggested that training SS on larger amounts of data might improve the unsupervised representation learned from the input space. Therefore, a straightforward experiment setting would be to train SS on all datasets discussed in Chapter 4, extract SS BNF features and acoustic unit sequences, and finally repeat all experiments made with SS-PTN, SSLM, and x-vector using SS BNFs.

We saw in Section 7.5 that MLA x-vector could not outperform the regular x-vector architecture. However, performance seemed to improve with each added attention module. Further experiments are required, with larger MLA x-vector models ($L > 5$ and attention based time pooling) before we may conclude if adding MLA modules into the x-vector architecture is more beneficial compared to simply increasing the size of the x-vector model by adding regular FC layers.

8.2 Other SLI models

This thesis approached SLI from an end-to-end perspective, assuming SLI is an multiclass classification problem solvable with discriminative deep learning models. While some approaches such as the x-vector architecture show promising results, acoustic-phonetic models are usually greatly outperformed by phonotactic approaches, as we discussed in Chapter 5. Even though we did not see promising results in Chapter 7 from PRLM or PTN, the superior results reported by e.g. Watanabe et al. (2017) and Ren et al. (2019) deserve more attention if the work done in this thesis is to be continued. Another approach would be to retrain CTC-PR with new datasets, while ensuring all acoustic-phonetic and phonotactic models are

trained on matching data.

At the time of writing this thesis, it is also an open research question how to identify languages from audio signals without applying pre-processing techniques (Fayek 2016; Shon et al. 2018). While SLI from raw audio might still be considered uncharted territory, the deep learning based approaches used in this thesis could provide a good starting point for classifying any kind of speech representation.

As discussed in Section 5.2, we did not use the x-vector architecture to its full potential and instead applied it as a discriminative SLI model. Snyder et al. (2018a) showed that the x-vector model is more accurate when it is trained on a large amount of speech data containing many languages, and then a backend classifier is trained on the x-vectors $\mathbf{x} \in \mathbb{R}^{512}$ extracted from layer 7 (see Table 6.6) of the trained x-vector model. This opens up interesting experiment settings, such as classifying the x-vectors using probabilistic linear discriminant analysis (Prince and Elder 2007) or some other backend classifiers. Furthermore, Leeuwen and Brümmer (2008) showed how to adapt linear discriminant analysis based backend SLI classifiers to small amounts of training data, which might enable better SLI on small datasets such as OGI-11L.

9. Conclusions

This thesis discussed spoken language identification (SLI) and ways to apply deep learning methods for SLI. Quantitative experiments were performed and they provided answers to most of the research questions discussed in Section 1.1, even though some results were inconclusive.

First, we conclude that we managed to reproduce results of three models on three datasets within 2 percentage points, measured with C_{avg}. These models and datasets are x-vector on AP19-OLR task 1, CNN on MGB-3, and BGRU on SBS.

Second, we saw that mean-normalization of all spectral channels over the time axis is on average more beneficial compared to mean-normalization followed by variance-normalization. However, there was no single VAD setting that worked well in all scenarios.

Third, we did not find one single model architecture that outperforms all other models. However, the x-vector architecture (see Section 6.3) provided best results on 3 datasets out of 5. In addition, the CNN model (see Section 6.3) provided best results for the other 2 datasets out of 5. Since the CNN and x-vector architectures are structurally similar, we can conclude that a design combining temporal convolutions, global statistics pooling over the time dimension, followed by a DNN for language classification seems to be a good approach for classifying log-scale Mel-spectra by language. However, in contrast to results from existing work discussed in Section 5.3, phonotactic models used in this thesis, e.g. PTN, PRLM, and x-vector with CTC-PR phoneme embeddings, did not outperform the acoustic-phonetic models. This implies that our phonotactic models could not reach an optimal level of performance and more work is required to investigate why this happened.

Finally, in Section 7.6 we saw that all models that utilized acoustic unit representations learned by the SS autoencoder, in fact produced worst results in almost all experiments. We therefore conclude that it is unlikely that the unsupervised acoustic unit representation could be used to replace a supervised phoneme representation and still provide good language discriminability.

References

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from https://tensorflow.org. URL: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf (visited on 2020-03-25).

Ali, Ahmed et al. (2016). "Automatic Dialect Detection in Arabic Broadcast Speech". In: *Proc. Interspeech 2016*, pp. 2934–2938. DOI: 10.21437/Interspeech.2016-1297.

Ali, Ahmed et al. (2019). "The MGB-5 Challenge: Recognition and Dialect Identification of Dialectal Arabic Speech". In: *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Alphonsa, Alice Celin et al. (July 2017). "Spectral feature based automatic tonal and non-tonal language classification". In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. IEEE, pp. 1271–1276.

Bahari, Mohamad Hasan et al. (July 2014). "Non-Negative Factor Analysis of Gaussian Mixture Model Weight Adaptation for Language and Dialect Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.7, pp. 1117–1129. ISSN: 2329-9304. DOI: 10.1109/TASLP.2014.2319159.

Bartz, Christian et al. (2017). "Language identification using deep convolutional recurrent neural networks". In: *International Conference on Neural Information Processing*. Springer, pp. 880–889.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer. ISBN: 978-0387-31073-2.

Bishop, Richard L. and Samuel I. Goldberg (1968). In: *Tensor analysis on manifolds*. New York, NY: The Macmillan. Chap. 1.

Brümmer, Niko and Johan du Preez (2006). "Application-independent evaluation of speaker detection". In: *Computer Speech & Language* 20.2. Odyssey 2004: The speaker and Language Recognition Workshop, pp. 230–275. ISSN: 0885-2308. DOI: 10.1016/j.csl.2005.08.001.

Campbell, William M. et al. (2004). "Language recognition with support vector machines". In: *Odyssey 2016*, pp. 285–288.

Canavan, Alexandra and George Zipperlen (1996). *CALLFRIEND speech corpus*. Philadelphia: Linguistic Data Consortium.

Cheng, Jyh-Min and Hsiao-Chuan Wang (Dec. 2004). "A method of estimating the equal error rate for automatic speaker verification". In: *2004 International Symposium on Chinese Spoken Language Processing*, pp. 285–288. DOI: 10.1109/CHINSL.2004.1409642.

Cole, Ronald and Yeshwant Muthusamy (1994). *OGI Multilanguage Corpus LDC94S17*. Web Download. Philadelphia: Linguistic Data Consortium.

Dehak, Najim et al. (May 2011a). "Front-End Factor Analysis for Speaker Verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798. ISSN: 1558-7916. DOI: 10.1109/TASL.2010.2064307.

Dehak, Najim et al. (Aug. 2011b). “Language recognition via i-vectors and dimensionality reduction”. In: *Proc. Interspeech 2011*, pp. 857–860.

Fayek, Haytham (Apr. 2016). *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What’s In-Between*. URL: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (visited on 2019-11-29).

Fernando, Sarith, Vidhyasaharan Sethu, and Eliathamby Ambikairajah (Sept. 2018). “Sub-band Envelope Features Using Frequency Domain Linear Prediction for Short Duration Language Identification”. In: *Proc. Interspeech 2018* (Hyderabad, India), pp. 1818–1822. DOI: 10.21437/Interspeech.2018-1805.

Frederiksen, Peter Sibbern et al. (Sept. 2018). “Effectiveness of Single-Channel BLSTM Enhancement for Language Identification”. In: *Proc. Interspeech 2018* (Hyderabad, India), pp. 1823–1827. DOI: 10.21437/Interspeech.2018-2458.

Gauvain, Jean-Luc and Chin-Hui Lee (Apr. 1994). “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains”. In: *IEEE Transactions on Speech and Audio Processing* 2.2, pp. 291–298. ISSN: 1558-2353. DOI: 10.1109/89.279278.

Gelly, Gregory and Jean-Luc Gauvain (Aug. 2017). “Spoken Language Identification Using LSTM-Based Angular Proximity”. In: *Proc. Interspeech 2017*, pp. 2566–2570. DOI: 10.21437/Interspeech.2017-1334.

Gelly, Gregory et al. (2016). “Language recognition for dialects and closely related languages”. In: *Odyssey 2016*, pp. 124–131.

Glembek, Ondej et al. (2008). “Advances in phonotactic language recognition”. In: *Proc. Interspeech 2008*, pp. 743–746.

Gonzalez-Dominguez, Javier et al. (Sept. 2014). “Automatic language identification using long short-term memory recurrent neural networks”. In: *Fifteenth Annual Conference of the International Speech Communication Association*, pp. 2155–2159.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016a). “Deep Feedforward Networks”. In: *Deep Learning*. MIT Press. Chap. 6, pp. 164–223. URL: http://www.deeplearningbook.org (visited on 2020-03-11).

– (2016b). “Optimization for Training Deep Models”. In: *Deep Learning*. MIT Press. Chap. 8, pp. 271–325. URL: http://www.deeplearningbook.org (visited on 2020-03-11).

– (2016c). “Practical Methodology”. In: *Deep Learning*. MIT Press. Chap. 11, pp. 416–437. URL: http://www.deeplearningbook.org (visited on 2020-03-11).

– (2016d). “Representation Learning”. In: *Deep Learning*. MIT Press. Chap. 15, pp. 524–554. URL: http://www.deeplearningbook.org (visited on 2020-03-11).

Graves, Alex et al. (2006). “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA). ICML ’06. ACM, pp. 369–376. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143891.

Hazen, Timothy J. and Victor W. Zue (1993). “Automatic language identification using a segment-based approach”. In: *Proc. EUROSPEECH 1993*, pp. 1303–1306.

He, Kaiming et al. (June 2016a). “Deep Residual Learning for Image Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, Liang et al. (2016b). “THU-EE System Description for NIST LRE 2015”. In: *Proc. Interspeech 2016*, pp. 3294–3298. DOI: 10.21437/Interspeech.2016-791.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

IANA - Language subtag registry (2019). URL: https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry (visited on 2019-11-29).

Jauhiainen, Tommi et al. (Aug. 2019a). “Automatic Language Identification in Texts: A Survey”. English. In: *Journal of Artificial Intelligence Research* 65, pp. 675–782. ISSN: 1076-9757.

Jauhiainen, Tommi et al. (June 2019b). “Language and Dialect Identification of Cuneiform Texts”. In: *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties*

and Dialects (Ann Arbor, Michigan). Association for Computational Linguistics, pp. 89–98. DOI: 10.18653/v1/W19-1409. URL: https://www.aclweb.org/anthology/W19-1409.

Kamusella, Tomasz (2016). "The History of the Normative Opposition of 'Language versus Dialect': From Its Graeco-Latin Origin to Central Europes Ethnolinguistic Nation-States". In: *Colloquia Humanistica* 5. DOI: 10.11649/ch.2016.011.

Karhila, Reima et al. (Sept. 2019). "Transparent Pronunciation Scoring Using Articulatorily Weighted Phoneme Edit Distance". In: *Proc. Interspeech 2019* (Graz, Austria), pp. 1866–1870. DOI: 10.21437/Interspeech.2019-1785.

Khurana, Sameer et al. (Aug. 2017). "QMDIS: QCRI-MIT Advanced Dialect Identification System". In: *Proc. Interspeech 2017*, pp. 2591–2595. DOI: 10.21437/Interspeech.2017-1391.

KingLine Data Center (2016). *AP16-OL7 Multilingual Database*. URL: www.speechocean.com (visited on 2019-11-01).

Kingma, Diederik P. and Jimmy Ba (May 2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations* (San Diego, CA, USA). URL: http://arxiv.org/abs/1412.6980.

Ko, Tom et al. (2015). "Audio augmentation for speech recognition". In: *Proc. Interspeech 2015*, pp. 3586–3589.

Kovács, György et al. (2017). "Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout". In: *Pattern Recognition Letters* 100, pp. 44–50. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2017.09.023.

Lander, Terri et al. (Sept. 1995). "The OGI 22 language telephone speech corpus". In: *EUROSPEECH 1995: Fourth European Conference on Speech Communication and Technology* (Madrid, Spain), pp. 817–820.

Leeuwen, David A. van and Niko Brümmer (2008). "Building language detectors using small amounts of training data". In: *Proc. Odyssey 2008*.

Li, Haizhou, Bin Ma, and Kong Aik Lee (May 2013). "Spoken Language Recognition: From Fundamentals to Practice". In: *Proceedings of the IEEE* 101.5, pp. 1136–1159. ISSN: 0018-9219. DOI: 10.1109/JPROC.2012.2237151.

Lin, Ruixi et al. (Sept. 2019). "Optimizing Voice Activity Detection for Noisy Conditions". In: *Proc. Interspeech 2019* (Graz, Austria), pp. 2030–2034. DOI: 10.21437/Interspeech.2019-1776.

Lopez-Moreno, Ignacio et al. (May 2014). "Automatic language identification using deep neural networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5337–5341. DOI: 10.1109/ICASSP.2014.6854622.

Lozano-Diez, Alicia et al. (2015). "An end-to-end approach to language identification in short utterances using convolutional neural networks". In: *Proc. Interspeech 2015*, pp. 403–407.

Ma, Zhanyu et al. (Jan. 2019). "Short Utterance Based Speech Language Identification in Intelligent Vehicles With Time-Scale Modifications and Deep Bottleneck Features". In: *IEEE Transactions on Vehicular Technology* 68.1, pp. 121–128. DOI: 10.1109/TVT.2018.2879361.

Martínez, David et al. (2011). "Language Recognition in iVectors Space". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 861–864.

Mateju, Lukas et al. (2018). "Using Deep Neural Networks for Identification of Slavic Languages from Acoustic Signal". In: *Proc. Interspeech 2018* (Hyderabad, India), pp. 1803–1807. DOI: 10.21437/Interspeech.2018-1165.

Mazzawi, Hanna et al. (Sept. 2019). "Improving Keyword Spotting and Language Identification via Neural Architecture Search at Scale". In: *Proc. Interspeech 2019* (Graz, Austria), pp. 1278–1282. DOI: 10.21437/Interspeech.2019-1916.

Miao, Xiaoxiao, Ian McLoughlin, and Yonghong Yan (Sept. 2019). "A New Time-Frequency Attention Mechanism for TDNN and CNN-LSTM-TDNN, with Application to Language Identification". In: *Proc. Interspeech 2019* (Graz, Austria), pp. 4080–4084. DOI: 10.21437/Interspeech.2019-1256.

Milde, Benjamin and Chris Biemann (Sept. 2019). “SparseSpeech: Unsupervised Acoustic Unit Discovery with Memory-Augmented Sequence Autoencoders”. In: *Proc. Interspeech 2019* (Graz, Austria), pp. 256–260. DOI: 10.21437/Interspeech.2019-2938.

Mohamed, Abdel-rahman (2014). “Deep Neural Network Acoustic Models for ASR”. PhD thesis. URL: http://hdl.handle.net/1807/44123.

Mozilla Common Voice (2020). URL: https://voice.mozilla.org/en (visited on 2020-02-10).

Muthusamy, Yeshwant K., Etienne Barnard, and Ronald A. Cole (Oct. 1994). “Reviewing automatic language identification”. In: *IEEE Signal Processing Magazine* 11.4, pp. 33–41. DOI: 10.1109/79.317925.

Muthusamy, Yeshwant K. and Ronald A. Cole (1992). “Automatic segmentation and identification of ten languages using telephone speech”. In: *Second International Conference on Spoken Language Processing*, pp. 1007–1010.

Muthusamy, Yeshwant K., Ronald A. Cole, and Beatrice T. Oshika (1992). “The OGI multilanguage telephone speech corpus”. In: *Second International Conference on Spoken Language Processing*.

Navratil, Jiri (Sept. 2006). “Recent advances in phonotactic language recognition using binary-decision trees”. In: *Proc. Interspeech 2006* (Pittsburgh, PA, USA).

Ng, Raymond W. M. et al. (2010). “Prosodic attribute model for spoken language identification”. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5022–5025. DOI: 10.1109/ICASSP.2010.5495070.

Nouza, Jan, Radek Safarik, and Petr Cerva (2016). “ASR for South Slavic Languages Developed in Almost Automated Way”. In: *Proc. Interspeech 2016*, pp. 3868–3872. DOI: 10.21437/Interspeech.2016-747.

Padi, Bharat, Anand Mohan, and Sriram Ganapathy (May 2019). “End-to-end Language Recognition Using Attention Based Hierarchical Gated Recurrent Unit Models”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5966–5970. DOI: 10.1109/ICASSP.2019.8683895.

Phillips, A. and M. Davis (2019). *Tags for Identifying Languages*. URL: https://tools.ietf.org/rfc/bcp/bcp47.txt (visited on 2019-11-29).

Pohjalainen, Jouni (2014). “Robust Methods for Speech Feature Extraction”. Doctoral Dissertation, 96 + app. 109. ISBN: 978-952-60-6006-4. URL: http://urn.fi/URN:ISBN:978-952-60-6006-4.

Povey, Daniel et al. (2011). “The Kaldi Speech Recognition Toolkit”. In: IEEE Catalog No.: CFP11SRW-USB. URL: http://infoscience.epfl.ch/record/192584.

Prince, Simon J.D. and James H. Elder (Oct. 2007). “Probabilistic Linear Discriminant Analysis for Inferences About Identity”. In: *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. DOI: 10.1109/ICCV.2007.4409052.

Ramus, Franck and Jacques Mehler (1999). “Language identification with suprasegmental cues: A study based on speech resynthesis”. In: *The Journal of the Acoustical Society of America* 105.1, pp. 512–521. DOI: 10.1121/1.424522.

Räsänen, Okko (2013). “Studies on unsupervised and weakly supervised methods in computational modeling of early language acquisition”. PhD thesis, 74 + app. 122. ISBN: 978-952-60-5096-6 (printed). URL: http://urn.fi/URN:ISBN:978-952-60-5097-3.

Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar (Apr. 2018). “On the Convergence of Adam and Beyond”. In: *6th International Conference on Learning Representations* (Vancouver, BC, Canada). URL: https://openreview.net/forum?id=ryQu7f-RZ.

Ren, Zongze, Guofu Yang, and Shugong Xu (Sept. 2019). “Two-Stage Training for Chinese Dialect Recognition”. In: *Proc. Interspeech 2019* (Graz, Austria), pp. 4050–4054. DOI: 10.21437/Interspeech.2019-1522.

Reynolds, Douglas A. (1995). “Speaker identification and verification using Gaussian mixture speaker models”. In: *Speech Communication* 17, pp. 91–108. ISSN: 0167-6393. DOI: https://doi.org/10.1016/0167-6393(95)00009-D.

Richardson, Fred, Douglas Reynolds, and Najim Dehak (2015). "Deep Neural Network Approaches to Speaker and Language Recognition". In: *IEEE Signal Processing Letters* 22.10, pp. 1671–1675. DOI: 10.1109/LSP.2015.2420092.

Ringbom, Håkan (2007). *Cross-linguistic similarity in foreign language learning*. Vol. 21. Multilingual Matters. ISBN: 978-1-85359-935-4.

Sadjadi, Seyed Omid et al. (2018). "The 2017 NIST Language Recognition Evaluation". In: *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 82–89. DOI: 10.21437/Odyssey.2018-12.

Sak, Haim, Andrew Senior, and Françoise Beaufays (2014). "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Fifteenth annual conference of the international speech communication association*.

Schultz, Tanja and Katrin Kirchhoff (June 2006). *Multilingual Speech Processing*. 1st ed. Elsevier Science and Technology, Burlington. ISBN: 9780120885015.

Seide, Frank et al. (Dec. 2011). "Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription". In: *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pp. 24–29. DOI: 10.1109/ASRU.2011.6163899.

Shon, Suwon, Ahmed Ali, and James Glass (June 2018). "Convolutional Neural Network and Language Embeddings for End-to-End Dialect Recognition". In: *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 98–104. DOI: 10.21437/Odyssey.2018-14.

Singer, Elliot et al. (2003). "Acoustic, phonetic, and discriminative approaches to automatic language identification". In: *EUROSPEECH 2003* (Geneva, Switzerland), pp. 1345–1348.

Snyder, David et al. (June 2018a). "Spoken Language Recognition using X-vectors". In: *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 105–111. DOI: 10.21437/Odyssey.2018-15.

Snyder, David et al. (Apr. 2018b). "X-Vectors: Robust DNN Embeddings for Speaker Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333. DOI: 10.1109/ICASSP.2018.8461375.

SoX - Sound eXchange (2015). URL: http://sox.sourceforge.net/ (visited on 2020-02-10).

Sutskever, Ilya et al. (2013). "On the Importance of Initialization and Momentum in Deep Learning". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning* (Atlanta, GA, USA). Vol. 28. ICML'13. JMLR.org, pp. 1139–1147.

Szegedy, Christian et al. (June 2016). "Rethinking the Inception Architecture for Computer Vision". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826.

Tang, Zhiyuan, Dong Wang, and Qing Chen (Nov. 2018a). "AP18-OLR Challenge: Three Tasks and Their Baselines". In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 596–600. DOI: 10.23919/APSIPA.2018.8659714.

Tang, Zhiyuan, Dong Wang, and Liming Song (Nov. 2019). "AP19-OLR Challenge: Three Tasks and Their Baselines". In: *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1917–1921. DOI: 10.1109/APSIPAASC47483.2019.9023321.

Tang, Zhiyuan et al. (Dec. 2017). "AP17-OLR challenge: Data, plan, and baseline". In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 749–753. DOI: 10.1109/APSIPA.2017.8282134.

Tang, Zhiyuan et al. (Jan. 2018b). "Phonetic Temporal Neural Model for Language Identification". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.1, pp. 134–144. ISSN: 2329-9290. DOI: 10.1109/TASLP.2017.2764271.

Tong, Audrey et al. (2016). "Summary of the 2015 NIST Language Recognition i-Vector Machine Learning Challenge". In: *Odyssey 2016: The Speaker and Language Recognition Workshop* (Bilbao, Spain), pp. 297–302. URL: http://www.isca-speech.org/archive/odyssey_2016/pdfs_stamped/74.pdf.

Tong, Rong et al. (May 2006). “Integrating Acoustic, Prosodic and Phonotactic Features for Spoken Language Identification”. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Vol. 1. DOI: 10.1109/ICASSP.2006.1659993.

Torres-Carrasquillo, Pedro A. et al. (2002). “Approaches to language identification using Gaussian mixture models and shifted delta cepstral features”. In: *Seventh International Conference on Spoken Language Processing*.

Vafeiadis, Anastasios et al. (Sept. 2019). “Two-Dimensional Convolutional Recurrent Neural Networks for Speech Activity Detection”. In: *Proc. Interspeech 2019* (Graz, Austria), pp. 2045–2049. DOI: 10.21437/Interspeech.2019-1354.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wakita, Hisashi (Apr. 1977). “Normalization of vowels by vocal-tract length and its application to vowel identification”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.2, pp. 183–192. ISSN: 0096-3518. DOI: 10.1109/TASSP.1977.1162929.

Wan, Li et al. (May 2019). “Tuplemax Loss for Language Identification”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5976–5980. DOI: 10.1109/ICASSP.2019.8683313.

Wang, Dong, Xuewei Zhang, and Zhiyong Zhang (2015). *THCHS-30 : A Free Chinese Speech Corpus*. URL: http://arxiv.org/abs/1512.01882.

Wang, Dong et al. (Dec. 2016). “AP16-OL7: A multilingual database for oriental languages and a language recognition baseline”. In: *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1–5. DOI: 10.1109/APSIPA.2016.7820796.

Wang, Shuai, Yanmin Qian, and Kai Yu (2017). “What Does the Speaker Embedding Encode?” In: *Proc. Interspeech 2017*, pp. 1497–1501. DOI: 10.21437/Interspeech.2017-1125.

Watanabe, Shinji, Takaaki Hori, and John R. Hershey (Dec. 2017). “Language independent end-to-end architecture for joint language identification and speech recognition”. In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 265–271. DOI: 10.1109/ASRU.2017.8268945.

WebRTC (Apr. 2020). URL: https://webrtc.org/ (visited on 2020-04-26).

Weston, Jason, Frédéric Ratle, and Ronan Collobert (2008). “Deep Learning via Semi-Supervised Embedding”. In: *Proceedings of the 25th International Conference on Machine Learning*. Helsinki, Finland, pp. 1168–1175. DOI: 10.1145/1390156.1390303.

Xu, Zhicun (Dec. 2018). “Audio Event Classification Using Deep Learning Methods”. en. MA thesis, pp. 67+6. URL: http://urn.fi/URN:NBN:fi:aalto-201812146460.

YouTube (Nov. 2019). URL: https://www.youtube.com/ (visited on 2019-11-26).

youtube-dl: Download videos from YouTube (Nov. 2019). URL: https://ytdl-org.github.io/youtube-dl/index.html (visited on 2019-11-26).

Yu, Changsong et al. (2018). “Multi-level attention model for weakly supervised audio classification”. In: *DCASE2018 Workshop on Detection and Classification of Acoustic Scenes and Events*. URL: http://epubs.surrey.ac.uk/849626/.

Yu, Dong and Li Deng (2015a). “Deep Neural Networks”. In: *Automatic Speech Recognition: A Deep Learning Approach*. London: Springer. Chap. 4, pp. 57–77. ISBN: 978-1-4471-5778-6. DOI: 10.1007/978-1-4471-5779-3.

– (2015b). “Gaussian Mixture Models”. In: *Automatic Speech Recognition: A Deep Learning Approach*. London: Springer. Chap. 2, pp. 13–21. ISBN: 978-1-4471-5778-6. DOI: 10.1007/978-1-4471-5779-3.

– (2015c). “Hidden Markov Models and the Variants”. In: *Automatic Speech Recognition: A Deep Learning Approach*. London: Springer. Chap. 3, pp. 23–54. ISBN: 978-1-4471-5778-6. DOI: 10.1007/978-1-4471-5779-3.

Zampieri, Marcos and Binyam Gebrekidan Gebre (Sept. 2012). “Automatic identification of language varieties: The case of Portuguese”. In: *Proceedings of the Conference on Natural*

Language Processing 2012 (Vienna, Austria), pp. 233–237. URL: http://hdl.handle.net/11858/00-001M-0000-000F-EC46-0.

Zazo, Ruben, Alicia Lozano-Diez, and Joaquin Gonzalez-Rodriguez (June 2016a). "Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks". In: *Odyssey 2016: The Speaker and Language Recognition Workshop* (Bilbao, Spain), pp. 231–236. URL: http://www.isca-speech.org/archive/odyssey_2016/pdfs_stamped/45.pdf.

Zazo, Ruben et al. (Jan. 2016b). "Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks". In: *PLOS ONE* 11.1, pp. 1–17. DOI: 10.1371/journal.pone.0146917.

Zhan, Puming and Alex Waibel (1997). *Vocal tract length normalization for large vocabulary continuous speech recognition*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

Zhang, Chunlei, Qian Zhang, and John H.L. Hansen (May 2019). "Semi-supervised Learning with Generative Adversarial Networks for Arabic Dialect Identification". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5986–5990. DOI: 10.1109/ICASSP.2019.8682629.

Zhao, Jingjing et al. (Nov. 2008). "Cortical competition during language discrimination". In: *NeuroImage* 43.3. Copyright - Copyright Elsevier Limited Nov 15, 2008; Last updated - 2015-03-28, pp. 624–633. DOI: 10.1016/j.neuroimage.2008.07.025.

Zissman, Marc A. (Jan. 1996). "Comparison of four approaches to automatic language identification of telephone speech". In: *IEEE Transactions on Speech and Audio Processing* 4.1, pp. 31–. ISSN: 1063-6676. DOI: 10.1109/TSA.1996.481450.

Zou, Fangyu et al. (June 2019). "A Sufficient Condition for Convergences of Adam and RMSProp". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11119–11127. DOI: 10.1109/CVPR.2019.01138.