UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE TECNOLOGÍA
ELECTRÓNICA Y DE LAS COMUNICACIONES

Deep Neural Network Architectures for Large-scale, Robust and Small-Footprint Speaker and Language Recognition

–TESIS DOCTORAL–

Autor: Ignacio López Moreno
(Ingeniero de Telecomunicación,
Universidad Politécnica de Madrid)

January 2017

Departamento:	Tecnología Electrónica y de las Comunicaciones Escuela Politécnica Superior Universidad Autónoma de Madrid (UAM)
Tesis:	Deep Neural Network Architectures for Large-scale, Robust and Small-Footprint Speaker and Language Recognition
Autor:	**Ignacio López Moreno** Ingeniero de Telecomunicación
Director:	**Joaquín González Rodríguez** Catedrático de Universidad Universidad Autónoma de Madrid
Director:	**Javier González Domínguez** Universidad Autónoma de Madrid
Fecha:	Abril, 2017
Tribunal:	**Luis A. Hernández Goméz** (Presidente) Universidad Politécnica de Madrid
	Doroteo Torre Toledano (Secretario) Universidad Autónoma de Madrid
	Pedro J. Moreno Mengibar (Vocal 1) Google Inc, NY, USA
	Nicholas Evans (Vocal 2) Eurecom, Biot, Francia
	Roberto Paredes Palacios (Vocal 3) Universidad Politécnica de Valencia

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This research was developed in collaboration with Google Inc., New York, and the ATVS biometric group (Universidad Autónoma de Madrid).

Abstract

Artificial neural networks are powerful learners of the information embedded in speech signals. They can provide compact, multi-level, nonlinear representations of temporal sequences and holistic optimization algorithms capable of surpassing former leading paradigms. Artificial neural networks are, therefore, a promising technology that can be used to enhance our ability to recognize speakers and languages–an ability increasingly in demand in the context of new, voice-enabled interfaces used today by millions of users. The aim of this thesis is to advance the state-of-the-art of language and speaker recognition through the formulation, implementation and empirical analysis of novel approaches for large-scale and portable speech interfaces. Its major contributions are: (1) novel, compact network architectures for language and speaker recognition, including a variety of network topologies based on fully-connected, recurrent, convolutional, and locally connected layers; (2) a bottleneck combination strategy for classical and neural network approaches for long speech sequences; (3) the architectural design of the first, public, multilingual, large vocabulary continuous speech recognition system; and (4) a novel, end-to-end optimization algorithm for text-dependent speaker recognition that is applicable to a range of verification tasks. Experimental results have demonstrated that artificial neural networks can substantially reduce the number of model parameters and surpass the performance of previous approaches to language and speaker recognition, particularly in the cases of long short-term memory recurrent networks (used to model the input speech signal), end-to-end optimization algorithms (used to predict languages or speakers), short testing utterances, and large training data collections.

Resumen

Las redes neuronales artificiales son sistemas de aprendizaje capaces de extraer la información embebida en las señales de voz. Son capaces de modelar de forma eficiente secuencias temporales complejas, con información no lineal y distribuida en distintos niveles semanticos, mediante el uso de algoritmos de optimización integral con la capacidad potencial de mejorar los sistemas aprendizaje automático existentes. Las redes neuronales artificiales son, pues, una tecnología prometedora para mejorar el reconocimiento automático de locutores e idiomas; siendo el reconocimiento de de locutores e idiomas, tareas con cada vez más demanda en los nuevos sistemas de control por voz, que ya utilizan millones de personas. Esta tesis tiene como objetivo la mejora del estado del arte de las tecnologías de reconocimiento de locutor y de idioma mediante la formulación, implementación y análisis empírico de nuevos enfoques basados en redes neuronales, aplicables a dispositivos portátiles y a su uso en gran escala. Las principales contribuciones de esta tesis incluyen la propuesta original de: (1) arquitecturas eficientes que hacen uso de capas neuronales densas, localmente densas, recurrentes y convolucionales; (2) una nueva estrategia de combinación de enfoques clásicos y enfoques basados en el uso de las denominadas *redes de cuello de botella*; (3) el diseño del primer sistema público de reconocimiento de voz, de vocabulario abierto y continuo, que es además multilingüe; y (4) la propuesta de un nuevo algoritmo de optimización integral para tareas de reconocimiento de locutor, aplicable también a otras tareas de verificación. Los resultados experimentales extraídos de esta tesis han demostrado que las redes neuronales artificiales son capaces de reducir el número de parámetros usados por los algoritmos de reconocimiento tradicionales, así como de mejorar el rendimiento de dichos sistemas de forma substancial. Dicha mejora relativa puede acentuarse a través del modelado de voz mediante redes recurrentes de memoria a largo plazo, el uso de algoritmos de optimización integral, el uso de locuciones de evaluation de corta duración y mediante la optimización del sistema con grandes cantidades de datos de entrenamiento.

I would like to dedicate this thesis to my parents Emilio and Cristina, to my sister Helena and to my other half Anne ...

Acknowledgements

The phrase "on the shoulders of giants" doesn't just remind me of the scientists who have extended the frontiers of human understanding, but of the friends and colleagues who have helped me reach this important academic milestone.

I would like to acknowledge Prof. Joaquín González Rodríguez, whose perseverance has at many points been stronger than mine since this endeavour began in 2003 (yup, 14 years ago). During this decade and a half, Joaquín's advice has helped me define my professional and academic career. Likewise, I would like to acknowledge Javier González Domínguez, without whose dedication and help many of the scientific results presented here would never have been published. I would also like to to extend my appreciation to Pedro Moreno for encouraging me to continue my work on speaker and language recognition at Google and, to Alex Gruenstein, for helping me bring together a professional and academic career focused on these topics.

I am lucky to have discovered giants in many corners of life, including Javier Franco Pedroso (thanks you for helping with the paperwork!), Daniel Ramos, Hasim Sak, Doroteo Torre and other present and former members of the ATVS group (among them Javier Galbally, Pedro Tomé, Ruben Vera, Alberto Montero, Julian Fierrez, Verónica Peña, María Puertas, Danilo Spada, Miriam Moreno, Marta Gómez, Almudena Gilperez, Ismael Mateos, Daniel Hernández, Alejandro Abejón, Lucas Pérez, Manuel Freire, and the list could go on and on).

I would also like to dedicate this thesis to my friends, those in NYC: Julio, Ramón, Sara, Marçal, Jesús, Beltran, Carlos, Xavi, Daniel, Vanesa, Luis, Alvaro, and those in Madrid: David, Joel, Miguel, Carmen, Emilio, Loa, Olivia, Mikel, Humber and Edu.

Last, but firstly, I would like to dedicate this thesis to my loving family and to my partner (and chief editor), Anne. Cheers!

Ignacio López Moreno.
NYC, January, 29th, 2017

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ANN	Artificial neural network
DNN	Deep feed-forward neural networks
EM	Expectation maximization
GMM	Gaussian mixture model
JFA	Joint factor analysis
LID	Automatic spoken language identification
LSTM	Long short-term memory recurrent neural networks
LVCSR	Large vocabulary continuous speech recognition
MAP	Maximum–a–posteriori
MFCC	Mel-frequency cepstral coefficients
PLDA	Probabilistic linear discriminant analysis
ReLu	Rectified linear units, $ReLu(x) = max(0,x)$
RNN	Recurrent neural networks
SDC	Shifted delta cepstral coefficients
SV	Automatic speaker verification
UBM	Universal background model
WER	Word error rate

Chapter 1

Introduction

In recent years, there has been increasing interest–in the scientific community and in private and governmental sectors–in automatic systems that leverage information from a variety of digital sources, including speech, images and documents. This increased interest is a result of the fast development of new algorithms, the availability of large, annotated collections of data, and an increase in the computational scale of modern processors that allows faster optimization of algorithms and the deployment of such technologies in our everyday life (for example, in portable devices).

The availability of computing scale has allowed us to rethink and question the previously accepted approaches to computer science in many disciplines. It has also allowed us to reconsider methods that, in the past, had been considered prohibitively expensive. Research has developed around a family of algorithms, namely artificial neural networks [4, 26, 43] (ANNs). ANNs have recently shown remarkable success in tackling some of the world's most difficult machine learning challenges, including in the areas of image recognition [41], translation [76], sentiment analysis in text [20], speech recognition [33], face recognition [66], document retrieval [42] and model visualization techniques [55], to name a few.

1.1 Artificial Neural Networks for Speech Recognition

ANNs have been investigated for many years as an approach to leverage information from speech signals [15, 74]. It was not until circa 2012, however, that neural networks revolutionized the field of speech recognition [33], setting a new precedent for other speech technologies that would follow.

During this technological revolution, ANNs have proven to be a particularly successful technique for characterizing the acoustic model component of speech recognition systems. ANN-based acoustic models are able to predict with remarkable accuracy the spoken phoneme in a short (approximately 100ms long) speech segment. Research has shown that ANNs are particularly successful in surpassing previous acoustic modeling approaches when very large amounts of training data are available, as the benefits of holistic optimisation tend to outweigh the benefits of optimisations based on prior knowledge. Additionally, fewer assumptions about the data representation are needed.

1.2 Demand for Speaker and Language Recognition

Today, large vocabulary continuous speech recognition (LVCSR) systems are becoming increasingly relevant for industry, tracking the technological trend toward increased human interaction using hands-free voice-operated devices [64]. For example, dialogue and command-and-control systems (e.g., Google Assistant, Apple's Siri and Amazon Echo) are now used by millions. As the transcription accuracy progressively improves in modern LVCSR systems, certain obstacles remain, diminishing user experience of these dialogue and command-and-control systems.

One such obstacle is the ability to authenticate the identity of each speaker. This ability is convenient for transparently personalizing the system's feedback and providing a layer of security by granting access only to known speakers. Speaker recognition can also help to improve the performance of speech recognition [67] and keyword spotting [8] systems. In such cases, a speaker independent model is adapted using a small amount of data from a single speaker, with the resulting speaker-specific model performing better on test data from that speaker.

For multilingual users, another perceived obstacle to natural interaction with devices is the monolingual character of LVCSR systems, meaning that users are limited to speaking a single, preset language. This could be considered a severe limitation as, according to several sources [70] [30] [73], multilingual speakers already outnumber monolingual speakers globally, and predictions point toward increasing numbers of multilingual speakers in the future.

1.3 Thesis Goals

The capacity to transparently and accurately recognize both the speaker and the language spoken is a desirable feature of 'smarter' and more natural, speech-based interfaces. This thesis addresses the problems associated with and contributions made toward language identification and speaker verification technologies in the context of global public speech services. The author's goals are to: 1) provide research results that improve speaker and language recognition technologies through the use of modern ANNs, and 2) make these technologies suitable for the most common portable devices, a challenge that imposes severe limitations on the algorithm, including small disk and memory usage (i.e., model footprint), small lapsed time between the user's end of interaction and the system's provided response (i.e., latency), and small battery and power consumption, which limit the algorithm's possible number of operations per second.

Chapter 2

Dissertation Structure and Claims

2.1 Dissertation Structure

This dissertation is presented in the format of a compendium of academic contributions, which includes journal papers, conference papers and patents. It is divided into two independent documents that comply with university regulations.

1. The first of the two documents consists of this thesis, which comprises six chapters: Chapter 1 is an introduction to the domains of speaker and language recognition for real-world applications. This chapter (chapter 2) describes the overall dissertation's structure and and claims. The definition and classical formulation of language and speaker recognition is summarized in chapter 3. Chapters 4 and 5 introduces artificial neural networks and summarizes the author's contributions to the fields of language and speaker recognition. Finally, conclusions and ideas for future research are presented in chapter 6.

2. The second document is divided into three appendices that are attached to this thesis and that list the author's academic contributions (three journal papers [Appendix A], seven conference papers [Appendix B], and four patents [Appendix C] relevant to language and speaker recognition). A brief description of each publication/patent is provided, along with a summary of the author's specific contributions to the work and the publication's/patent's full text.

2.2 Claims

As summarized in this dissertation, the author has contributed to the fields of language and speaker recognition through the formulation, implementation and empirical analysis of the following novel neural network approaches:

Contributions to Language Recognition

- A novel, feed-forward deep neural network (DNN) system for large-scale language recognition. This approach, when compared with alternative approaches, reduced the error rate in short utterances by more than a factor of three [23, 45].
- A combination strategy for a DNN-based language recognition system and multiple large vocabulary continuous speech recognition (LVCSR) backends, with minimum negative impact in latency, CPU load and word error rate (WER) [21].
- The first successful long short-term memory (LSTM) model for large-scale language recognition. By using LSTM models, the number of parameters required by the language recognition model was reduced by a factor of twenty [24].
- A novel bottleneck strategy for DNN-based language recognition. When combined with state-of-the-art speaker recognition techniques, this approach showed remarkable performance on both short and long utterances [44].

Contributions to Speaker Recognition

- The first successful feed-forward DNN system for large-scale, text-dependent speaker recognition, a technique that achieved performance comparable to other standard benchmarks [71].
- A novel approach using locally connected layers that succeeded in reducing the number of parameters and computations of a DNN-based, text-dependent speaker recognition system by a factor of three, with no negative effects on performance [9].
- A novel "end-to-end" optimization protocol for speaker recognition. This protocol enables all of a system’s components to be trained with a single algorithm that uses the same evaluation metric during both training and testing. This approach led to remarkable improvements in text-dependent speaker recognition, reducing errors by more than half [31].

Additional engineering contributions of the author include: $a)$ novel contributions to classic i-vector techniques [46, 65, 67]; $b)$ the acquisition of large speaker and language datasets; and $c)$ collaborating in the implementation of inference engines suitable for Google's production systems, which are available to millions of users. The author has also been a major contributor to the launch of the first multilingual speech recognition service and the Trusted Voice service (a speech-based biometric service for unlocking android devices).

Chapter 3

Language and Speaker Recognition

This chapter provides a formal definition of language identification and speaker verification, a brief introduction to the classical formulations thereof, and a rationale for why these classical approaches may be sub-optimal in real-world applications.

3.1 Supervised Learning

Language and speaker recognition, at its core, is a problem of machine learning [5, 14]. The problems of machine learning can be considered variations on data-driven function estimators $f(x)$. In *supervised* learning, a set of pairs (X,Y) is provided as inputs $X = (x_1, x_2, \ldots,)$ and outputs $Y = (y_1, y_2, \ldots,)$, with the goal of training a model $f : x \mapsto y$ that approximates the function's behavior in a generalizable fashion. This is different from *unsupervised* learning, which is used for unlabeled datasets with no target outputs $(X, -)$ or rewards from the environment. *Semi-supervised* learning is a class of supervised learning that makes use of labeled (X,Y) and unlabeled $(X, -)$ data for training (typically a small amount of labeled data and a large amount of unlabeled data), and is generally motivated by the fact that large, labeled datasets are difficult to acquire.

When the elements of Y correspond to independent classes, the learning problem is referred to as *classification* (e.g., a color recognition problem comprising targets such as green, red, blue), whereas, in *regression* problems, Y stands for continuous values sampled from a target function (e.g., energy demand in the electrical grid).

In supervised learning, modelling the distribution $p(y|x)$ is a natural approach for classifying a given example x into a class y, which is why algorithms that model this directly are called *discriminative* algorithms. *Generative* algorithms, on the other hand, model $p(x,y)$, which, by applying Bayes rule, can be transformed into $p(y|x)$ and then used for classification. Discriminative models generally outperform generative models in classification tasks [37];

however, modeling $p(x,y)$ can be used for other purposes, such as to generate likely (x,y) pairs.

3.2 Open-Set and Closed-Set Classification

This thesis deals exclusively with supervised classification problems. This subset of machine learning problems can be further divided into two categories: *open-set* and *closed-set* classification. The distinction between open-set and closed-set classification is mainly a technical one. Although the definitions are conceptually similar, these two types of machine learning problems often have very different solutions.

In closed-set classification, often referred to as *identification* (Fig. 3.1), the set of classes (Y) with which the algorithm is trained is identical to the set of classes (Y') that exist in a disjointed testing dataset. One common criticism of systems trained to optimize an identification metric is that they may underperform in cases where they have to learn to reject examples from unseen classes.

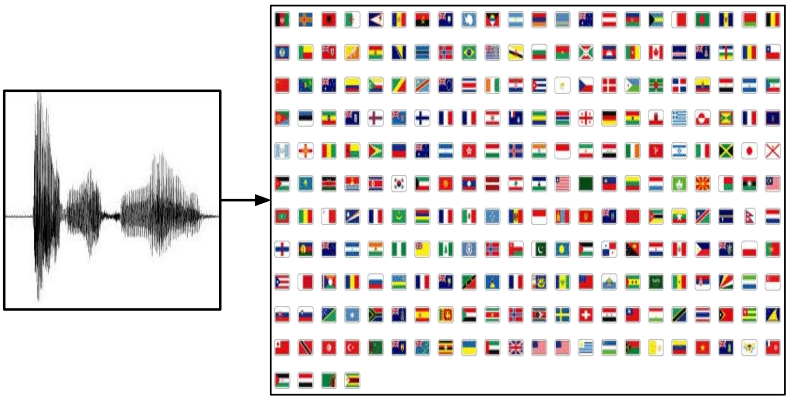


Fig. 3.1 A depiction of language recognition as a closed-set recognition problem (i.e., *identification* problem). Language identification aims to determine which language is being spoken out of a list of known candidates.

Open-set classification covers the more general classification problem in which two sets (Y and Y') overlap partially or not at all. One approach to addressing this problem is to formulate it as a *verification* problem where the goal is to determine (yes or no) if two samples belong to the same category (Fig. 3.2). Generally, verification systems are trained with large quantities of labeled examples in the hope that they will generalize learning to the unseen classes of Y' by using a very limited set of labeled reference examples (i.e., *enrollment* examples).

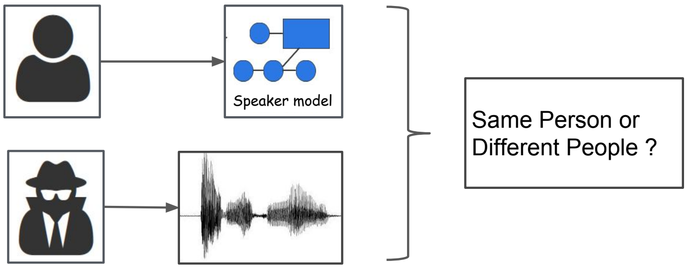


Fig. 3.2 Speaker recognition depicted as a verification problem. Speaker verification aims to determine the speaker's identity based on one or more reference examples that are usually represented by a parametric model.

3.3 Language Identification

Automatic language recognition refers to the process of automatically determining the language in a given speech sample [56], with output categories such as *English* or *Spanish*. In language recognition problems, the list of languages with which the algorithm will be evaluated is typically known at training time. Thus, the problem is often referred to as a language identification (LID) problem (Fig. 3.1).

In general, language discriminant information is spread across different structures or levels of the speech signal, ranging from low-level, short-term acoustic and spectral features to high-level, long-term features (i.e. phonotactic, prosodic). However, even though several high-level approaches are used to get meaningful information from complementary sources [16, 51, 79], most language identification systems still rely on acoustic modelling [22, 68], which mostly targets short-term features. This is because acoustic models tend to have better scalability and computational efficiency.

LID is used in several applications, including multilingual speech recognition systems [21], translation systems and emergency call routing (where the response time of a fluent native operator may be critical) [1, 56].

3.4 Speaker Verification

Automatic speaker recognition is the process of verifying, based on a speaker's known utterances, whether an utterance belongs to that speaker or not (e.g., *same speaker* or *different speaker*). In speaker recognition problems, it is typically required that a system be generalizeable to recognize any arbitrary speaker, rather than just those speakers found in the training set Y. Thus, as the list of candidate speakers is typically unknown at training time, this problem is often referred to as the speaker verification (SV) problem (Fig. 3.2).

As in the case of LID, speaker discriminant information is spread across different levels of speech as a result of morphological and behavioural patterns [36]. Morphological information originates from physical differences in speech organs, such as the size and shape of the vocal tract, lungs and larynx, which mostly effect short-term features. Behavioral information, on the other hand, originates from characteristic manners of speaking, such as the use of a particular rhythm, intonation, pronunciation style and vocabulary, which mostly influences long-term features. Currently, most successful automatic SV approaches extract discriminant information using acoustic models that primarily target short-term features and morphological differences among individuals [13, 17, 39].

When the lexicon of spoken utterances is constrained to a single word or phrase across all users, the process is referred to as global password, text-dependent speaker verification. By constraining the lexicon, text-dependent speaker verification aims to compensate for phonetic variability, which poses a significant challenge to the process of speaker verification [2]. The speaker recognition part of this thesis is focused on text-dependent speaker verification with the global password ”Ok Google.” The choice of this particularly short, approximately 0.6 second-long global password is due to its use in Google’s Keyword Spotting system [58] as an entry-point to Google’s Voice Search system [64].

Applications of SV include forensics [25, 61], authentication in security-critical systems [49], the personalization of speech interfaces [67], and when searching for speakers in large databases [65].

3.5 Classical Approaches

I-vector-based systems have become the standard approach for LID [52] and SV [13], and have grown in popularity in a wide range of other fields, including personalized speech recognition [67], emotion recognition [47], age estimation [3] and intelligibility assessment [50].

One i-vector can be understood as a fixed-size compact representation of an utterance, which is derived from an unsupervised data-driven algorithm capable of capturing the accumulated long-term effects of multiple sources embedded in speech. I-vector-based systems are therefore semi-supervised systems that use i-vectors as a way to parametrize an utterance, while the classification task is performed by a subsequent recognition backend.

3.5.1 Feature Extraction

The first step in LID and SV systems consists of providing a time-frequency representation of the speech signal that is derived from a short-term frequency analysis of the waveform and produces the so-called short-term, low-level acoustic features.

Specifically, the audio is segmented into sliding windows of typically 25ms that have 10ms overlap. Each window is then transformed into a vector of frequency coefficients and further transformed using algorithms that seek to reflect the human auditory frequency analysis, such as mel-frequency cepstral coefficients (MFCCs) [12] or shifted delta cepstral (SDC) [69]. SV systems tend to use MFCC features together with their first-order temporal differences, while LID systems use SDC features more frequently.

MFCC and SDC are approaches to pre-process audio that are not data-driven, but rather based on prior knowledge. They are designed to discard information in the audio that is considered irrelevant and express the remaining information in a form that facilitates differentiation among languages and speakers.

3.5.2 I-vector Extraction

The i-vector model is an unsupervised generative model derived from the joint factor analysis (JFA) framework [38]. It assumes that every utterance is generated by a different GMM that represents its distribution of short-term acoustic features, an idea first proposed for GMM-supervectors [60]. The i-vector is a compact representation of the parameters of such GMM supervectors and efficiently captures the inter- and intra-dependencies across Gaussian components.

The i-vector is formulated as follows: First, the prior distribution of a generality of short-term acoustic features is modelled by a single GMM model using an expectation-maximization algorithm (EM). This initial GMM model is known as a universal background model (UBM). Utterance-specific supervectors are then obtained by a maximum a posteriori (MAP) estimate of the mean of a posterior distribution computed over the UBM [38]. The relationship between the utterance supervector **m** and the utterance i-vector **w** is formulated as:

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{3.1}$$

where **u** is the mean GMM supervector and **T** is a low-rank rectangular matrix representing the bases spanning the sub-space, which contains most of the variability in the supervector space.

It is worth mentioning that equation 3.1 simply denotes the mean of the distribution of $p(M|w)$ imposed by the i-vector model. However, during i-vector estimation, it is frequently more useful to formulate $p(w|M)$. The Bayes rule estimation of $p(w|M)$ for two types of priors of $p(w)$ is detailed in [40]. The i–vector $\mathbf{w}$ is then a MAP estimate of a low-dimensional latent variable. The estimation algorithm for the low-rank matrix $\mathbf{T}$ is presented in [38].

Typical values for this approach range from the use of $a = 1024\ldots2048$ Gaussian components for the GMM models, $b = 400\ldots600$ dimensional i-vectors, and $c = 24\ldots56$ dimensional features (MFCC or SDC). The total number of parameters of an i-vector system can be estimated by $ac(b+2)$, which means that i-vector models often require $10\ldots70$ millions of parameters, a relatively large number for small embedded devices.

3.5.3 Classification Backends

The i-vector is therefore a compact representation of a whole utterance that is derived as a point estimate of the latent variables in a factor analysis model [13, 40]. However, i-vector-based systems still require a supervised backend classifier to accomplish LID or SV.

The simplest of those backends is a non-parametric approach that uses the cosine kernel between the test i-vector $\mathbf{w}$ and the mean language, or speaker i-vector $\mathbf{w}'$ [13]. The resulting value is then compared to a threshold in order to make the final decision.

Recently, more sophisticated backends have shown to better deal with unwanted variability factors affecting performance. In particular, classical approaches for LID typically use the Gaussian backend [52], whereas probabilistic linear discriminant analysis (PLDA) [17, 59] is more frequently used for SV problems.

The Gaussian backend used for LID characterizes the i-vectors of each language by estimating a single Gaussian distribution via maximum likelihood, where the covariance matrix is shared among languages and is equal to the within-class covariance matrix of the training data. During evaluation, every new utterance is evaluated against the model of all candidate languages.

Alternativelly, the PLDA backend used in SV problems can be seen as a special unimodal case of joint factor analysis (JFA) [38] that takes a combination of two i-vectors as the input and directly evaluates the likelihood ratio between the two verification hypotheses (i.e., whether the two i-vectors were generated by the same speaker or not).

3.6 Limitations of Classical Approaches

While proven to be successful in a variety of scenarios, i-vector-based approaches suffer from four major drawbacks when used in the context of real-world applications:

1. In real-time applications, most of the costs associated with i-vector computation occur after completion of the utterance, which introduces an undesirable latency to the application's user. This latency effect is particularly problematic when the application requires an implementation embedded in a portable device with relatively small computational power.

2. Research largely develops assuming the existence of long segments of audio. In some applications, however, it is difficult to find collaborating speakers willing to speak for more than a few seconds in a systematic way. Short utterances, despite being easier to collect, have a particularly negative effect on the i-vector estimation (which is a point estimate) and its robustness quickly degrades as the amount of data used to derive it decreases.

3. I-vector systems cope with incomplete assumptions about the speech data's representation, including the assumption of pseudo-independence of neighbouring segments of speech (even for segments as close together as a few tens of milliseconds) and the assumption that the distribution of the time-frequency information of the speech signal can be compactly estimated with a GMM with a finite number of components with diagonal covariance matrices [33].

4. I-vector systems break down LID and SV problems into more tractable, but loosely connected subproblems or stages, which includes the computation of the i-vector and backend models and the extraction of hand-crafted features. These problems are therefore not jointly optimized to reduce the same metrics as those used during testing.

3.7 The Role of Classical Approaches in this Dissertation

In this thesis, the author shows that SV and LID systems can rely on ANNs to overcome the limitations that exist in classical i-vector-based approaches. Performance, latency and model size benchmarks were used to conduct side-by-side comparisons between classical and ANN-based approaches using large-scale public and private datasets. Benchmarks used include i-vector systems that are followed by the cosine distance backend to model languages in [23, 24, 45], and speakers in [31, 71]. A variety of Gaussian backends were developed for LID in [44, 45], whereas PLDA was used for speaker recognition in [31].

Additionally, this thesis explores how novel ANN systems and classical approaches can complement each other by combining independently-leveraged discriminant information. For example, score-level fusion was used for SV in [71] and score-level fusion and calibration was used for LID in [24]. In [44], the author proposed using ANNs as a sophisticated, trainable feature extractor whose sequence of outputs can be used to replace or augment SDC features for classical systems. In [67], i-vectors are used as an additional input signal to ANNs used for speech recognition to facilitate speaker, channel and background normalization.

Finally, novel classification backends for classical LID systems were proposed in [46] and scoring optimization techniques for large-scale speaker recognition were proposed in [65].

Chapter 4

Artificial Neural Networks for Speaker and Language Recognition

In Chapter 3, we reviewed the definition of LID and SV and the classical approaches used to address these challenges. This chapter provides a rationale for why ANNs are a potential alternative to these classical approaches and presents the most relevant approaches to modeling LID and SV using ANNs.

4.1 ANN-based and GMM-based Models

Most current SV and LID systems use some form of hand-crafted features, such as MFCC or SDC, followed by some form of GMM- and i-vector-based representation of an utterance's features. Recent findings in the fields of speech recognition and deep learning have shown that remarkable accuracy improvements can be achieved through the use of modern ANNs and low-level mel-filterbank features, which replace GMM schemes trained with, for example, MFCC features [33]. One interpretation of these improvements is that ANNs may be considered universal approximators that, using a single algorithm, can holistically optimize the tasks of feature extraction and classification [10] (Figure 4.1).

In [33], additional advantages of ANNs, as compared to GMM models, have been listed. Among them: a) Multilayer ANNs use a multi-level distributed representation of the input, which makes the ANN an exponentially more compact model than a GMM model and allows the modeling of larger input examples, with additional contextual information, without incurring a linear increment of the number of parameters; b) ANNs have the potential to create better models of data that lie on or near a non-linear manifold; c) ANNs do not require detailed assumptions about the input data's distribution [54]; d) Using computer

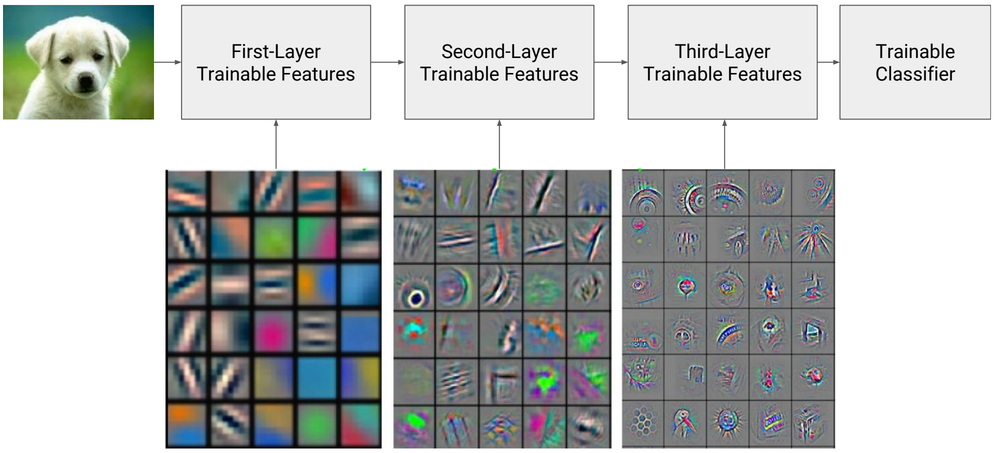


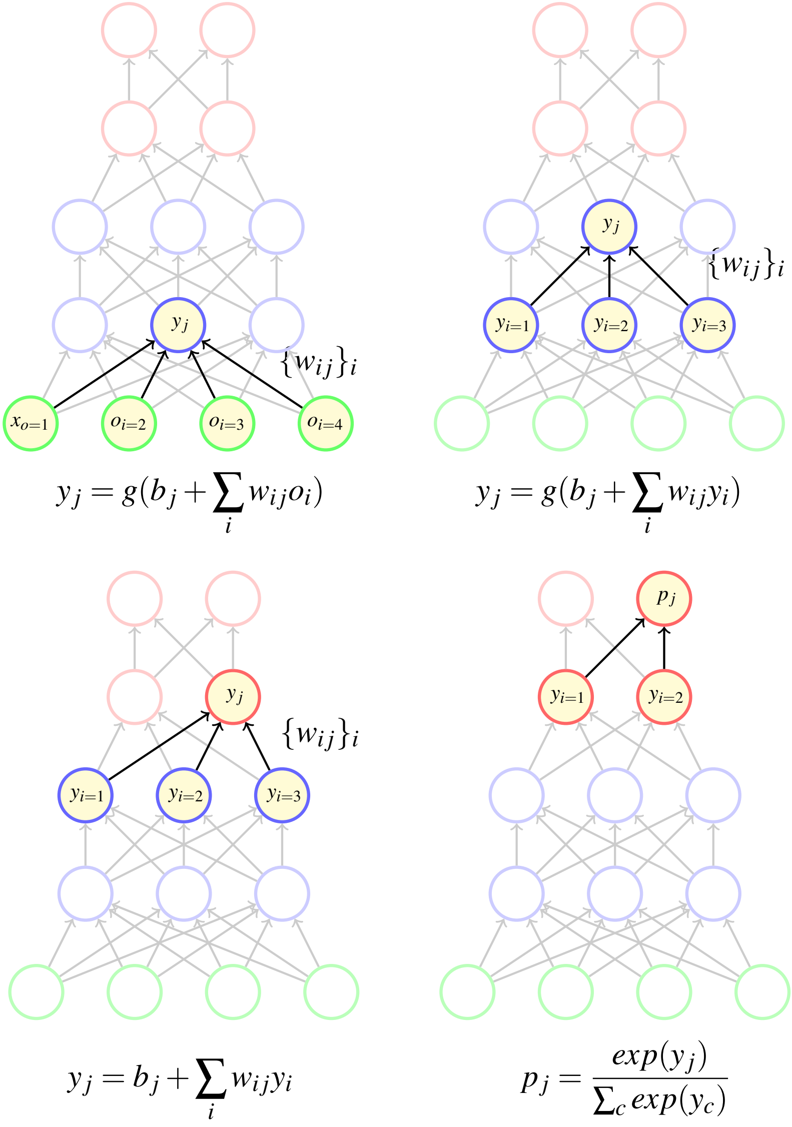
Fig. 4.1 Example of feature visualization produced by a network trained on Imagenet. Figure adapted from [78].

scale and regularization, ANNs can successfully exploit large amounts of data, resulting in more robust models without lapsing into overtraining; and e) ANNs have shown to be robust to low-precision quantization of weights, a technique used to reduce a model's footprint [53].

These are particularly interesting features to be considered when developing real-time, embedded architectures and have motivated the use of ANNs as an alternative to GMM models for LID and SV. Next, we describe the novel ANN-based architectures developed by the author for practical implementation.

4.2 Deep Neural Networks

The term *Artificial Neural Network* refers to the system formed from the interconnections between simple mathematical units, often called nodes or neurons. ANNs admit a recurrent formulation where each unit j computes a function $f : \{y_i\}_i \mapsto y_j$ that outputs an activation y_j from the activations received from other units $\{y_i\}_i$, where i is an index that iterates over the received connections. Typically, $f(\cdot)$ is defined in terms of a weighted sum of the activations of other units that output an activation, which is a nonlinear function $g(\cdot)$ of that sum (top of Figure 4.2).



$$y_j = g(b_j + \sum_i w_{ij} o_i)$$

$$y_j = g(b_j + \sum_i w_{ij} y_i)$$

$$y_j = b_j + \sum_i w_{ij} y_i$$

$$p_j = \frac{exp(y_j)}{\sum_c exp(y_c)}$$

Fig. 4.2 A depiction of the set of operations computed by a DNN during inference time. An input layer (green) provides external features $\{o_i\}_{i=1...4}$. The DNN has two hidden layers and two nodes per layer (blue) computing activations y, and a softmax classifier (red) that computes the predicted probabilities p over two possible classes. i is an index over the input nodes and j is a label of the current node. There is a one-to-one map between nodes and classes in the output layer. The softmax classifier is divided into two layers according to equations 4.1, 4.2 and 4.4.

$$y_j = g(x_j) \tag{4.1}$$

$$x_j = b_j + \sum_i w_{ij} y_i \tag{4.2}$$

The terms in $\{w_{ij}\}$ refer to the weights associated with the incoming activations received at unit j, whereas b_j is a bias term associated with that unit. An example of a nonlinear activation function $g(\cdot)$ is the rectified linear units (ReLu) function, where $ReLU(x) = max(0,x)$. ReLU is one of the most frequently used nonlinear activation functions in speech recognition algorithms [11].

Typically, these units are hierarchically arranged in layers, where each node in a layer receive its inputs $\{y_i\}_i$ from units in the previous layer. The input layer receives its inputs from provided features $\{o_i\}_i$, such as the filter-bank energies of one frame of speech. If each node in the layer l receives inputs from all the units in the layer $l-1$, then the layer l is referred to as a fully-connected layer. Other alternatives are listed in section 4.3.

These networks can be trained to approximate a desired output function by back-propagating the gradients of an error function [26]. Such error functions, known as fitting functions, are the result of comparing the network output to a desired target value t_j provided for each training input example. An example of a fitting function is the cross-entropy loss frequently used in identification problems:

$$C = -\sum_c t_c \log p_c \tag{4.3}$$

Here, c is an index over all of the targets, p_c is the estimated probability for the target class c, and t_c represents the desired target probability for the current evaluated example, taking a value of either 1 (true class) or 0 (false class). The cross-entropy optimization involves minimizing the negative log likelihood of the training set and, at the same time, reducing the Kullback–Leibler divergence between the distributions t and p. At its optimal value (when $t = p$), equation 4.3 is referred to as the entropy function, which is associated with the optimal bit encoder used in information theory [26].

One common way to map hidden units into probabilities, as required by the cross-entropy function (Equation 4.3), is to configure the output layer as a *softmax* layer where:

$$p = \frac{exp(x)}{\sum_c exp(x_c)} \tag{4.4}$$

A softmax layer is an exponential model that can also be formulated as a two-layer model that follows equations 4.1 and 4.2, where the first layer is a linear layer such that $g(x) = x$

and, in the second layer, $g(\cdot)$ is computed according to equation 4.4 (Figure 4.2). It is worth noting that, in the softmax output layer, there is a one-to-one correspondence between nodes and classes ($p_j = y_j$).

ANNs defined in this framework are known as feed-forward ANNs, whereas DNNs are feed-forward ANNs with more than one hidden layer between the input and output layers.

4.3 Alternative DNN Architectures

Modern DNNs are used extensively for the acoustic modeling of speech recognizers [33] and various other tasks [41, 43, 66, 76]. However, not all DNNs follow the formulation presented in Section 4.2, as DNNs are among some of the most heterogeneous pattern recognition models.

Several possible alternative architectures include: 1) cases where DNNs admit different nonlinear activation functions (Equation 4.1), such as logistic sigmoid functions, linear functions, and hyperbolic tangent functions [48]; 2) different types of hidden layers (Equation 4.2), such as space invariant convolutional layers generally used for image and video processing [41, 43], dropout and maxout layers [27] used for regularized training, and the size-efficient locally connected layers introduced in this thesis [9]; and 3) different types of fitting functions and output layers (Equations 4.3 and 4.4), such as restricted Boltzmann machines [32] used for unsupervised problems, the triplet loss used for clustering [66], the MMI optimization criteria used for sequence training [72], the logistic layer used for regression problems [26], and the end-to-end layer used for verification and also introduced in this thesis [31]. There are also multiple possible modifications of the standard back-propagation algorithm, some of which aim to optimize the learning rate, the algorithm parallelization and the estimation of gradients [26].

4.4 Recurrent Neural Networks

One limitation of feed-forward ANNs is that they assume that input examples are *i.i.d.* variables. Given this assumption, feed-forward ANNs may fail to model the long-time dependencies of time sequences. Recurrent Neural Networks (RNNs) are an attempt to address this issue. RNNs define a family of ANNs with recurrent, hidden self-connections that store the temporal state of the network, which changes with the input to the network at each time step (Figure 4.3). Such recurrent connections provide a powerful mechanism to model temporal sequences, such as the speech signal and its complex long-range correlations.

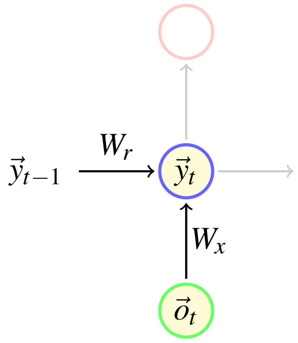


Fig. 4.3 Depiction of a recurrent neural network.

For simplicity, we will now use the vectorial notation to refer to the fact that observations $\vec{o}$ and outputs $\vec{p}$ are multidimensional, and that multiple nodes can be present in each layer. With this notation, RNNs compute a mapping from an input sequence of observations $(\vec{o}_1, ..., \vec{o}_T)$ to an output sequence $(\vec{p}_1, ..., \vec{p}_T)$. In its simplest form, the RNN iteratively calculates from time step $t = 1$ to T, the activations for network units, according to the following equation:

$$\vec{y}_t = g(W_r\vec{y}_{t-1} + W_x\vec{o}_t + \vec{b}) \tag{4.5}$$

$$\vec{p}_t = \phi(W_y\vec{r}_t + b_y) \tag{4.6}$$

where $\vec{y}_0$ is a vector typically initialized with zeros; $g(\cdot)$ is either the ReLu or the hyperbolic tangent activation function; and ϕ is the softmax output activation function (Equation 4.4). As in the case of DNNs, RNNs can also be arranged in multilayer "deep" architectures where the input to each node in a layer is the vector of activations $\vec{y}$ of the layer below.

RNN models are typically trained using some version of the back-propagation through time (BPTT) learning algorithm [62, 63, 75], where the RNN is transformed into a feed-forward ANN as deep as the number of elements in the sequence. This type of process is known as *unrolling* the RNN. This unrolled structure of RNNs can be helpful in illustrating how RNNs are used to model different types of problems involving sequences and when using deep architectures (Figure 4.4).

4.5 Long Short-Term Memory Networks

Long short-term memory networks (LSTM) are a type of RNNs [18, 19, 35] that have recently been shown to outperform the state-of-the-art of feed-forward DNN systems for

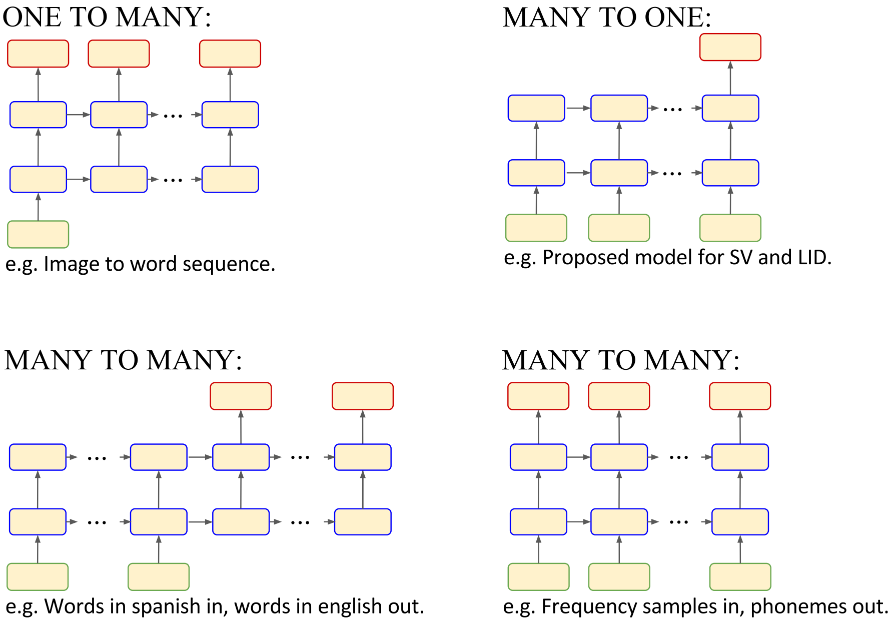

ONE TO MANY:
e.g. Image to word sequence.
MANY TO ONE:
e.g. Proposed model for SV and LID.
MANY TO MANY:
e.g. Words in spanish in, words in english out.
MANY TO MANY:
e.g. Frequency samples in, phonemes out.

Fig. 4.4 Illustration of unrolled RNN networks used in multiple applications.

acoustic modeling in LVCSR [28, 62]. LSTM have special network units called *memory blocks* in the recurrent hidden layer that make them a more powerful tool to model sequence data than feed-forward neural networks and conventional RNNs. The memory blocks contain memory cells with self-connections storing the temporal state of the network, in addition to special multiplicative units called *gates* to control the flow of information.

The modern LSTM RNN architecture [18, 19, 35] is shown in Figure 4.5. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the rest of the network. The forget gate scales the internal state of the cell before adding it as input through self-recurrent connection to the cell, therefore adaptively forgetting or resetting the cell’s memory. In addition, the LSTM RNN architecture contains "peephole connections" that connect the internal cells to the gates in the same cell to learn the precise timing of the outputs [19]. With this architecture, LSTM RNNs compute a mapping from the input sequence to an output sequence according to the following set of operations:

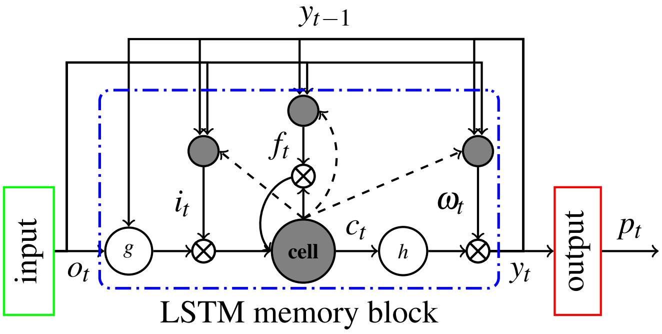


Fig. 4.5 Depiction of a single member block of a long short-term memory recurrent neural network architecture.

$$\vec{i}_t = \sigma(W_{ix}\vec{o}_t + W_{ir}\vec{y}_{t-1} + W_{ic}\vec{c}_{t-1} + \vec{b}_i) \tag{4.7}$$

$$\vec{f}_t = \sigma(W_{fx}\vec{o}_t + W_{fr}\vec{y}_{t-1} + W_{fc}\vec{c}_{t-1} + \vec{b}_f) \tag{4.8}$$

$$\vec{c}_t = \vec{f}_t \odot \vec{c}_{t-1} + \vec{i}_t \odot tanh(W_{cx}\vec{o}_t + W_{cr}\vec{y}_{t-1} + \vec{b}_c) \tag{4.9}$$

$$\vec{\omega}_t = \sigma(W_{ox}\vec{o}_t + W_{or}\vec{y}_{t-1} + W_{oc}\vec{c}_t + \vec{b}_o) \tag{4.10}$$

$$\vec{y}_t = \vec{\omega}_t \odot tanh(\vec{c}_t) \tag{4.11}$$

where the W terms denote weight matrices (e.g., W_{ix} is the matrix of weights from the input gate to the input); W_{ic}, W_{fc}, W_{oc} are diagonal weight matrices for peephole connections; the b terms denote bias vectors (b_i is the input gate bias vector); σ is the logistic sigmoid function; and $\vec{i}$, $\vec{f}$, $\vec{\omega}$ and $\vec{c}$ are the input gate, forget gate, output gate and cell activation vectors, respectively, all of which are the same size as the cell output activation vector $\vec{y}$; $\odot$ is the element-wise product of the vectors; and *tanh* is the hyperbolic tangent activation function for cell inputs and cell outputs.

In [34], it is shown that the BPTT optimization algorithm is a more stable algorithm for training LSTM networks than conventional RNN networks. LSTM's gating strategy does not suffer from the so-called "vanishing gradient" problem, which in conventional RNNs causes the value produced from activation functions to shrink rapidly or explode, depending largely on the length of the input sequence. This is a feature that makes LSTM networks a good candidate model for LID and SV problems with input sequences of arbitrary length.

Chapter 5

Original Contributions

Previous chapters have provided an introduction to the concepts most relevant to this thesis, including LID (Section 3.3), SV (Section 3.4), classical approaches to LID and SV (Section 3.5), DNNs (Section 4.2) and LSTMs (Section 4.5). This chapter summarizes the author's contributions to LID and SV problems, building on previous references and linking to the author's most relevant publications.

5.1 Contributions to LID Problems Using ANNs

The author's contributions to LID are focused on the development of language recognition technology with applications for multilingual continuous LVCSR systems, as well as speech-to-speech translation systems.

[23, 45] introduce novel discriminative approaches for addressing LID problems using DNNs, the architecture of which is depicted in figure 5.1. We demonstrate that DNNs are particularly suitable for performing LID in the context of real-time applications, due to their capacity to emit language posteriors at each new frame of the test utterance (Figure 5.3). Furthermore, combining frame-level posteriors into utterance-level scores leads to remarkable improvements for short utterances when compared to state-of-the-art approaches (Figure 5.2). Finally, we analyse various key aspects of the system, such as the amount of required training data, the number of hidden layers, the significant relevance of contextual information and the effect of test utterance duration.

Similarly, [24] presents a novel approach for addressing LID using LSTM, which is motivated by their better ability to model sequences with respect to the feed-forward networks used in previous works. We show that LSTM RNNs can effectively achieve better performance than the best DNN system, with an order of magnitude fewer parameters, and found that a *many to one* architecture was critical for model convergence (Figure 4.4). Table 5.1

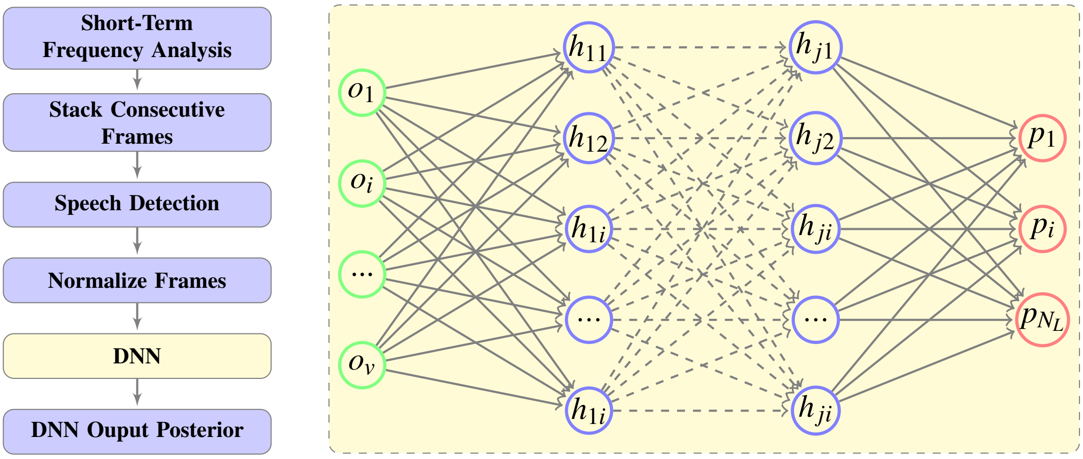


Fig. 5.1 Example of a DNN model used for LID (right) and the pipeline process from the waveform to the DNN output posteriors (left). The input (i.e., visible units) of the DNN is a fixed-size vector $o \in \mathbb{R}^v$. The output (i.e., output units) is a vector $p \in \mathbb{R}^{N_l}$ with the predicted values of probability for each language. There are $j * i$ hidden units arranged in j layers. DNN outputs are computed every 10ms over a sliding window of time-frequency coefficients and further combined into a final score for the complete utterance [23].

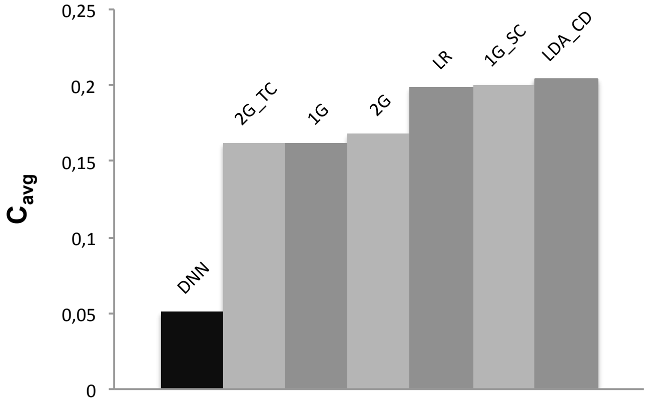


Fig. 5.2 Performance comparison of the DNN-based LID system and multiple i-vector-based systems using state-of-the-art Gaussian backends. All system are evaluated on a test set formed by audio logs sampled from Google speech recognition services in 34 languages. Systems are ranked according to the C_{avg} (average cost) error metric defined in NIST [7, 57]. Figure extracted from [45].

summarizes the performance of DNN- and LSTM-based systems for models of different sizes.

In [44], we propose an architecture where the DNN is used to extract bottleneck features that are then used as inputs for a state-of-the-art i-vector system. This DNN-based system

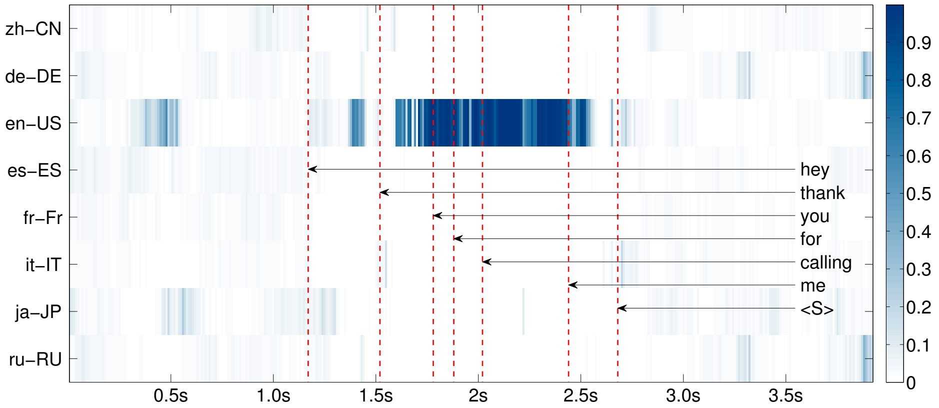


Fig. 5.3 Frame-level probabilities of a DNN-based LID system (8 languages selected) evaluated over an English-USA (4s) test utterance. DNN outputs are computed every 10ms over a sliding window of time-frequency coefficients and further combined into a final score for the complete utterance [23].

	Size	$EER_{avg}(\%)$	C_{avg}
i-vector (Cosine Backend)	∼16M	15.89	0.1968
DNN_2_layers	∼8M	11.61	0.1727
DNN_4_layers	∼21M	8.79	0.1292
DNN_8_layers	∼48M	9.58	0.1376
LSTM RNN	∼**1M**	8.35	0.0944
Fusion	∼94M	**6.47**	**0.0649**

Table 5.1 LID performance for a subset of the LRE'09 evaluation (3s test segments, 8 languages) [57]. Systems are ranked according to the equal error rate averaged per language (EER_{avg}) and the average cost (C_{avg}) defined in NIST [7, 57]. We show that LSTM networks can achieve equal or better performance than DNN models that are one order of magnitude larger, and can achieve much better performance than classical i-vector systems. Best performance is achieved by combining results from all previous systems. Table adapted from [24].

is shown to outperform other state-of-art systems when dealing with utterances of different lengths (Figure 5.2).

Finally, in [21], we build on previous advances using ANNs for LID to present an end-to-end multi-language LVCSR architecture developed and deployed at Google that allows users to select arbitrary combinations of spoken languages. We leverage their previous advances in LID and a novel method of real-time language selection to achieve similar recognition accuracy and nearly identical latency characteristics as a monolingual system.

	$EER_{avg}(\%)$		
	03 seconds	10 seconds	30 seconds
i-vector (Gaussian Backend)	15.74	5.30	2.33
DNN_4_layers	13.49	6.38	4.37
Bottleneck	13.52	5.58	3.24
i-vector + DNN_4_layers	11.93	3.80	1.94
i-vector + Bottleneck	**11.19**	**3.61**	**1.85**

Table 5.2 LID performance for the full LRE'09 evaluation [57]. Results are sorted by test duration. Systems are ranked according to the equal error rate averaged per language (EER_{avg}). We show that for 30s-long utterances, the i-vector system tends to outperform the best DNN, while the combination of i-vector and bottleneck systems outperforms all other approaches. Table adapted from [44].

5.2 Contributions to SV Problems Using ANNs

The author's contributions to SV are focused on the development of speaker recognition technology that has applications for continuous voice authentication systems used in portable devices.

[71] presents a novel approach to addressing SV problems using DNNs; in this case, for a small footprint text-dependent speaker verification task. In the development stage, a DNN is trained to classify speakers at the frame level and, during speaker enrollment, the trained DNN is used to extract speaker-specific features from the last hidden layer. The average of these speaker features (or d-vector) is taken as the speaker model. At the evaluation stage, a d-vector is extracted for each utterance and compared to the enrolled speaker model to make a verification decision using the cosine distance backend. Initial experimental results conducted in [71] show that the performance of the d-vector SV system is reasonably good compared to an i-vector system, and further gains can be achieved by combining both systems. Furthermore, experiments conducted in [31] have shown that, when additional training data is available, a d-vector system similar to the one presented here (referred to as DNN/sofmax in Table 5.4), can achieve more than 15% better performance than a state-of-the-art i-vector/PLDA system.

In [9], we exploit locally-connected networks (LCN) and convolutional neural networks (CNN) to better model the local time-frequency correlations of the speech signal as an alternative to the fully-connected DNN used in previous work. We demonstrate that both an LCN and CNN can reduce the total model footprint and latency to 30% of the original size of the model presented in [71], with minimal impact on performance (Table 5.3). This reduction in the model size and latency allowed for the deployment of the first voice biometric unlock mechanism for Android.

	Size	Multiplies	EER(%)
Fully Connected	787k	787k	**3.88**
LCN	**234k**	**234k**	4.02
CNN	**234k**	345k	4.04

Table 5.3 Performance comparison of fully-connected, LCN and CNN approaches used for the first hidden layer of a text-dependent SV system. We show that LCN and CNN can reduce the total model footprint and latency (multiplies) to 30% of the original value, with only a small impact on performance. Table adapted from [9].

SV system	Size	EER(%)
i-vector (Cosine Backend) [13, 71]	12M	5.77
i-vector (PLDA Backend) [17]	12M	4.66
DNN, softmax [9]	813k	3.86
DNN, end-to-end [31]	813k	1.87
LSTM, end-to-end [31]	1M	**1.36**

Table 5.4 A comparison of the performance of the end-to-end models and other technologies used for text-dependent SV. We show that end-two-end optimization can largely outperform previously used approaches and is particularly useful in combination with *many to one* LSTM models (Figure 4.4). Table adapted from [31].

Finally, in [31], we define a new fitting function for verification problems and use it to optimize both DNN and LSTM models, namely *end-to-end* optimization. Using this fitting function, we define an integrated approach to SV that maps a test utterance and a few enrollment utterances directly to a single score for verification, using the same evaluation protocol as at test time. Such an approach results in simple and efficient systems that require little domain-specific knowledge, require few model assumptions and result in remarkable performance improvements compared to other state-of-the-art approaches with a similar or larger number of parameters (Table 5.4.)

5.3 Contributions to Classical Approaches.

The author's contributions to classical SV and LID approaches are focused on the application of i-vector models to the speech recognition domain, as well as on developing new backend systems with applications for LID and SV.

[67] shows that using the utterance i-vectors as input features to a DNN used for speech recognition provides the network with valuable information that brings about a roughly 4% relative reduction in word error rate (WER). This technique can be applied on any utterance, without requiring any speaker information or speaker adaptation or model storage. The technique has been shown to combine well with model adaptation, delivering an overall 9% WER reduction for models that are small enough to be run in real-time on portable devices, making this an ideal technique for speaker-adapted models.

The work in [46] proposes novel classification backends for classical LID systems using i-vector features. The backends use the von Mises-Fisher distribution to assign language conditioned probabilities to language data. The two proposed methods use kernel density functions and mixture models. The experiments show that von Mises-Fisher mixture models can outperform the baseline cosine backend for short-duration utterances by 25% for most of the considered experimental conditions.

Finally, in [65], we propose a fast retrieval method for SV in large data sets. The research is based on combining two approaches that interact via the cosine distance: locality sensitive hashing (LHS), which enables fast nearest neighbor search, and i-vectors, which provide good identification accuracy. Results on a realistic, large data set from YouTube show that speedups of one to two orders of magnitude can be achieved, while sacrificing only a small fraction of the identification accuracy. This approach is a promising candidate for large-scale SV and could also be useful for other large-scale clustering applications of i-vectors or d-vectors.

Chapter 6

Conclusions and Future Work

The aim of this thesis was to advance the state-of-the-art in speaker and language recognition technologies for application in large-scale and portable speech interfaces. This research was motivated by recent findings in the fields of speech recognition and deep learning (Chapter 4), and showed that holistic optimizations can overcome the incomplete assumptions of classical, prior knowledge-based approaches to language and speaker recognition (Section 3.6). The author formulated, implemented and conducted empirical analyses of novel deep learning approaches for language and speaker recognition, with the aim of addressing an important gap in the literature (Chapter 5). These novel approaches were optimized and tested in the context of several new speech interfaces that were being developed and launched at Google, including a multilingual large vocabulary continuous speech recognition system, a speech-to-speech translation system, and an always-on voice authentication system for portable devices. It was largely thanks to the improvements that resulted from this research that the performance, footprint and latency needs of these new speech interfaces were satisfied.

This research demonstrates that feed-forward deep neural networks (DNNs) [23, 45] and long short-term memory networks (LSTMs) [24] directly trained to discern languages can obtain significantly improved results (as compared to classical i-vector-based systems) when working with short-duration utterances. The proposed architectures are able to generate an accurate local decision regarding the language spoken in every single frame, with no perceivable latency. The research also demonstrated that straight-forward, incremental decisions can be made at any point during an utterance, making this approach particularly suitable for multilingual speech recognition systems that incrementally combine and emit results from individual language and speech recognizers [21]. In the case of applications involving utterances over ten seconds in length, a bottleneck-based combination of ANNs and i-vector systems was found to provide additional gains [44].

In the domain of speaker recognition, this research shows that DNNs can successfully be used to map utterances directly to vectors of hidden variables–namely d-vectors–which can be used for classifying speakers with simple classification backends [71]. The d-vector model is different from the classical i-vector model because it follows a speaker discriminant optimization process with fewer assumptions about the underlying data representation. In addition, the research shows that locally connected networks substantially reduce the model footprint and latency by exploiting the exponentially compacted representation of the DNN model parameters [9]. Finally, the research summarized in [31] shows that the novel end-to-end loss function presented for verification problems can jointly optimize all components involved in the d-vector model by using a single algorithm that, during training time, optimizes the same metric as that used for evaluation. We tested the proposed end-to-end loss function for the internal ”Ok Google” benchmark, showing that both DNN and LSTM models can achieve remarkable improvements over all current state-of-the-art approaches.

Future research should be conducted to develop additional architectures and optimization functions that can efficiently capture the long-term information of long (up to 10-minute) speech segments. Promising initial results have already been seen when using deep LSTM models, which could be further combined with bidirectional LSTMs [29] and attention models [77]. Additionally, research is recommended to investigate the use of the end-to-end optimization function to solve verification problems with inputs of variable length, such as the problems of text-independent speaker verification, speaker clustering, and diarization [6]. Finally, research is recommended to develop scalable multilingual systems that can provide multilingual transcriptions of utterances involving two or more languages spoken in the same sentence.

References

[1] Ambikairajah, E., Li, H., Wang, L., Yin, B., and Sethu, V. (2011). Language identification: A tutorial. *Circuits and Systems Magazine, IEEE*, 11(2):82–108.

[2] Aronowitz, H., Hoory, R., Pelecanos, J. W., and Nahamoo, D. (2011). New developments in voice biometrics for user authentication. In *INTERSPEECH*, pages 17–20.

[3] Bahari, M. H., McLaren, M., van Leeuwen, D. A., et al. (2014). Speaker age estimation using i-vectors. *Engineering Applications of Artificial Intelligence*, 34:99–108.

[4] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[5] Bishop, C. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition.

[6] Bredin, H. (2016). Tristounet: Triplet loss for speaker turn embedding. *arXiv preprint arXiv:1609.04301*.

[7] Brümmer, N. (2010). *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, Citeseer.

[8] Chen, G., Parada, C., and Heigold, G. (2014). Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091. IEEE.

[9] Chen, Y.-h., Lopez-Moreno, I., Sainath, T., Visontai, M., Alvarez, R., and Parada, C. (2015). Locallyconnected and convolutional neural networks for small footprint speaker recognition. *Interspeech, Dresden, Germany*.

[10] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[11] Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613. IEEE.

[12] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366.

[13] Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

[14] Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. John Wiley and Sons Inc, New York City, New York, USA.

[15] Farrell, K. R., Mammone, R. J., and Assaleh, K. T. (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on speech and audio processing*, 2(1):194–205.

[16] Ferrer, L., Scheffer, N., and Shriberg, E. (2010). A Comparison of Approaches for Modeling Prosodic Features in Speaker Recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 4414–4417.

[17] Garcia-Romero, D. and Espy-Wilson, C. Y. (2011). Analysis of i-vector length normalization in speaker recognition systems. In *Interspeech*, pages 249–252.

[18] Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

[19] Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2003). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143.

[20] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

[21] Gonzalez-Dominguez, J., Eustis, D., Lopez-Moreno, I., Senior, A., Beaufays, F., and Moreno, P. J. (2015a). A real-time end-to-end multilingual speech recognition architecture. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):749–759.

[22] Gonzalez-Dominguez, J., Lopez-Moreno, I., Franco-Pedroso, J., Ramos, D., Toledano, D., and Gonzalez-Rodriguez, J. (2010). Multilevel and Session Variability Compensated Language Recognition: ATVS-UAM Systems at NIST LRE 2009. *Selected Topics in Signal Processing, IEEE Journal of*, 4(6):1084–1093.

[23] Gonzalez-Dominguez, J., Lopez-Moreno, I., Moreno, P. J., and Gonzalez-Rodriguez, J. (2015b). Frame-by-frame language identification in short utterances using deep neural networks. *Neural Networks*, 64:49–58.

[24] Gonzalez-Dominguez, J., Lopez-Moreno, I., Sak, H., Gonzalez-Rodriguez, J., and Moreno, P. J. (2014). Automatic language identification using long short-term memory recurrent neural networks. In *INTERSPEECH*, pages 2155–2159.

[25] Gonzalez-Rodriguez, J., Rose, P., Ramos, D., Toledano, D. T., and Ortega-Garcia, J. (2007). Emulating dna: Rigorous quantification of evidential weight in transparent and testable forensic speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2104–2115.

[26] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[27] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. (2013). Maxout networks. *ICML (3)*, 28:1319–1327.

[28] Graves, A., Jaitly, N., and Mohamed, A. (2013a). Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

[29] Graves, A., Jaitly, N., and Mohamed, A.-r. (2013b). Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

[30] Grosjean, F. (2010). *Bilingual: Life and Reality*. Harvard University Press.

[31] Heigold, G., Moreno, I., Bengio, S., and Shazeer, N. (2016). End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119. IEEE.

[32] Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926.

[33] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[34] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

[35] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[36] Jain, A., Flynn, P., and Ross, A. A. (2007). *Handbook of biometrics*. Springer Science & Business Media.

[37] Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841.

[38] Kenny, P. (2005). Joint factor analysis of speaker and session variability: Theory and algorithms. *CRIM, Montreal,(Report) CRIM-06/08-13*.

[39] Kenny, P. (2010). Bayesian speaker verification with heavy-tailed priors. In *Odyssey*, page 14.

[40] Kenny, P., Oullet, P., Dehak, N. Gupta, V., and Dumouchel, P. (2008). A Study of Interspeaker Variability in Speaker Verification. *IEEE Trans. on Audio, Speech and Language Processing*, 16(5):980–988.

[41] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[42] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

[43] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[44] Lopez-Moreno, I., Gonzalez-Dominguez, J., Martinez, D., Plchot, O., Gonzalez-Rodriguez, J., and Moreno, P. J. (2016). On the use of deep feedforward neural networks for automatic language identification. *Computer Speech and Language*, 40:46 – 59.

[45] Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martinez, D., Gonzalez-Rodriguez, J., and Moreno, P. (2014). Automatic language identification using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5337–5341. IEEE.

[46] Lopez-Moreno, I., Ramos, D., Gonzalez-Dominguez, J., and Gonzalez-Rodriguez, J. (2011). Von mises–fisher models in the total variability subspace for language recognition. *IEEE Signal Processing Letters*, 18(12):705–708.

[47] Lopez-Otero, P., Docio-Fernandez, L., and Garcia-Mateo, C. (2014). ivectors for continuous emotion recognition. *Training*, 45:50.

[48] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30.

[49] Markowitz, J. A. (2000). Voice biometrics. *Communications of the ACM*, 43(9):66–73.

[50] Martınez, D., Green, P., and Christensen, H. (2013). Dysarthria intelligibility assessment in a factor analysis total variability space. In *Proceedings of Interspeech*.

[51] Martinez, D., Lleida, E., Ortega, A., and Miguel, A. (2013). Prosodic features and formant modeling for an ivector-based language recognition system. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6847–6851.

[52] Martınez, D., Plchot, O., Burget, L., Glembek, O., and Matejka, P. (2011). Language recognition in ivectors space. *Proceedings of Interspeech, Firenze, Italy*, pages 861–864.

[53] McGraw, I., Prabhavalkar, R., Alvarez, R., Arenas, M. G., Rao, K., Rybach, D., Alsharif, O., Gruenstein, A., Parada, C., et al. (2016). Personalized speech recognition on mobile devices. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5955–5959. IEEE.

[54] Mohamed, A., Dahl, G., and Hinton, G. (2012). Acoustic Modeling using Deep Belief Networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14–22.

[55] Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20.

[56] Muthusamy, Y., Barnard, E., and Cole, R. (1994). Reviewing automatic language identification. *Signal Processing Magazine, IEEE*, 11(4):33–41.

[57] NIST (2009). The 2009 NIST SLR Evaluation Plan. www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf.

[58] Prabhavalkar, R., Alvarez, R., Parada, C., Nakkiran, P., and Sainath, T. N. (2015). Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4704–4708. IEEE.

[59] Prince, S. J. and Elder, J. H. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

[60] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41.

[61] Rose, P. (2006). Technical forensic speaker recognition: Evaluation, types and testing of evidence. *Computer Speech & Language*, 20(2):159–191.

[62] Sak, H., Senior, A., and Beaufays, F. (2014a). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *ArXiv e-prints*.

[63] Sak, H., Senior, A., and Beaufays, F. (2014b). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In *submitted to INTERSPEECH 2014*.

[64] Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Kamvar, M., and Strope, B. (2010). "your word is my command": Google search by voice: A case study. In *Advances in Speech Recognition*, pages 61–90. Springer.

[65] Schmidt, L., Sharifi, M., and Moreno, I. L. (2014). Large-scale speaker identification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1650–1654. IEEE.

[66] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

[67] Senior, A. and Lopez-Moreno, I. (2014). Improving dnn speaker independence with i-vector inputs.

[68] Torres-Carrasquillo, P., Singer, E., Gleason, T., McCree, A., Reynolds, D., Richardson, F., and Sturim, D. (2010). The MITLL NIST LRE 2009 Language Recognition System. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4994–4997.

[69] Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., Greene, R. J., Reynolds, D. A., and Deller Jr, J. R. (2002). Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *INTERSPEECH*.

[70] Tucker, G. (1999). *A Global Perspective on Bilingualism and Bilingual Education*. ERIC (Collection). ERIC Clearinghouse on Languages and Linguistics.

[71] Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-Dominguez, J. (2014). Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056. IEEE.

[72] Veselỳ, K., Ghoshal, A., Burget, L., and Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, pages 2345–2349.

[73] Waggoner, D. (1993). The Growth of Multilingualism and the Need for Bilingual Education: What Do We Know so Far? *Bilingual Research Journal*, 17(1-2):1–12.

[74] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

[75] Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for online training of recurrent network trajectories. *Neural Computation*, 2:490–501.

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

[77] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

[78] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer.

[79] Zissman, M. (1996). Comparison of Four Approaches to Automatic Language Identification of Telephone Speech. *IEEE Trans. Acoust., Speech, Signal Processing*, 4(1):31–44.

Appendix A

Full Text of Journal Papers

A.1 On the Use of Deep Feedforward Neural Networks for Automatic Language Identification

TITLE	**On the use of deep feedforward neural networks for automatic language identification.**	
JOURNAL	Computer Speech and Language.	
IMPACT-FACTOR	5-year 1.324, Q2	Computer Science
AUTHORS	Ignacio López Moreno	UAM / Google Inc.
	Javier González Domínguez	UAM
	David Martínez	Univ. of Zaragoza
	Oldřich Plchot	Brno Univ. of Technology
	Joaquin González Rodríguez	UAM
	Pedro J. Moreno	Google Inc.
DOI	10.1016/j.csl.2016.03.001	
DATE	2016/10	
PUBLISHER	Elsevier Science Publishers	Amsterdam, The Netherlands

Summary: This work focuses on improving the performance of Deep Neural Network (DNN) -based spoken language identification systems for utterances of arbitrary length. Here, a DNN is used as a sophisticated feature extractor, whose generated features are further modeled using a Bayesian approach. The proposed method is suitable for both short and long utterances.

Contributions: The candidate proposed the novel idea of using DNN-based language recognition models as sophisticated feature extractors and further modeled them using i-vectors; provided the necessary code to generate neural networks for language recognition using public databases; generated the candidate DNN models with optimized hyperparameters; computed inference results of the evaluation data; and coordinated the institutions involved.

Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 40 (2016) 46–59

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

On the use of deep feedforward neural networks for automatic language identification

Ignacio Lopez-Moreno [a,*], Javier Gonzalez-Dominguez [b], David Martinez [c], Oldřich Plchot [d], Joaquin Gonzalez-Rodriguez [b], Pedro J. Moreno [a]

[a] *Google Inc., New York, USA*
[b] *ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain*
[c] *I3A, Zaragoza, Spain*
[d] *Brno University of Technology, Brno, Czech Republic*

Received 30 October 2015; accepted 18 March 2016
Available online 6 May 2016

Abstract

In this work, we present a comprehensive study on the use of deep neural networks (DNNs) for automatic language identification (LID). Motivated by the recent success of using DNNs in acoustic modeling for speech recognition, we adapt DNNs to the problem of identifying the language in a given utterance from its short-term acoustic features. We propose two different DNN-based approaches. In the first one, the DNN acts as an end-to-end LID classifier, receiving as input the speech features and providing as output the estimated probabilities of the target languages. In the second approach, the DNN is used to extract bottleneck features that are then used as inputs for a state-of-the-art i-vector system. Experiments are conducted in two different scenarios: the complete NIST Language Recognition Evaluation dataset 2009 (LRE'09) and a subset of the Voice of America (VOA) data from LRE'09, in which all languages have the same amount of training data. Results for both datasets demonstrate that the DNN-based systems significantly outperform a state-of-art i-vector system when dealing with short-duration utterances. Furthermore, the combination of the DNN-based and the classical i-vector system leads to additional performance improvements (up to 45% of relative improvement in both EER and C_{avg} on 3s and 10s conditions, respectively).

Keywords: LID; DNN; Bottleneck; i-vectors

1. Introduction

Automatic language identification (LID) refers to the process of automatically determining the language of a given speech sample (Muthusamy et al., 1994). The need for reliable LID is continuously growing due to a number of factors, including the technological trend toward increased human interaction using hands-free, voice-operated devices and the need to facilitate the coexistence of multiple different languages in an increasingly globalized world (Gonzalez-Dominguez et al., 2014).

* Corresponding author at: Google Inc, 76 Ninth Ave. P.C. 10011, New York, NY. Tel.: 9174054991.
E-mail address: elnota@google.com (I. Lopez-Moreno).

http://dx.doi.org/10.1016/j.csl.2016.03.001

Driven by recent developments in speaker verification, current state-of-the-art technology in acoustic LID systems involves using i-vector front-end features followed by diverse classification mechanisms that compensate for speaker and session variabilities (Brummer et al., 2012; Li et al., 2013; Sturim et al., 2011). An i-vector is a compact representation (typically from 400 to 600 dimensions) of a whole utterance, derived as a point estimate of the latent variable in a factor analysis model (Dehak et al., 2011; Kenny et al., 2008). While proven to be successful in a variety of scenarios, i-vector based approaches have two major drawbacks. First, i-vectors are point estimates and their robustness quickly degrades as the duration of the utterance decreases. Note that the shorter the utterance, the larger the variance of the posterior probability distribution of the latent variable; and thus, the larger the i-vector *uncertainty*. Second, in real-time applications, most of the costs associated with i-vector computation occur after completion of the utterance, which introduces an undesirable latency.

Motivated by the prominence of deep neural networks (DNNs), which surpass the performance of the previous dominant paradigm, Gaussian mixture models (GMMs), in diverse and challenging machine learning applications – including acoustic modeling (Hinton et al., 2012; Mohamed et al., 2012), visual object recognition (Ciresan et al.), and many others (Yu and Deng, 2011) – we previously introduced a successful LID system based on DNNs in Lopez-Moreno et al.. Unlike previous works on using neural networks for LID (Cole et al., 1989; Leena et al., 2005; Montavon, 2009), this paper represented, to the best of our knowledge, the first time a DNN scheme was applied at large scale for LID and was benchmarked against alternative state-of-the-art approaches. Evaluated using two different datasets – the NIST LRE’09 (3s task) and Google 5M LID – this scheme demonstrated significantly improved performance compared to several i-vector-based state-of-the-art systems (Lopez-Moreno et al.). This scheme has also been successfully applied as a front-end stage for real-time multilingual speech recognition, as described in (Gonzalez-Dominguez et al., 2014).

This article builds on our previous work by extensively evaluating and comparing the use of DNNs for LID with an i-vector baseline system in different scenarios. We explore the influence of several factors on the DNN architecture configuration, such as the number of layers, the importance of including the temporal context and the duration of test segments. Further, we present a hybrid approach between the DNN and the i-vector system – the *bottleneck* system – in an attempt to take the best from both approaches. In this hybrid system, a DNN with a bottleneck hidden layer (40 dimensions) acts as a new step in the feature extraction before the i-vector modeling strategy is implemented. Bottleneck features have recently been used in the context of LID (Jiang et al., 2014; Matějka et al., 2014; Richardson et al.). In these previous works, the DNN models were optimized to classify the phonetic units of a specific language, following the standard approach of an acoustic model for automatic speech recognition. Unlike in these previous works, here we propose using the bottleneck features from a DNN directly optimized for language recognition. In this new approach, *i*) the DNN optimization criterion is coherent with the LID evaluation criterion, and *ii*) the DNN training process does not require using transcribed audio, which is typically much harder to acquire than language labels. Note that the transcription process involves handwork from experts that are familiarized with specific guidelines (e.g. transcriptions provided in the written domain, or the spoken domain); it is slow, as each utterance typically contains about 2 words/sec and moreover, word level transcriptions needs to be mapped into frame level alignments before a DNN such as the one used in previous works can be trained. That requires bootstrapping from another pre-existing ASR system, typically a GMM-based acoustic model iteratively trained from scratch. Instead, in the process of training lang-id networks, no previous alignments are needed, only one label per utterance is required and annotation guidelines are significantly simpler. Overall, that facilitates the adoption of a bottleneck lang-id system, which has the additional advantage that targets language discrimination in all its intermediate stages.

For this study, we conducted experiments using two different datasets: i) a subset of LRE’09 (8 languages) that comprises equal quantities of data for each target language, and ii) the full LRE’09 evaluation dataset (23 languages), which contains significantly different amounts of available data for each target language. This approach enabled us to assess the performance of all the proposed systems in cases of both controlled and uncontrolled conditions.

The rest of this paper is organized as follows: Sections 2 and 3 present the i-vector baseline system and the architecture of the DNN-based system. In Section 4, we describe the proposed bottleneck scheme. In Sections 5 and 6, we outline fusion and calibration, and the datasets used during experimentation. Results are then presented in Section 7. Finally, Section 8 summarizes final conclusions and potential future lines of this work.

2. The baseline I-vector based system

2.1. Feature extraction

The input audio to our system is segmented into windows of 25ms with 10ms overlap. 7 Mel-frequency cepstral coefficients (MFCCs), including C_0, are computed on each frame (Davis and Mermelstein, 1980). Vocal tract length normalization (VTLN) (Welling et al., 1999), cepstral mean and variance normalization, and RASTA filtering (Hermansky and Morgan, 1994) are applied on the MFCCs. Finally, shifted delta cepstra (SDC) features are computed in a 7-1-3-7 configuration (Torres-Carrasquillo et al., 2002), and a 56-dimensional vector is obtained every 10 ms by stacking the MFCCs and the SDC of the current frame. The feature sequence of each utterance is converted into a single i-vector with the i-vector system described next.

2.2. I-vector extraction

I–vectors (Dehak et al., 2011) have become a standard approach for speaker identification, and have grown in popularity also for language recognition (Brummer et al., 2012; Dehak et al., 2011; Martinez et al., 2011; McCree, 2014). Apart from language and speaker identification, i–vectors have been shown to be useful also for several different classification problems including emotion recognition (Xia and Liu, 2012), and intelligibility assessment (Martínez et al., 2013). An i–vector is a compact representation of a Gaussian Mixture Model (GMM) supervector (Reynolds et al., 2000), which captures most of the GMM supervectors variability. It is obtained by a Maximum–A–Posteriori (MAP) estimate of the mean of a posterior distribution (Kenny, 2007). In the i–vector framework, we model the utterance-specific supervector **m** as:

$$\mathbf{m} = \mathbf{u} + \mathbf{T}\mathbf{w}, \tag{1}$$

where **u** is the UBM GMM mean supervector and **T** is a low-rank rectangular matrix representing the bases spanning the sub-space, which contains most of the variability in the supervector space. The i–vector is then a MAP estimate of the low-dimensional latent variable **w**. In our experiments, we have used a GMM containing 2048 Gaussian components with diagonal covariance matrices and the dimensionality of i-vectors was set to 600.

2.3. Classification backends

For classification, the i-vectors of each language are used to estimate a single Gaussian distribution via maximum likelihood, where the covariance matrix is shared among languages and is equal to the within-class covariance matrix of the training data. During evaluation, every new utterance is evaluated against the models of all the languages. Further details can be found in (Martinez et al., 2011).

3. The DNN-based LID system

Recent findings in the field of speech recognition have shown that significant accuracy improvements over classical GMM schemes can be achieved through the use of DNNs. DNNs can be used to generate new feature representations or as final classifiers that directly estimate class posterior scores. Among the most important advantages of DNNs is their multilevel distributed representation of the model's input data (Hinton et al., 2012). This fact makes the DNN an exponentially more compact model than GMMs. Further, DNNs do not impose assumptions on the input data distribution (Mohamed et al., 2012) and have proven successful in exploiting large amounts of data, achieving more robust models without lapsing into overtraining. All of these factors motivate the use of DNNs in language identification. The rest of this section describes the architecture and practical application of our DNN system.

3.1. Architecture

The DNN system used in this work is a fully-connected feed-forward neural network with rectified linear units (ReLU) (Zeiler et al., 2013). Thus, an input at level j, x_j, is mapped to its corresponding activation y_j (input of the layer above) as:

$$y_j = ReLU(x_j) = \max(0, x_j) \tag{2}$$

$$x_j = b_j + \sum_i w_{ij} y_i \tag{3}$$

where i is an index over the units of the layer below and b_j is the bias of the unit j.

The output layer is then configured as a *softmax*, where hidden units map input y_j to a class probability p_j in the form:

$$p_j = \frac{\exp(y_j)}{\sum_l \exp(y_l)} \tag{4}$$

where l is an index over all of the target classes (languages, Fig. 2).

As a cost function for backpropagating gradients in the training stage, we use the cross-entropy function defined as:

$$C = -\sum_j t_j \log p_j \tag{5}$$

where t_j represents the target probability of the class j for the current evaluated example, taking a value of either 1 (true class) or 0 (false class).

3.2. Implementing DNNs for language identification

From the conceptual architecture explained above, we built a language identification system to work at the frame level as follows:

As the input of the net, we used the same features as the i-vector baseline system (56 MFCC-SDC). Specifically, the input layer was fed with 21 frames formed by stacking the current processed frame and its ±10 left/right neighbors. Thus, the input layer comprised a total number of 1176 (21 × 56) visible units, v.

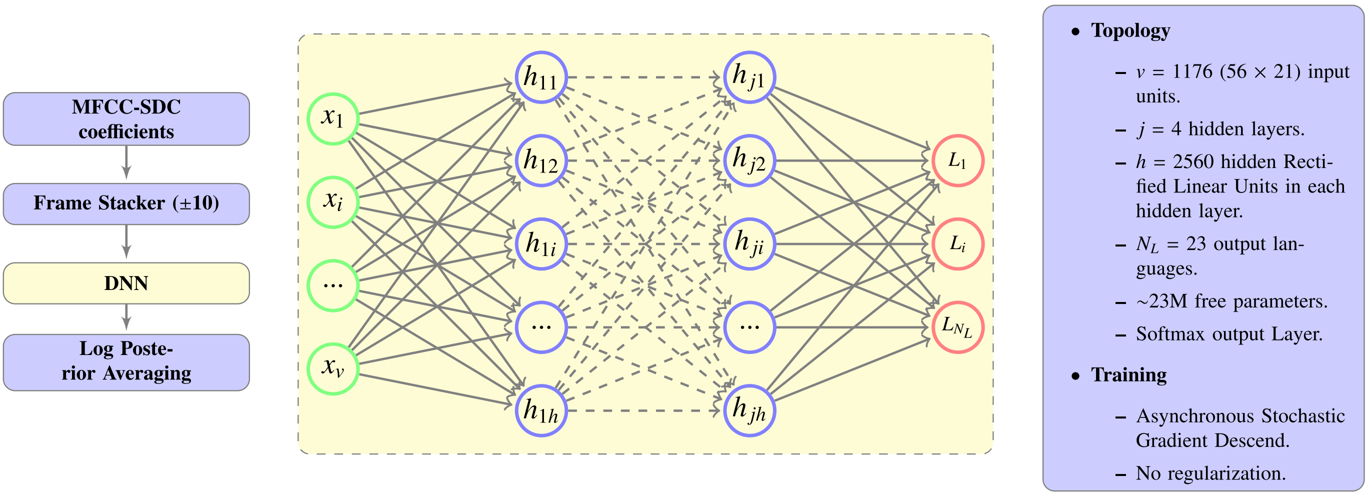
On top of the input layer, we stacked a total number of N_{hl} (4) hidden layers, each containing h (2560) units. Then, we added the softmax layer, whose dimension (s) corresponds to the number of target languages (N_L), plus one extra output for the out-of-set (*OOS*) languages. This OOS class, devoted to unknown test languages, could later allow us to use the system in open-set identification scenarios.

Overall, the net was defined by a total of w free parameters (weights + bias), $w = (v+1)h + (N_{hl}-1)(h+1)h + (h+1)s$ (~23M). The complete topology of the network is depicted in Fig. 1.

In terms of the training procedure, we used asynchronous stochastic gradient descent within the DistBelief framework (Dean et al., 2012), which uses computing clusters with thousands of machines to train large models. The learning rate and minibatch size were fixed to 0.001 and 200 samples.[1]

Note that the presented architecture works at the frame level, meaning that each single frame (plus its corresponding context) is fed-forward through the network, obtaining a class posterior probability for all of the target languages. This fact makes the DNNs particularly suitable for real-time applications because, unlike other approaches (i.e. i-vectors), we can potentially make a decision about the language at each new frame. Indeed, at each frame, we can combine the evidence from past frames to get a single similarity score between the test utterance and the target

[1] We define sample as the input of the DNN: the feature representation of a single frame besides those from its adjacent frames forming the context.



- **Topology**
  - v = 1176 (56 × 21) input units.
  - j = 4 hidden layers.
  - h = 2560 hidden Rectified Linear Units in each hidden layer.
  - N_L = 23 output languages.
  - ~23M free parameters.
  - Softmax output Layer.
- **Training**
  - Asynchronous Stochastic Gradient Descend.
  - No regularization.

Fig. 1. Pipeline process from the waveform to the final score (left). DNN topology (middle). DNN description (right).

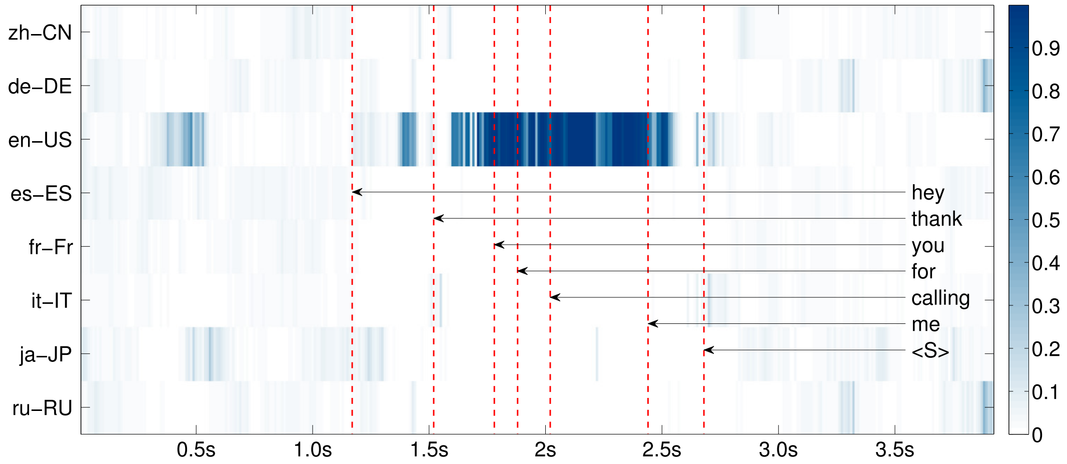


Fig. 2. Frame level probabilities of a DNN-based LID system (8 languages selected) evaluated over an English-USA (4s) test utterance.

languages. A simple way of doing this combination is to assume that frames are independent and multiply the posterior estimates of the last layer. The score s_l for language l of a given test utterance is computed by multiplying the output probabilities p_l obtained for all of its frames; or equivalently, accumulating the logs as:

$$s_l = \frac{1}{N}\sum_{t=1}^{N} \log p(L_l|x_t, \theta) \tag{6}$$

where $p(L_l|x_t, \theta)$ represents the class probability output for the language l corresponding to the input example at time t, x_t by using the DNN defined by parameters θ.

4. Bottleneck features: A hybrid approach

Another interesting way to leverage the discriminative power of a DNN is through the use of *bottleneck features* (Fontaine et al., 1997; Grézl et al., 2009). Typically, in speech recognition, bottleneck features are extracted from a DNN trained to predict phonetic targets, by either using the estimated output probabilities (Hermansky et al., 2000) or the activations of a narrow hidden layer (Grézl et al., 2007), the so-called bottleneck layer. The bottleneck features represent a low-dimensional non-linear transformation of the input features, ready to use for further classification.

Utilizing this approach, we extracted bottleneck features from the DNN directly trained for LID, as explained in Section 3, and replaced the last complete hidden layer with a bottleneck layer of 40 dimensions. Then, we modeled those new bottleneck features by using an i-vector strategy. That is, we replaced the standard MFCC-SDC features with bottleneck features as the input of our i-vector baseline system.

The underlying motivation of this hybrid architecture is to take the best from both the DNN and the i-vector system approaches. On one hand, we make use of the discriminative power of the DNN model and its capability to learn better feature representations; on the other, we are still able to leverage the generative modeling introduced by the i-vector system.

5. Fusion and calibration

We used multiclass logistic regression in order to combine and calibrate the outputs of individual LID systems (Brümmer and van Leeuwen, 2006). Let $s_{kL}(x_i)$ be the log-likelihood score for the recognizer k and language L for utterance x_i. We derive combined scores as

$$\hat{s}_L(x_i) = \sum_{k=1}^{K} \alpha_k s_{kL}(x_i) + \beta_L \tag{7}$$

Note that this is just a generic version of the product rule combination, parameterized by α and β. Defining a multiclass logistic regression model for the class posterior as

$$P(L|\hat{s}_L(x_i)) = \frac{\exp(\hat{s}_L(x_i))}{\sum_l \exp(\hat{s}_l(x_i))} \tag{8}$$

we found α and β to maximize the global log-posterior in a held-out dataset of I utterances

$$Q(\alpha_1, \ldots, \alpha_K, \beta_1, \ldots \beta_N) = \sum_{i=1}^{I} \sum_{l=1}^{N_L} \delta_{tl} P(L|\hat{s}_l(x_i)) \tag{9}$$

being

$$\delta_{iL} \begin{cases} w_L, & \text{if } x_i \in L \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

where w_l $(l = 1, \ldots, N_L)$ is a weight vector that normalizes the number of samples for every language in the development set (typically, $w_L = 1$ if an equal number of samples per language is used). This fusion and calibration procedure was conducted using the FoCal (Multi-class) toolkit (Brümmer).

6. Databases and evaluation metrics

6.1. Databases

We evaluate all proposed systems in the framework of the NIST LRE 2009 (LRE’09) evaluation. The LRE’09 includes data from two different audio sources: Conversational Telephone Speech (CTS) and, unlike previous LRE evaluations, telephone speech from broadcast news, which was used for both training and test purposes. Broadcast data were obtained via an automatic acquisition system from “Voice of America” news (VOA) that mixed telephone and non-telephone speech. Up to 2TB of 8kHz raw data containing radio broadcast speech, with corresponding language and audio source labels, were distributed to participants, and a total of 40 languages (23 target and 17 out of set) were included. While the VOA corpus contains over 2000 hours of labeled audio, only the labels from a fraction of about 200 hours were manually verified by the Linguistic Data Consortium (LDC).

Due to the large disparity in the amounts of available training material by language and type of audio source, we created two different evaluation sets from LRE’09: LRE09_FULL and LRE09_BDS. LRE09_FULL corresponds to

Table 1
Distribution of training hours per languages and eval files in used datasets LRE09_BDS and LRE09_FULL.

	LRE09_BDS				LRE09_FULL			
	Train (#hours)	Eval (#files)			Train (#hours)	Eval (#files)		
	–	03s	10s	30s	–	03s	10s	30s
amha	–	–	–	–	6h	411	391	379
bosn	–	–	–	–	2h	371	359	331
cant	–	–	–	–	39h	349	347	375
creo	–	–	–	–	6h	347	312	307
croa	–	–	–	–	1.5h	390	369	364
dari	–	–	–	–	6h	397	382	369
engi	–	–	–	–	18h	511	533	580
engl	200h	383	369	373	127h	866	836	913
fars	200h	338	338	338	38h	385	383	391
fren	200h	395	395	395	35.5h	401	394	388
geor	–	–	–	–	5h	403	396	398
haus	–	–	–	–	6h	430	382	345
hind	–	–	–	–	69h	640	614	668
kore	–	–	–	–	77.5h	460	450	453
mand	200h	404	390	387	244h	977	971	1028
pash	200h	406	391	388	6h	404	391	388
port	–	–	–	–	6h	457	377	339
russ	200h	257	253	254	66h	484	479	523
span	200h	402	383	370	114h	398	383	370
turk	–	–	–	–	6h	396	394	392
ukra	–	–	–	–	1h	403	383	375
urdu	200h	358	344	339	13h	386	372	371
viet	–	–	–	–	56h	282	279	315
OOS	200h	–	–	–	2510h	–	–	–
TOTAL	1800h	2943	2863	2844	3458.5h	10548	10177	10362

the original LRE'09 evaluation, which includes the original test files and all development training files for each language.[2] LRE09_BDS, on the other hand, is a balanced subset of 8 languages from automatically labeled VOA audio data. While the LRE09_FULL set uses data from the manually annotated part of the VOA corpus, the LRE09_BDS contains audio from both automatically and manually annotated parts. This dual evaluation approach served two purposes: i) LRE09_FULL, which is a standard benchmark, allowed us to generate results that could be compared with those of other research groups, and ii) LRE09_BDS allowed us to conduct new experiments using a controlled and balanced dataset with more hours of data for each target language. This approach may also help identify a potentially detrimental effect on the LRE09_FULL DNN-based systems due to the lack of data in some target languages. This is important because we previously found that the relative performance of a DNN versus an i-vector system is largely dependent on the amount of available data (Lopez-Moreno et al.).

Table 1 summarizes the specific training and evaluation data per language used in each dataset.

6.2. Evaluation metrics

Two different metrics were used to assess the performance of the proposed techniques. As the main error measure to evaluate the capabilities of one-vs.-all language detection, we used C_{avg} (average detection cost), as defined in the LRE 2009 (Brummer, 2010; NIST, 2009) evaluation plan. C_{avg} is a measure of the cost of making incorrect decisions and, therefore, considers not only the discrimination capabilities of the system, but also the ability of setting optimal thresholds (i. e., calibration). Further, the well-known metric Equal Error Rate (EER) is a calibration-insensitive metric that indicates the error rate at the operating point where the number of false alarms and the number of false rejections

[2] We used the training dataset defined by the I3A research group (University of Zaragoza) in its participation in the LRE'11 evaluation (Martínez et al., 2011).

are equal. Since our problem is a detection task where a binary classification is performed for each language, the final EER is the average of the EERs obtained for each language.

7. Results

In this section, we present a comprehensive set of experiments that compare and assess the two systems of interest, as well as a combined version of the two. Besides the i-vector-based baseline system, we evaluate the following three family of systems:

- **DNN** refers to the end-to-end deep neural network based system presented in Section 3, which is used as a final classifier to predict language posteriors.
- **DNN_BN**: refers to an end-to-end DNN system where the last hidden layer is replaced by a bottleneck layer. This DNN is used as a final classifier to predict language posteriors.
- **BN** refers to the i-vector system where the inputs are bottleneck features, as explained in Section 4.

Individual systems vary in the number of layers used (4 or 8 layers) and the size of their input context (0, 5 or 10 left/right frames). Hereafter, we will use the family name to refer a specific system {DNN, DNN_BN, BN}, followed by a set of suffixes {4L, 8L} and the {0-0C, 5-5C, 10-10C} to denote the number of layers and input context, respectively. For instance, the system name DNN_BN_4L_5-5C refers to a DNN system with 4 layers where the last hidden layer is a bottleneck layer, which uses an input of 11 concatenated frames (5 to the left and 5 to the right of the central frame). Note that the difference between DNN_BN and BN is that in the first, the DNN with a bottleneck layer is used directly as an end-to-end classifier, while in the second the DNN is used to extract bottleneck features which are used as input to an i-vector system.

7.1. Results using LRE09_BDS

7.1.1. DNN vs i-vector system

As the starting point of this study, we compare the performance of the proposed DNN architecture (with 4 layers and input context of ±10 frames) and the i-vector baseline system. Fig. 3 shows the difference in performance using test segments with a duration of 3s, 10s and 30s. The trend of the lines in the figure illustrates one of the main conclusions of this work: the DNN system significantly outperforms the i-vector system for short duration utterances (3s), while the i-vector system is more robust for test utterance over 10s.

Unlike i-vectors, the DNN system does not process the complete test utterance at once. Instead, posterior scores are computed at each individual frame and combined as if each frame was independent (Eq. 6). This is a frame-by-frame strategy that allows for providing continuous labels for a data stream, which may be beneficial in real time applications (Gonzalez-Dominguez et al., 2014).

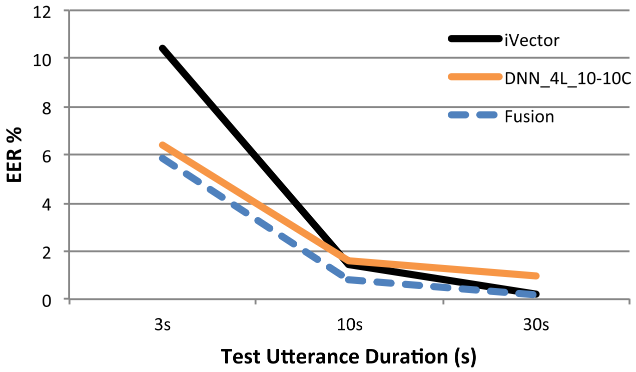


Fig. 3. DNN versus i-vector system performance (average EER) in function of test utterance segment duration (LRE09_BDS corpus).

Table 2
Performance for individual and fusion systems – average EER in % and $C_{avg} \times 100$ – on the balanced *LRE*09*_BDS* dataset by test duration. All DNN family systems {DNN, DNN_BN and BN} come from a DNN with 4 layers and context of ±10 frames.

	Equal error rate (%)/ C_{avg} (×100)		
	LRE09_BDS		
	3s	10s	30s
i-vector	10.20/10.39	1.45/1.54	0.21/0.28
DNN	6.42/6.79	1.62/1.72	0.94/0.94
DNN_BN	6.43/6.61	1.65/1.68	0.97/0.98
BN	6.73/7.03	1.16/1.20	0.43/0.56
i-vector + DNN	5.86/6.17	0.92/1.03	0.20/0.28
i-vector + DNN_BN	5.87/6.02	0.91/1.06	0.19/0.27
i-vector + BN	**5.82/6.16**	**0.81/0.85**	**0.18/0.20**

7.1.2. Bottlenecks

Next, we explore the use of bottleneck features in the bottleneck system. As previously stated, it is a hybrid DNN/i-vector system where the DNN model acts as a feature extractor, whose features are used by the i-vector model instead of the standard MFCC-SDC. Specifically, we present the results of a bottleneck system that uses a DNN model with 4 layers, where the last hidden layer was replaced by a 40 dimensional bottleneck layer. Table 2 compares the results from the hybrid bottleneck system with those of the standalone alternative approaches presented previously. Results show significantly improved performance for 10s and 30s utterances when using the bottleneck system (BN), as compared with the DNN system without bottleneck (DNN) (28% and 54% relative improvement in EER, respectively), while results for 3s utterances are similar. With respect to the i-vector, results obtained with the BN system are better in 3s and 10s (20% and 34% relative improvement in EER, respectively), whereas in 30s i-vectors still obtain better performance. Again, these results demonstrate the robustness of the i-vector system when evaluating longer test segments. They also suggest that further research into this area could lead to improved results when combining the strengths of DNN and i-vector systems.

We also analyze the loss in performance of our standalone neural network system when reducing the number of nodes in the last hidden layer from 2560 (DNN system) to 40 nodes used by the DNN_BN system. That is, the DNN_BN system uses the same network that extracts the BN features, but is used as an end-to-end classifier. Results collected in Table 2 show that there is not a significant difference in performance when reducing the number of nodes in the last hidden layer. This result demonstrates that bottleneck features are an accurate representation of the frame-level information; at least, comparable to that presented in the complete last hidden layer of the conventional DNN architecture.

7.1.3. Temporal context and number of layers

Another important aspect in the DNN system configuration is the temporal context of the spectral features used as the input to the DNN. Until now, we have used a fixed right/left context of ±10 frames respectively. That is, the input of our network, as mentioned in Section 3, is formed by stacking the features of every frame with its corresponding 10 adjacent frames to the left and right. The motivation behind using temporal contexts with a large number of frames lies in the idea of incorporating additional high-level information into our system (i.e. phonetic, phonotactic and prosodic information). This idea has been widely and successfully implemented in language identification in the past, using long-term phonotactic/prosodic tokenizations (Ferrer et al., 2010; Reynolds et al., 2003) or, in acoustic approaches, by using shifted-delta-cepstral features (Torres-Carrasquillo et al., 2002).

Table 3 presents the performance for contextual windows of size 0, ± 5 and ± 10 frames. Unlike the results we presented in Gonzalez-Dominguez et al. (2015), where we found that the window size was critical to model the contextual information, here just small and non-uniform gains were found. This result can be explained by the fact that, unlike the PLP features used in Gonzalez-Dominguez et al. (2015), the MFCC-SDC features in this paper already include some degree of temporal information.

In addition, we evaluate the effect of increasing the number of layers to eight, doubling the number of weights in the network from 22.7M to 48.9M. The results of this evaluation are summarized in Table 3. The 8 layers topology achieved only small gains for DNN and DNN_BN in 3s segments, so we opted to keep the original 4-layer DNN as our reference DNN system.

Table 3
Performance of DNN-based systems as a function of the number of layers and temporal context used. Results on LRE09_BDS are reported as average EER on the 8 languages (%).

	Equal error rate (%)																	
	4 Layers									8 Layers								
	0C-0C			5C-5C			10C-10C			0C-0C			5C-5C			10C-10C		
	3s	10s	30s	3s	10s	30s	3s	10s	30s	3s	10s	30s	3s	10s	30s	3s	10s	30
DNN	6.8	2.00	1.07	6.66	1.64	0.95	6.42	1.62	0.94	6.90	1.94	0.98	6.40	1.72	0.92	6.23	1.61	0.94
DNN_BN	7.21	2.17	1.16	6.77	1.83	1.04	6.43	1.65	0.97	6.87	1.91	0.96	**6.17**	1.71	0.98	6.36	1.58	0.85
BN	7.43	1.32	0.41	7.07	**1.13**	**0.38**	6.73	1.16	0.43	9.36	2.01	0.68	8.74	1.43	0.59	9.17	1.84	0.76

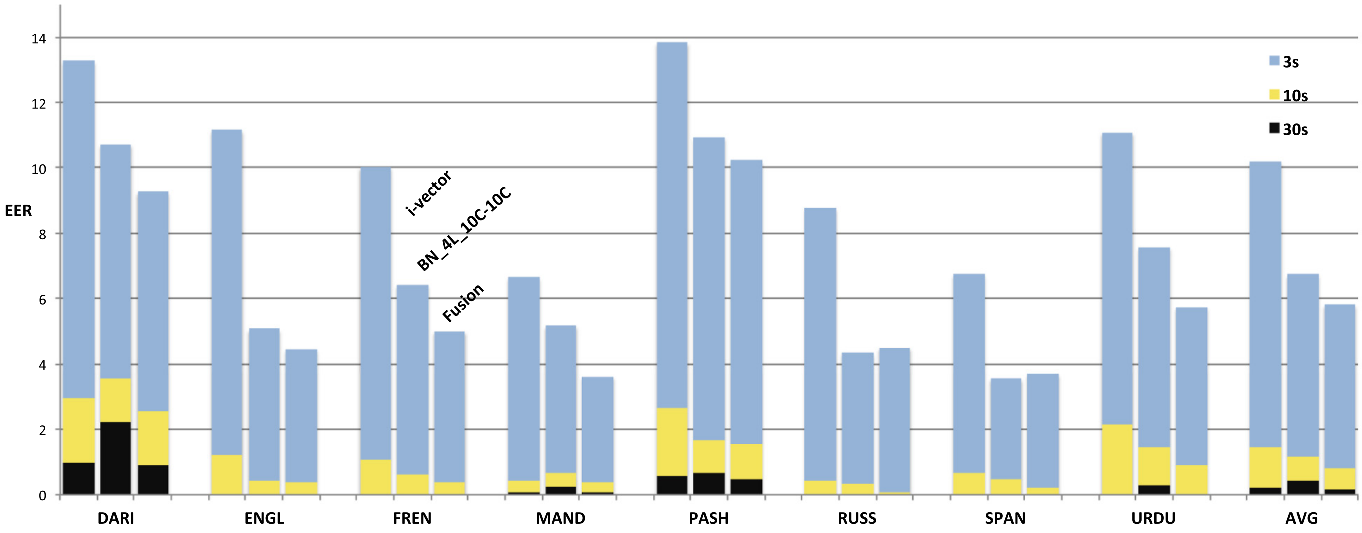


Fig. 4. i-vector, BN and fusion system performance comparison (average EER) per language on LRE09_BDS dataset. Errors bars for 30s, 10s and 3s are superimposed, and therefore, representing the actual error for every condition.

7.1.4. Fusion and results per language

The different optimization strategies of DNN and i-vector systems (discriminative vs. generative) and the different results observed in the evaluated tasks have demonstrated the complementarity of these two approaches. Such complementarity suggests that further gains may be achieved through a score level combination of the systems presented above, the results of which are presented in the last three rows of Table 2. We performed a score-level combination of the baseline i-vector system and various neural network-based systems by means of multiclass logistic regression (Section 5). The fusion of the i-vector and bottleneck systems achieves the best performance, with relative improvements over the standalone i-vector system of 42%/40%, 44%/44% and 14%/28% in terms of EER/C_{avg} for the 3s, 10s and 30s evaluated conditions, respectively. Moreover, results are consistent across all the languages and test duration conditions as shown in Fig. 4. These results confirm that when bottleneck and i-vector systems are combined, they consistently outperform the baseline i-vector system, although the relative improvement diminishes as the test duration increases (Fig. 5).

7.2. Results using LRE09_FULL

To properly train a DNN system, we ideally need large and balanced amounts of data for each language. In this section, we evaluate the implications of having an unbalanced training dataset. Specifically, we mirror the experiments in the above section, instead using the entire LRE09_FULL dataset (see Table 1 for the distribution of this dataset). One of the possible approaches for dealing with an unbalanced dataset is to build a bottleneck system in the following way: First, generate a balanced subset of utterances from the most represented languages to train a network that

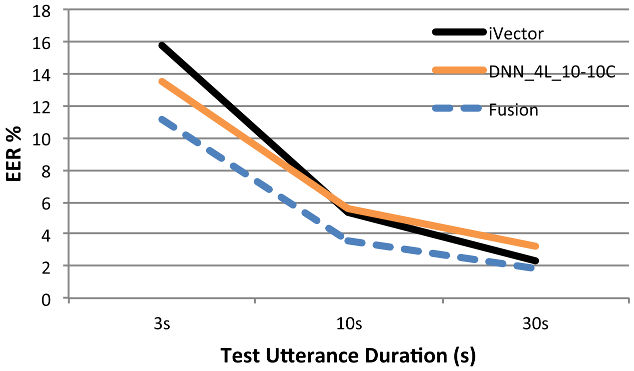


Fig. 5. DNN vs i-vector system performance (EER) in function of test utterance segment duration (LRE09_FULL corpus).

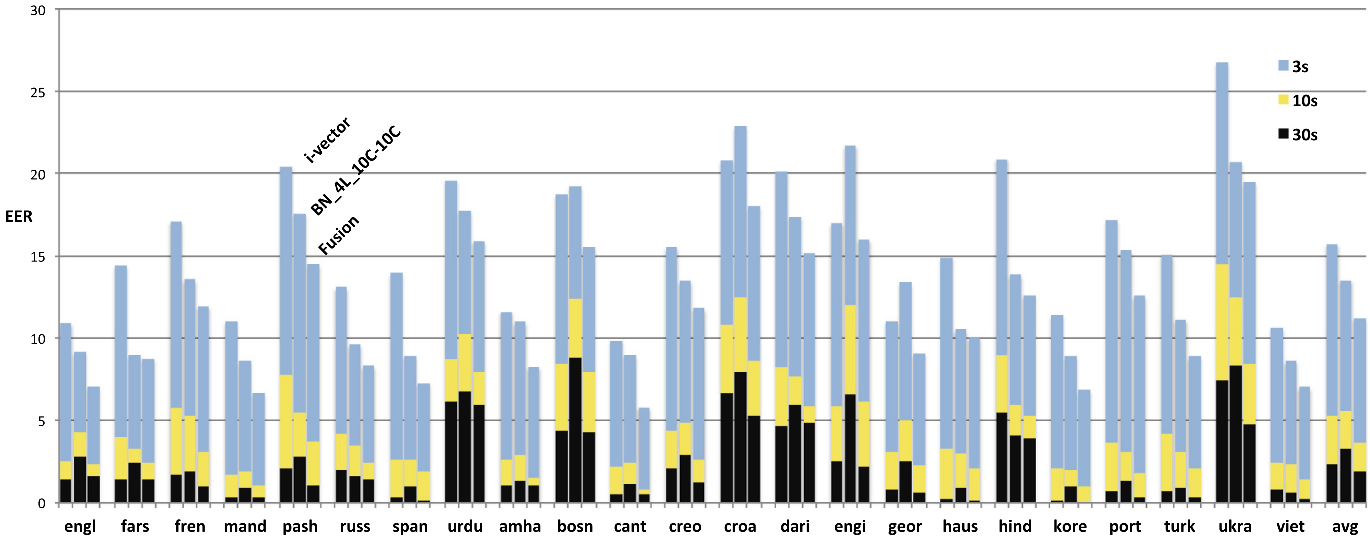


Fig. 6. i-vector, BN and fusion system performance comparison (average EER) per language on LRE09_FULL dataset. The DNN used to extract bottleneck features was trained just with the 8 target languages of LRE09_BDS (on the left of the vertical dashed line). Errors bars for 30s, 10s and 3s are superimposed, and therefore, representing the actual error for every condition.

includes a bottleneck layer. This network may or may not contain all of the target languages. Then, using the previous network, compute the bottleneck features from the original unbalanced dataset to optimize the remaining stages involved in the i-vector system. We simulated the unbalanced data scenario by using the eight-language DNN from Section 7.1 to compute bottleneck features over our LRE09_FULL training set. While one could consider using non-overlapping sets for the DNN and i-vector optimization to avoid overfitting, we opted to use the entire unbalanced dataset due to data scarcity in our training material for LRE09_FULL.

Fig. 6 depicts the performance of the bottleneck system trained as explained above, the i-vector system, and their fusion, for each of the 3 conditions (3s, 10s, 30s) and the 23 target languages. The vertical line separates the performance for the languages included (left) and excluded (right) during the DNN optimization process. The results show that, despite overall good performance, the bottleneck system performs much better for the languages involved in the DNN training.

The second approach used was to train a new DNN model using all the target languages in the LRE09_FULL evaluation. Note that, in this case, unequal amounts of training data were used to optimize each of the 23 DNN outputs. The results of this second approach are shown in Fig. 7. By comparing Figs. 7 and 6, in which the only underlying difference is the training data used for the DNN model, we see that data imbalance may be an issue in the standalone DNN system, but it is not an issue when using the bottleneck system. Moreover, the bottleneck system seems to benefit from matching the target classes of the underlying DNN model with the target languages in the language recognition evaluation (see, for instance, the performance improvements on Georgian, Korean or Ukranian).

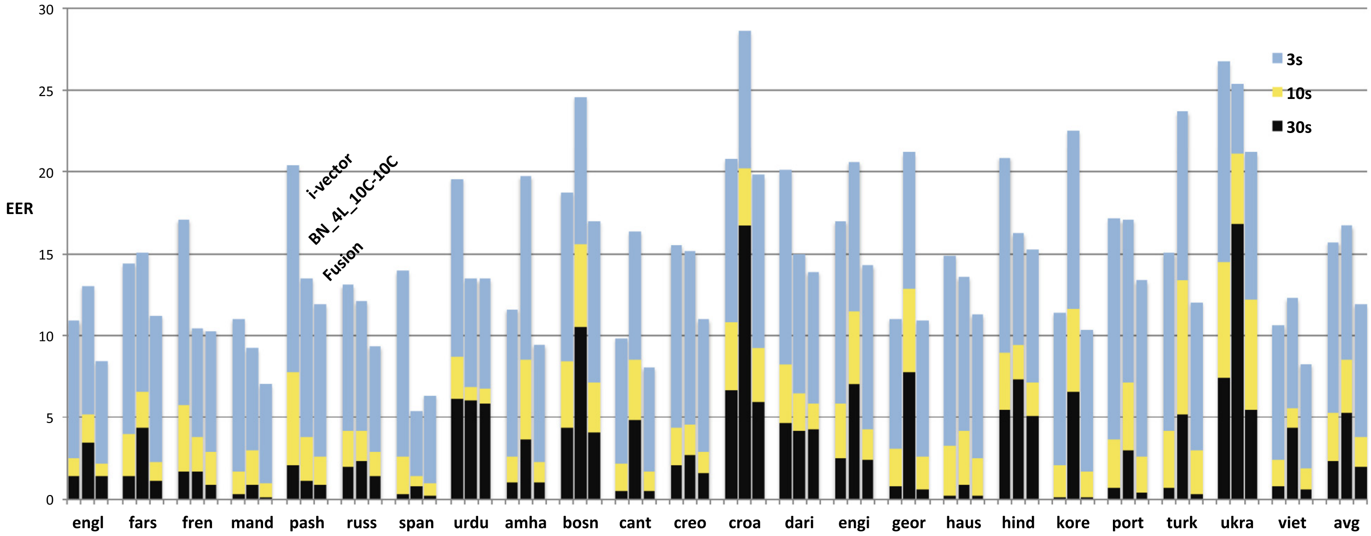


Fig. 7. i-vector, BN and fusion systems performance comparison (average EER) per language on LRE09_FULL dataset. The DNN used to extract bottleneck features was trained with all the 23 target languages. Errors bars for 30s, 10s and 3s are superimposed, and therefore, representing the actual error for every condition.

Table 4
Performance for individual and fusion systems – average EER in % and $C_{avg} \times 100$ – on full *LRE09_FULL* dataset by test duration. All DNN family systems {DNN, DNN_BN and BN} come from a DNN with 4 layers and context of ±10 frames.

	Equal error rate (%)/C_{avg} (×100)		
	LRE09_FULL		
	03s	10s	30s
i-vector	15.74/16.37	5.30/6.24	2.33/2.90
DNN_BN	13.49/14.21	6.38/7.11	4.37/4.88
BN	13.52/14.19	5.58/6.21	3.24/3.77
i-vector + DNN_BN	11.93/12.76	3.80/4.59	1.94/2.28
i-vector + BN	**11.19/11.87**	**3.61/4.13**	**1.85/2.21**

Finally, the results for the LRE09_FULL dataset for all the individual systems proposed, including fusions with the baseline i-vector system, are summarized in Table 4. Note that all DNNs shown in this table were trained using data of the 23 target languages. The conclusion remains the same as in the case of the LRE09_BDS dataset (Table 2), but with modest performance improvements. Specifically, when fusing the i-vector and the bottleneck systems, we achieved improvements of 29%/27%, 32%/34% and 21%/24% in terms of EER/C_{avg} for the 3s, 10s and 30s evaluated conditions.

8. Summary

In this work, we presented an extensive study of the use of deep neural networks for LID. Guided by the success of DNNs for acoustic modeling, we explored their capability to learn discriminative language information from speech signals.

First, we showed how a DNN directly trained to discern languages obtains significantly improved results with respect to our best i-vector system when dealing with short-duration utterances. This proposed DNN architecture is able to generate a local decision about the language spoken in every single frame. These local decisions can then be combined into a final decision at any point during the utterance, which makes this approach particularly suitable for real-time applications.

Next, we introduced the LID optimized bottleneck system as a hybrid approach between the proposed DNN and i-vector systems. Here, a DNN optimized to classify languages is seen as a front-end that extracts a more suitable representation (in terms of discrimination) of feature vectors. On contrary to previous bottleneck approaches for LID, where the DNN was trained to recognize the phonetic units of a given language, in this work, the DNN optimization criterion is coherent with the LID objective. Moreover, the DNN model requires only language labels which are much easier to obtain than the speech transcriptions.

We observed that the most desirable scenario is to train the DNN with a balanced dataset that includes all the target languages. In the case of not being able to fulfill this requirement, it is preferable to include data from all target languages during the DNN optimization stage, even if some languages contain more training hours than others. In addition, fusion results show that DNN-based systems provide complementary information to the baseline i-vector system. In particular, the combination of the i-vector and bottleneck systems result in a relative improvement of up to 42%/40%, 44%/44% and 14%/28% and 29%/27%, 32%/34% and 21%/24% for the balanced dataset LRE09_BDS and the whole LRE'09 evaluation respectively, in terms of EER/*C_avg* and for the 3s, 10s, and 30s test conditions.

We believe that the performance of the DNN could be improved further. In the future, we plan to experiment with other topologies/activation functions and other input features, such as filterbank energies. Further, for the sake of comparison, in this work we chose i-vector modeling as the strategy to model the bottleneck features. It is a future line of this work to experiment with different modeling schemes that might fit better for those bottleneck features .

Acknowledgment

The authors would like to thank Daniel Garcia Romero for helpful suggestions and valuable discussions. Javier Gonzalez-Dominguez worked in this project during his stay is Google supported by the Google Visitor Faculty program. This work was partly supported by the Spanish government through project TIN2011-28169-C05-02 and by the Czech Ministry of Interior project No. VG20132015129 "ZAOM" and by the Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602".

References

Brummer, N., 2010. Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech (Ph.D. thesis). Department of Electrical and Electronic Engineering, University of Stellenbosch.

Brummer, N., Cumani, S., Glembek, O., Karafiát, M., Matejka, P., Pesan, J., et al., 2012. Description and analysis of the Brno276 System for LRE2011. In: Proceedings of Odyssey 2012: The Speaker and Language Recognition Workshop. International Speech Communication Association, pp. 216–223.

Brümmer, N., 2010. Fusion and calibration toolkit [software package]. <http://sites.google.com/site/nikobrummer/focal>.

Brümmer, N., van Leeuwen, D., 2006. On calibration of language recognition scores. In: Proc. of Odyssey. San Juan, Puerto Rico.

Ciresan, D., Meier, U., Gambardella, L., Schmidhuber, J., 2010. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition, CoRR abs/1003.0358.

Cole, R., Inouye, J., Muthusamy, Y., Gopalakrishnan, M., 1989. Language identification with neural networks: a feasibility study. In: Communications, Computers and Signal Processing, 1989. Conference Proceeding, IEEE Pacific Rim Conference on, pp. 525–529. doi:10.1109/PACRIM.1989.48417.

Davis, S., Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. 28, 357–366.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., et al., 2012. Large scale distributed deep networks. In: Bartlett, P., Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 25, pp. 1232–1240.

Dehak, N., Torres-Carrasquillo, P.A., Reynolds, D.A., Dehak, R., 2011. Language recognition via i-vectors and dimensionality reduction. In: INTERSPEECH, ISCA, pp. 857–860.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front-end factor analysis for speaker verification. IEEE Trans. Audio Speech Lang. Process. 19 (4), 788–798.

Ferrer, L., Scheffer, N., Shriberg, E., 2010. A comparison of approaches for modeling prosodic features in speaker recognition. In: International Conference on Acoustics, Speech, and Signal Processing, pp. 4414–4417. doi:10.1109/ICASSP.2010.5495632.

Fontaine, V., Ris, C., Boite, J.-M., Initialis, M.S., 1997. Nonlinear discriminant analysis for improved speech recognition. In: Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997. Rhodes, Greece.

Gonzalez-Dominguez, J., Eustis, D., Lopez Moreno, I., Senior, A., Beaufays, F., Moreno, P., 2014. A real-time end-to-end multilingual speech recognition architecture. IEEE J. Sel. Top. Signal Process. (99), 1. doi:10.1109/JSTSP.2014.2364559.

Gonzalez-Dominguez, J., Lopez-Moreno, I., Moreno, P.J., Gonzalez-Rodriguez, J., 2015. Frame-by-frame language identification in short utterances using deep neural networks. Neural Netw. 64, 49–58, Special Issue on Deep Learning of Representations. doi:10.1016/j.neunet.2014.08.006.

Grézl, F., Karafiát, M., Kontár, S., Černocký, J., 2007. Probabilistic and bottle-neck features for LVCSR of meetings. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007). IEEE Signal Processing Society, pp. 757–760.

Grézl, F., Karafiát, M., Burget, L., 2009. Investigation into bottle-neck features for meeting speech recognition. In: INTERSPEECH 2009, International Speech Communication Association, pp. 2947–2950.

Hermansky, H., Morgan, N., 1994. RASTA processing of speech. IEEE Trans. Speech Audio Process. 2 (4), 578–589.

Hermansky, H., Ellis, D.P.W., Sharma, S., 2000. Tandem connectionist feature extraction for conventional HMM systems. In: PROC. ICASSP, pp. 1635–1638.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29 (6), 82–97. doi:10.1109/MSP.2012.2205597.

Jiang, B., Song, Y., Wei, S., Liu, J.H., McLoughlin, I.V., Dai, L.-R., 2014. Deep bottleneck features for spoken language identification. PLoS ONE 9 (7), e100795. doi:10.1371/journal.pone.0100795.

Kenny, P., 2007. Joint factor analysis of speaker and session variability: theory and algorithms. <http://www.crim.ca/perso/patrick.kenny/FAtheory.pdf>.

Kenny, P., Oullet, P., Dehak, V., Gupta, N., Dumouchel, P., 2008. A study of interspeaker variability in speaker verification. IEEE Trans. Audio Speech Lang. Process. 16 (5), 980–988.

Leena, M., Srinivasa Rao, K., Yegnanarayana, B., 2005. Neural network classifiers for language identification using phonotactic and prosodic features. In: Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on, pp. 404–408. doi:10.1109/ICISIP.2005.1529486.

Li, H., Ma, B., Lee, K.A., 2013. Spoken language recognition: from fundamentals to practice. P. IEEE 101 (5), 1136–1159. doi:10.1109/JPROC.2012.2237151.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martinez, D., Gonzalez-Rodriguez, J., Moreno, P., 2014. Automatic language identification using deep neural networks. Acoustics, Speech, and Signal Processing, IEEE International Conference on.

Martínez, D., Villalba, J., Ortega, A., Lleida, E., 2011. I3A language recognition system description for NIST LRE 2011. In: NIST 2011 LRE Workshop Booklet. Atlanta, Georgia, USA.

Martínez, D., Green, P.D., Christensen, H., 2013. Dysarthria intelligibility assessment in a factor analysis total variability space. In: INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association. Lyon, France, pp. 2133–2137. August 25–29.

Martinez, D., Plchot, O., Burget, L., Glembek, O., Matejka, P., 2011. Language recognition in iVectors space. In: INTERSPEECH. ISCA, pp. 861–864.

Matějka, P., Zhang, L., Ng, T., Mallidi, H.S., Glembek, O., Ma, J., et al., 2014. Neural network bottleneck features for language identification. In: Speaker Odyssey, pp. 299–304.

McCree, A., 2014. Multiclass discriminative training of i-vector language recognition. In: IEEE Odyssey: The Speaker and Language Recognition Workshop. Joensu, Finland.

Mohamed, A., Dahl, G., Hinton, G., 2012. Acoustic modeling using deep belief networks. IEEE Trans. Audio Speech Lang. Process. 20 (1), 14–22. doi:10.1109/TASL.2011.2109382.

Mohamed, A.-R., Hinton, G.E., Penn, G., 2012. Understanding how deep belief networks perform acoustic modelling. In: ICASSP. IEEE, pp. 4273–4276.

Montavon, G., 2009. Deep learning for spoken language identification. In: NIPS Workshop on Deep Learning for Speech Recognition and Related Applications.

Muthusamy, Y., Barnard, E., Cole, R., 1994. Reviewing automatic language identification. IEEE Signal Process. Mag. 11 (4), 33–41. doi:10.1109/79.317925.

NIST, 2009. The 2009 NIST SLR Evaluation Plan. <www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf>.

Reynolds, D., Quatieri, T., Dunn, R., 2000. Speaker verification using adapted Gaussian mixture models. Dig. Sig. Process. 10 (1/2/3), 19–41.

Reynolds, D., Andrews, W., Campbell, J., Navratil, J., Peskin, B., Adami, A., et al., 2003. The SuperSID project: exploiting high-level information for high-accuracy speaker recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 784–787.

Richardson, F., Reynolds, D., Dehak, N., 2015. A Unified Deep Neural Network for Speaker and Language Recognition arXiv:1504.00923. <http://arxiv.org/abs/1504.00923>.

Sturim, D., Campbell, W., Dehak, N., Karam, Z., McCree, A., Reynolds, D., et al., 2011. The MIT LL 2010 speaker recognition evaluation system: scalable language-independent speaker recognition. In: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pp. 5272–5275. doi:10.1109/ICASSP.2011.5947547.

Torres-Carrasquillo, P.A., Singer, E., Kohler, M.A., Deller, J.R., 2002. Approaches to language identification using Gaussian mixture models and shifted delta cepstral features. In: ICSLP, vol. 1, pp. 89–92.

Welling, L., Kanthak, S., Ney, H., 1999. Improved methods for vocal tract normalization. In: Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP ‘99. Phoenix, Arizona, USA, pp. 761–764. March 15–19.

Xia, R., Liu, Y., 2012. Using i-vector space model for emotion recognition. In: INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association. Portland, Oregon, USA, pp. 2230–2233. September 9–13.

Yu, D., Deng, L., 2011. Deep learning and its applications to signal and information processing [exploratory DSP]. IEEE Signal Process. Mag. 28 (1), 145–154. doi:10.1109/MSP.2010.939038.

Zeiler, M., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., et al., 2013. On rectified linear units for speech processing. In: 38th International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver.

A.2 A Real-Time End-to-End Multilingual Speech Recognition Architecture

Title	**A Real-Time End-to-End Multilingual Speech Recognition Architecture.**	
Journal	IEEE Journal of Selected Topics in Signal Processing.	
Impact-Factor	5-year 2.569, Q1	Engineering, Electrical & Electronic
Authors	Javier González Domínguez	UAM
	David Eustis	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
	Andrew Senior	Google Inc.
	Françoise Beaufays	Google Inc.
	Pedro J. Moreno	Google Inc.
Doi	10.1109/JSTSP.2014.2364559	
Date	2015/04	
Publisher	IEEE	New York, NY, US

Summary: This article presents the design, implementation and performance analysis of the first public and mutilingual Large Vocabulary Continuous Speech Recognition (LVCSR) system. Our mutilingual LVCSR system combines decisions made based on a DNN-based language identification system and N parallel individual DNN-based LVCSR recognizers. The design focus on minimizing Word Error Rate and Language Identification Error Rate across languages, while supporting under real-time online decoding.

Contributions: The candidate contributed with the collection of approximately one year of labeled data (audio and parameters) from real voice queries; provided the necessary code to generate neural networks for language recognition; generated candidate DNN models with optimized hyperparameters and provided a production environment where the language classifier could be tested in terms of user latency and server cpu load (necessary to validate the system providing metrics of real time factor and multilingual speech recognition performance.)

A Real-Time End-to-End Multilingual Speech Recognition Architecture

Javier Gonzalez-Dominguez, *Member, IEEE*, David Eustis, Ignacio Lopez-Moreno, *Member, IEEE*, Andrew Senior, *Senior Member, IEEE*, Françoise Beaufays, *Senior Member, IEEE*, and Pedro J. Moreno, *Senior Member, IEEE*

***Abstract*—Automatic speech recognition (ASR) systems are used daily by millions of people worldwide to dictate messages, control devices, initiate searches or to facilitate data input in small devices. The user experience in these scenarios depends on the quality of the speech transcriptions and on the responsiveness of the system. For multilingual users, a further obstacle to natural interaction is the monolingual character of many ASR systems, in which users are constrained to a single preset language. In this work, we present an end-to-end multi-language ASR architecture, developed and deployed at Google, that allows users to select arbitrary combinations of spoken languages. We leverage recent advances in language identification and a novel method of real-time language selection to achieve similar recognition accuracy and nearly-identical latency characteristics as a monolingual system.**

***Index Terms*—Automatic speech recognition (ASR), deep neural network (DNN), language identification (LID), multilingual.**

I. Introduction

AUTOMATIC speech recognition (ASR) has become increasingly relevant to date, tracking the explosive growth of mobile devices. The use of voice as a natural and convenient method of human-device interaction is especially applicable to hands-free scenarios (e.g., while driving) and interaction with small form-factor devices (e.g., wearables). The quality of the user experience in these scenarios is primarily affected by the transcription accuracy and real-time responsiveness of the ASR system.

For multilingual users, another obstacle to natural interaction is the common monolingual character of ASR systems, in which users can speak in only a single preset language. According to several sources [1]–[3], multilingual speakers already outnumber monolingual speakers, and predictions point to a larger number of multilingual speakers in the future. The capacity to transparently recognize multiple spoken languages is, therefore, a desirable feature of ASR systems.

Several architectures have been considered to achieve multilingual speech recognition. One technique has been to train a universal speech model capable of recognizing multiple languages. Efforts in this direction are presented in [4]–[6]. This approach seeks to exploit similarities among languages and dialects, and lends itself to an easily deployable system. However, universal models tend to be larger and higher in perplexity relative to their monolingual equivalents, leading to potentially adverse effects on transcription accuracy and decoding latency.

Other architectures have attempted to detect the language of an utterance as a preprocessing step, through the use of language identification (LID) classifiers [7] [8]. Here, the outcome of the LID classification determines which of several monolingual speech recognizers is activated. The main drawbacks of this method are the latency introduced by the LID step, and the propagation of language classification errors to the final transcription.

Here, we present an integrated end-to-end multilingual architecture that builds upon the work described in [9]. In this architecture, monolingual speech recognizers decode the input signal simultaneously in each of the selected languages, while the LID system tries to determine which language is spoken. A decision is then made, based on the LID decision and on the confidence scores of the individual recognizers as to which recognition result best matches the user input. This architecture avoids the extra latency that an early language decision would introduce, and benefits from the extra scores from the recognizers to better decide which result to return to the user. These benefits come however with an increased processing cost since the input is recognized multiple times.

In this paper, we further improve upon [9] in two major directions. First, we replace the LID classifier with a more accurate Deep Neural Network (DNN) based classifier. Second, we propose a method to dynamically combine confidence scores and LID decisions from partial recognition results together with various timeout strategies to maintain the streaming nature of the monolingual system and to greatly reduce computing costs. One such strategy is shown to perform with near-ideal characteristics in both dimensions, minimizing latency and misrecognitions caused by language confusions.

Both the ASR and LID backend engines are based on DNNs [10]–[12]. During recent years, DNNs have achieved outstanding performance in diverse and challenging machine learning applications. Those including acoustic modelling [13] [14], visual object recognition [15], and many others [16]. Compared to previous acoustic modelling based on GMMs, the

Manuscript received July 01, 2014; accepted October 01, 2014. Date of publication October 23, 2014; date of current version May 12, 2015. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Raphael Cendrillon.

J. Gonzalez-Dominguez is with Google, Inc., New York, NY 10011 USA, and also with the ATVS Biometric Recognition Group, Universidad Autonoma de Madrid, 28046 Madrid, Spain (e-mail: jgd@google.com; javier.gonzalez@uam.es).

D. Eustis, I. Lopez-Moreno, A. Senior, F. Beaufays, and P. J. Moreno are with Google, Inc, New York, NY 10011 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2014.2364559

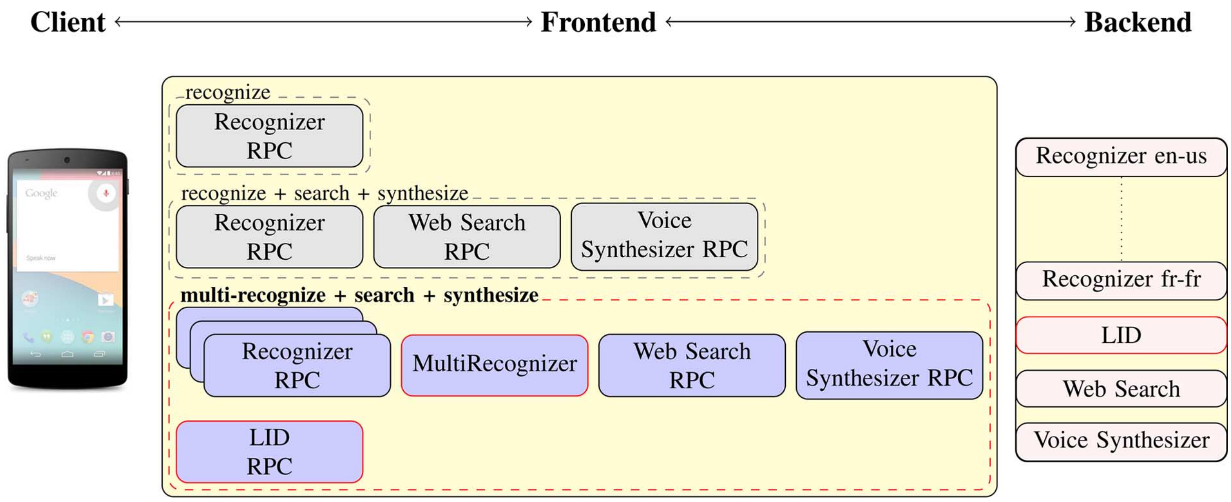


Fig. 1. Diagram showing components of the complete architecture: Client, Frontend and Backends. Within the Frontend, there are several activity pipelines composed of modules, each of which communicates with a single Backend. In the "multi-recognize + search + synthesize" activity pipeline, we introduce a MultiRecognizer module responsible for language selection.

use of DNNs presents several advantages. First, unlike GMMs, DNNs employ a multilevel distributed representation of the input [14]. This fact makes DNNs exponentially more compact than GMMs [17]. Second, DNNs, being purely discriminative, do not impose any assumptions about the input data distribution. Further, DNNs have proved successful in exploiting large amounts of data, achieving more robust models without lapsing into overfitting. We leverage this last point to employ large quantities of training data recorded from user traffic.

The rest of this paper is organized as follows. Section II presents the overall architecture of the system. Section III describes the frontend component which is responsible for language selection. Section IV describes the backend components, with Section IV-A focusing on the LID classifier and Section IV-B focusing on the monolingual speech recognizers. In Section V we present the databases and evaluation metrics used to assess system performance. We present results in Section VI. Finally, conclusions are outlined in Section VII.

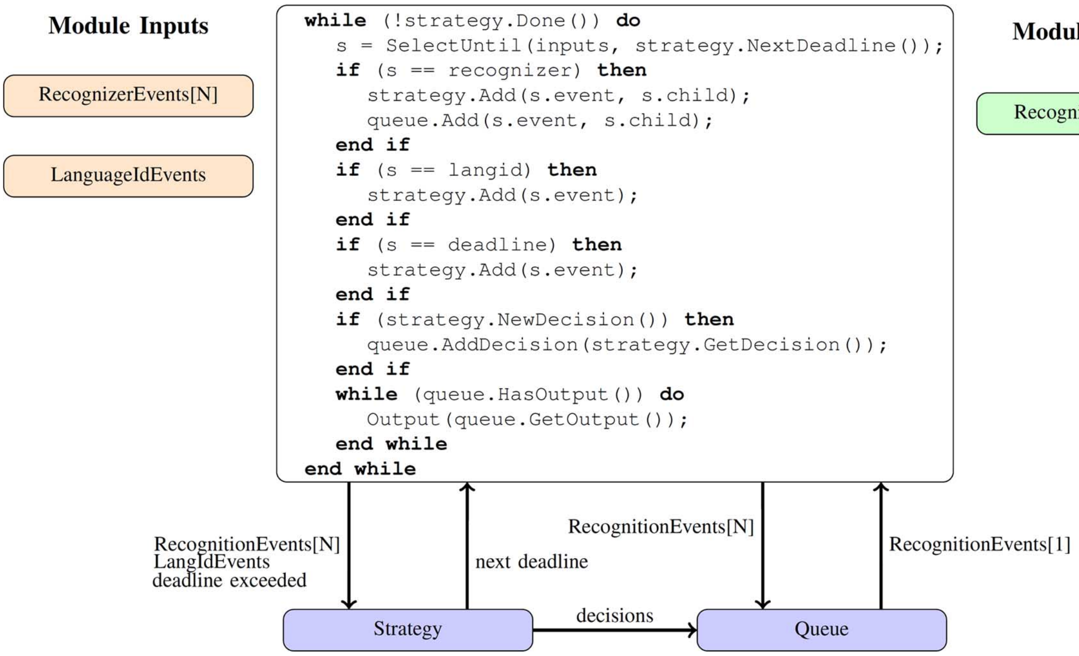
II. Overall Architecture

The end-to-end multilingual speech recognition system consists of the following components represented in Fig. 1:

- ***Client*** : a mobile phone, browser, or similar internet connected device capable of recording audio, issuing voice requests and displaying results.
- ***Frontend***: an HTTP server exposing a voice recognition API, which serves as an intermediary between the client and the backend(s). The frontend supports several activity pipelines. For example, one activity called "recognize" simply converts an audio request to a text transcription. A more complex activity pipeline called "recognize + search + synthesize" executes a web search based on the transcription, and then synthesizes an audio summary for the user. Frontend activity pipelines are composed of modules, each of which performs some task such as communicating with a backend over RPC. In the multi-language configuration, we add an additional MultiRecognizer module which is responsible for making language selection decisions.
- ***Backends***:
  - **LID Backend**: a Remote Procedure Call (RPC) server that accepts an audio stream and returns language identification scores for many languages, as described in Section IV-A.
  - **Speech Recognizer Backend**: hereafter Recognizer, an RPC server that accepts an audio stream and streams back transcription results for a particular language (e.g., US English), as described in Section IV-B .
  - **Web Search Backend**: an RPC server that accepts a recognition transcription and retrieves search results. The results sometimes include a text summary which is intended to be read back to the user. Search results are also determined by the selected language, which affects the search corpus, query normalization procedure, and result summarization procedure.
  - **Voice Synthesizer Backend**: an RPC server that accepts text (in this case the search summary), and produces a speech waveform. Again, voice synthesis is influenced by the selected language. For example, depending on which of two dialects is selected, the speech output will be accented toward that dialect.

All components communicate via bi-directional streams. The Client begins sending audio segments to the Frontend as soon an utterance begins. The Frontend forwards those segments to a Recognizer to obtain partial transcriptions results, which are then returned to the Client for display. Typically, the Client receives several partial transcriptions before the user finishes speaking an utterance, followed by a final transcription. For the purpose of display, each subsequent transcription overrides the previous, providing real-time feedback to the user.

In the single-language voice recognition system, the Client provides one spoken language preference which directs the Frontend to the corresponding Recognizer. In the multi-language system, the client may provide several language candidates, causing the Frontend to forward the audio stream to multiple Recognizers in parallel as well as to the LID



```
while (!strategy.Done()) do
  s = SelectUntil(inputs, strategy.NextDeadline());
  if (s == recognizer) then
    strategy.Add(s.event, s.child);
    queue.Add(s.event, s.child);
  end if
  if (s == langid) then
    strategy.Add(s.event);
  end if
  if (s == deadline) then
    strategy.Add(s.event);
  end if
  if (strategy.NewDecision()) then
    queue.AddDecision(strategy.GetDecision());
  end if
  while (queue.HasOutput()) do
    Output(queue.GetOutput());
  end while
end while
```

Fig. 2. Pseudocode illustrating how the MultiRecognizer accepts input streams from the LID classifier and monolingual recognizers, and emits an output stream emulating a universal recognizer.

Backend. The Frontend then makes real-time language selection decisions based on all of these backend results and some strategy. Only transcription results from the selected language are returned to the Client, thus emulating a universal speech recognizer. In the event that the language selection decision changes mid-stream, the Frontend emits enqueued transcription responses for the new selected language, causing the Client to display the transcription for the new language.

III. Multilingual Recognizer Frontend

A. MultiRecognizer Module

The MultiRecognizer module encapsulates the language selection logic in the Frontend. It accepts as inputs i) a stream of transcription results from multiple Recognizers, where each transcription includes a confidence score indicating the probability that the transcription is correct and ii) a stream of language classification scores from the LID Backend. Its output is a single stream of transcription results. As the MultiRecognizer module receives transcription results from the Recognizers it adds them to a queue. This queue withholds results until appropriate language selection decisions have been made. A Strategy class examines all Recognizer and LID input events and outputs language selection decisions. The decisions are designated either partial or final, and cause the queue to release partial or final transcription results for the selected language. Released results are then returned to the client. Pseudocode for this process is depicted in Fig. 2.

The Strategy class must decide which language to select, and also importantly, *when* to emit its decisions. It will generally make more accurate decisions as more information becomes available. However, delaying a decision may introduce latency, particularly because Recognizers typically respond more slowly when decoding audio from an unexpected language. Note that latency is incurred precisely when the correct Recognizer has finished, but the Strategy is waiting for results from incorrect ones. In principle, latency could also be incurred waiting for classification scores from the LID Backend, but in practice this typically responds faster than any Recognizer, and furthermore can be tuned to provide frequent partial responses (e.g., every 200 ms) which are cumulatively averaged.

Formally, after an input event at time t, the Strategy class updates partial scores for language i, $s_{i,t}$, as a combination of the speech transcription confidence and the LID Backend output probability for that language. In this work, we use linear weighting to combine these as

$$s_{i,t} = \alpha p(\mathcal{W}i|L_i, t_1, \theta_{ASR_i}) + \beta \frac{1}{T_2} \sum_{t=0}^{t_2} p(L_i|t_2, \theta_{LID}) \quad (1)$$

where,

$p(W_i|L_i, t_1, \theta_{ASR_i})$ is the confidence of the last transcription received at time $t_1 \leqq t$ using the Recognizer model for language i, θ_{ASR_i}.

$p(L_i|t_2, \theta_{LID})$ is the language identification output probability for language i, at time $t_2 \leqq t$, produced for the LID model θ_{LID}.

and α and β are adjustable parameters to weight the influence of the Recognizers and the LID system.

Note that one of t_1 or t_2 will be equal to t depending on whether the input event is produced by a Recognizer or by the LID Backend. If the input event was produced by the LID Backend, we do not compute $s_{i,t}$ for languages, i, where the corresponding Recognizer has not yet returned any transcription. The final score for a given language, s_{iT_f}, is also updated as in (1) when a final response for the corresponding Recognizer is received ($t = T_f$).

After the update step, the Strategy class tests for certain decision triggers. In this work we consider three timeout strategies for the final decision trigger. Those are:

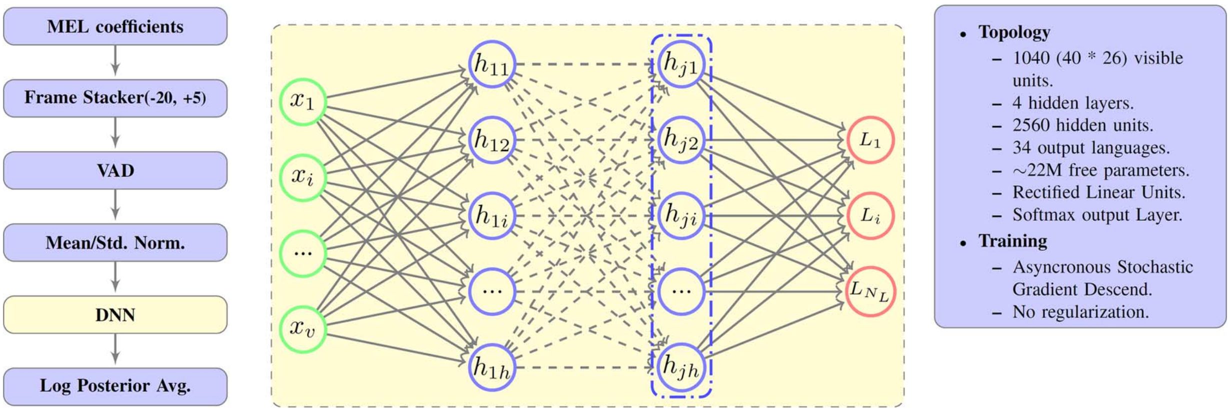


Fig. 3. Pipeline process from the waveform to the final score (left). DNN topology (middle). DNN description (right).

- *Infinite timeout strategy*: always wait for the final response from all Recognizers.
- *Constant timeout strategy*: wait for the first final response from any Recognizer, then timeout after some constant duration (τ_{cte}). Setting τ_{cte} to 1s provided the best experimental results.
- *Variable timeout strategy*: modify the constant timeout duration by examining the difference between the highest available final score and the highest available partial score from a Recognizer that we are still waiting on. Let I be the set of languages that have already received a final Recognizer response and let J be the remaining languages that we are still waiting on; formally we set the variable timeout τ as:

$$\tau = \tau_{cte} \max(1 - \gamma s_\delta, 0) \tag{2}$$

with

$$s_\delta = \max_I(s_{i,T_{fi}}) - \max_J(s_{j,t}). \tag{3}$$

Thus, when s_δ is positive, the timeout duration (τ) is shorter (or zero). On the other hand when s_δ is negative, τ is longer. We tune γ to control the variability of the timeout. Note that if either s_δ or γ is zero, the equation degenerates to the Constant timeout strategy. Setting τ_{cte} to 1s and γ to 2.0 provided the best experimental results.

Given one of the above timeout strategies, the Strategy class uses the following triggers for emitting partial or final language decisions:

- *Partial language decision*: if there exists a previous partial language decision, l, then a new partial decision is triggered when $\arg\max_i(s_{it}) \neq l$. Otherwise, if l does not yet exist, then the trigger is $\max(s_{it}) > 0$.
- *Final language decision*: a final decision is produced when either there is a final response from all the Recognizers ($s_{i,T_{fi}} > 0, \forall i$) or the timeout from the given timeout strategy is exceed ($t > \tau$).

IV. Backends

We describe in this section the language identification and monolingual speech recognition backend engines.

A. Deep Neural Networks for LID

1) Architecture: The DNN used for LID is a fully-connected feed-forward neural network with hidden units implemented as rectified linear units (ReLU). Thus, an input at level j, x_j, is mapped to its corresponding activation y_j (input of the layer above) as

$$y_j = ReLU(x_j) = \max(0, x_j) \tag{4}$$

$$x_j = b_j + \sum_i w_{ij} y_i \tag{5}$$

where i is an index over the units of the layer below and b_j is the bias of the unit j.

The output layer is then configured as a *softmax*, where hidden units map input x_j to a class probability p_j in the form

$$p_j = \frac{\exp(x_j)}{\sum_l \exp(x_l)} \tag{6}$$

where l is an index over all the classes.

As a cost function for backpropagating gradients in the training stage, we use the cross-entropy function defined as

$$C = -\sum_j t_j \log p_j \tag{7}$$

where t_j represents the target probability of the class j for the current evaluated example, taking a value of either 1 (true class) or 0 (false class).

2) Implementing DNN for LID: From the conceptual architecture explained above, we construct a language identification system to work at the frame level as follows.

As the input of the net we used 40 mel filterbanks. Specifically, the input layer was fed with 26 frames formed by stacking the current processed frame and its -20, $+5$ left/right neighbors. Thus, the input layer comprised a total number of $1040 (26 \times 40)$ visible units, v. The reason behind using an asymmetric context is to reduce the undesirable latency on processing every frame in a real-time scenario. Note that just 5 frames in the future are required ($\sim$ 50 ms).

On top of the input layer, we stacked a total number of N_{hl} (4) hidden layers, each containing h (2560) units. Then, we added the softmax layer, whose dimension (s) corresponds to the number of target languages (N_L).

Overall, the net was defined by a total of w free parameters (weights+bias), $w = (v+1)h + (N_{hl}-1)(h+1)h + (h+1)s$. The complete topology of the network is depicted in Fig. 3.

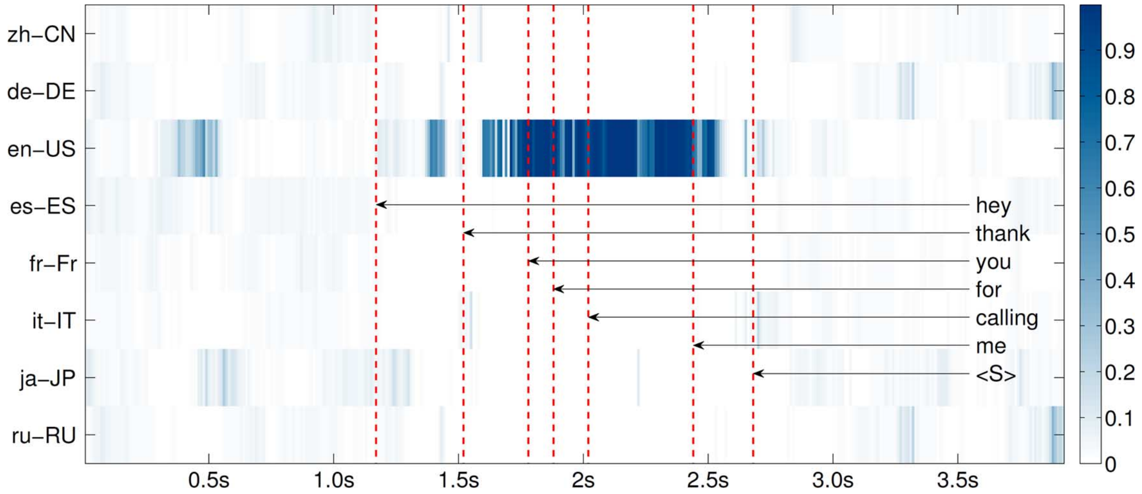


Fig. 4. Color map of frame by frame DNN-based LID system output probabilities (8 languages selected) for an English-USA (4s) test utterance.

Regarding the training procedure, we used asynchronous stochastic gradient descent within the DistBelief framework [10], a software framework that uses computing clusters with thousands of machines. This distributed allows us to exploit large amounts of data to train large models. The learning rate and minibatch size were fixed to 0.001 and 200 samples [1].

Note that the presented architecture works at the frame level, meaning that each single frame (plus its corresponding context) is fed-forward through the network, obtaining a class posterior probability for all of the target languages. This fact makes DNNs particularly suitable for real-time applications since, unlike other approaches (i.e., i-vectors), we can potentially make a decision about the language at each new frame. Indeed, at each frame, we can combine the evidence from past frames to get a single similarity score between the test utterance and the target languages. A simple way of doing this combination is to assume that frames are independent and multiply the posterior estimates of the last layer. The score s_l for the language l of a given test utterance is computed by multiplying the output probabilities p_l obtained for all its frames; or equivalently, accumulating the logs as

$$s_l = \frac{1}{N}\sum_{t=1}^{N} logp(L_l|x_t,\theta) \tag{8}$$

where $p(L_l|x_t,\theta)$ represents the class probability output for the language l corresponding to the input example at time t, x_t by using the DNN defined by parameters θ.

Fig. 4 shows the frame-by-frame DNN outputs (log probabilities $\log p(L_l|x_t,\theta)$), produced for an American English test utterance of 4s duration[2]. The color intensity of each cell (frame/language) represents the output probability of the given frame to belong to the corresponding language.

B. Speech Recognizers

For ASR, we use the same features and similar neural network architectures as are described in Section IV-A for language identification. For each language, we use exactly the same 26-frame asymmetric window of normalized 40-dimensional filterbank energy features, feeding into a network with eight hidden layers of 2560 ReLUs each. The final layer is again a softmax, but here trained to estimate posteriors for 14,000 context dependent triphone states. Such a network has 85 million trainable parameters. The network is trained with asynchronous gradient descent, first to a cross-entropy criterion and then with sequence discriminative training. [18]. Each language's neural network is trained on thousands of hours of speech, using anonymized live traffic utterances. For some languages these are manually-transcribed and in others they are machine transcribed using an earlier generation recognizer. Each language uses a pruned 5-gram language model with on-the-fly rescoring using a much larger 5-gram language model. Numerous optimizations are made to enable decoding in real time on conventional hardware [19].

V. Databases and Evaluation Metrics

1) Google 5M Language Identification Corpus: We generated the Google 5M LID corpus dataset by randomly picking anonymized queries from several Google speech recognition services such as Voice Search or the Speech Android API. Following the user's phone Voice Search language settings, we labelled a total of ~ 5 million utterances, 150 k utterances by 34 different locales (25 languages+9 dialects) yielding ~ 87,5 h of speech per language and a total of ~ 2975 h. A held-out test set of 1k utterances per language was created while the remainder was used for training and development. Involved languages and data description is presented in Figs. 5 and 6.

Non-speech queries were discarded. Selected queries ranged from 1s up to 10s in duration with average speech content of 2.1s. Fig. 6 shows the duration distribution before and after doing this activity detection process.

2) Google Multilang Corpus: In order to constrain the combinatorial explosion of language tuples to be evaluated, we selected 8 languages for the Google Multilang Speech Recognition Corpus. This list is given in Fig. 5. The Google Multilang Speech Recognition Corpus is a subset of the Google 5M LID corpus, containing 1k manually-annotated utterances per language. We refer to [20] for more details on the training data used in our LVCSR systems.

[1] We define *sample* as the input of the DNN: the feature representation of a single frame besides those from its adjacent frames forming the context.

[2] For the sake of clarity, just 8 out of 34 outputs (languages) are showed.

Locale	Language	5M LID	Multilang
ar-EG	Arabic (Egypt)	✓	
ar-GULF	Arabic (Persian Gulf)	✓	
ar-LV	Arabic (Levant)	✓	
bg-BG	Bulgarian	✓	
cs-CZ	Czech	✓	
de-DE	German	✓	✓
en-GB	English (U.K.)	✓	
en-IN	English (India)	✓	
en-US	English (USA)	✓	✓
en-ZA	English (South Africa)	✓	
es-419	Spanish (Latin Am.)	✓	
es-AR	Spanish (Argentina)	✓	
es-ES	Spanish (Spain)	✓	✓
fi-FI	Finish	✓	
fr-FR	French	✓	✓
he-IL	Hebrew (Israel)	✓	
hu-HU	Hungarian	✓	
id-ID	Indonesian	✓	
it-IT	Italian	✓	✓
ja-JP	Japanese	✓	✓
ko-KR	Korean (South Korea)	✓	
ms-MY	Malay	✓	
nl-NL	Dutch	✓	
pt-BR	Portuguese (Brazilian)	✓	
pt-PT	Portuguese (Portugal)	✓	
ro-RO	Romanian	✓	
ru-RU	Russian	✓	✓
sk-SK	Slovak	✓	
sr-RS	Serbian	✓	
sv-SE	Sweden	✓	
tr-TR	Turkish	✓	
zh-CN	Chinese (Mandarin)	✓	✓
zh-TW	Chinese (Taiwan)	✓	
zh-HK	Chinese (Cantonese)	✓	

Fig. 5. List of the Google 5M LID and Google Multilang languages considered.

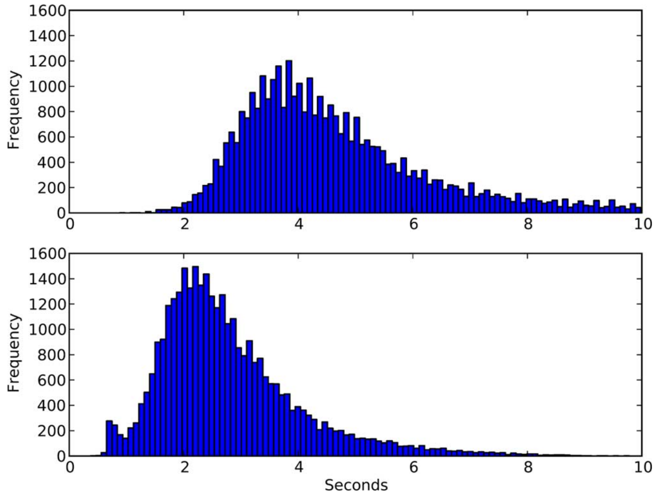


Fig. 6. Histograms of durations of the Google 5M LID and Google Multilang Speech Recognition test utterances. Original speech signals (above) and after voice activity detection (below).

It is worth mentioning that focusing on tuples with up to 8 languages also reduces the computational load on our system. In particular, running many monlingual speech recognizers in parallel is computationally expensive. In the final deployed system, we artificially restrict the number of languages that can be selected, both to improve accuracy and to protect the production system from excessive load.

3) Evaluation Metrics: Given an utterance with a known language and transcription, we evaluate the multilingual recognition system on three metrics:

- Language Id Accuracy: the ratio of the cases in which the top scored language is the true language out of a pool of N candidate languages.
- Word Error Rate (WER): the edit distance between the expected transcription and the system's transcription.
- Real-time Factor (RTF): the ratio of the speech recognition response time to the utterance duration. We examine both mean RTF (average over all utterances), and 90th percentile RTF.

Accuracy is the most direct measurement of the system's language selection performance, but it penalizes language-ambiguous utterances, consisting for example of names and loanwords, where multiple recognizers may produce similar transcripts. The WER metric typically produces a weaker penalty on utterances where the language is ambiguous. It is important to note that the language selection itself may be an important output of the system, independent of the transcription. This is true, for example, in the "multi $-$ recognize $+$ search $+$ synthesize" activity pipeline shown in Fig. 1 where the selected language influences search results and voice synthesis.

For this work we developed a real-time evaluation framework that streams test utterances to the system at their natural playback rate and records responses from the system. A typical test scenario involves choosing some language tuple, sampling test utterances for each language in the tuple, and querying the system with all utterances, providing the unordered tuple as the language candidates. Thus, the system has an equal prior probability of selecting any of the languages in the tuple. The framework collects Accuracy, WER, and RTF for each utterance in the test set.

VI. Results

In this section we aim to study the perceived performance of the multi-language speech recognition system, as compared to a corresponding monolingual system in which the user selects the true spoken language for each utterance in advance. In a multi-language environment, language uncertainty will normally contribute to degraded performance on speech recognition accuracy and latency. In the following subsections we present data measuring the performance of our system along those dimensions.

A. Language Identification Performance

In previous works we showed our DNN-based LID system outperforms previous state-of-the-art approaches [12]. Over the Google 5M LID corpus, our DNN-based system surpassed previous state-of-the-art i-vector-based systems [21], [22] by 70% relative. Fig. 10 (in appendix A) depicts the confusion matrix across languages and dialects. This matrix is built by taking hard identification decisions (i.e., selecting the language with highest

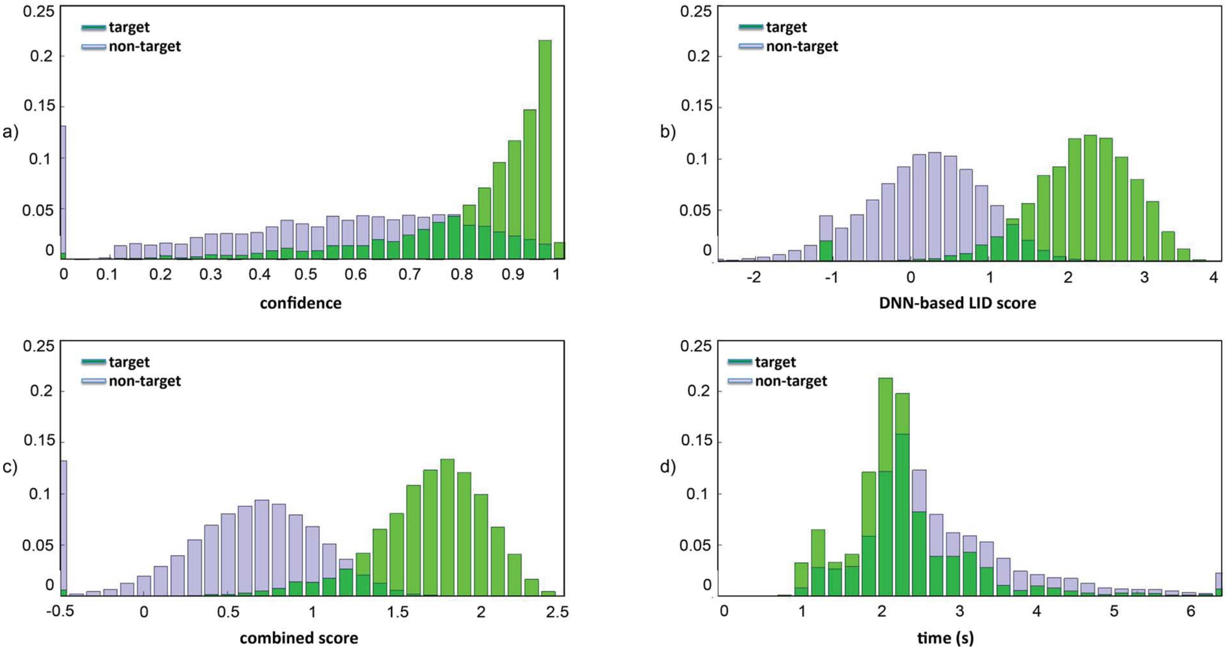


Fig. 7. Language identification over the Google Multilang Corpus. Distribution of target/non-target scores by using a) ASR confidences, b) the DNN-based LID system, c) linear weighting of confidences and DNN-based LID system; and d) as a function of the response time of the speech recognizers.

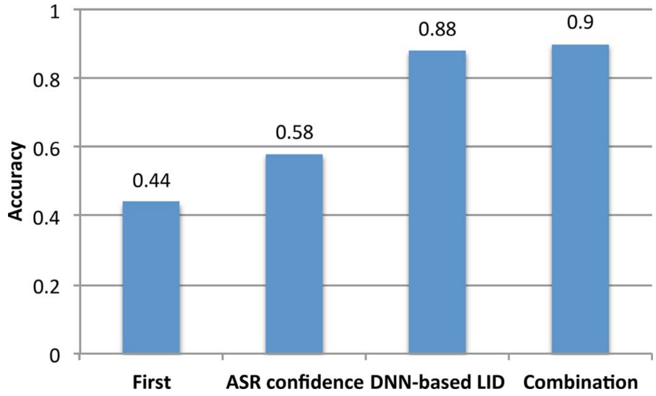


Fig. 8. Language identification performance for individual and combined systems. $N = 8$ over the Google Multilang Corpus.

associated DNN-output). On average the system achieves an accuracy of $\sim$ 80%.

However, confusion submatrices around dialects (i.e ar-EG/ar-GULF/ar-LV) illustrate the difficulty of dialect identification using only acoustic features [23]. This suggests that complementary high-level features (e.g., phonetic, phonotactic or prosodic) would be helpful, particularly when dealing with dialects or non-native accented speech. [24]–[26]. With this in mind, we use ASR confidence scores to complement the acoustic LID classifier, since these confidence scores incorporate language model cost.

Fig. 7 shows various score distributions for trials labelled either target or non-target depending on whether the score corresponds to the correct language for the given utterance. It shows the individual distributions for ASR confidence (7.a) and DNN-based LID scores after applying Z-norm [27] (7.b), as well as the linear weighted combination of the two (7.c). The set of ASR confidence scores with value zero represents trials where a speech recognizer did not emit a transcription. As expected, this occurs most often in non-target trials. The average LID accuracy achieved by these three systems (ASR confidences, DNN-based LID and Combination) for $N = 8$ over the Google Multilang Corpus is presented in Fig. 8. The DNN-only classifier outperformed ASR confidence-only classifier (88% vs 58% of accuracy), but the weighted combination achieved the best results (90% of accuracy).

In Figs. 7 and 8, we also examine ASR response times as a potential signal for language identification. Supplying a wrong-language utterance to a speech recognizer typically results in a wider effective lattice beam, reflecting a greater degree of uncertainty, and therefore a slower average response time. This effect can be observed in (7.d), where we show the distribution of final response times for target and non-target trials. Note that the non-target distribution is shifted to the right ($\sim$ 0.5 s). As a further illustration, in Fig. 8 we present language identification accuracy for a system that simply selects the first language recognizer to return a final transcription. This system achieved an accuracy of 44%, which is significantly better than chance (12.5% when $N = 8$). These results endorse the use of response times as a signal for language identification. In our system, this is exploited through the use of timeout strategies which seek to reduce latency without adversely affecting accuracy.

B. Monolingual Speech Recognition Performance

The monolingual speech recognition performance defines the ceiling for the multilingual recognizer in terms speech recognition performance and latency. We computed WER and RTF for all 8 languages in the Google Multilang Corpus, which are presented in Table I. There is significant variation in the WER metric across languages. This is due to actual variations in recognition quality, caused for example by differences in the type or amount of data used in training [20], and also synthetic variations in the WER metric computation due to text normalization and word tokenization procedures that vary

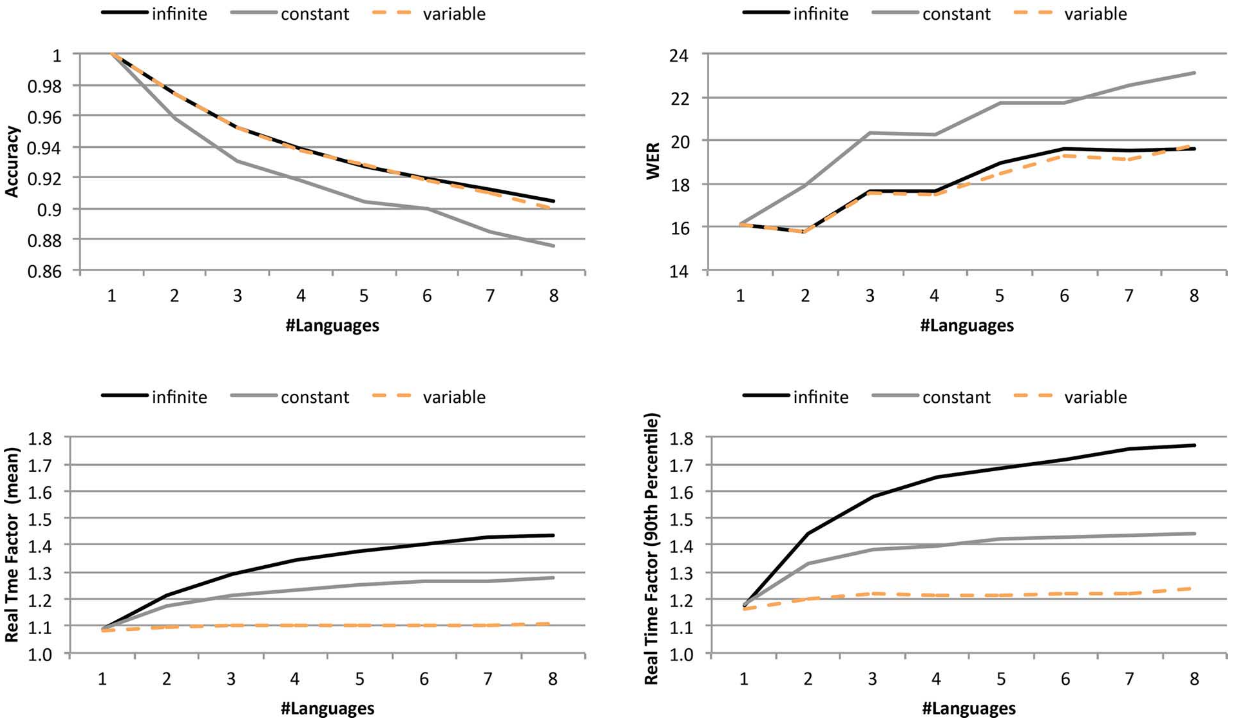


Fig. 9. Language identification accuracy, Word Error Rate (WER) and Real Time Factors (RTF) for the multilingual recognizer as a function of the number of candidate recognition languages.

by language. Thus, it is difficult to draw conclusions from a comparison of absolute WER values across languages. However, WER remains an useful metric for examining relative performance of language selection strategies over the same set of test utterances, such as the results presented in Fig. 9.

C. Multilingual Speech Recognition Performance

In order to examine the performance of the system across a range of language combinations, we adopt the following procedure using test utterances from the Google Multilang Corpus:

- Select a set of N languages.
- Sample U utterances per language.
- Select a tuple size interval $[K_{\min}, K_{\max}]$, bounded by $[1, N]$.
- Select the number of language combinations, M, to test for each tuple size in the interval.
- For each tuple size, k, randomly select M language combinations of that size, or at most the number of k-combinations (N choose k).
- For each language combination, send U utterances from each of the k languages to the multilang system, giving all languages as candidates; record the metrics described in Section V-A3.

The resulting number of trials is:

$$\sum_{k=K_{\min}}^{K_{\max}} U \times \min(M, \binom{N}{k} \times k).$$

In the results below, $N = 8, K = [1, 8], M = 8, U = 1000$, for a total of 232,000 trials. We held the adjustable weighting parameters α and β constant and varied only the timeout strategy (Constant, Infinite, and Variable) as described in Section III-A. The results from all trials with same number of candidate languages were averaged to produce the graphs in Fig. 9.

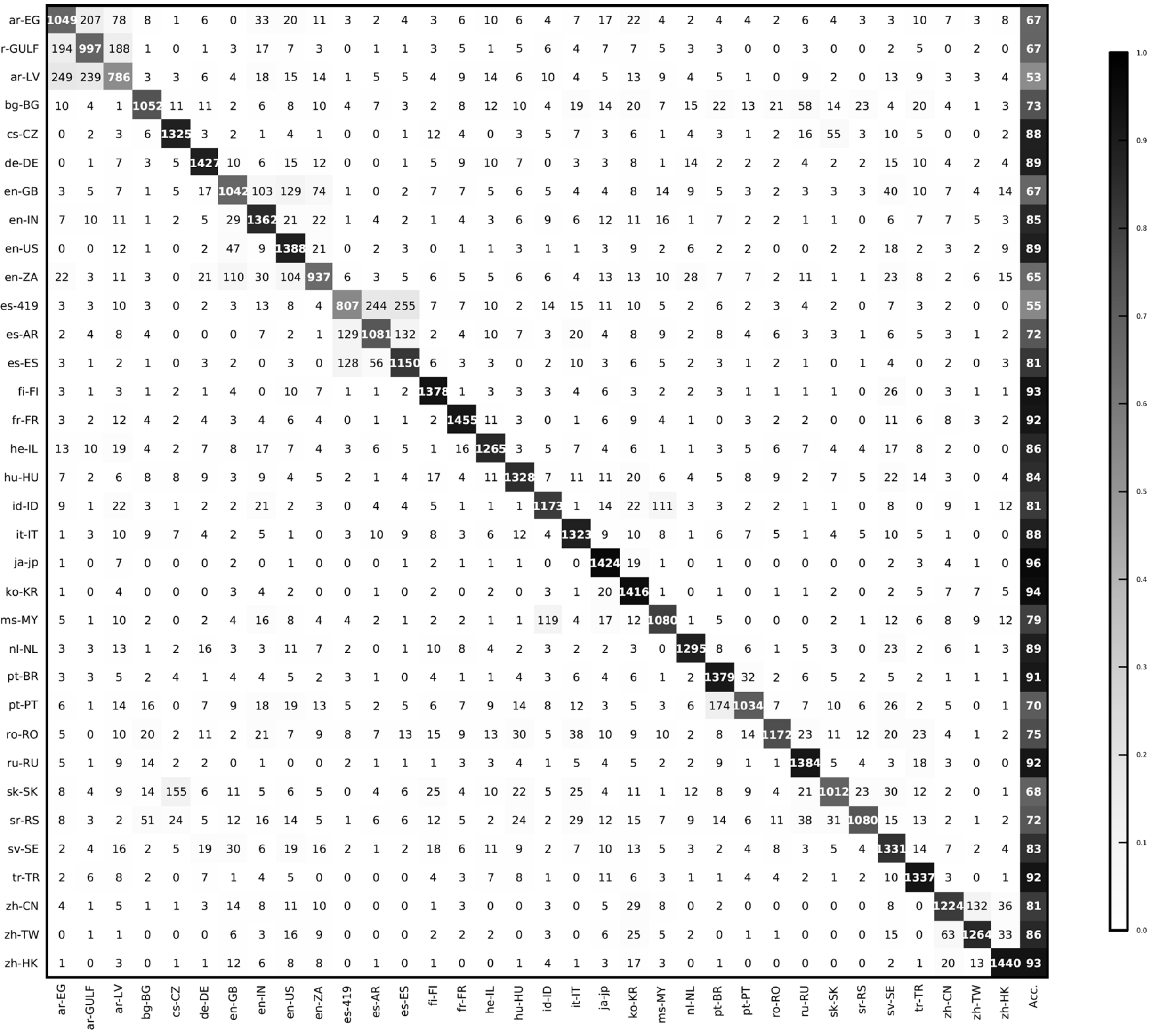
TABLE I
WORD ERROR RATE (WER) AND REAL TIME FACTOR (RTF) OF MONOLINGUAL SPEECH RECOGNIZERS PERFORMANCE

Locale	WER	RT 50%	RT 90%
de-DE	15.58	1.08	1.17
en-US	9.38	1.09	1.19
es-ES	15.05	1.05	1.09
fr-FR	12.88	1.13	1.28
it-IT	12.19	1.07	1.14
ja-JP	20.13	1.08	1.16
ru-RU	25.72	1.11	1.21
zh-CN	18.14	1.09	1.17
Avg.	16.13	1.09	1.18

Looking at the plots for the Infinite timeout strategy, we see that it provides the best Accuracy and WER out of all the strategies, but also had the worst performance on RTF. The plots for the Constant timeout strategy show improvement on RTF at the cost of Accuracy and WER.

The Variable timeout strategy was clearly the best of the three, closely tracking the Accuracy and WER curve of the Infinite timeout strategy, while performing close to the monolingual ceiling on the RTF metric. Indeed, the difference in 90th percentile RTF for $N = 1$ vs $N = 8$ was barely perceptible for the Variable timeout strategy at $\sim 6\%$ relative, compared to a very noticeable $\sim 51\%$ relative for the Infinite timeout strategy.

Under the Infinite (or Variable) strategy, recognition slowly degrades as the number of candidate languages increases. For


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

	ar-EG	ar-GULF	ar-LV	bg-BG	cs-CZ	de-DE	en-GB	en-IN	en-US	en-ZA	es-419	es-AR	es-ES	fi-FI	fr-FR	he-IL	hu-HU	id-ID	it-IT	ja-jp	ko-KR	ms-MY	nl-NL	pt-BR	pt-PT	ro-RO	ru-RU	sk-SK	sr-RS	sv-SE	tr-TR	zh-CN	zh-TW	zh-HK	Acc.
ar-EG	1049	207	78	8	1	6	0	33	20	11	3	2	4	3	6	10	6	4	7	17	22	4	2	4	4	2	6	4	3	3	10	7	3	8	67
ar-GULF	194	997	188	1	0	1	3	17	7	3	0	1	1	3	5	1	5	6	4	7	7	5	3	3	0	0	3	0	0	2	5	0	2	0	67
ar-LV	249	239	786	3	3	6	4	18	15	14	1	5	5	4	9	14	6	10	4	5	13	9	4	5	1	0	9	2	0	13	9	3	3	4	53
bg-BG	10	4	1	1052	11	11	2	6	8	10	4	7	3	2	8	12	10	4	19	14	20	7	15	22	13	21	58	14	23	4	20	4	1	3	73
cs-CZ	0	2	3	6	1325	3	2	1	4	1	0	0	1	12	4	0	3	5	7	3	6	1	4	3	1	2	16	55	3	10	5	0	0	2	88
de-DE	0	1	7	3	5	1427	10	6	15	12	0	0	1	5	9	10	7	0	3	3	8	1	14	2	2	2	4	2	2	15	10	4	2	4	89
en-GB	3	5	7	1	5	17	1042	103	129	74	1	0	2	7	7	5	6	5	4	4	8	14	9	5	3	2	3	3	3	40	10	7	4	14	67
en-IN	7	10	11	1	2	5	29	1362	21	22	1	4	2	1	4	3	6	9	6	12	11	16	1	7	2	2	1	1	0	6	7	7	5	3	85
en-US	0	0	12	1	0	2	47	9	1388	21	0	2	3	0	1	1	3	1	1	3	9	2	6	2	2	0	0	2	2	18	2	3	2	9	89
en-ZA	22	3	11	3	0	21	110	30	104	937	6	3	5	6	5	5	6	6	4	13	13	10	28	7	7	2	11	1	1	23	8	2	6	15	65
es-419	3	3	10	3	0	2	3	13	8	4	807	244	255	7	7	10	2	14	15	11	10	5	2	6	2	3	4	2	0	7	3	2	0	0	55
es-AR	2	4	8	4	0	0	0	7	2	1	129	1081	132	2	4	10	7	3	20	4	8	9	2	8	4	6	3	3	1	6	5	3	1	2	72
es-ES	3	1	2	1	0	3	2	0	3	0	128	56	1150	6	3	3	0	2	10	3	6	5	2	3	1	2	1	0	1	4	0	2	0	3	81
fi-FI	3	1	3	1	2	1	4	0	10	7	1	1	2	1378	1	3	3	3	4	6	3	2	2	3	1	1	1	1	0	26	0	3	1	1	93
fr-FR	3	2	12	4	2	4	3	4	6	4	0	1	1	2	1455	11	3	0	1	6	9	4	1	0	3	2	2	0	0	11	6	8	3	2	92
he-IL	13	10	19	4	2	7	8	17	7	4	3	6	5	1	16	1265	3	5	7	4	6	1	1	3	5	6	7	4	4	17	8	2	0	0	86
hu-HU	7	2	6	8	8	9	3	9	4	5	2	1	4	17	4	11	1328	7	11	11	20	6	4	5	8	9	2	7	5	22	14	3	0	4	84
id-ID	9	1	22	3	1	2	2	21	2	3	0	4	4	5	1	1	1	1173	1	14	22	111	3	3	2	2	1	1	0	8	0	9	1	12	81
it-IT	1	3	10	9	7	4	2	5	1	0	3	10	9	8	3	6	12	4	1323	9	10	8	1	6	7	5	1	4	5	10	5	1	0	0	88
ja-jp	1	0	7	0	0	0	2	0	1	0	0	0	1	2	1	1	1	0	0	1424	19	1	0	1	0	0	0	0	0	2	3	4	1	0	96
ko-KR	1	0	4	0	0	0	3	4	2	0	0	1	0	2	0	2	0	3	1	20	1416	1	0	1	0	1	1	2	0	2	5	7	7	5	94
ms-MY	5	1	10	2	0	2	4	16	8	4	4	2	1	2	2	1	1	119	4	17	12	1080	1	5	0	0	0	2	1	12	6	8	9	12	79
nl-NL	3	3	13	1	2	16	3	3	11	7	2	0	1	10	8	4	2	3	2	2	3	0	1295	8	6	1	5	3	0	23	2	6	1	3	89
pt-BR	3	3	5	2	4	1	4	4	5	2	3	1	0	4	1	1	4	3	6	4	6	1	2	1379	32	2	6	5	2	5	2	1	1	1	91
pt-PT	6	1	14	16	0	7	9	18	19	13	5	2	5	6	7	9	14	8	12	3	5	3	6	174	1034	7	7	10	6	26	2	5	0	1	70
ro-RO	5	0	10	20	2	11	2	21	7	9	8	7	13	15	9	13	30	5	38	10	9	10	2	8	14	1172	23	11	12	20	23	4	1	2	75
ru-RU	5	1	9	14	2	2	0	1	0	0	2	1	1	1	3	3	4	1	5	4	5	2	2	9	1	1	1384	5	4	3	18	3	0	0	92
sk-SK	8	4	9	14	155	6	11	5	6	5	0	4	6	25	4	10	22	5	25	4	11	1	12	8	9	4	21	1012	23	30	12	2	0	1	68
sr-RS	8	3	2	51	24	5	12	16	14	5	1	6	6	12	5	2	24	2	29	12	15	7	9	14	6	11	38	31	1080	15	13	2	1	2	72
sv-SE	2	4	16	2	5	19	30	6	19	16	2	1	2	18	6	11	9	2	7	10	13	5	3	2	4	8	3	5	4	1331	14	7	2	4	83
tr-TR	2	6	8	2	0	7	1	4	5	0	0	0	0	4	3	7	8	1	0	11	6	3	1	1	4	4	2	1	2	10	1337	3	0	1	92
zh-CN	4	1	5	1	1	3	14	8	11	10	0	0	0	1	3	0	0	3	0	5	29	8	0	2	0	0	0	0	0	8	0	1224	132	36	81
zh-TW	0	1	1	0	0	0	6	3	16	9	0	0	0	2	2	2	0	3	0	6	25	5	2	0	1	1	0	0	0	15	0	63	1264	33	86
zh-HK	1	0	3	0	1	1	12	6	8	8	0	1	0	1	0	0	1	4	1	3	17	3	0	1	0	0	0	0	0	2	1	20	13	1440	93

Fig. 10. Confusion matrix obtained by evaluating the DNN-based LID system on the Google 5M LID corpus.

$N = 3$ candidate languages the degradation is around $\sim 10\%$ relative $- \sim 1.5\%$ absolute-. Even for $N = 8$, the performance degradation is still acceptable, $\sim$ 24% relative $-$ $\sim$ 3.5% absolute-.

VII. Summary

Through this paper we presented a novel end-to-end multilingual speech recognizer architecture developed at Google. This architecture supports multiple languages, allowing users to naturally interact with the system in several languages.

The language detection relies on the combination of a specific DNN-based LID classifier and the transcription confidences emitted by the individual speech recognizers. Thus, complementing the acoustic information exploited by the DNN-based LID classifier with the high-level information associated to the language model of the speech recognizer.

Unlike other approaches, this architecture establishes a mechanism to perform language selection in nearly real time. This allows users to transparent switch among different languages under the appearance of using a monolingual ASR.

We assessed the system in terms of both accuracy (speech recognition and LID performance) and response time in a large database including real traffic data and 34 languages. Results show that the proposed architecture is capable of managing multiple languages without significant impact on accuracy and latency compared to our monolingual speech recognizers.

References

[1] G. Tucker and A. Tucker, “A global perspective on bilingualism and bilingual education, ser. ERIC (Collection),” ERIC Clearinghouse on Languages and Linguistics, 1999 [Online]. Available: http://books.google.com/books?id=sEB5tgAACAAJ

[2] F. Grosjean, *Bilingual: Life and reality*. Harvard, MA, USA: Harvard Univ. Press, 2010 [Online]. Available: http://books.google.com/books?id=XgRum7AWOoUC

[3] D. Waggoner, “The growth of multilingualism and the need for bilingual education: What do we know so far?,” *Bilingual Res. J.* vol. 17, no. 1–2, pp. 1–12, 1993 [Online]. Available: http://dx.doi.org/10.1080/15235882.1993.10162645

[4] T. Schultz and A. Waibel, “Language independent and language adaptive large vocabulary speech recognition,” in *Proc. ICSLP*, 1998, vol. 1998, pp. 1819–1822.

[5] H. Lin, L. Deng, J. Droppo, D. Yu, and A. Acero, "Learning methods in multilingual speech recognition," in *Proc. NIPS*, Vancouver, BC, Canada, 2008.

[6] H.-A. Chang, Y. H. Sung, B. Strope, and F. Beaufays, "Recognizing English queries in Mandarin voice search," in *Proc. IEEE Int. Acoust., Speech, Signal Process. (ICASSP), Conf.*, May 2011, pp. 5016–5019.

[7] T. Niesler and D. Willett, "Language identification and multilingual speech recognition using discriminatively trained acoustic models," *Multilingual Speech Lang. Process.*, 2006.

[8] H. Caesar, "Integrating language identification to improve multilingual speech recognition," *Idiap, Idiap-RR Idiap-RR-24–2012*, no. 7, 2012.

[9] H. Lin, J. T. Huang, F. Beaufays, B. Strope, and Y. H. Sung, "Recognition of multilingual speech in mobile applications," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Mar. 2012, pp. 4881–4884.

[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Cambridge, MA, USA: MIT Press, 2012, pp. 1232–1240.

[11] G. Heigold, V. Vanhoucke, A. W. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 8619–8623.

[12] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 5337–5341.

[13] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[14] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[15] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, vol. Abs/1003.0358, 2010.

[16] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing, Exploratory DSP," *IEEE Signal Processing Mag.*, vol. 28, no. 1, pp. 145–154, Jan. 2011.

[17] Y. Bengio, *Learning deep architectures for AI*, ser. Foundations and Trends(r) in Mach. Learning. Delft, The Netherlands: Now Publishers, 2009 [Online]. Available: http://books.google.com/books?id=cq5ewg7FniMC

[18] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Florence, Italy, 2014, pp. 5587–5591.

[19] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

[20] O. Kapralova, J. Alex, E. Weinstein, P. Moreno, and O. Siohan, "A big data approach to acoustic model training corpus selection," in *Proc. Interspeech*, 2014.

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

[22] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in iVectors space," in *Proc. Interspeech. ISCA*, 2011, pp. 861–864.

[23] P. A. Torres-Carrasquillo, D. E. Sturim, D. A. Reynolds, and A. McCree, "Eigen-channel compensation and discriminatively trained Gaussian mixture models for dialect and accent recognition," in *Proc. Interspeech*, 2008, pp. 723–726.

[24] F. Biadsy, "Automatic dialect and accent recognition and its application to speech recognition," Ph.D. dissertation, Columbia Univ., New York, NY, USA, 2011.

[25] W. Baker, D. Eddington, and L. Nay, "Dialect identification: The effects of region of origin and amount of experience," *Amer. Speech*, vol. 84, no. 1, pp. 48–71, 2009.

[26] G. Liu, Y. Lei, and J. H. Hansen, "Dialect identification: Impact of difference between read versus spontaneous speech," in *EUSIPCO '10*, 2003–2006.

[27] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Process.*, vol. 10, no. 1/2/3, pp. 42–54, 2000.

Javier Gonzalez-Dominguez received his M.S. degree in computer science in 2005 from Universidad Autonoma de Madrid, Spain. In 2005, he joined the Biometric Recognition Group—ATVS at Universidad Autonoma de Madrid (UAM) as a Ph.D. student. In 2007, he obtained the postgraduate master in computer science and electrical engineering from UAM and received a FPI research fellowship from the Spanish Ministerio de Educacion y Ciencia. His research interests are focused on robust speech, speaker and language recognition. He has been recipient of several awards as the Microsoft Best student paper at SIG-IL 2009 conference ant the Google best Ph.D. Thesis Award at IberSpeech 2012. He has actively participated and led several ATVS systems submitted to the NIST speaker and language evaluation recognition since 2006. During his Ph.D. pursuit he had been member of several research sites as SAIVT-QUT (2008, Brisbane, Australia), TNO (2009, Utrecht, The Netherlands) and Google Inc. Research (2010 New York, USA). Since 2012, he is an Assistant Professor at UAM. During the 2013–2014 term, he is a visiting scientist at the Speech Team in Google Research New York.

David Eustis is a Senior Staff Software Engineer in the Speech Recognition Group at Google where he develops features and infrastructure for Google's Speech Serving System. David joined Google in 2005. Prior to Speech, he worked in Google's Geographic Data Group where he led the development of Google's Geographic Entity Database, and was an early contributor to Google's Distributed Geographic Data Processing Pipelines. He received his Sc.B. degree in computer science and applied math in 2005 from Brown University and was awarded the Susan Colver Rosenberger Senior Prize in Computer Science. At Brown, he participated in research with the Brown Graphics Group at their CAVE virtual reality environment, and interned at the Los Alamos National Lab's CAVE environment as part of a BGG-LANL collaboration.

Ignacio Lopez-Moreno is a Software Engineer in the Speech Recognition Group at Google where he develops language and speaker recognition technologies and services. He received his M.S. degree in electrical engineering in 2009 from Universidad Politecnica de Madrid (UPM). He is currently pursuing his Ph.D. degree with the Biometric Recognition Group—ATVS at Universidad Autonoma de Madrid. He has participated several technology evaluations, such as NIST speaker and language recognition evaluations carried out since 2005. His research interests include speaker verification, language identification, pattern recognition, speech processing, statistics and forensic evaluation of the evidence. He has been recipient of several awards and distinctions, such as the IBM Research Best Student Paper in 2009.

Andrew Senior received his Ph.D. from Cambridge University for his thesis "Recurrent Neural Networks for Offline Cursive Handwriting Recognition." He is a research scientist at Google, New York, where he works in the speech team on deep and recurrent neural networks for acoustic modeling. Before joining Google he worked at IBM Research in the areas of handwriting, audio-visual speech, face and fingerprint recognition as well as video privacy protection and visual tracking.

Françoise Beaufays is a Senior Staff Research Engineer and Manager at Google. She received a B.S. degree in mechanical and electrical engineering from the University of Brussels (ULB) and a Ph.D. in electrical engineering from Stanford University, with a Ph.D. minor in Italian and French literature. She spent the last 20 years working on speech recognition, focusing on building products that make their users happier and more productive.

Pedro J. Moreno is a Research Scientist at Google Inc. working in the New York office. His research interests are speech and multimedia indexing and retrieval, speech and speaker recognition and applications of machine learning to multimedia. Before joining Google, he worked in the areas of text processing, image classification, bioinformatics and robust speech recognition at HP labs where he was on the lead researchers developing SpeechBot, one of the first audio search engines freely available on the web. He received a Ph.D. in electrical and computer engineering from Carnegie Mellon University and was a former Fulbright scholar.

A.3 Frame-by-Frame Language Identification in Short Utterances Using Deep Neural Networks

TITLE	**Frame-by-Frame Language Identification in Short Utterances Using Deep Neural Networks.**	
JOURNAL	Neural Networks, Special Issue on Deep Learning of Representations.	
IMPACT-FACTOR	5-year 3.216, Q1	Computer Science
AUTHORS	Javier González Domínguez	UAM
	Ignacio López Moreno	UAM / Google Inc.
	Pedro J. Moreno	Google Inc.
	Joaquin González Rodríguez	UAM
DOI	10.1016/j.neunet.2014.08.006	
DATE	2015/04	
PUBLISHER	Elsevier Science Publishers	Amsterdam, The Netherlands

Summary: This article proposes the usage of DNN to the Spoken Language Identification task. We explore multiple DNN topologies to recognize over 30 languages form actual Google voice queries, showing that DNN architectures can significantly outperform alternative state-of-the-art approaches.

Contributions: The candidate proposed neural networks as a novel domain of application for language recognition; provided the necessary code to generate neural networks for language recognition; provided the necessary code to generate the baseline i-vector systems; collected approximately one year of labeled data (audio and parameters) from real voice queries; generated baseline i-vector and candidate DNN-based systems with optimized hyperparameters; and collaborated in validating the system with experimental results.

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2015 Special Issue

Frame-by-frame language identification in short utterances using deep neural networks

Javier Gonzalez-Dominguez [a,b,*], Ignacio Lopez-Moreno [a], Pedro J. Moreno [a], Joaquin Gonzalez-Rodriguez [b]

[a] *Google Inc., NY, USA*

[b] *ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain*

ARTICLE INFO

Article history:
Available online 3 September 2014

Keywords:
DNNs
Real-time LID
i-vectors

ABSTRACT

This work addresses the use of deep neural networks (DNNs) in automatic language identification (LID) focused on short test utterances. Motivated by their recent success in acoustic modelling for speech recognition, we adapt DNNs to the problem of identifying the language in a given utterance from the short-term acoustic features. We show how DNNs are particularly suitable to perform LID in real-time applications, due to their capacity to emit a language identification posterior at each new frame of the test utterance. We then analyse different aspects of the system, such as the amount of required training data, the number of hidden layers, the relevance of contextual information and the effect of the test utterance duration. Finally, we propose several methods to combine frame-by-frame posteriors. Experiments are conducted on two different datasets: the public NIST Language Recognition Evaluation 2009 (3 s task) and a much larger corpus (of 5 million utterances) known as Google 5M LID, obtained from different Google Services. Reported results show relative improvements of DNNs versus the *i*-vector system of 40% in LRE09 3 second task and 76% in Google 5M LID.

1. Introduction

Automatic language identification (LID) refers to the process of automatically determining the language in a given speech sample (Muthusamy, Barnard, & Cole, 1994). The need for reliable LID is continuously growing due to several factors. Among them, the technological trend towards increased human interaction using hands-free, voice-operated devices and the need to facilitate the coexistence of a multiplicity of different languages in an increasingly globalized world.

In general, language discriminant information is spread across different structures or levels of the speech signal, ranging from low-level, short-term acoustic and spectral features to high-level, long-term features (i.e. phonotactic, prosodic). However, even though several high-level approaches are used as meaningful complementary sources of information (Ferrer, Scheffer, & Shriberg, 2010; Martinez, Lleida, Ortega, & Miguel, 2013; Zissman, 1996), most LID systems still include or rely on acoustic modelling (Gonzalez-Dominguez et al., 2010; Torres-Carrasquillo et al., 2010), mainly due to their better scalability and computational efficiency.

Indeed, computational cost plays an important role, as LID systems commonly act as a pre-processing stage for either machine systems (i.e. multilingual speech processing systems) or human listeners (i.e. call routing to a proper human operator) (Li, Ma, & Lee, 2013). Therefore, accurate and efficient behaviour in real-time applications is often essential, for example, when used for emergency call routing, where the response time of a fluent native operator is critical (Ambikairajah, Li, Wang, Yin, & Sethu, 2011; Muthusamy et al., 1994). In such situations, the use of high-level speech information may be prohibitive, as it often requires running one speech/phonetic recognizer per target language (Zissman & Berkling, 2001). Lightweight LID systems are especially necessary in cases where the application requires an implementation embedded in a portable device.

Driven by recent developments in speaker verification, the current state of the art in acoustic LID systems involves using *i*-vector front-end features followed by diverse classification mechanisms that compensate speaker and session variabilities (Brummer et al., 2012; Li et al., 2013; Sturim et al., 2011). The *i*-vector is a compact representation (typically from 400 to 600 dimensions) of a

[*] Corresponding author at: ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain. Tel.: +34 914977558.
E-mail address: javier.gonzalez@uam.es (J. Gonzalez-Dominguez).

http://dx.doi.org/10.1016/j.neunet.2014.08.006
0893-6080/© 2014 Elsevier Ltd. All rights reserved.

whole utterance, derived as a point estimate of the latent variables in a factor analysis model (Dehak, Torres-Carrasquillo, Reynolds, & Dehak, 2011; Kenny, Oullet, Dehak, Gupta, & Dumouchel, 2008). However, while proven to be successful in a variety of scenarios, *i*-vector-based approaches suffer from two major drawbacks when coping with real-time applications. First, the *i*-vector is a point estimate and its robustness quickly degrades as the amount of data used to derive it decreases. Note that the smaller the amount of data, the larger the variance of the posterior probability distribution of the latent variables, and thus, the larger the *i*-vector *uncertainty*. Second, in real-time applications, most of the costs associated with *i*-vector computation occur after completion of the utterance, which introduces an undesirable latency.

Motivated by the prominence of deep neural networks (DNNs), which surpass the performance of the previous dominant paradigm, Gaussian mixture models (GMMs), in diverse and challenging machine learning applications – including acoustic modelling (Hinton et al., 2012; Mohamed, Dahl, & Hinton, 2012), visual object recognition (Ciresan, Meier, Gambardella, & Schmidhuber, 2010), and many others (Yu & Deng, 2011) – we previously introduced a successful LID system based on DNNs in Lopez-Moreno et al. (2014). Unlike previous works on using shallow or convolutional neural networks for small LID tasks (Cole, Inouye, Muthusamy, & Gopalakrishnan, 1989; Leena, Srinivasa Rao, & Yegnanarayana, 2005; Montavon, 2009), this was, to the best of our knowledge, the first time that a DNN scheme was applied at a large scale for LID and benchmarked against alternative state-of-the-art approaches. Evaluated using two different datasets—the NIST LRE 2009 (3 s task) and Google 5M LID—this scheme significantly outperformed several *i*-vector-based state-of-the-art systems (Lopez-Moreno et al., 2014).

In the current study, we explore different aspects that affect DNN performance, with a special focus on very short utterances and real-time applications. We believe that the DNN-based system is a suitable candidate for this kind of application, as it could potentially generate decisions at each processed frame of the test speech segment, typically every 10 ms. Through this study, we assess the influence of several factors on the performance, namely: (a) the amount of required training data, (b) the topology of the network, (c) the importance of including the temporal context, and (d) the test utterance duration. We also propose several blind techniques to combine frame-by-frame posteriors obtained from the DNN to get identification decisions.

We conduct the experiments using the following LID datasets: a dataset built from Google data, hereafter, Google 5M LID corpus and the NIST Language Recognition Evaluation 2009 (LRE'09). First, by means of the Google 5M LID corpus, we evaluate the performance in a real application scenario. Second, we check if the same behaviour is observed in a familiar and standard evaluation framework for the LID community. In both cases, we focus on short test utterances (up to 3 s).

The rest of this paper is organized into the following sections. Section 2 defines a reference system based on *i*-vectors. The proposed DNN system is presented in Section 3. The experimental protocol and datasets are described in Section 4. Next, we examine the behaviour of our scheme over a range of configuration parameters in both the task and the neural network topology. Finally, Sections 6 and 7 are devoted to presenting the conclusions of the study and potential future work.

2. Baseline system: *i*-vector

Currently, most acoustic approaches to perform LID rely on *i*-vector technology (Dehak, Kenny, Dehak, Dumouchel, & Ouellet, 2011). All such approaches, while sharing *i*-vectors as a feature representation, differ in the type of classifier used to perform the final language identification (Martinez, Plchot, Burget, Glembek, & Matejka, 2011). In the rest of this section we describe: (a) the *i*-vector extraction procedure, (b) the *i*-vector classifier used in this study, and (c) the configuration details of our baseline *i*-vector system. This system will serve us as the baseline system.

2.1. I-vector extraction

Based on the MAP adaptation approach in a GMM framework (Reynolds, 1995), utterances in language or speaker recognition are typically represented by the accumulated zero- and centred first-order Baum–Welch statistics, N and F, respectively, computed from a Universal Background Model (UBM) λ. For the UBM mixture $m \in 1, \dots, C$, with mean, μ_m, the corresponding zero- and centred first-order statistics are aggregated over all frames of the utterance as

$$N_m = \sum_t p(m|o_t, \lambda) \tag{1}$$

$$F_m = \sum_t p(m|o_t, \lambda)(o_t - \mu_m), \tag{2}$$

where $p(m|o_t, \lambda)$ is the Gaussian occupation probability for the mixture m given the spectral feature observation $o_t \in \Re^D$ at time t.

The total variability model, hereafter TV, can be seen as a classical FA generative model (Bishop, 2007), with observed variables given by the *supervector* (CD × 1) of stacked statistics $F = \{F_1, F_2, \dots, F_C\}$. In the TV model, the vector of hidden variables $w \in \Re^L$ is known as the utterance *i*-vector. Observed and hidden variables are related by the rectangular low rank matrix $T \in \Re^{CD\times L}$

$$N^{-1}F = Tw, \tag{3}$$

where the zero-order statistics N are represented by a block diagonal matrix $\in \Re^{CD\times CD}$, with C diagonal $D \times D$ blocks. The mth component block is the matrix $N_m I_{(D\times D)}$. Given the imposed Gaussian distributions of $p(w)$ and $p(F|w)$, it can be seen that the mean of the posterior $p(w|F)$ is given by

$$w = (I + T^t \Sigma^{-1} NT)^{-1} T^t \Sigma^{-1} F, \tag{4}$$

where $\Sigma \in \Re^{CD\times CD}$ is the diagonal covariance matrix of F. The TV model is thus a data driven model with parameters $\{\lambda, T, \Sigma\}$. Kenny et al. (2008) provides a more detailed explanation of the derivation of these parameters, using the EM algorithm.

2.2. Classification

Since T constrains all the variabilities (i.e. language, speaker, session), and it is shared for all the language models/excerpts, the *i*-vectors, w, can be seen as a new input feature to classify. Further, several classifiers—either discriminative (i.e. Logistic Regression) or generative (i.e. the Gaussian classifier and linear discriminant analysis)—can be used to perform classification (Martinez et al., 2011). In this study, we utilized LDA, followed by cosine distance (LDA_CS), as the classifier.

Even though using a more sophisticated classifier (Lopez-Moreno et al., 2014) would have resulted in slightly increased performance, we chose the LDA_CS considering the trade-off between performance and computational time efficiency. In this framework, the similarity measure (score) of the two given *i*-vectors, w_1 and w_2, is obtained as

$$S_{w_1,w_2} = \frac{(A^t w_1)(A^t w_2)}{\sqrt{(A^t w_1)(A^t w_1)}\sqrt{(A^t w_2)(A^t w_2)}} \tag{5}$$

where A is the LDA matrix.

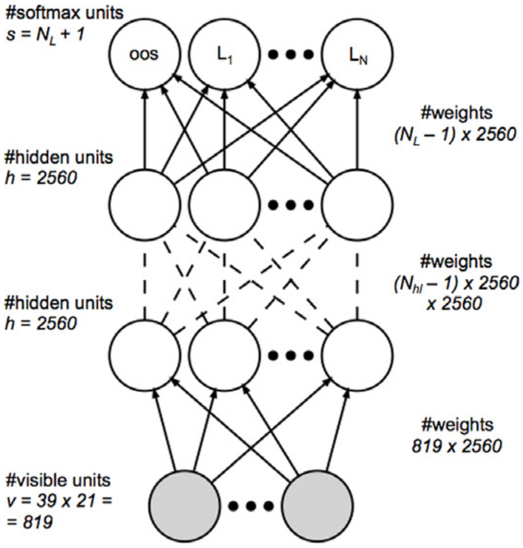


Fig. 1. DNN network topology.

2.3. Feature extraction and configuration parameters

As input features for this study we used perceptual linear predictive (PLP) coefficients (Hermansky, 1990). In particular, 13 PLP coefficients augmented with *delta* and *delta–delta* features (39 dimensions total) were extracted with a 10 ms frame rate over 25 ms long windows. From those features, we built a Universal Background Model of 1024 components. The total variability matrix was trained by using PCA and a posterior refinement of 10 EM iterations (Dehak, Kenny et al., 2011), keeping just the top 400 eigenvectors. We then derived the *i*-vectors using the standard methodology presented in Section 2.1. In addition, we filtered out silence frames by using an energy-based voice activity detector.

3. DNN as a language identification system

Recent findings in the field of speech recognition have shown that significant accuracy improvements over classical GMM schemes can be achieved through the use of deep neural networks, either to generate GMM features or to directly estimate acoustic model scores. Among the most important advantages of DNNs is their multilevel distributed representation of the input (Hinton et al., 2012). This fact makes the DNN an exponentially more compact model than GMMs. In addition DNNs do not require detailed assumptions about the input data distribution (Mohamed, Hinton, & Penn, 2012) and have proven successful in exploiting large amounts of data, reaching more robust models without lapsing into overtraining. All of these factors motivate the use of DNN in language identification. The rest of this section describes the architecture and the practical implementation of the DNN system.

3.1. Architecture

The DNN used in this work is a fully-connected feed-forward neural network with hidden units implemented as rectified linear units (ReLUs). Thus, an input at level j, x_j, is mapped to its corresponding activation y_j (input of the layer above) as

$$y_j = \mathrm{ReLU}(x_j) = \max(0, x_j) \tag{6}$$

$$x_j = b_j + \sum_i w_{ij} y_i \tag{7}$$

where i is an index over the units of the layer below and b_j is the bias of the unit j.

The output layer is then configured as a *softmax*, where hidden units map input x_j to a class probability p_j in the form

$$p_j = \frac{\exp(x_j)}{\sum_l \exp(x_l)} \tag{8}$$

where l is an index over all the classes.

As a cost function for backpropagating gradients in the training stage, we use the cross-entropy function defined as

$$C = -\sum_j t_j \log p_j \tag{9}$$

where t_j represents the target probability of the class j for the current evaluated example, taking a value of either 1 (true class) or 0 (false class).

3.2. Implementing DNN for language identification

From the conceptual architecture explained above, we built a language identification system to work at the frame level as follows.

As the input of the net we used the same features as the *i*-vector baseline system (39 PLP). Specifically, the input layer was fed with 21 frames formed by stacking the current processed frame and its ±10 left/right neighbours. Thus, the input layer comprised a total number of 819 (21 × 39) visible units, v.

On top of the input layer, we stacked a total number of N_{hl} (8) hidden layers, each containing h (2560) units. Then, we added the softmax layer, whose dimension (s) corresponds to the number of target languages (N_L) plus one extra output for the out-of-set (*OOS*) languages. This OOS class, devoted to non-known test languages not seen in training time, could in future allow us to use the system in open-set identification scenarios. Overall, the net was defined by a total of w free parameters (weights + bias), $w = (v+1)h + (N_{hl}-1)(h+1)h + (h+1)s$ ($\sim$48M). The complete topology of the network is depicted in Fig. 1.

Regarding the training procedure, we used asynchronous stochastic gradient descent within the DistBelief framework (Dean et al., 2012), a software framework that uses computing clusters with thousands of machines to train large models. The learning rate and minibatch size were fixed to 0.001 and 200 samples.[1]

Note that the presented architecture works at the frame level, meaning that each single frame (plus its corresponding context) is fed-forward through the network, obtaining a class posterior probability for all of the target languages. This fact makes the DNNs particularly suitable for real-time applications since, unlike other approaches (i.e. *i*-vectors), we can potentially make a decision about the language at each new frame. Indeed, at each frame, we can combine the evidence from past frames to get a single similarity score between the test utterance and the target languages. A simple way of doing this combination is to assume that frames are independent and multiply the posterior estimates of the last layer. The score s_l for the language l of a given test utterance is computed by multiplying the output probabilities p_l obtained for all its frames, or equivalently, accumulating the logs as

$$s_l = \frac{1}{N}\sum_{t=1}^{N} \log p(L_l|x_t, \theta) \tag{10}$$

where $p(L_l|x_t, \theta)$ represents the class probability output for the language l corresponding to the input example at time t, x_t by using the DNN defined by parameters θ.

[1] We define sample as the input of the DNN: the feature representation of a single frame besides those from its adjacent frames forming the context.

Table 1
List of the Google 5M LID (above) and LRE'09 (below) languages considered.

Locale/Abbrev.	Language
Google 5M	
ar-EG	Arabic (Egypt)
ar-GULF	Arabic (Persian Gulf)
ar-LEVANT	Arabic (Levant)
bg-BG	Bulgarian
cs-CZ	Czech
de-DE	German
en-GB	English (United Kingdom)
en-IN	English (India)
en-US	English (USA)
en-ZA	English (South Africa)
es-419	Spanish (Latin America/Caribbean)
es-AR	Spanish (Argentina)
es-ES	Spanish (Spain)
fi-FI	Finish
fr-FR	French
he-IL	Hebrew (Israel)
hu-HU	Hungarian
id-ID	Indonesian
it-IT	Italian
ja-JP	Japanese
ko-KR	Korean (South Korea)
ms-MY	Malay
nl-NL	Dutch
pt-BR	Portuguese (Brazilian)
pt-PT	Portuguese (Portugal)
ro-RO	Romanian
ru-RU	Russian
sk-SK	Slovak
sr-RS	Serbian
sv-SE	Sweden
tr-TR	Turkish
zh-cmn-Hans-CN	Chinese (Mandarin)
zh-cmn-Hant-TW	Chinese (Taiwan)
zh-yue-hant-HK	Chinese (Cantonese)
LRE'09	
en	English (USA)
es	Spanish (Latin America/Caribbean)
fa	Farsi
fr	French
ps	Pashto
ru	Russian
ur	Urdu
zh	Chinese (Mandarin)

Table 2
Data description of the Google 5M LID and LRE09 subcorpus.

Database	$\#N_L$	Train (h)	Test (#files)	Test length (avg. on s)
Google 5M	34	2975	51 000	4.2
LRE09_VOA_3s	8	1600	2 916	3
LRE09_VOA_realtime	8	1600	11 276	(0.1 s–3 s)

4. Datasets and evaluation metrics

We conducted experiments on two different databases following the standard protocol provided by NIST in LRE 2009 (NIST, 2009). Particularly, we used the LRE'09 corpus and a corpus generated from Google Voice services. This followed a two-fold goal: first, to evaluate the proposed methods with a large collection of real application data, and second, to provide a benchmark comparable with other related works in the area by using the well-known LRE'09 framework.

4.1. Databases

4.1.1. Google 5M LID corpus

We generated the Google 5M LID corpus dataset by randomly picking anonymized queries from several Google speech recognition services such as Voice Search or the Speech Android API. Following the user's phone Voice Search language settings, we labelled a total of ∼5 million utterances, 150 k utterances by 34 different locales (25 languages + 9 dialects) yielding ∼87,5 h of speech per language and a total of ∼2975 h. A held-out test set of 1 k utterances per language was created while the remainder was used for training and development. Involved languages and data description are presented in Tables 1 and 2 respectively.

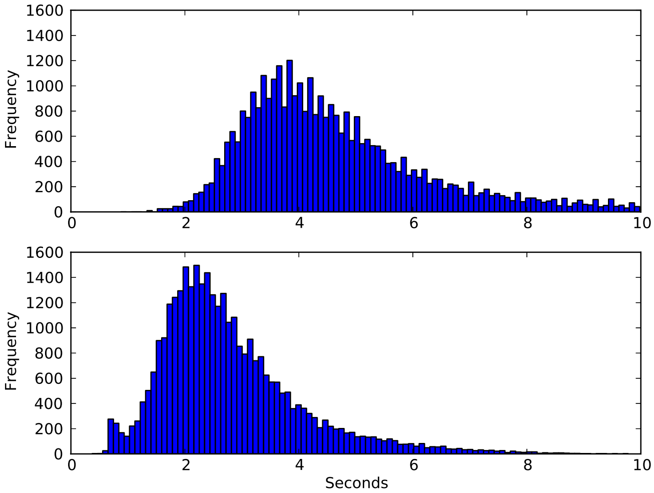


Fig. 2. Histograms of durations of the Google 5M LID test utterances. Original speech signals (above) and after voice activity detection (below).

An automatic speech recognition system was used to discard non-speech queries. Selected queries ranged from 1 up to 10 s in duration with average speech content of 2.1 s. Fig. 2 shows the duration distribution before and after doing this activity detection process.

Privacy issues do not allow Google to link the user identity with the spoken utterance and therefore, determining the exact number of speakers involved in this corpus is not possible. However, it is reasonable to consider that the total number of speakers is very large.

4.1.2. Language recognition evaluation 2009 dataset

The LRE evaluation in 2009 included, for the first time, data coming from two different audio sources. Besides Conversational Telephone Speech (CTS), used in the previous evaluations, telephone speech from broadcast news was used for both training and test purposes. Broadcast data were obtained via an automatic acquisition system from "Voice of America" news (VOA) where telephone and non-telephone speech are mixed.

Due to the large disparity on training material for every language (from ∼10 to ∼950 h), out of the 40 initial target languages (Liu, Zhang, & Hansen, 2012) we selected 8 representative languages for which up to 200 h of audio were available: US English (en), Spanish (es), Dari (fa), French (fr), Pashto (ps), Russian (ru), Urdu (ur), Chinese Mandarin (zh) (Table 1). Further, to avoid misleading result interpretation due to the unbalanced mix of CTS and VOA, all the data considered in this dataset were part of VOA.

As test material in LRE'09, we used a subset of the NIST LRE 2009 3 s condition evaluation set (as for training, we also discarded CTS test segments), yielding a total of 2916 test segments of the 8 selected languages. That makes a total of 23 328 trials. We refer this test dataset as LRE09_VOA_3s_test. For evaluating performance in real-time conditions, we used the VOA test segments for all the LRE'09 conditions (3 s, 10 s, 30 s) with at least 3 s of speech (according to our voice activity detector) that made a total of 11 276 files. Then we cut these recordings to build different duration subsets ranging from 0.1 to 3 s of speech. Specifically, we

Table 3
System performance (EER %) comparison per language on LRE09_VOA_3s_test. The *I*-vector baseline system vs. the DNN_8layers_200h system.

	Equal Error Rate (EER in %)								Avg.
	en	es	fa	fr	ps	ru	ur	zh	
Iv_200h	17.22	10.92	20.03	15.30	19.98	14.87	18.74	10.09	15.89
DNN_8layers_200h	**8.65**	**3.74**	**17.22**	**7.53**	**16.01**	**5.59**	**13.10**	**4.82**	**9.58**

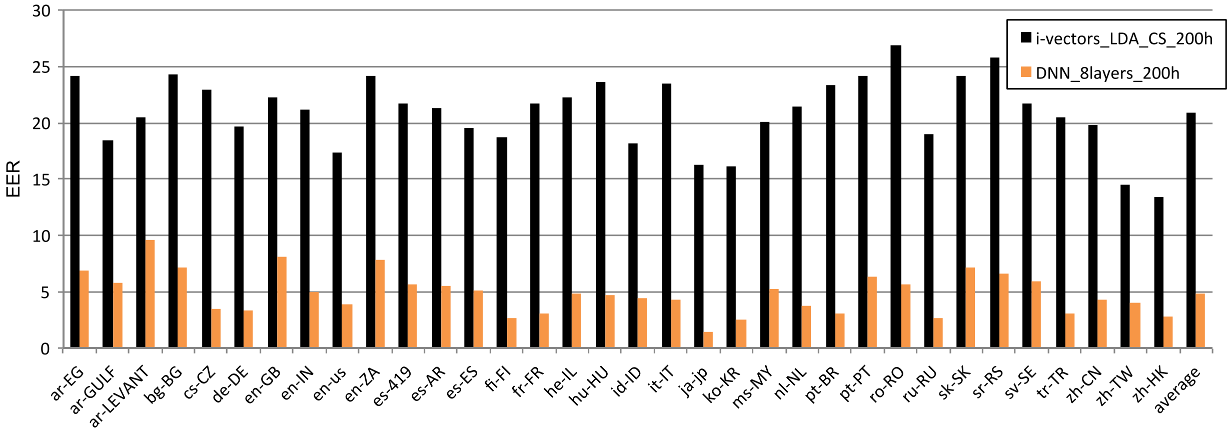


Fig. 3. System performance (EER %) comparison per language on Google 5M LID corpus. The *I*-vector baseline system vs. the DNN_8layers_200h system.

came up with 8 datasets of 11 276 files with durations: 0.1 s, 0.2 s, 0.5 s, 1 s, 1.5 s, 2.0 s, 2.5 s, and 3.0 s. We refer those test datasets as the LRE09_VOA_realtime_test benchmark.

4.2. Evaluation metrics

In order to assess the performance we used the accuracy and equal error rate (EER)[2] metrics. Language identification rates are measured in terms of accuracy, understanding this as the % of correctly identified trials when making hard decisions (by selecting the top scored language) language detection rates are measured in terms of per-language EER and for the sake of clarity we do not deal with the problem of setting optimal thresholds (calibration) as we previously did in Lopez-Moreno et al. (2014).

5. Experimental results

5.1. Global performance

We start our study by comparing the performance of the proposed DNN scheme with the baseline *i*-vector system on the LRE09_VOA_3s_test corpus. Table 3 summarizes this comparison in terms of EER. Results show how the DNN approach largely outperforms the *i*-vector system, obtaining up to a ∼40% relative improvement. An even larger improvement is obtained on the Google 5M corpus, where we found an average relative gain of ∼76% (see Fig. 3). Those results are especially remarkable since they are found on short test utterances and demonstrate the ability of the DNN to exploit discriminative information in large datasets.

It is also worth analysing the errors made by the DNN system as a function of the *similarity* of the different languages. We present in Fig. 7 the confusion matrix obtained using the DNN system on the Google 5M LID corpus. Confusion submatrices around dialects (i.e. ar-EG/ar-GULF/ar-LEVANT) illustrate the difficulty of dialect identification from spectral features in short utterances (Torres-Carrasquillo, Sturim, Reynolds, & McCree, 2008). These results suggest that exploiting just acoustic information might be not enough to reach accurate identification when dealing with dialects (Baker, Eddington, & Nay, 2009; Biadsy, 2011; Liu, Lei, & Hansen, 2010).

5.2. Number of hidden layers and training material

In this section, we evaluate two related aspects when training a DNN: the number of hidden layers and the amount of training material used. On the one hand, we want to exploit the ability showed by DNNs to improve the recognition performance while increasing the training, avoiding overfitting. On the other hand, we aim to get the lightest architecture possible without losing accuracy.

We started by fixing the available training material to its maximum in LRE'09 (200 h per language) and then reducing the number of hidden layers from 8 (DNN_8layers_200h) to 4 (DNN_4layers_200h) and 2 (DNN_2layers_200h). Table 4 summarizes those results. The net with 4 hidden layers seems to be more discriminative than the 2 hidden layers, and more interestingly, than the one with 8 hidden layers. In particular, on average, the DNN_4layers_200h outperforms by ∼8% in terms of EER the DNN_8layers_200h system, using half as many parameters.

As a further step, we swept the number of hours used per language from 1 to 200 h for the three nets. Fig. 4 shows the % accuracy as a function of the training hours per language. As expected, the bigger the amount of training data, the better the performance. However, the slope of this gain degrades when reaching 100 h per language. Indeed, from the 2 layer system, increasing the training material incurs in a minor degradation mostly due to underfitting. Again, it is clear from the results the need for a convenient tradeoff between the training data and number of parameters to optimize. In particular, our best configuration contains ∼21M parameters for ∼648M training samples.

5.3. Real-time identification

Taking now as reference the net with best performance so far (DNN_4_200h) we explored the performance degradation when limiting the test duration. The goal is to gain some insight about

[2] EER is the point on ROC or DET curve where false acceptance and true reject rates are equal.

Table 4
Effect of using different numbers of hidden layers. System performance (EER %) per language on LRE09_VOA_test_3s.

	Equal Error Rate (EER in %)								Avg.
	en	es	fa	fr	ps	ru	ur	zh	
DNN_2layers_200h	12.66	5.04	19.67	8.60	17.84	8.75	14.78	5.54	11.61
DNN_4layers_200h	**8.53**	**3.58**	**16.19**	**5.82**	**15.42**	6.38	**11.24**	**3.16**	**8.79**
DNN_8layers_200h	8.65	3.74	17.22	7.53	16.01	**5.59**	13.10	4.82	9.58

Table 5
Effect of using different left/right input contexts for the DNN_4layers_200h system. System performance (EER %) on the LRE09_VOA_realtime_test (3 s).

	Equal Error Rate (EER in %)								Avg.
	en	es	fa	fr	ps	ru	ur	zh	
No context	19.07	9.65	24.82	13.17	21.64	14.28	19.39	12.38	16.80
±10	8.42	**3.62**	15.89	**5.46**	14.54	6.31	**10.05**	**3.47**	8.47
±20	**7.71**	3.88	**15.49**	6.11	**12.90**	**6.09**	10.50	4.00	**8.33**
±30	9.44	4.53	16.24	7.95	14.40	7.96	12.07	5.23	9.72
±40	12.05	5.08	17.41	9.71	15.47	9.14	13.10	6.27	11.03
±50	9.85	5.71	19.26	8.80	14.54	7.76	13.37	6.51	10.72

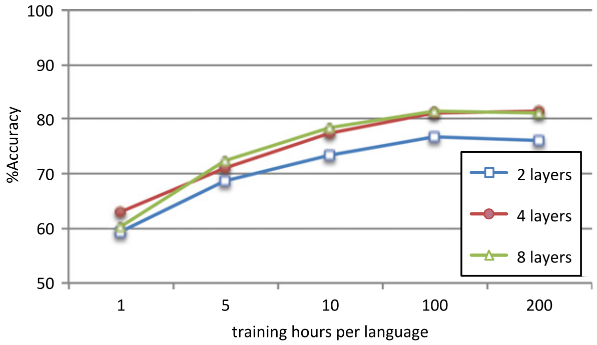


Fig. 4. DNN system performance (% accuracy) in function of the training time per language and the number of hidden layers. Results on LRE09_VOA_3s_test.

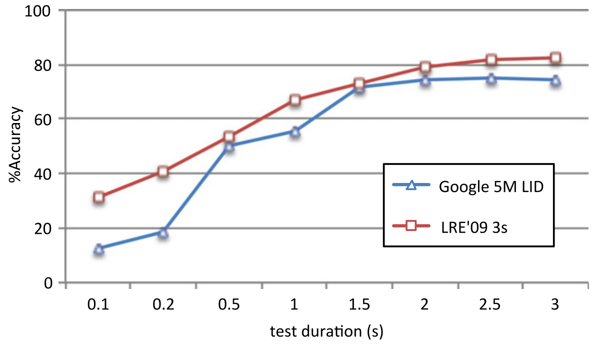


Fig. 5. DNN_4layers_200h system performance (% accuracy) in function of the test utterance duration. Results on LRE09_VOA_realtime_3s.

how long a test utterance must be to consider the identification *accurate*, a main concern in real-time applications.

Fig. 5 shows the average accuracy as a function of the test durations, for both test corpus LRE09_VOA_realtime_test and Google 5M LID. We highlight here two main points. Notice first that up to 0.5 s of speech (according to our voice activity detection) the identification accuracy is very poor (rates under 50% accuracy). Very quick decisions can lead systems to a bad user experience in real-time applications. Second, as expected, the larger the test duration, the better the performance. However, this practically saturates after 2 s. This suggests that a decision can be taken at this point without significant loss of accuracy even when we increase the number of target languages from 9 to 34.

A more detailed analysis per language can be seen in Table 7 for all the 34 languages involved in the Google 5M LID corpus, where we show that the previous conclusion holds true also for each individual language. Confusion matrices on the LRE09_VOA_realtime_test are also collected in Figs. A.8–A.11.

5.4. Temporal context

So far we have been using a fixed right/left context of ±10 frames respectively. That is, the input of our network, as mentioned in Section 3, is formed by stacking the features of every frame with its corresponding 10 to the left and 10 to the right neighbours. We explore in this section the effect of including a shorter/wider context for language identification.

The motivation behind using temporal information from a large number of frames lies in the idea of incorporating additional high-level information (i.e. phonetic, phonotactic and prosodic information). This idea has been largely and successfully exploited in language identification by using long-term phonotactic/prosodic tokenizations (Ferrer et al., 2010; Reynolds et al., 2003) or, in acoustic approaches, by using shifted-delta-cepstral features (Torres-Carrasquillo, Singer, Kohler, & Deller, 2002).

We modify the input of the network by stacking each frame with a symmetric context that ranges from 0 to 50 left and right neighbour frames; that is, we sweep from a context-free scheme to a maximum context that spans 0.5 s to the left and 0.5 s to the right (a total of 1 s context).

Table 5 summarizes the obtained results on the LRE09_VOA_realtime_test (3 s subcorpus) using the DNN_4_200h network. The importance of the context is apparent from first two rows. We observe a relative improvement of ∼49% from the ±10 context scheme with respect to the context-free one. We find the lowest EER when using ±20 frames of context. After this value the EER increases. This behaviour can be explained by understanding that as we demand our net to learn more 'high-level' rich features, we are also increasing the size of the input, therefore forcing the net to learn more complex features from the same amount of data. Fig. 6 collectsthe top 10 filters for a given minibatch (those which produce highest activations in the minibatch) extracted from the first hidden layer for the DNN_4_200h network. The distribution of those weights evidences how the DNN is using the context information.

Although the number of parameters of the input layer is affected by the size of the contextual window, the input layer represents less than the 25% of the model size. Thus, it seems that DNNs can lead to better modelling of the contextual information than competing approaches, such as GMM-based systems, which are traditionally more affected by the *curse of dimensionality*. Note that

Table 6
Comparison of different frame combination schemes for the DNN_4layers_200h. System performance (EER %) per language. Results on LRE09_VOA_3s_test.

	Equal Error Rate (EER in %)								Avg.
	en	es	fa	fr	ps	ru	ur	zh	
Product	**8.53**	**3.58**	**16.19**	**5.82**	**15.42**	6.38	**11.24**	**3.16**	**8.79**
Voting	12.66	5.04	19.67	8.60	17.84	8.75	14.78	5.54	11.61
Entropy	8.65	3.74	17.22	7.53	16.01	**5.59**	13.10	4.82	9.58

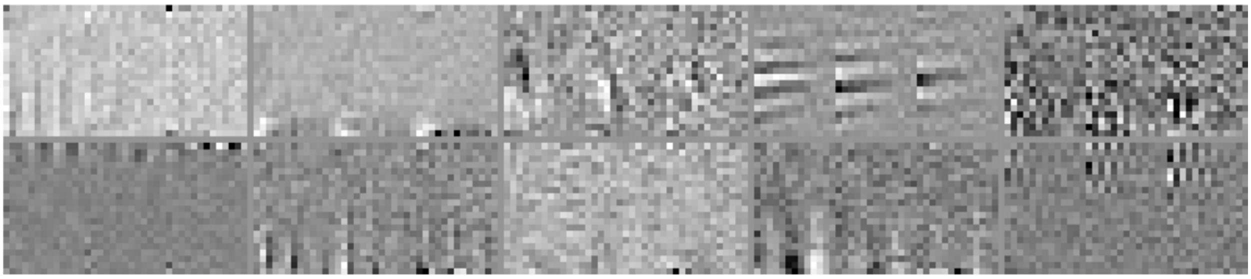
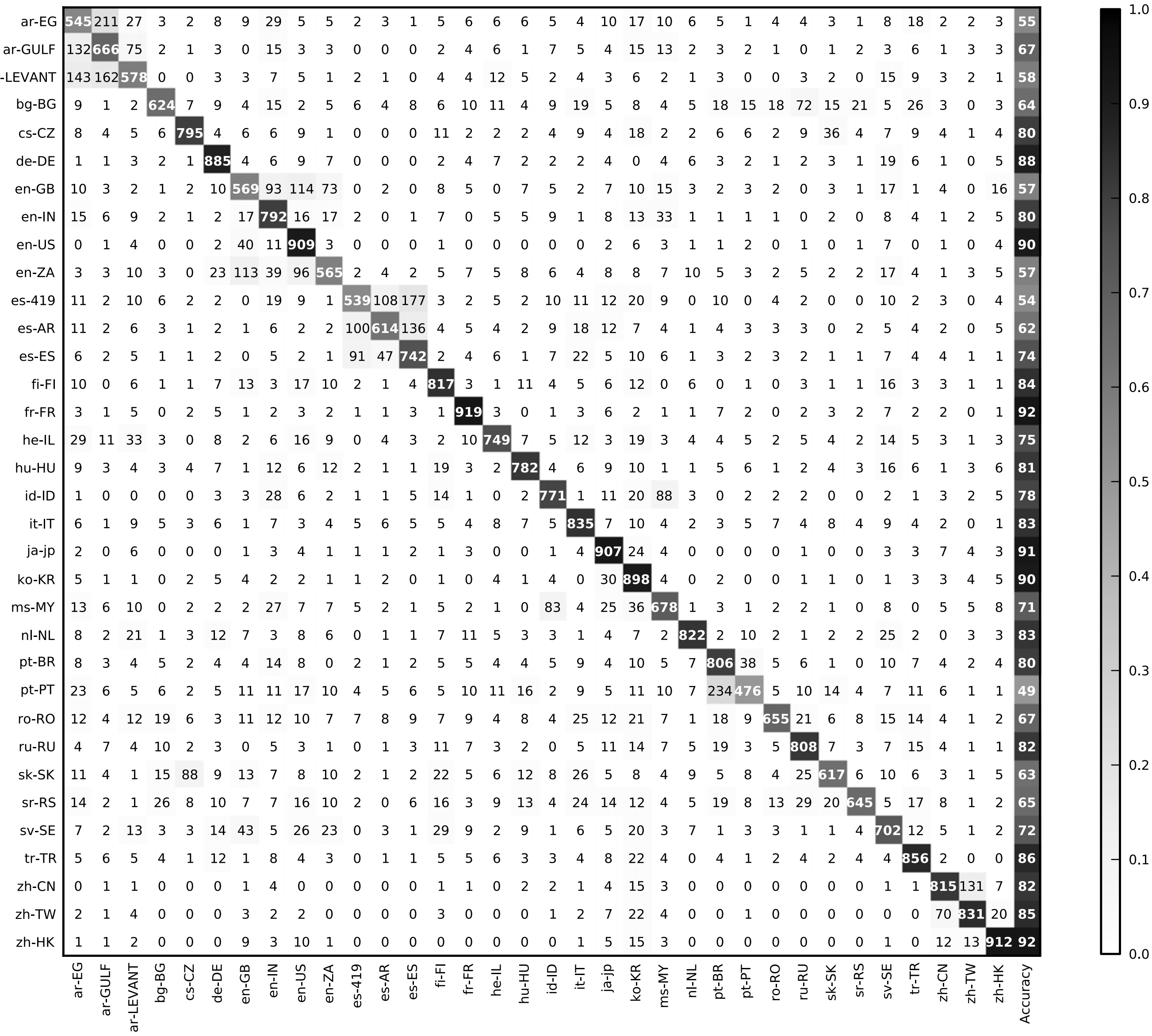


Fig. 6. Visualization of top 10 filters (those which produce highest activations in the given minibatch) of the first hidden layer using a ±10 context. Each filter is composed of 21 rows (number of frames stacked as input) and 39 columns (feature dimension).



	ar-EG	ar-GULF	ar-LEVANT	bg-BG	cs-CZ	de-DE	en-GB	en-IN	en-US	en-ZA	es-419	es-AR	es-ES	fi-FI	fr-FR	he-IL	hu-HU	id-ID	it-IT	ja-jp	ko-KR	ms-MY	nl-NL	pt-BR	pt-PT	ro-RO	ru-RU	sk-SK	sr-RS	sv-SE	tr-TR	zh-CN	zh-TW	zh-HK	Accuracy
ar-EG	545	211	27	3	2	8	9	29	5	5	2	3	1	5	6	6	6	5	4	10	17	10	6	5	1	4	4	3	1	8	18	2	2	3	55
ar-GULF	132	666	75	2	1	3	0	15	3	3	0	0	0	2	4	6	1	7	5	4	15	13	2	3	2	1	0	1	2	3	6	1	3	3	67
ar-LEVANT	143	162	578	0	0	3	3	7	5	1	2	1	0	4	4	12	5	2	4	3	6	2	1	3	0	0	3	2	0	15	9	3	2	1	58
bg-BG	9	1	2	624	7	9	4	15	2	5	6	4	8	6	10	11	4	9	19	5	8	4	5	18	15	18	72	15	21	5	26	3	0	3	64
cs-CZ	8	4	5	6	795	4	6	6	9	1	0	0	0	11	2	2	2	4	9	4	18	2	2	6	6	2	9	36	4	7	9	4	1	4	80
de-DE	1	1	3	2	1	885	4	6	9	7	0	0	0	2	4	7	2	2	2	4	0	4	6	3	2	1	2	3	1	19	6	1	0	5	88
en-GB	10	3	2	1	2	10	569	93	114	73	0	2	0	8	5	0	7	5	2	7	10	15	3	2	3	2	0	3	1	17	1	4	0	16	57
en-IN	15	6	9	2	1	2	17	792	16	17	2	0	1	7	0	5	5	9	1	8	13	33	1	1	1	1	0	2	0	8	4	1	2	5	80
en-US	0	1	4	0	0	2	40	11	909	3	0	0	0	1	0	0	0	0	0	2	6	3	1	1	2	0	1	0	1	7	0	1	0	4	90
en-ZA	3	3	10	3	0	23	113	39	96	565	2	4	2	5	7	5	8	6	4	8	8	7	10	5	3	2	5	2	2	17	4	1	3	5	57
es-419	11	2	10	6	2	2	0	19	9	1	539	108	177	3	2	5	2	10	11	12	20	9	0	10	0	4	2	0	0	10	2	3	0	4	54
es-AR	11	2	6	3	1	2	1	6	2	2	100	614	136	4	5	4	2	9	18	12	7	4	1	4	3	3	3	0	2	5	4	2	0	5	62
es-ES	6	2	5	1	1	2	0	5	2	1	91	47	742	2	4	6	1	7	22	5	10	6	1	3	2	3	2	1	1	7	4	4	1	1	74
fi-FI	10	0	6	1	1	7	13	3	17	10	2	1	4	817	3	1	11	4	5	6	12	0	6	0	1	0	3	1	1	16	3	3	1	1	84
fr-FR	3	1	5	0	2	5	1	2	3	2	1	1	3	1	919	3	0	1	3	6	2	1	1	7	2	0	2	3	2	7	2	2	0	1	92
he-IL	29	11	33	3	0	8	2	6	16	9	0	4	3	2	10	749	7	5	12	3	19	3	4	4	5	2	5	4	2	14	5	3	1	3	75
hu-HU	9	3	4	3	4	7	1	12	6	12	2	1	1	19	3	2	782	4	6	9	10	1	1	5	6	1	2	4	3	16	6	1	3	6	81
id-ID	1	0	0	0	0	3	3	28	6	2	1	1	5	14	1	0	2	771	1	11	20	88	3	0	2	2	2	0	0	2	1	3	2	5	78
it-IT	6	1	9	5	3	6	1	7	3	4	5	6	5	5	4	8	7	5	835	7	10	4	2	3	5	7	4	8	4	9	4	2	0	1	83
ja-jp	2	0	6	0	0	0	1	3	4	1	1	1	2	1	3	0	0	1	4	907	24	4	0	0	0	0	1	0	0	3	3	7	4	3	91
ko-KR	5	1	1	0	2	5	4	2	2	1	1	2	0	1	0	4	1	4	0	30	898	4	0	2	0	0	1	1	0	1	3	3	4	5	90
ms-MY	13	6	10	0	2	2	2	27	7	7	5	2	1	5	2	1	0	83	4	25	36	678	1	3	1	2	2	1	0	8	0	5	5	8	71
nl-NL	8	2	21	1	3	12	7	3	8	6	0	1	1	7	11	5	3	3	1	4	7	2	822	2	10	2	1	2	2	25	2	0	3	3	83
pt-BR	8	3	4	5	2	4	4	14	8	0	2	1	2	5	5	4	4	5	9	4	10	5	7	806	38	5	6	1	0	10	7	4	2	4	80
pt-PT	23	6	5	6	2	5	11	11	17	10	4	5	6	5	10	11	16	2	9	5	11	10	7	234	476	5	10	14	4	7	11	6	1	1	49
ro-RO	12	4	12	19	6	3	11	12	10	7	7	8	9	7	9	4	8	4	25	12	21	7	1	18	9	655	21	6	8	15	14	4	1	2	67
ru-RU	4	7	4	10	2	3	0	5	3	1	0	1	3	11	7	3	2	0	5	11	14	7	5	19	3	5	808	7	3	7	15	4	1	1	82
sk-SK	11	4	1	15	88	9	13	7	8	10	2	1	2	22	5	6	12	8	26	5	8	4	9	5	8	4	25	617	6	10	6	3	1	5	63
sr-RS	14	2	1	26	8	10	7	7	16	10	2	0	6	16	3	9	13	4	24	14	12	4	5	19	8	13	29	20	645	5	17	8	1	2	65
sv-SE	7	2	13	3	3	14	43	5	26	23	0	3	1	29	9	2	9	1	6	5	20	3	7	1	3	3	1	1	4	702	12	5	1	2	72
tr-TR	5	6	5	4	1	12	1	8	4	3	0	1	1	5	5	6	3	3	4	8	22	4	0	4	1	2	4	2	4	4	856	2	0	0	86
zh-CN	0	1	1	0	0	0	1	4	0	0	0	0	0	1	1	0	2	2	1	4	15	3	0	0	0	0	0	0	0	1	1	815	131	7	82
zh-TW	2	1	4	0	0	0	3	2	2	0	0	0	0	3	0	0	0	1	2	7	22	4	0	0	1	0	0	0	0	0	0	70	831	20	85
zh-HK	1	1	2	0	0	0	9	3	10	1	0	0	0	0	0	0	0	0	1	5	15	3	0	0	0	0	0	0	0	1	0	12	13	912	92

Fig. 7. Confusion matrix obtained by evaluating the DNN_8_200h system on the Google 5M LID corpus.

Table 7
System performance (accuracy %) by language and test utterance duration on Google 5M Database.

% Accuracy		Test utterance duration (s)							
Locale	Language	0.1	0.2	0.5	1.0	1.5	2.0	2.5	3.0
ar-EG	Arabic (Egypt)	12	14	38	41	52	56	**58**	57
ar-GULF	Arabic (Persian)	19	22	46	53	61	64	**69**	68
ar-LEVANT	Arabic (Levant)	43	51	54	48	**65**	64	61	62
bg-BG	Bulgarian	7	11	38	51	65	64	68	**72**
cs-CZ	Czech	2	6	45	69	71	75	79	**81**
de-DE	German	18	27	62	72	84	88	87	**89**
en-GB	English (United)	4	12	31	39	49	**54**	**54**	**54**
en-IN	English (India)	27	30	56	63	73	74	76	**78**
en-US	English (USA)	22	29	62	70	85	87	89	**91**
en-ZA	English (South)	4	7	34	45	46	51	56	**57**
es-419	Spanish (Latin)	6	8	25	41	47	50	52	**55**
es-AR	Spanish (Argentina)	6	8	35	50	53	56	58	**61**
es-ES	Spanish (Spain)	5	9	48	54	67	70	72	**73**
fi-FI	Finish	14	23	55	75	76	80	**82**	**82**
fr-FR	French	14	25	69	83	90	93	94	**95**
he-IL	Hebrew (Israel)	4	10	46	60	60	67	68	**70**
hu-HU	Hungarian	8	16	48	71	72	80	**82**	**82**
id-ID	Indonesian	13	21	45	62	69	72	75	**76**
it-IT	Italian	8	13	42	58	75	78	80	**81**
ja-JP	Japanese	18	25	68	87	89	91	94	**95**
ko-KR	Korean (South)	16	25	68	91	89	91	**92**	**92**
ms-MY	Malay	17	25	44	59	63	70	**72**	**72**
nl-NL	Dutch	6	12	56	68	76	80	80	**81**
pt-BR	Portuguese (Brazilian)	11	18	47	74	74	78	80	**81**
pt-PT	Portuguese (Portugal)	6	8	28	53	42	43	48	**49**
ro-RO	Romanian	7	12	34	43	56	61	64	**66**
ru-RU	Russian	5	11	52	70	83	**85**	**85**	**85**
sk-SK	Slovak	10	13	30	40	48	51	55	**58**
sr-RS	Serbian	6	9	35	54	55	59	60	**62**
sv-SE	Sweden	10	16	42	62	65	70	**73**	71
tr-TR	Turkish	5	10	55	78	79	82	83	**85**
zh-cmn-Hans-CN	Chinese (Mandarin)	12	16	54	76	75	76	80	**82**
zh-cmn-Hant-TW	Chinese (Taiwan)	12	22	63	78	80	83	83	**85**
zh-yue-hant-HK	Chinese (Cantonese)	15	25	68	81	88	91	90	**91**

the relative gains reported in this analysis (∼50%) surpass previous attempts reported in the literature in including contextual information using the GMM paradigm (Torres-Carrasquillo et al., 2002). We refer also to Li and Narayanan (2014) for a extensive comparison of different features in language identification over an *i*-vector-based framework.

5.5. *Frame-by-frame posteriors combination*

One of the features that make DNNs particularly suitable for real-time applications is their ability to generate frame-by-frame posteriors. Indeed we can derive decisions about the language identification at each frame. Here we aim to study how we can combine frame posteriors into a single utterance-level score.

Probably the most standard way to perform this combination is assuming frame independence and using the product rule (see Section 3). That is, simply compute the product of the posteriors frame-by-frame as the new single score vector. Another common and simple approach used in the literature is plurality voting, where, at each frame, the language associated with the highest posterior receives a single *vote* while the rest receive none. The voting scheme aims to control the negative effect of outlier scores. The score for a given language l, s_l, is then computed by counting the received votes over all the frames as

$$s_l = \sum_{t=1}^{N} \delta(p(L_l|x_t, \theta)), \tag{11}$$

with δ function defined as

$$\delta(p(L_l|x_t, \theta)) \begin{cases} 1, & \text{if } l = \arg\max_l (p(L_l|x_t, \theta)) \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

A more interesting approach, among *blind* techniques (no need for training), is to weight the posteriors of every frame as a function of the entropy of its posterior distribution. The idea here is to penalize those frames whose distribution of posteriors across the set of languages tends to be uniform (high entropy). This approach was successfully applied in Misra, Bourlard, and Tyagi (2003), resulting in a performance improvement when working with mismatched training and test datasets. The resulting score for language l, s_l, is computed as

$$s_l = \sum_{t=1}^{N} \log\left(\frac{1}{h_t} p(L_l|x_t, \theta)\right) \tag{13}$$

where the weight for frame t is the inverse of its entropy

$$h_t = -\sum_{l=1}^{N} p(L_l|x_t, \theta) \log_2 p(L_l|x_t, \theta). \tag{14}$$

Table 6 compares these three different combination schemes: product, voting and entropy on the LRE09_VOA_3s_test corpus. Results show a better performance of the simple product rule compared to the other approaches, with voting the worst choice. This result suggests that making binary decisions at a frame level leads to a performance degradation. Although the entropy scheme does not help in this scenario, it should be considered when working with more noisy environments.

6. Conclusion

In this work, we present a detailed analysis of the use of deep neural networks (DNNs) for automatic language identification (LID) of short utterances. Guided by the success of DNNs for acoustic modelling in speech recognition, we explore the capacity of DNNs to learn language information embedded in speech signals.

Through this study, we also explore the limits of the proposed scheme for real-time applications, evaluating the accuracy of the system when using very short test utterances (≤3 s). We find, for our proposed DNN scheme, that while more than 0.5 s is needed to obtain over 50% accuracy rates, 2 s are enough to reach accuracy rates of over 90%. Further, we experiment with the amount of training material, the number of hidden layers and the combination of frame posteriors. We also analyse the relevance of including the temporal context, which is critical to achieving high performance in LID.

Results using NIST LRE 2009 (8 languages selected) and Google 5M LID datasets (25 languages + 9 dialects) demonstrate that DNNs outperform current state-of-art *i*-vector-based approaches when dealing with short test durations. Finally, we demonstrated that using a frame-by-frame approach, DNNs can be successful applied for real-time applications.

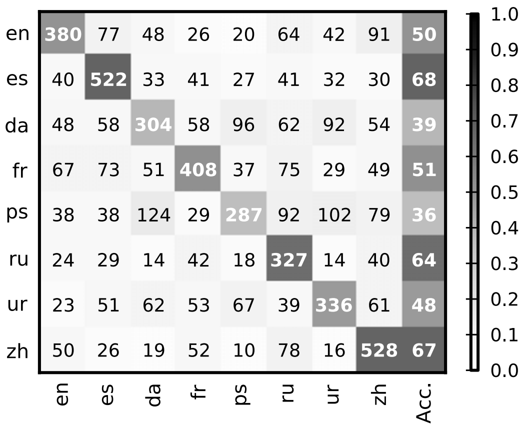
7. Future work

We identified several areas where further investigation is needed. Among them, establishing a more appropriate combination of frame posteriors obtained in DNNs; exploring different fusions among DNNs and *i*-vector systems (Saon, Soltau, Nahamoo, & Picheny, 2013); and dealing with unbalanced training data. Note that even though we proposed different ways of combining posteriors, all of them are blind techniques (no need for training). This fact is due to we focused on real-time applications and simple approaches. However, other data-driven methods could be more appropriate for combining posteriors.

Further other neural network architectures should also be explored. For instance, recurrent neural networks might be an elegant solution to incorporate contextual information. Also, convolutional neural networks might help to reduce the number of parameters of our model.

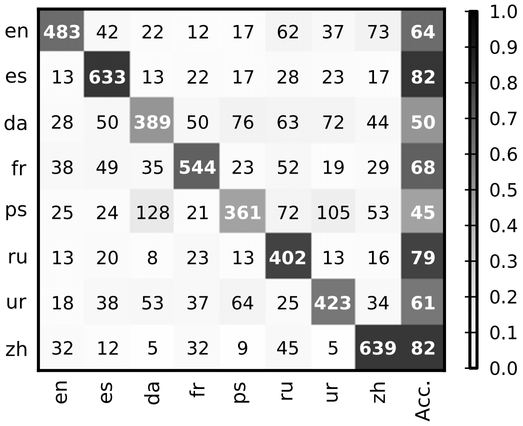
Another promising approach is the use of the activations of the last hidden layer as bottleneck features. Then, *i*-vector-based systems or another classification architecture could be trained over those bottleneck features, rather than over classical features, such as PLP or MFCC.

Appendix. Extended results



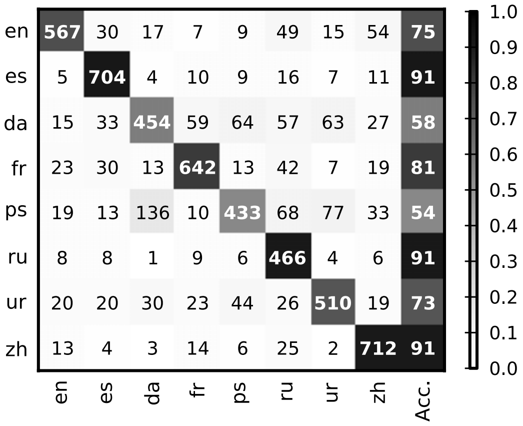
	en	es	da	fr	ps	ru	ur	zh	Acc.
en	380	77	48	26	20	64	42	91	50
es	40	522	33	41	27	41	32	30	68
da	48	58	304	58	96	62	92	54	39
fr	67	73	51	408	37	75	29	49	51
ps	38	38	124	29	287	92	102	79	36
ru	24	29	14	42	18	327	14	40	64
ur	23	51	62	53	67	39	336	61	48
zh	50	26	19	52	10	78	16	528	67

Fig. A.8. DNN_4layers_200h confusion matrix on LRE'09 (0.5 s test).



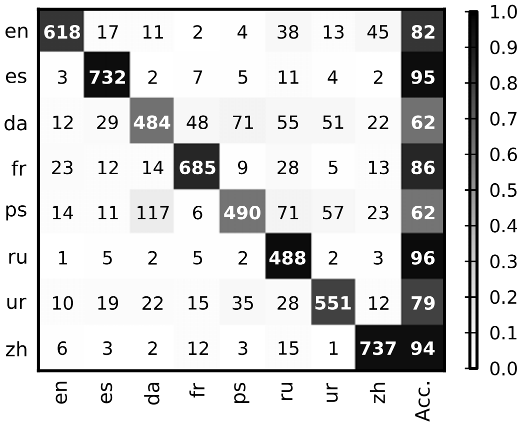
	en	es	da	fr	ps	ru	ur	zh	Acc.
en	483	42	22	12	17	62	37	73	64
es	13	633	13	22	17	28	23	17	82
da	28	50	389	50	76	63	72	44	50
fr	38	49	35	544	23	52	19	29	68
ps	25	24	128	21	361	72	105	53	45
ru	13	20	8	23	13	402	13	16	79
ur	18	38	53	37	64	25	423	34	61
zh	32	12	5	32	9	45	5	639	82

Fig. A.9. DNN_4layers_200h confusion matrix on LRE'09 (1 s test).



	en	es	da	fr	ps	ru	ur	zh	Acc.
en	567	30	17	7	9	49	15	54	75
es	5	704	4	10	9	16	7	11	91
da	15	33	454	59	64	57	63	27	58
fr	23	30	13	642	13	42	7	19	81
ps	19	13	136	10	433	68	77	33	54
ru	8	8	1	9	6	466	4	6	91
ur	20	20	30	23	44	26	510	19	73
zh	13	4	3	14	6	25	2	712	91

Fig. A.10. DNN_4layers_200h confusion matrix on LRE'09 (2 s test).



	en	es	da	fr	ps	ru	ur	zh	Acc.
en	618	17	11	2	4	38	13	45	82
es	3	732	2	7	5	11	4	2	95
da	12	29	484	48	71	55	51	22	62
fr	23	12	14	685	9	28	5	13	86
ps	14	11	117	6	490	71	57	23	62
ru	1	5	2	5	2	488	2	3	96
ur	10	19	22	15	35	28	551	12	79
zh	6	3	2	12	3	15	1	737	94

Fig. A.11. DNN_4layers_200h confusion matrix on LRE'09 (3 s test).

References

Ambikairajah, E., Li, H., Wang, L., Yin, B., & Sethu, V. (2011). Language identification: A tutorial. *IEEE Circuits and Systems Magazine*, *11*(2), 82–108. http://dx.doi.org/10.1109/MCAS.2011.941081.

Baker, W., Eddington, D., & Nay, L. (2009). Dialect identification: The effects of region of origin and amount of experience. *American Speech*, *84*(1), 48–71.

Biadsy, F. (2011). *Automatic dialect and accent recognition and its application to speech recognition*. (Ph.D. thesis). Columbia University.

Bishop, C. (2007). *Information science and statistics, Pattern recognition and machine learning* (1st ed.). Springer.

Brummer, N., Cumani, S., Glembek, O., Karafiát, M., Matejka, P., Pesan, J., et al. (2012). Description and analysis of the Brno276 system for LRE2011. In *Proceedings of Odyssey 2012: The speaker and language recognition workshop* (pp. 216–223). International Speech Communication Association.

Ciresan, D., Meier, U., Gambardella, L., & Schmidhuber, J. (2010). Deep big simple neural nets excel on handwritten digit recognition, CoRR abs/1003.0358.

Cole, R., Inouye, J., Muthusamy, Y., & Gopalakrishnan, M. (1989). Language identification with neural networks: A feasibility study. In *IEEE Pacific Rim conference on communications, computers and signal processing, 1989. Conference proceeding.* (pp. 525–529) http://dx.doi.org/10.1109/PACRIM.1989.48417.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., et al. (2012). Large scale distributed deep networks. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1232–1240).

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, *19*(4), 788–798.

Dehak, N., Torres-Carrasquillo, P. A., Reynolds, D. A., & Dehak, R. (2011). Language recognition via *i*-vectors and dimensionality reduction. In *INTERSPEECH* (pp. 857–860). ISCA.

Ferrer, L., Scheffer, N., & Shriberg, E. (2010). A comparison of approaches for modeling prosodic features in speaker recognition. In *International conference on acoustics, speech, and signal processing* (pp. 4414–4417) http://dx.doi.org/10.1109/ICASSP.2010.5495632.

Gonzalez-Dominguez, J., Lopez-Moreno, I., Franco-Pedroso, J., Ramos, D., Toledano, D., & Gonzalez-Rodriguez, J. (2010). Multilevel and session variability compensated language recognition: ATVS-UAM systems at NIST LRE 2009. *IEEE Journal of Selected Topics in Signal Processing*, *4*(6), 1084–1093. http://dx.doi.org/10.1109/JSTSP.2010.2076071.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, *87*(4), 1738–1752.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. http://dx.doi.org/10.1109/MSP.2012.2205597.

Kenny, P., Oullet, P., Dehak, V., Gupta, N., & Dumouchel, P. (2008). A study of interspeaker variability in speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, *16*(5), 980–988.

Leena, M., Srinivasa Rao, K., & Yegnanarayana, B. (2005). Neural network classifiers for language identification using phonotactic and prosodic features. In *Proceedings of 2005 international conference on intelligent sensing and information processing, 2005* (pp. 404–408) http://dx.doi.org/10.1109/ICISIP.2005.1529486.

Li, H., Ma, B., & Lee, K. A. (2013). Spoken language recognition: From fundamentals to practice. *Proceedings of the IEEE*, *101*(5), 1136–1159. http://dx.doi.org/10.1109/JPROC.2012.2237151.

Li, M., & Narayanan, S. (2014). Simplified supervised *i*-vector modeling with application to robust and efficient language identification and speaker verification, Computer Speech and Language.

Liu, G., Lei, Y., & Hansen, J. H. (2010). Dialect identification: Impact of difference between read versus spontaneous speech. In *EUSIPCO-2010* (pp. 2003–2006).

Liu, G., Zhang, C., & Hansen, J. H. L. (2012). A linguistic data acquisition front-end for language recognition evaluation. In *Proc. Odyssey, Singapore*.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martinez, D., Gonzalez-Rodriguez, J., & Moreno, P. (2014). Automatic language identification using deep neural networks. In *IEEE International conference on acoustics, speech, and signal processing*.

Martinez, D., Lleida, E., Ortega, A., & Miguel, A. (2013). Prosodic features and formant modeling for an ivector-based language recognition system. In *2013 IEEE International conference on acoustics, speech and signal processing, ICASSP* (pp. 6847–6851) http://dx.doi.org/10.1109/ICASSP.2013.6638988.

Martinez, D., Plchot, O., Burget, L., Glembek, O., & Matejka, P. (2011). Language recognition in ivectors space. In *INTERSPEECH* (pp. 861–864). ISCA.

Misra, H., Bourlard, H., & Tyagi, V. (2003). New entropy-based combination rules in HMM/ANN multi-stream ASR. In *2003 IEEE International conference on acoustics, speech, and signal processing, 2003. Proceedings, ICASSP'03, Vol. 2* (pp. II-741-4) http://dx.doi.org/10.1109/ICASSP.2003.1202473.

Mohamed, A., Dahl, G., & Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech and Language Processing*, *20*(1), 14–22. http://dx.doi.org/10.1109/TASL.2011.2109382.

Mohamed, A. Rahman, Hinton, G. E., & Penn, G. (2012). Understanding how deep belief networks perform acoustic modelling. In *ICASSP* (pp. 4273–4276). IEEE.

Montavon, G. (2009). Deep learning for spoken language identification. In *NIPS Workshop on deep learning for speech recognition and related applications*.

Muthusamy, Y., Barnard, E., & Cole, R. (1994). Reviewing automatic language identification. *IEEE Signal Processing Magazine*, *11*(4), 33–41. http://dx.doi.org/10.1109/79.317925.

NIST, 2009. The 2009 NIST SLR Evaluation Plan. www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf.

Reynolds, D. (1995). Speaker identification and verification using Gaussian mixture speaker models. *Speech Communication*, *17*(1–2), 91–108.

Reynolds, D., Andrews, W., Campbell, J., Navratil, J., Peskin, B., & Adami, A. et al. (2003). The SuperSID project: Exploiting high-level information for high-accuracy speaker recognition. In *IEEE International conference on acoustics, speech, and signal processing, Vol. 4* (pp. 784–787).

Saon, G., Soltau, H., Nahamoo, D., & Picheny, M. (2013). Speaker adaptation of neural network acoustic models using *i*-vectors. In *2013 IEEE Workshop on automatic speech recognition and understanding, ASRU* (pp. 55–59) http://dx.doi.org/10.1109/ASRU.2013.6707705.

Sturim, D., Campbell, W., Dehak, N., Karam, Z., McCree, A., & Reynolds, D. et al. (2011). The MIT LL 2010 speaker recognition evaluation system: Scalable language-independent speaker recognition. In *2011 IEEE International conference on acoustics, speech and signal processing, ICASSP* (pp. 5272–5275) http://dx.doi.org/10.1109/ICASSP.2011.5947547.

Torres-Carrasquillo, P., Singer, E., Gleason, T., McCree, A., Reynolds, D., & Richardson, F. et al. (2010). The MITLL NIST LRE 2009 language recognition system. In *2010 IEEE International conference on acoustics speech and signal processing, ICASSP* (pp. 4994–4997) http://dx.doi.org/10.1109/ICASSP.2010.5495080.

Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., & Deller, J. R. (2002). Approaches to language identification using Gaussian mixture models and shifted delta cepstral features. In *ICSLP, Vol. 1* (pp. 89–92).

Torres-Carrasquillo, P. A., Sturim, D. E., Reynolds, D. A., & McCree, A. (2008). Eigen-channel compensation and discriminatively trained Gaussian mixture models for dialect and accent recognition. In *INTERSPEECH* (pp. 723–726).

Yu, D., & Deng, L. (2011). Deep learning and its applications to signal and information processing [exploratory DSP]. *IEEE Signal Processing Magazine*, *28*(1), 145–154. http://dx.doi.org/10.1109/MSP.2010.939038.

Zissman, M. (1996). Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *4*(1), 31–44.

Zissman, M. A., & Berkling, K. (2001). Automatic language identification. *Speech Communication*, *35*(1–2), 115–124.

Appendix B

Full Text of Conference Papers

B.1 End-to-End Text-Dependent Speaker Verification

TITLE	**End-to-End Text-Dependent Speaker Verification.**	
CONFERENCE	ICASSP 2016,	Shanghai, China.
AUTHORS	Georg Heigold	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
AUTHORS	Samy Bengio	Google Inc.
AUTHORS	Noam M. Shazeer	Google Inc.
DOI	10.1109/ICASSP.2016.7472652	
DATE	2016/03	
PUBLISHER	IEEE	New York, NY, US

Summary: In this paper we present a data-driven, integrated approach to speaker verification, which maps a test utterance and a few reference utterances directly to a single score for verification and jointly optimizes the system's components using the same evaluation protocol and metric as at test time. The joint optimization uses asynchronous gradient decent to estimate the parameters of either DNN or LSTM networks. This new approach provided remarkable improvements over alternative baseline models.

Contributions: The candidate collaborated in the proposal of the novel idea of using a loss function that directly optimizes all the stages of a speaker recognition system; generated a labeled dataset (with audio and parameters) from over 20 million recordings and 100,000 speakers; and developed the necessary tools to collect the data. The candidate collaborated in generating LSTM and DNN models with optimized hyperparameters; and validated the system with comparative results from baseline and candidate models.

End-to-End Text-Dependent Speaker Verification

Georg Heigold[*]

Saarland University & DFKI, Germany
georg.heigold@dfki.de

Ignacio Moreno, Samy Bengio, Noam Shazeer

Google Inc., USA
{elnota,bengio,noam}@google.com

Abstract

In this paper we present a data-driven, integrated approach to speaker verification, which maps a test utterance and a few reference utterances directly to a single score for verification and jointly optimizes the system's components using the same evaluation protocol and metric as at test time. Such an approach will result in simple and efficient systems, requiring little domain-specific knowledge and making few model assumptions. We implement the idea by formulating the problem as a single neural network architecture, including the estimation of a speaker model on only a few utterances, and evaluate it on our internal "Ok Google" benchmark for text-dependent speaker verification. The proposed approach appears to be very effective for big data applications like ours that require highly accurate, easy-to-maintain systems with a small footprint.

Index Terms: speaker verification, end-to-end training, deep learning.

1. Introduction

Speaker verification is the process of verifying, based on a speaker's known utterances, whether an utterance belongs to the speaker. When the lexicon of the spoken utterances is constrained to a single word or phrase across all users, the process is referred to as global password text-dependent speaker verification. By constraining the lexicon, text-dependent speaker verification aims to compensate for phonetic variability, which poses a significant challenge in speaker verification [1]. At Google, we are interested in text-dependent speaker verification with the global password "Ok Google." The choice of this particularly short, approximately 0.6 seconds long global password relates to the Google Keyword Spotting system [2] and Google VoiceSearch [3] and facilitates the combination of the systems.

In this paper, we propose to directly map a test utterance together with a few utterances to build the speaker model, to a single score for verification. All the components are jointly optimized using a verification-based loss following the standard speaker verification protocol. Compared to existing approaches, such an end-to-end approach may have several advantages, including the *direct modeling* from utterances, which allows for capturing long-range context and reduces the complexity (one vs. number of frames evaluations per utterance), and the *direct and joint estimation*, which can lead to better and more compact models. Moreover, this approach often results in considerably simplified systems requiring fewer concepts and heuristics.

More specifically, the contributions of this paper include:

- formulation of end-to-end speaker verification architecture, including the estimation of a speaker model on a few utterances (Section 4);
- empirical evaluation of end-to-end speaker verification, including comparison of frame (i-vectors, d-vectors) and utterance-level representations (Section 5.2) and analysis of the end-to-end loss (Section 5.3);
- empirical comparison of feedforward and recurrent neural networks (Section 5.4).

This paper focuses on text-dependent speaker verification for small footprint systems, as discussed in [4]. But the approach is more general and could be used similarly for text-independent speaker verification.

In previous studies, the verification problem is broken down into more tractable, but loosely connected subproblems. For example, the combination of i-vector and probabilistic linear discriminant analysis (PLDA) [5, 6] has become the dominant approach, both for text-independent speaker verification [7, 8, 5, 6] and text-dependent speaker verification [9, 10, 11]. Hybrid approaches that include deep learning based components have also proved to be beneficial for text-independent speaker recognition [12, 13, 14]. For small footprint systems, however, a more direct deep learning modeling may be an attractive alternative [15, 4]. To the best of our knowledge, recurrent neural networks have been applied to related problems such as speaker identification [16] and language identification [17], but not to the speaker verification task. The proposed neural network architecture can be thought of as joint optimization of a generative-discriminative hybrid and is in the same spirit as deep unfolding [18] for adaptation.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of speaker verification in general. Section 3 describes the d-vector approach. Section 4 introduces the proposed end-to-end approach to speaker verification. An experimental evaluation and analysis can be found in Section 5. The paper is concluded in Section 6.

2. Speaker Verification Protocol

The standard verification protocol can be divided into the three steps: training, enrollment, and evaluation, which we describe in more detail next.

Training In the training stage, we find a suitable internal speaker representation from the utterance, allowing for a simple scoring function. In general, this representation depends on the type of the model (e.g., Gaussian subspace model or deep neural network), the representation level (e.g., frame or utterance), and the model training loss (e.g., maximum likelihood or softmax). State-of-the art representations are a summary of frame-level information, such as i-vectors [7, 8] and d-vectors (Section 3).

Enrollment In the enrollment stage, a speaker provides a few utterances (see Table 1), which are used to estimate a speaker model. A common choice is to average the i-vectors [19] or d-vectors [15, 4] of these utterances.

[*] Work done while the author was at Google.

978-1-4799-9988-0/16/$31.00 ©2016 IEEE

Evaluation During the evaluation stage, the verification task is performed and the system is evaluated. For verification, the value of a scoring function of the utterance X and the test speaker spk, $S(X, spk)$, is compared against a pre-defined threshold. We accept if the score exceeds the threshold, i.e., the utterance X comes from speaker spk, and reject otherwise. In this setup, two types of error can occur: false reject and false accept. Clearly, the false reject rate and the false accept rate depend on the threshold. When the two rates are equal, the common value is called equal error rate (EER).

A simple scoring function is the cosine similarity between the speaker representation $f(X)$ of an evaluation utterance X (see paragraph "Training") and the speaker model m_{spk} (see paragraph "Enrollment"):

$$S(X, spk) = [f(X)^\top m_{spk}]/[\|f(X)\| \; \|m_{spk}\|].$$

PLDA has been proposed as a more refined, data-driven scoring approach.

3. D-Vector Baseline Approach

D-vectors are derived from a deep neural network (DNN), as speaker representation of an utterance. A DNN consists of the successive application of several non-linear functions in order to transform the speaker utterance into a vector where a decision can be easily made. Fig. 1 depicts the topology of our baseline DNN. It includes a locally-connected layer [4] and several fully connected layers. All layers use ReLU activation except the last, which is linear. During the training stage, the parameters of the DNN are optimized using the softmax loss, which, for convenience, we define to comprise a linear transformation with weight vectors w_{spk} and biases b_{spk}, followed by the softmax function and the cross-entropy loss:

$$l_{\text{softmax}} = -\log \frac{\exp(w_{spk}^\top y + b_{spk})}{\sum_{\tilde{spk}} \exp(w_{\tilde{spk}}^\top y + b_{\tilde{spk}})}$$

where the activation vector of the last hidden layer is denoted by y and spk denotes the correct speaker. The normalization is over all competing training speakers $\tilde{spk}$.

After the training stage is completed, the parameters of the DNN are fixed. Utterance d-vectors are obtained by averaging the activation vectors of the last hidden layer for all frames of an utterance. Each utterance generates one d-vector. For enrollment, the speaker model is given by the average over the d-vectors of the enrollment utterances. Finally, during the evaluation stage, the scoring function is the cosine similarity between the speaker model d-vector and the d-vector of a test utterance.

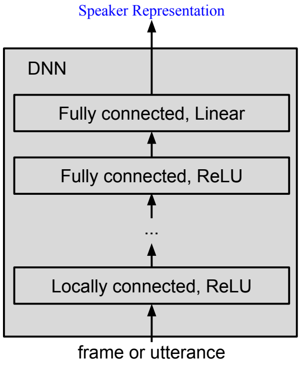


Figure 1: *Deep neural network (DNN) with a locally-connected layer followed by fully-connected layers.*

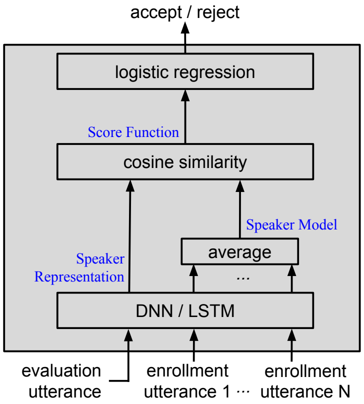


Figure 2: *End-to-end architecture: the input is an "evaluation" utterance and up to N "enrollment" utterances, which the network maps to a single output node (accept/reject). The "enrollment" utterances are used to estimate the speaker model.*

Criticism about this baseline approach includes the limited context of the d-vectors derived from (a window of) frames and the type of the loss. The softmax loss attempts to discriminate between the true speaker and all competing speakers but does not follow the standard verification protocol in Section 2. As a result, heuristics and scoring normalization techniques becomes necessary to compensate for inconsistencies. Moreover, the softmax loss does not scale well with more data as the computational complexity is linear in the number of training speakers and requires a minimum amount of data per speaker to estimate the speaker-specific weights and biases. The complexity issue (but not the estimation issue) can be alleviated by candidate sampling [20].

Similar concerns can be expressed over the alternative speaker verification approaches, where some of the component blocks are either loosely connected or not directly optimized following the speaker verification protocol. For example, GMM-UBM [7] or i-vector models does not directly optimize a verification problem; the PLDA [5] model is not followed a refinement of the i-vector extraction; or long contextual features may be ignored by frame-based GMM-UBM models [7].

4. End-To-End Speaker Verification

In this section, we integrate the steps of the speaker verification protocol (Section 2) into a single network, see Fig. 2. The input of this network consists of an "evaluation" utterance and a small set of "enrollment" utterances. The output is a single node indicating accept or reject. We jointly optimized this end-to-end architecture using DistBelief [21], a predecessor of TensorFlow [22]. In both tools, complex computational graphs such as the one defined by our end-to-end topology can be decomposed into a sequence of operations with simple gradients such as sums, divisions, and cross products of vectors. After the training step, all network weights are kept fixed, except for the bias of the one-dimensional logistic regression (Fig. 2) which is manually tuned on the enrollment data. Apart from this, nothing is done in the enrollment step as the speaker model estimation is part of the network. At test time, we feed an evaluation utterance and the enrollment utterances of a speaker to be tested to the network, which directly outputs the decision.

We use neural networks to obtain the speaker representation

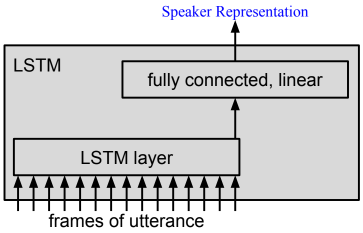


Figure 3: *Long short-term memory recurrent neural network (LSTM) with a single output.*

of an utterance. The two types of networks we use in this work are depicted in Figs. 1 and 3: a deep neural network (DNN) with locally-connected and fully connected layers as our baseline DNN in Section 3 and a long short-term memory recurrent neural network (LSTM) [23, 24] with a single output. DNNs assume a fixed-length input. To comply with this constraint, we stack the frames of a sufficiently large window of fixed length over the utterance and use them as the input. This trick is not needed for LSTMs but we use the same window of frames for better comparability. Unlike vanilla LSTMs which have multiple outputs, we only connect the last output to the loss to obtain a single, utterance-level speaker representation.

The speaker model is the average over a small number of ”enrollment” representations (Section 2). We use the same network to compute the internal representations of the ”test” utterance and of the utterances for the speaker model. The actual number of utterances per speaker available in training typically is much larger (a few hundred or more) than in enrollment (fewer than ten), see Table 1. To avoid a mismatch, we sample for each training utterance only a few utterances from the same speaker to build the speaker model at training time. In general, we cannot assume to have N utterances per speaker. To allow for a variable number of utterances, we pass a weight along with the utterance to indicate whether to use the utterance.

Finally, we compute the cosine similarity between the speaker representation and the speaker model, $S(X, spk)$, and feed it to a logistic regression including a linear layer with a bias. The architecture is optimized using the end-to-end loss

$$l_{e2e} = -\log p(target) \qquad (1)$$

with the binary variable $target \in \{\text{accept}, \text{reject}\}$, $p(\text{accept}) = (1+\exp(-wS(X, spk)-b))^{-1}$, and $p(\text{reject}) = 1-p(\text{accept})$. The value $-b/w$ corresponds with the verification threshold.

The input of the end-to-end architecture are $1 + N$ utterances, i.e., an utterance to be tested and up to N different utterances of the same speaker to estimate the speaker model. To achieve a good tradeoff between data shuffling and memory, the input layer maintains a pool of utterances to sample $1+N$ utterances from for each training step and gets refreshed frequently for better data shuffling. As a certain number of utterances of the same speaker is needed for the speaker model, the data is presented in small groups of utterances of the same speaker.

5. Experimental Evaluation

We evaluate the proposed end-to-end approach on our internal ”Ok Google” benchmark.

5.1. Data Sets & Basic Setup

We tested the proposed end-to-end approach on a set ”Ok Google” utterances collected from anonymized voice search logs. For improved noise robustness, we perform multistyle training. The data were augmented by artificially adding in car and cafeteria noise at various SNRs, and simulating different distances between the speaker and the microphone, see [2] for further details. Enrollment and evaluation data include only real data. Table 1 shows some data set statistics.

Table 1: Data set statistics.

	#utterances (#augmented)	#speakers	#utts / spk
train_2M	2M (9M)	4k	>500
train_22M	22M (73M)	80k	>150
enrollment	18k	3k	1-9
evaluation	20k	3k	3-5

The utterances are forced aligned to obtain the ”Ok Google” snippets. The average length of these snippets is around 80 frames, for a frame rate of 100 Hz. Based on this observation, we extracted the last 80 frames from each snippet, possibly padding or truncating frames at the beginning of the snippet. The frames consist of 40 log-filterbanks (with some basic spectral subtraction) each.

For DNNs, we concatenate the 80 input frames, resulting in a 80x40-dimensional feature vector. Unless specified otherwise, the DNN consists of 4 hidden layers. All hidden layers in the DNN have 504 nodes and use ReLU activation except the last, which is linear. The patch size for the locally-connected layer of the DNN is 10×10. For LSTMs, we feed the 40-dimensional feature vectors frame by frame. We use a single LSTM layer with 504 nodes without a projection layer. The batch size is 32 for all experiments.

Results are reported in terms of equal error rate (EER), without and with t-norm score normalization [25].

5.2. Frame-Level vs. Utterance-Level Representation

First, we compare frame-level and utterance-level speaker representations, see Table 2. Here, we use a DNN as described in Fig. 1 with a softmax layer and trained on train_2M (Table 1) with 50% dropout [26] in the linear layer. The utterance-level approach outperforms the frame-level approach by 30%. Score normalization gives a substantial performance boost (up to 20% relative) in either case. For comparison, two i-vector baselines are shown. The first baseline is based on [6], and uses 13 PLPs with first-order and second-order derivatives, 1024 Gaussians, and 300-dimensional i-vectors. The second baseline is based on [27] with 150 eigenvoices. The i-vector+PLDA baseline should be taken with a grain of salt as the PLDA model was only trained on a subset of the 2M_train data set (4k speakers and 50 utterances per speaker) due to limitations of our current implementation.[1] Also, this baseline does not include other refining techniques such as ”uncertainty training” [10] that have been reported to give substantial additional gains under certain

Table 2: Equal error rates for frame-level and utterance-level speaker representations.

		EER (%)	
level	system	raw	t-norm
frame	i-vector [6]	5.77	5.11
	i-vector+PLDA [27]	4.66	4.89
	DNN, softmax [4]	3.86	3.32
utterance	DNN, softmax	2.90	2.28

[1] However, training with only 30 utterances per speaker gives almost the same results.

conditions. Note that compared to [15], we have improved our d-vectors significantly [4].

5.3. Softmax vs. End-to-End Loss

Next, we compare the softmax loss (Section 2) and end-to-end loss (Section 4) for training utterance-level speaker representations. Table 3 shows the equal error rates for the DNN in Fig. 1. If trained on the small training set (train_2M), the error rates on the raw scores are comparable for the different loss functions. While dropout gives a 1% absolute gain for softmax, we did not observe a gain from dropout for the end-to-end loss. Similarly, t-normalization helps by 20% for softmax, but not at all for the end-to-end loss. This result is in agreement with the degree of consistency between the training loss and the evaluation metric. In particular, the end-to-end approach assuming a global threshold in training (see Eq. (1)), can implicitly learn normalized scores that are invariant under different noise conditions etc. and makes score normalization redundant. When using the softmax DNN for initialization of the end-to-end training, the error rate is reduced from 2.86% to 2.25%, suggesting an estimation problem.

If trained on the larger training set (train_22M), the end-to-end loss clearly outperforms softmax, see Table 3. To reasonably scale the softmax layer to 80k speaker labels, we employed candidate sampling, similar to [20]. Again, t-normalization helps by 20% for softmax and softmax can catch up with the other losses, which do not benefit from t-normalization. The initialization for end-to-end training (random vs. "pre-trained" softmax DNN) does not make a difference in this case.

Although the step time for the end-to-end approach is larger than for softmax with candidate sampling because the speaker model is computed on the fly, the overall convergence times are comparable.

Table 3: Equal error rates for different losses, * is with candidate sampling.

	EER (%), raw / t-norm	
loss	train_2M	train_22M
softmax	2.90 / 2.28	2.69 / 2.08*
end-to-end	2.86 / 2.85	2.04 / 2.14

The optimal choice of the number of utterances used to estimate the speaker model in training, referred to as the speaker model size, depends on the (average) number of enrollment utterances. In practice, however, smaller speaker model sizes may be more attractive to reduce the training time and make the training harder. Fig. 4 shows the dependency of the test equal error rate on the speaker model size, i.e., the number of utterances used to estimate the speaker model. There is a relatively broad optimum around a model size of 5 with 2.04% equal error rate, compared to 2.25% for a model size of 1. This model size is close to the true average model size, which is 6 for our enrollment set. Similar trends are observed for the other configurations in this paper (not shown). This indicates the consistency of the proposed training algorithm with the verification protocol and suggests that task-specific training tends to be better.

5.4. Feedforward vs. Recurrent Neural Networks

So far we focused on the "small footprint" DNN in Fig. 1 with one locally-connected and three fully-connected hidden layers. Next, we explore larger and different network architectures, regardless of their size and computational complexity. The results are summarized in Table 4. Compared to the small footprint DNN, the "best" DNN uses an additional hidden layer and gives a 10% relative gain. The LSTM in Fig. 3 adds another 30% gain over this best DNN. The number of parameters is comparable to that of the DNN but the LSTM involves about ten times more multiplications and additions. More hyperparameter tuning will hopefully bring the computational complexity further down to make it feasible. Slightly worse error rates are achieved with the softmax loss (using t-normalization, candidate sampling, dropout, and possibly early stopping, which were all not needed for the end-to-end approach). On train_2M, we observed similar relative gains in error rate over the respective DNN baselines.

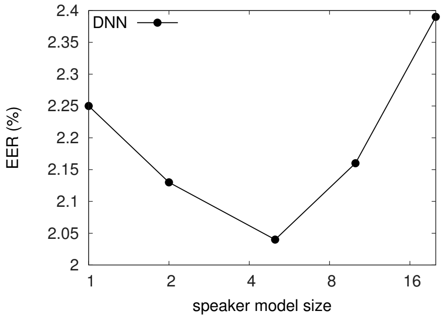


Figure 4: *Speaker model size vs. equal error rate (EER).*

Table 4: Equal error rates for different model architectures using end-to-end training, † is with t-norm score normalization and trained only on the smaller training set.

	EER (%)
frame-level DNN baseline	3.32†
DNN, "small footprint"	2.04
DNN, "best"	1.87
LSTM	1.36

6. Summary & Conclusion

We proposed a novel end-to-end approach to speaker verification, which directly maps the utterance to a score and jointly optimizes the internal speaker representation and the speaker model using the same loss for training and evaluation. Assuming sufficient training data, the proposed approach improved our best small footprint DNN baseline from over 3% to 2% equal error rate on our internal "Ok Google" benchmark. Most of the gain came from the utterance-level vs. frame-level modeling. Compared to other losses, the end-to-end loss achieved the same or slightly better results but with fewer additional concepts. In case of softmax, for example, we obtained comparable error rates only when using score normalization at runtime, candidate sampling to make training feasible, and dropout in training. Furthermore, we showed that the equal error rate can further be reduced to 1.4% using a recurrent neural network instead of a simple deep neural network, although at higher computational runtime cost. By comparison, a reasonable but not fully state-of-the-art i-vector/PLDA system gave 4.7%. Clearly, more comparative studies are needed. Nevertheless, we believe that our approach demonstrates a promising new direction for big data verification applications.

7. References

[1] H. Aronowitz, R. Hoory, J. W. Pelecanos, and D. Nahamoo, "New developments in voice biometrics for user authentication," in *Interspeech*, Florence, Italy, Aug. 2011, pp. 17 – 20.

[2] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, Apr. 2015, pp. 4704–4708.

[3] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, ""Your word is my command": Google search by voice: A case study," in *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer, 2010, ch. 4, pp. 61–90.

[4] Y. Chen, I. Lopez-Moreno, and T. Sainath, "Locally-connected and convolutional neural networks for small footprint speaker recognition," in *Interspeech*, Dresden, Germany, Sep. 2015.

[5] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, Jul. 2010.

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

[7] D. Reynolds, T. Quoter, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.

[8] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1435–1447, 2007.

[9] H. Aronowitz, "Text-dependent speaker verification using a small development set," in *Proc. Odyssey Speaker and Language Recognition Workshop*, Singapore, Jun. 2012.

[10] T. Stafylakis, P. Kenny, P. Ouellet, P. Perez, J. Kockmann, and P. Dumouchel, "Text-dependent speaker recognition using PLDA with uncertainty propagation," in *Interspeech*, Lyon, France, Aug. 2013.

[11] A. Larcher, K.-A. Lee, B. Ma, and H. Li, "Phonetically-constrained PLDA modeling for text-dependent speaker verification with multiple short utterances," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

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

[13] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 1695–1699.

[14] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, 2005.

[15] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014.

[16] S. Parveen, A. Qadeer, and P. Green, "Speaker recognition with recurrent neural networks," in *Interspeech*, Beijing, China, Oct 2000, pp. 16–20.

[17] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Interspeech*, Singapore, Sep. 2014, pp. 2155–2159.

[18] J. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *CoRR*, vol. abs/1409.2574, 2014.

[19] C. Greenberg, D. Bansé, G. Doddington, D. Garcia-Romero, J. Godfrey, T. Kinnunen, A. Martin, A. McCree, M. Przybocki, and D. Reynolds, "The NIST 2014 speaker recognition i-vector machine learning challenge," in *Odyssey 2014: The Speaker and Language Recognition Workshop*, Joensuu, Finland, Jun. 2014.

[20] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," *CoRR*, vol. abs/1412.2007, 2014.

[21] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, Singapore, Sep. 2014.

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

[26] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[27] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Interspeech*, Florence, Italy, Aug. 2011.

B.2 Locally-Connected and Convolutional Neural networks for Small Footprint Speaker Recognition

TITLE	**Locally-Connected and Convolutional Neural networks for Small Footprint Speaker Recognition.**	
CONFERENCE	Interspeech 2015,	Dresden, Germany.
AUTHORS	Yu-hsin Chen	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
	Tara N. Sainath	Google Inc.
	Mirkó Visontai	Google Inc.
	Mirkó Visontai	Google Inc.
	Raziel Alvarez	Google Inc.
	Carolina Parada	Google Inc.
PAGES	1136-1140	
DATE	2015/09	
PUBLISHER	ISCA	Grenoble, France

Summary: This work compares the performance of deep Locally-Connected Networks (LCN) and Convolutional Neural Networks (CNN) for text-dependent speaker recognition. These topologies model the local time-frequency correlations of the speech signal better, using only a fraction of the number of parameters of a fully connected Deep Neural Network (DNN) used in previous works.

Contributions: The candidate proposed the novel idea of using LCN layers as a mechanism to reduce model footprint and latency without significant performance degradation; generated a labeled dataset (with audio and parameters) from over 2 million recordings and over 10,000 speakers; and developed the necessary tools to collect the data. The candidate provided the necessary code to generate LCN models; optimized hyperparamaters for the generated LCN models; and collaborated in validating the performance with experimental results for the tasks of speaker recognition and key-word spotting.

Locally-Connected and Convolutional Neural Networks for Small Footprint Speaker Recognition

Yu-hsin Chen, Ignacio Lopez-Moreno, Tara N. Sainath,
Mirkó Visontai, Raziel Alvarez, Carolina Parada

Google Inc., USA

yjc@google.com, elnota@google.com, tsainath@google.com,
mirkov@google.com, raziel@google.com, carolinap@google.com

Abstract

This work compares the performance of deep Locally-Connected Networks (LCN) and Convolutional Neural Networks (CNN) for text-dependent speaker recognition. These topologies model the local time-frequency correlations of the speech signal better, using only a fraction of the number of parameters of a fully connected Deep Neural Network (DNN) used in previous works. We show that both a LCN and CNN can reduce the total model footprint to 30% of the original size compared to a baseline fully-connected DNN, with minimal impact in performance or latency. In addition, when matching parameters, the LCN improves speaker verification performance, as measured by equal error rate (EER), by 8% relative over the baseline without increasing model size or computation. Similarly, a CNN improves EER by 10% relative over the baseline for the same model size but with increased computation.

1. Introduction

Speaker Verification (SV) is the process of verifying, based on a speaker's known utterances, whether an utterance belongs to the speaker. When the lexicon of the spoken utterances is constrained to a single word or phrase across all users, the process is referred to as *global* password Text-Dependent Speaker Verification (TD-SV). By constraining the lexicon, TD-SV compensates for phonetic variability, which poses a significant challenge in SV [1]. At Google, we target a global password TD-SV where the spoken password is given by: *"Ok Google"*. This particularly short, approximately 0.6 seconds long global password was chosen as it relates to the Google Keyword Spotting system [2] and Google VoiceSearch [3], facilitating the combination of all three systems.

Our goal is to create a small footprint TD-SV system that can run in real-time in space-constrained mobile platforms. Our constraints are a) total number of model parameters must be small (e.g. 0.8M parameters), and b) total number of operations must be small (e.g. 1.5M multiplications), in order to keep latency below 40ms on most platforms. Previous work [4] introduced our baseline system and compared it to the more standard i-vector approach. This system used a fully-connected Deep Neural Network (DNN) to extract a speaker-discriminative feature, "d-vector", from each utterance. Utterance d-vectors were incrementally computed frame by frame, and improved latency by avoiding the computational costs associated with the latent variables of a factor analysis model [5], which occurred after utterance completion.

In this paper, we explore alternative architectures to the fully-connected feed-forward DNN architecture used to compute d-vectors, with the goal of improving the equal error rate (EER) of the SV system while limiting and even reducing the number of parameters and latency. We explore locally-connected (LCN) and convolutional neural network (CNN) [6] architectures; these architectures focus on exploiting the local correlations of the speech signal. Both LCNs and CNNs are based on local receptive fields (i.e. patches), whose characteristic shape is sparse but locally dense. LCNs and CNNs have been widely used in image processing [7] and more recently in speech processing too [8, 9, 10]. Unlike in previous works, this paper uses LCNs and CNNs to directly compute speaker discriminative features while simultaneously constraining the size and latency of the model. In this paper we show that LCNs and CNNs can reduce the number of parameters in the first hidden layer by an order of magnitude with minimal performance degradation. We also show that for the same number of parameters, LCNs and CNNs can achieve better performance than fully-connected layers. Finally, we propose applying LCNs over CNNs in our global password TD-SV system because LCNs have lower latency.

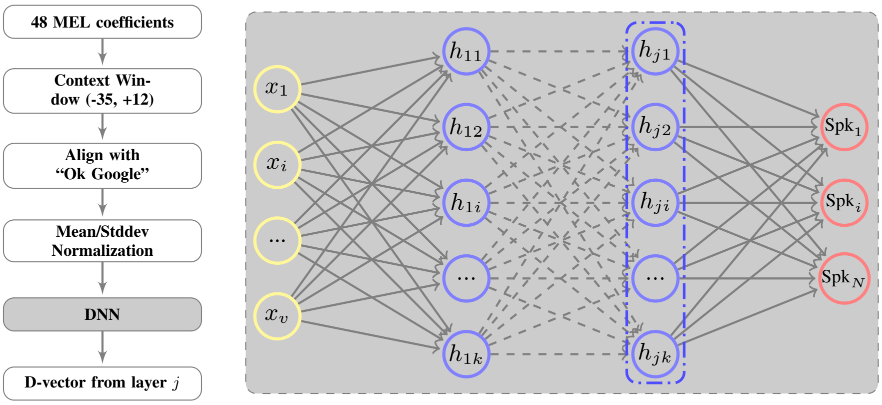
This paper is organized as follows: Section 2 describes the baseline fully-connected d-vector system. Section 3 describes the LCNs and CNNs that are explored in this paper. Section 4 presents the results of two experiments: the first experiment compares models that differ only in the first hidden layer, while the second experiment compares models of the same size. Section 5 concludes the paper.

2. d-vector Baseline Model

Figure 1 contains the complete topology of the baseline fully-connected DNN and its position in the SV pipeline. Let x^t be the input features of the input layer at time t. x^t is formed by stacking q-dimensional mel-filterbank vectors by l contextual vectors to the left and r contextual vectors to the right; the total number of stacked frames is $l + r + 1$. Therefore, there are $v = q(l + r + 1)$ visible units per input x^t. The hidden layers contain units with a rectified linear unit (ReLU) activation. Each hidden layer contains k units.

The output of the DNN is a softmax layer which corresponds to the number of speakers in the development set, N. Each input has a target label, which is an integer corresponding to speaker identity. See [4] for details. The DNN is trained using the cross-entropy criterion.

For enrollment, the parameters of the DNN are fixed. We derive the d-vector speaker feature from output activations of the last hidden layer (before the softmax layer). To compute the d-vector, for every input x^t of a given utterance, we compute



Topology

- $c = 48$ mel-filterbanks.
- $l = 35, r = 12$ context frames.
- $v = 2304$ visible units.
- $M = 4$ hidden layers.
- $k = 256$ hidden units.
- $N = 3200$ output speakers.
- $w = 787$k model weights (excluding output layer).
- Rectified Linear Units.
- Softmax output Layer.

Training

- Stochastic Gradient Descent.

Figure 1: Pipeline process from the waveform to the final score (left). DNN topology (middle). DNN description (right).

the output activations h_j^t of the last hidden layer j, using standard feed-forward propagation. Then we take the element-wise maximum of activations to form the compact representation of that utterance, the d-vector $\vec{d}$. Thus, the i^{th} component of the k-dimensional d-vector $\vec{d}$ is given by:

$$\vec{d}_i = \max_t (h_{ji}^t) \tag{1}$$

Note that in the computation of $\vec{d}$ we do not use any of the parameters in the output layer, which can be discarded. Thus, for M hidden layers, the number of total weights w in real-time system is given by:

$$w = vk + (M-1)k^2 \tag{2}$$

Each utterance generates exactly one d-vector. For enrollment, a speaker provides a few utterances of the global password; the d-vector from each of these utterances is averaged together to form a speaker model that is used for speaker verification, similar to the original i-vector model [11].

During evaluation, the scoring function is the cosine distance between the speaker model d-vector and the d-vector of an evaluation utterance.

Layers	Patch	Depth	Weights	Multiplies	EER
4	48 × 48	256	787k	787k	3.88
3			721k	721k	4.16
	48 × 48		787k	787k	3.88
4	20 × 48	256	442k	442k	4.05
	5 × 48		258k	258k	5.04
4	48 × 48	256	787k	787k	3.88
		128	344k	344k	5.53

Table 1: Baseline results for various configurations of fully-connected networks: with variable number of layers (*top*), with variable context sizes (*middle*) and with variable number of nodes (*bottom*.) The "Weights" column is the number of weights in each model, and represents the model footprint. The "Multiplies" column corresponds to the number of multiplications required for computing the feed-forward neural net, and represents the latency impact.

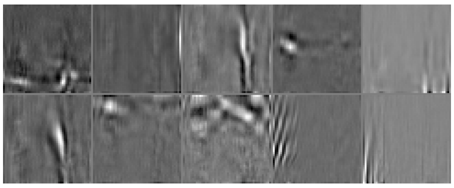


Figure 2: Weight matrices of first fully-connected layer in DNN. The weight matrices are sparse with well-localized non-zero weights.

3. Optimizing Local Connections

In order for our SV system to run in real-time on space-constrained platforms, the size of the DNN feature extractor must be small. However, in a fully-connected model with large number of visible units v, the term vk dominates over the rest of terms in Eq. 2; the first hidden layer accounts for most of the parameters. For example, our baseline model is a fully-connected DNN model with $v = 48 \times 48$ input elements and $k = 256$ hidden nodes in each of $M = 4$ hidden layers, such that the input layer accounts for the 75% of the model parameters. Similarly, in the previous work [4], the input layer accounted for 70% of the network parameters. Direct methods to reduce DNN size include reducing the number of hidden layers, reducing the input size by using fewer stacked context frames, and reducing the number of hidden nodes per layer; however, Table 1 shows that reducing the number of layers, context size, or hidden units strongly hurts performance. Therefore, in order to limit model size, this paper focuses on reducing the size of the first hidden layer using alternative architectures.

Although the first hidden layer contains most of our baseline fully-connected DNN model's weights, the weight matrices of the first fully-connected hidden layer are very sparse and low-rank; Fig. 2 shows visualizations of the weight matrices from the first hidden layer. Previous works have taken note of DNN sparsity and attempted to train networks that are less sparse [12], or iteratively prune low-value weights [13]. We observe that the sparse non-zero weights are clumped close together, not scattered throughout the matrix, such that a small patch could span over the well-localized non-zero weights. This is important because we rely heavily on parallel SIMD operations (as in [14]) to efficiently compute neural nets using small dense matrices rather than large and sparse matrices. In this work, we seek to use LCN and CNN layers to take advantage of the sparse and local nature of the DNN to constrain the model size while improving performance.

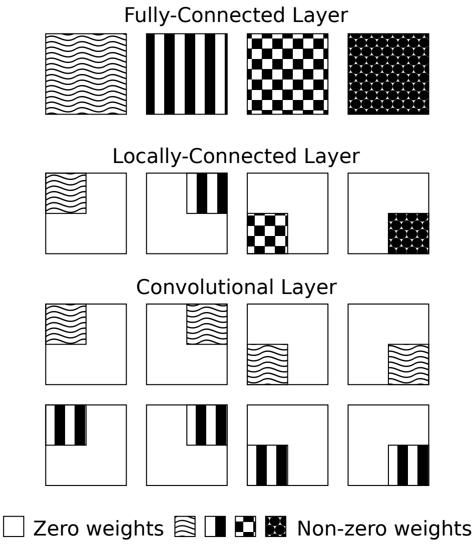


Figure 3: In a fully-connected input layer, each filter contains non-zero weights for each input element. In a LCN input layer, each filter is only non-zero for a subset of the input elements, and different filters may cover different subsets of the input. While each filter in a LCN layer covers only one patch of the input, each filter in a CNN layer covers all the patches in the input through convolution. Each patterned square corresponds to a filter matrix.

3.1. Locally Connected DNNs

To reduce the model size, we experiment with explicitly enforcing sparsity in the first hidden layer by using a LCN layer [6]. When using local connections, each of the hidden activations is the result of processing a locally-connected "patch" of v, rather than all of v as done in fully-connected DNNs. Fig. 3 compares the weight matrices of a fully-connected layer and a LCN layer, emphasizing how a LCN layer is equivalent to a sparse fully-connected layer.

Previous works suggested that any reasonable tiling of the input space, including random patches, could be sufficient to obtain high performance [15, 16]. Thus, as more sophisticated approaches may not be necessary, we use LCN with square patches of size $p \times p$ that perfectly tile the input elements in a grid with no gaps. Let v be the number of input features, p the width and length of the square patch, $n = v/p^2$ the number of patches over the input and f_{lcn} is the number of filters over each patch. Then, the total number of filters used by the LCN layer is given by nf_{lcn}, while the number of weights in the network is:

$$w = vf_{\text{lcn}} + nf_{\text{lcn}}k + (M-2)k^2 \tag{3}$$

Here k denotes the number of nodes of the rest of the hidden layers in the network. Note by comparing (2) and (3) that the variables f_{lcn} and n offer finer control over the number of parameters in the network. The first two hidden layers are influenced by f_{lcn}, while remaining hidden layers have k^2 weights. One interpretation of local connections is that they enforce patch-based sparse matrices when training; given the sparse filters in the first fully-connected hidden layer (Figure 3), local connections are a natural fit. By using a LCN layer, we are implementing a sparse-coding with hand-crafted bases.

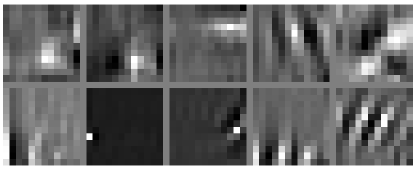


Figure 4: Filters from LCN layer with 12x12 patches.

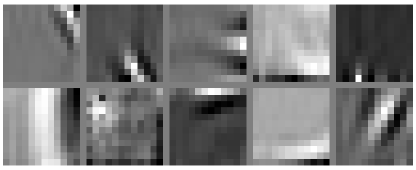


Figure 5: Filters from CNN layer with 12x12 patches.

3.2. Convolutional Neural Networks

As Figure 4 shows, several LCN filters appear similar, suggesting further compression is possible. This motivates us to look at CNNs to reduce model size further. Like LCN, CNN also define a topology where local receptive fields, or patches, are used to model the local correlations in the input [7]. However, unlike LCN layers—where each filter is applied to a single patch in the input—in CNN layers, filters are convolved, such that all filters are applied to every input patch. (Figure 3). This approach may be interpreted as using a unique set of f_{cnn} filters repeated over all patches, versus using n sets of localized filters, each of size f_{lcn}, as in LCN. As several LCN filters appeared similar in Figure 4, this strategy of sharing filters suggests that further compression is possible. Furthermore, previous work suggested that CNNs are particularly good in handling noisy or reverberant conditions [17, 18].

CNN layers take orders of magnitude more multiplications to compute than similarly sized fully-connected or LCN layers. In order to keep latency under 40ms on our target platforms, we limit our experiments to CNN configurations with 1.5M multiplications; under this constraint, the only configurations we consider are filters that shift with very large strides of size p when convolving. We do not using pooling layers, as they reduce speaker variance [19]. Given a 48×48 input, we present experiment results for CNN layers with 4 24×24 patches, 16 12×12 patches, or 64 6×6 patches.

We compute the number of weights in a model with CNN first hidden layer as follows. Let v be the number of input features, p the width and length of square patch filter, $n = v/p^2$ the number of patches, f_{cnn} be the number filters from first hidden layer, and k be the number of nodes in the rest of the hidden layers; then the number of weights for a CNN model is

$$w = f_{\text{cnn}}p^2 + nf_{\text{cnn}}k + (M-2)k^2$$

Unlike fully-connected and LCN models, the number of multiplications necessary to compute the CNN model is not equal to the number of model weights. The number of multiplications required to compute a CNN model is

$$vf_{\text{cnn}} + nf_{\text{cnn}}k + (M-2)k^2$$

Some of the filters learned by CNN layer can be seen in Figure 5. The CNN filters appear to be smoother and sparser than the LCN filters in Figure 4.

4. Experiment results

The experiments are performed on a small footprint global password TD-SV task. The spoken password in all our datasets is given by "Ok Google" and samples are collected from anonymized real traffic from the Google KWS system [2]. The training set for our neural networks contains 3,200 anonymized speakers speaking, with an average of ∼745 repetitions per speaker. Repetitions are recorded in multiple sessions in a wide variety of environments, including multiple devices and languages. A non-overlapped set of 3,000 speakers are present for enrollment and evaluation. Each speaker in the evaluation set enrolls with 3 to 9 utterances and it is evaluated with 7 positive utterances. In our results, all possible trials were considered, leading to ∼21k target trials and ∼6.3M non-target trials. Results are reported in Equal Error Rate (EER). We found that relative differences with other operating points are preserved.

The hidden layers generally contain 256 nodes, except in Section 4.3, when our experiment calls for matching the number of parameters between different model architectures. The focus of this paper is on experimenting with variations of the first hidden layer, which processes the input frames.

4.1. Baseline system

Our baseline system is a fully-connected neural network with 4 fully-connected hidden layers of 256 nodes each, described in Figure 1 and Section 2. Our baseline system is similar to our DNN in previous work [4], but more recent experiments suggested the following optimizations: a) maxout layers have been replaced by fully-connected layers with rectified linear units, b) in Eq. 1, the dimension-wise max function replaces the average function used before c) visible input elements are given by matrices of 48×48 elements instead of 41×40, which gives us more flexibility in the configuration of patches. Note that 48×48 facilitates the definition of square patches as it is divisible by 24, 12, 8, 6, 4, 3 and 2.

4.2. Compressing first hidden layer

We experiment with only modifying the first hidden layer, fixing the last three hidden layers as fully-connected layers with 256 nodes, 66k weight parameters each. For LCN layers and CNN layers, we experiment with three patch sizes: $24 \times 24, 12 \times 12, 6 \times 6$. In order to achieve 256 output nodes from the first hidden layer, the depth of each layer is varied with the type of layer and patch size. For example, a fully-connected layer with depth of 256 would have 256 output nodes. A LCN layer with 24×24 patch size with depth of 64 would generate 4 patches with depth 64, for a total of 256 output nodes as well.

Table 2 shows the configuration and equal error rate (EER) for each experimental model, as well model footprint and latency information. This experiment shows that the baseline fully-connected first hidden layer can be reduced from 590k parameters to 37k (6% of baseline layer) parameters with about 4% increase in EER by using a LCN layer with 12×12 patches or a CNN layer with 24×24 patches. For 4% increase in EER, we have LCN and CNN models that are 30% the size of the baseline model; in this experiment, the best LCN model and the best CNN model have the same number of parameters and similar EER.

4.3. Improve performance given size constraint

Section 4.2 focuses on reducing model size, allowing the EER to increase above that of the baseline model. In this section, we focus on closely matching the model size across different experimental models and decreasing EER. The model size is important for resource-constrained platforms. To match the model size, the first hidden layer is no longer constrained to have 256 hidden units, allowing us to increase the depth of the LCN and CNN layers. The last two hidden layers are fully-connected, have 256 inputs and outputs, and contain 66k weights.

	Patch	Depth	Weights	Multiplies	EER
Fully	48 × 48	256	**787k**	787k	**3.88**
LCN	24 × 24	64	345k	345k	4.11
	12 × 12	16	**234k**	234k	**4.02**
	6 × 6	4	206k	206k	4.54
CNN	24 × 24	64	**234k**	345k	**4.04**
	12 × 12	16	199k	234k	4.24
	6 × 6	4	197k	206k	4.45

Table 2: Compare fully-connected, LCN, and CNN first hidden layer. First hidden layer has 256 outputs, while the remaining hidden layers have 256 inputs and 256 outputs. "Weights" corresponds to model size. "Multiplies" corresponds to latency.

Table 3 shows the EER, number of weights (model size), and number of multiplications (latency) for each experimental model. When parameters are matched, every LCN and CNN experimental model has smaller EER than that of the baseline fully-connected model. With approximately the same number of weights and multiplications, LCN model with 12×12 patches has EER that is 8% lower than baseline model. With approximately the same number of weights and 90% more multiplications, CNN model with 24×24 patches has EER that is 10% lower than the baseline model. When the number of model parameters is held constant, CNN models have better performance than LCN models.

	Patch	Depth	Weights	Multiplies	EER
Fully	48 × 48	256	787k	787k	3.88
LCN	24 × 24	197	787k	787k	3.71
	12 × 12	102	784k	784k	**3.60**
	6 × 6	35	786k	786k	3.75
CNN	24 × 24	411	789k	1499k	**3.52**
	24 × 24	154	785k	1117k	3.75
	24 × 24	40	788k	879k	3.87

Table 3: Match total number of parameters, holding last 2 hidden layers constant while varying the first 2 hidden layers. "Weights" corresponds to model size. "Multiplies" corresponds to latency.

5. Conclusions

In this paper, we compare two alternative neural network layer architectures to a fully-connected baseline for small footprint text-dependent speaker verification. Both LCN and CNN layers can be used to shrink model size to 30% of baseline with a 4% relative increase in EER (Table 2). When model size is held constant, CNN model is preferred because it reduces baseline EER by 10% relatively, versus 8% for LCN model of the same size (Table 3). If latency, which corresponds to number of model multiplications, is constrained, then the LCN model is preferred because it uses 52% fewer multiplications than CNN model, though LCN model has slightly higher EER.

6. References

[1] H. Aronowitz, R. Hoory, J. W. Pelecanos, and D. Nahamoo, "New developments in voice biometrics for user authentication." in *INTERSPEECH*, 2011, pp. 17–20.

[2] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. N. Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks." in *to appear at ICASSP*, 2015.

[3] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, "“your word is my command": Google search by voice: A case study," in *Advances in Speech Recognition*. Springer, 2010, pp. 61–90.

[4] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, pp. 4052–4056.

[5] T. Stafylakis, P. Kenny, P. Ouellet, J. Perez, M. Kockmann, and P. Dumouchel, "Text-dependent speaker recognition using plda with uncertainty propagation," *matrix*, vol. 500, p. 1, 2013.

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

[7] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, 2004, pp. II–97.

[8] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8614–8618.

[9] M. McLaren, Y. Lei, N. Scheffer, and L. Ferrer, "Application of convolutional neural networks to speaker recognition in noisy conditions," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[10] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer, "Application of convolutional neural networks to language identification in noisy conditions," in *Proc. Speaker Odyssey Workshop (submitted)*, 2014.

[11] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 788 – 798, February 2011.

[12] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting Sparseness In Deep Neural Networks For Large Vocabulary Speech Recognition," in *ICASSP 2012*. IEEE SPS, March 2012.

[13] B. Hassibi, D. G. Stork, and S. C. R. Com, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993, pp. 164–171.

[14] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[15] A. Coates and A. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 921–928.

[16] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *ICCV*. IEEE, 2009, pp. 2146–2153.

[17] M. McLaren, Y. Lei, N. Scheffer, and L. Ferrer, "Application of convolutional neural networks to speaker recognition in noisy conditions," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, H. Li, H. M. Meng, B. Ma, E. Chng, and L. Xie, Eds. ISCA, 2014, pp. 686–690.

[18] H. Lee, P. T. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks." in *NIPS*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1096–1104.

[19] T. N. Sainath, A. rahman Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr." in *ICASSP*. IEEE, 2013, pp. 8614–8618.

B.3 Automatic Language Identification Using Long Short-Term Memory Recurrent Neural Networks

TITLE	**Automatic Language Identification Using Long Short-Term Memory Recurrent Neural Networks.**	
CONFERENCE	Interspeech 2014	Singapore.
AUTHORS	Javier González Domínguez	UAM
	Ignacio López Moreno	UAM / Google Inc.
	Hasim Sak	Google Inc.
	Joaquin González Rodríguez	UAM
	Pedro J. Moreno	Google Inc.
PAGES	2155 - 2159	
DATE	2014/09	
PUBLISHER	ISCA	Grenoble, France

Summary: This article introduces LSTM networks in the Spoken Language Identification task. LSTM networks are recurrent network networks particularly suitable to model sequences. We explore the configurations under which, LSTM-based architectures can outperform alternative approaches, including DNN-based models.

Contributions: Collaborated in the proposal of the novel idea of using LSTM models for the task of language recognition; collaborated in providing the necessary code to generate LCN models. collected approximately one year of labeled data (audio and parameters); collaborated in providing the necessary tools to generate and optimize topologies for the LSTM models; and collaborated in generating LSTM models for language recognition.

Automatic Language Identification using Long Short-Term Memory Recurrent Neural Networks

Javier Gonzalez-Dominguez[1,2], *Ignacio Lopez-Moreno*[1], *Haşim Sak*[1],
Joaquin Gonzalez-Rodriguez[2], *Pedro J. Moreno*[1]

[1]Google Inc., New York, USA
[2]ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain
javier.gonzalez@uam.es, jgd@google.com

Abstract

This work explores the use of Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) for automatic language identification (LID). The use of RNNs is motivated by their better ability in modeling sequences with respect to feed forward networks used in previous works. We show that LSTM RNNs can effectively exploit temporal dependencies in acoustic data, learning relevant features for language discrimination purposes. The proposed approach is compared to baseline i-vector and feed forward Deep Neural Network (DNN) systems in the NIST Language Recognition Evaluation 2009 dataset. We show LSTM RNNs achieve better performance than our best DNN system with an order of magnitude fewer parameters. Further, the combination of the different systems leads to significant performance improvements (up to 28%).

1. Introduction

The problem of automatic language identification (LID) can be defined as the process of automatically identifying the language of a given spoken utterance [1]. LID is daily used in several applications such as multilingual translation systems or emergency call routing, where the response time of a fluent native operator might be critical [1] [2].

Even though several high level approaches based on phonotactic and prosody are used as meaningful complementary sources of information [3][4][5], nowadays, many state-of-the-art LID systems still include or rely on acoustic modelling [6][7]. In particular, guided by the advances on speaker verification, the use of i-vector extractors as a front-end followed by diverse classification mechanisms has become the state-of-the-art in acoustic LID systems [8][9].

In [10] we found Deep feed forward Neural Networks (DNNs) to surpass i-vector based approaches when dealing with very short test utterances ($\leq$3s) and large amount of training material is available ($\geq$20h per language). Unlike previous works on using neural networks for LID [11] [12] [13], this was, to the best of our knowledge, the first time that a DNN scheme was applied at large scale for LID, and benchmarked against alternative state-of-the-art approaches.

Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) [14, 15, 16] have recently been shown to outperform the state of the art DNN systems for acoustic modeling in large vocabulary speech recognition [17, 18]. Recurrent connections and special network units called memory blocks in the recurrent hidden layer in LSTM RNNs make them a more powerful tool to model sequence data than feed forward neural networks and conventional RNNs. The memory blocks contain memory cells with self-connections storing the temporal state of the network which changes with the input to the network at each time step. In addition, they have multiplicative units called gates to control the flow of information into the memory cell and out of the cell to the rest of the network. This allows the network to model temporal sequences such as speech signals and their complex long-range correlations.

In this paper, we propose LSTM RNNs for automatic language identification. Our motivation is that LSTM RNNs' effectiveness in modeling temporal dependencies in the acoustic signal can help learning long-range discriminative features over the input sequence for language identification. To assess the proposed method's performance we experiment on the NIST Language Recognition Evaluation 2009 (LRE'09). We focus on short test utterances (up to 3s). We show that LSTM RNNs perform better than feed forward neural networks with an order of magnitude fewer parameters. Besides, they learn complementary features to DNNs and we get significant improvements by combining the scores from DNN and LSTM RNN systems.

The rest of this paper is organized as follows. Section 2 presents the i-vector based baseline system and the feed forward DNN architecture. Section 3 is devoted to present the LSTM RNN architecture. The fusion and calibration procedure is presented in Section 4. The experimental protocol and datasets used are then described in section 5. Results are discussed in section 6. Finally, conclusions are presented in Section 7.

2. Baseline Systems

To establish a baseline framework, we built a classical i-vector based acoustic system and three different DNNs based LID systems by varying the number of hidden layers. Baseline systems are summarized below and we refer to [10] for a more detailed description.

2.1. i-vector Based LID Systems

The i-vector system follows the standard procedure described in [8]. It is based on an Universal Background Model consisting of 1024 Gaussian components, trained on 39 dimensional PLP coefficients (13 + Δ + $\Delta\Delta$). From Baum-Welch statistics computed over this UBM, we derived a Total Variability (TV) space of 400 dimensions. Our TV model is trained using a PCA followed by 10 EM iterations.

We adopted a classical classification scheme based on Linear Discriminant Analysis followed by Cosine Distance (LDA_CS). Thus, the similarity measure (score) for a given test utterance i-vector w, and the mean i-vector w_L of the language

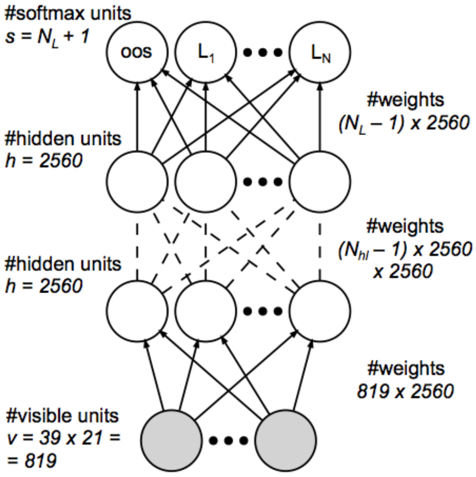


Figure 1: DNN network topology

L is given by

$$S_{w,w_L} = \frac{(A^t w)(A^t w_L)}{\sqrt{(A^t w)(A^t w)}\sqrt{(A^t w_L)(A^t w_L)}} \quad (1)$$

being A is the LDA matrix.

In [10] we provided a more detailed comparison between state-of-the-art i-vector and DNN -based LID system over the Google 5M dataset. In this work we opted for a LDA_CS baseline as it is a widely used technique and offers comparable results with the DNN model on the public LRE'09 dataset [10].

The total number of parameters of the i-vector system accounts for the TV and LDA matrices. It is given by $NFD + D(N_L - 1)$, being N, F, D and N_L the number of Gaussians components (1024), the feature dimension (39) the i-vector dimensions (400) and the number of languages (8). In our model, this makes a total of ∼16M of parameters.

2.2. DNN-based LID System

The DNN architecture used in this work is a fully connected feed-forward neural network [19]. The hidden layers contain units with rectified linear activation functions. The output is configured as a softmax layer with a cross-entropy cost function. Each hidden layer contains h (2560) units while the output layer dimension (s) corresponds to the number of target languages (N_L) plus one extra output for the out-of-set (oos) languages.

The DNN works at frame level, using the same features as the baseline systems described above (39 PLP). The input layer is fed with 21 frames formed by stacking the current processed frame and its ±10 left-right context. Therefore, there are 819 (21 × 39) visible units, v. The number of total weights w, considering N_{hl} hidden layers can be then easily computed as $w = vh + (N_{hl} - 1)hh + sh$. Figure 1 represents the complete topology of the network.

Regarding the training parameters, we used asynchronous stochastic gradient descent within the DistBelief framework [20]. We also fixed the learning rate and minibatch size to 1e-03 and 200 samples.

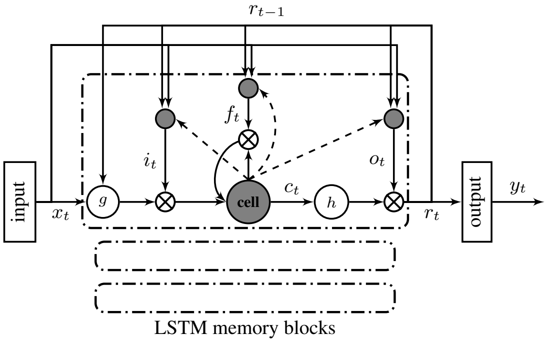


Figure 2: Long Short-Term Memory recurrent neural network architecture. A single memory block is shown for clarity.

Finally, we computed the output scores at utterance level by respectively averaging the log of the softmax output of all its frames (i.e. log of the predicted posterior probabilities).

3. Long Short-Term Memory RNNs

The modern LSTM RNN architecture [14, 15, 16] is shown in Figure 2. The LSTM contains special units called *memory blocks* in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the rest of the network. The forget gate scales the internal state of the cell before adding it as input to the cell through self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the LSTM RNN architecture contains *peephole connections* from its internal cells to the gates in the same cell to learn precise timing of the outputs [16].

With this architecture, LSTM RNNs compute a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$. They calculate the activations for network units using the following equations iteratively from the time step $t = 1$ to T:

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \quad (2)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}r_{t-1} + W_{fc}c_{t-1} + b_f) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$r_t = o_t \odot tanh(c_t) \quad (6)$$

$$y_t = \phi(W_{yr}r_t + b_y) \quad (7)$$

where the W terms denote weight matrices (e.g. W_{ix} is the matrix of weights from the input gate to the input), W_{ic}, W_{fc}, W_{oc} are diagonal weight matrices for peephole connections, the b terms denote bias vectors (b_i is the input gate bias vector), σ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the cell output activation vector r, $\odot$ is the element-wise product of the vectors, $tanh$ is the hyperbolic tangent activation function for cell inputs and cell outputs, and ϕ is the softmax output activation function for the LSTM RNN models used in this paper.

The LSTM RNN architecture that we used in this paper contains 512 memory cells. Different than DNNs, the input to the network is just 39-dimensional PLP features calculated at a given time step with no stacking of acoustic frames. The total number of parameters N ignoring the biases can be calculated as $N = n_i \times n_c \times 4 + n_c \times n_c \times 4 + n_c \times n_o + n_c \times 3$, where n_c is the number of memory cells, n_i is the number of input units, and n_o is the number of output units.

We trained the LSTM RNN model using asynchronous stochastic gradient descent (ASGD) and the truncated backpropagation through time (BPTT) learning algorithm [21] within a distributed training framework [18, 22]. Activations are forward propagated for a fixed step time of 20 over a subsequence of an input utterance, the cross entropy gradients are computed for this subsequence and backpropagated to its start. For better randomization of gradients in ASGD and stability of training, we split the training utterances into random chunks of length between 2.5 and 3 seconds. We also set the same target language id sparsely for a chunk, 1 in every 5 frames for the experiments in this paper. The errors are only calculated for the frames that we set a target language id. We used 100 machines for distributed training and in each machine 4 concurrent threads each processing a batch of 4 subsequences. We used an exponentially decaying learning rate of 1e-04. For scoring, we computed an utterance level score for each target language by averaging the log of the softmax outputs for that target language of all the frames in an utterance.

4. Fusion and Calibration

We used multiclass logistic regression in order to combine and calibrate the output of individual LID systems [23]. Let $s_{kL}(x_t)$ be the log-likelihood score for the recognizer k and language L for utterance x_t. We derive combined scores as

$$\hat{s_L}(x_t) = \sum_{k=1}^{K} \alpha_k s_{kL}(x_t) + \beta_L \qquad (8)$$

Note that this is just a generic version of the product rule combination, parametrized by α and β. Defining a multiclass logistic regression model for the class posterior as

$$P(L|\hat{s_L}(x_t)) = \frac{exp(\hat{s_L}(x_t))}{\sum_l exp(\hat{s}_l(x_t))} \qquad (9)$$

we found α and β to maximize the global log-posterior in a held-out dataset

$$Q(\alpha_1, ..., \alpha_K, \beta_1, ...\beta_N) = \sum_{t=1}^{T} \sum_{l=1}^{N_L} \delta_{tl} P(L|\hat{s}_l(x_t)) \qquad (10)$$

being

$$\delta_{tL} \begin{cases} w_L, & \text{if } x_t \in L \\ 0, & \text{otherwise.} \end{cases} \qquad (11)$$

where w_l ($l = 1, ..., N_L$) is a weight vector which normalizes the number of samples for every language in the development set (typically, $w_L = 1$ if an equal number of samples per language is used). This fusion and calibration procedure was conducted through the FoCal toolkit [24].

5. Datasets and Evaluation Metrics

5.1. The NIST Language Recognition Evaluation dataset

The LRE evaluation in 2009 included data coming from two different audio sources. Besides Conversational Telephone Speech (CTS), used in the previous evaluations, telephone speech from broadcast news was used for both training and test purposes. Broadcast data were obtained via an automatic acquisition system from “Voice of America” news (VOA) where telephone and non-telephone speech is mixed. Up to 2TB of 8KHz raw data containing radio broadcast speech, with the corresponding language and audio source labels were distributed to participants; and a total of 40 languages (23 target and 17 out of set) were included.

Due to the large disparity on training material for every language (from ∼10 to ∼950 hours) and also, for the sake of clarity, we selected 8 representative languages for which up to 200 hours of audio are available: US English (en), Spanish (es), Dari (fa), French (fr), Pashto (ps), Russian (ru), Urdu (ur), Chinese Mandarin (zh). Further, to avoid misleading result interpretation due to the unbalanced mix of CTS and VOA, all the data considered in this dataset belongs to VOA.

For evaluation, we used a subset of the official NIST LRE 2009 3s condition evaluation set (as for training, we also discarded CTS test segments), yielding a total of 2916 test segments of the 8 selected languages. That makes a total of 23328 trials.

5.2. Evaluation metrics

In order to assess the performance, two different metrics were used. As the main error measure to evaluate the capabilities of one-vs.-all language detection, we use C_{avg} (average cost) as defined in the LRE 2009 [25][26] evaluation plan. C_{avg} is a measure of the cost of taking bad decisions, and therefore it considers not only discrimination, but also the ability of setting optimal thresholds (i. e. calibration). Further, the well-known metric Equal Error Rate (EER) is used to show the performance, when considering only scores of each individual language. Detailed information can be found in the LRE’09 evaluation plan [25].

6. Experimental Results

6.1. Standalone systems performance

In [10] we found DNNs to outperform several different i-vector based systems when dealing with short test durations and large amount of training data (>20h per language). We followed up those experiments by first exploring the use of LSTM RNNs as a natural approach to exploit useful temporal information for LID; and second exploring the effect of varying the number of hidden layers in the DNN architecture.

Table 1 summarizes the results obtained in terms of EER (per language and on average) and C_{avg}. At a first glance, we highlight two major results. First, the proposed LSTM RNN architecture better performance than our best DNN system with 4 hidden layers. This fact is particularly interesting taking into account that the proposed LSTM RNN contains 20 times fewer parameters (see *Size* column in Table 1). Additionally, note from the C_{avg} values that the scores produced by the LSTM RNN model are better calibrated than those produced by DNN or i-vector systems. Second, both neural networks approaches (DNNs and LSTM RNN) surpass the i-vector system performance by ∼47% and ∼52% in EER and C_{avg} respec-

		Equal Error Rate (EER in %)									
	Size	en	es	fa	fr	ps	ru	ur	zh	EER_{avg}	C_{avg}
i-vector_LDA_CS	∼16M	17.22	10.92	20.03	15.30	19.98	14.87	18.74	10.09	15.89	0.1968
DNN_2_layers	∼8M	12.66	5.04	19.67	8.60	17.84	8.75	14.78	5.54	11.61	0.1727
DNN_4_layers	∼21M	8.53	3.58	16.19	5.82	15.42	6.38	11.24	3.16	8.79	0.1292
DNN_8_layers	∼48M	8.65	3.74	17.22	7.53	16.01	5.59	13.10	4.82	9.58	0.1376
LSTM RNN	∼1M	5.81	3.23	17.46	6.42	12.52	6.16	9.91	5.30	8.35	0.0944
Fusion1	∼22M	5.19	2.16	13.67	4.12	10.82	3.98	8.20	3.91	6.51	0.0693
Fusion2	∼38M	5.34	2.09	12.80	4.24	9.83	4.39	7.62	3.76	**6.26**	0.0654
Fusion3	∼94M	5.42	2.95	12.01	4.40	10.98	4.01	8.20	3.76	6.47	**0.0649**

Table 1: Systems performance per language and average metrics on LRE'09 subset (3s test segments)

tively. This result confirms the ability of the proposed neural networks architectures to exploit discriminative information in large datasets.

A further analysis regarding the optimal *depth* of the DNN system shows that the 4-hidden layer topology outperforms the 2-hidden layers one and more interestingly, the 8-hidden layers topology. In particular, the DNN_4_layers achieved, in average, ∼8% better EER than the DNN_8_layers despite of using just half of the parameters.

6.2. Systems combination performance

In this section we aim to analyze the score correlation among LSTM, DNN and i-vector systems and in particular, how that can lead to a good combination strategy. For this purpose defined three different groups of systems and combined them using the multiclass logistic regression framework presented in Section 4. The three groups defined bellow represent various tradeoffs between performance and number of parameters.

- **Fusion1**: this group is composed by the DNN_4_layers and LSTM RNN systems. This combination strategy allow us to evaluate the fusion capabilities of the proposed DNN and LSTM RNN architectures.
- **Fusion2**: this group incorporates the i-vector system to the compound Fusion1 system. It analyzes the complementarity between neural networks and a classical i-vector approach.
- **Fusion3**: this group includes DNN_2_layers and DNN_8_layers to the systems in Fusion2 to explore a global fusion for all the developed systems.

Fusion results are collected in Table 1. As observed, the combination of the best DNN system and LSTM (Fusion1) gets a >25% gain of performance in terms of C_{avg} with respect to our best individual LSTM RNN system. This fact shows that despite of the presumable similarity of the approaches (both neural nets trained via ASGD), they produce uncorrelated scores that can be successfully combined. Further improvements achieved by Fusion2 highlights the degree of complementary between i-vectors, DNN and LSTM RNN systems. This result is particularly interesting taking into account the gap of performance between Fusion1 and i-vector_LDA_CS. Finally, we present the fusion of all the developed systems in Fusion3. As expected, different DNN topologies exploit correlated information, which turns into a not significant improvement over Fusion2.

7. Conclusions

In this work, we proposed a new approach to Automatic Language Identification (LID) based on Long Short Term Memory (LSTM) Recurrent Neural Networks (RNNs). Motivated by the recent success of Deep Neural Networks (DNNs) for LID, we explored LSTM RNNs as a natural architecture to include temporal contextual information within a neural network system.

We compared the proposed system with an i-vector based system and different configurations of feed forward DNNs. Results on NIST LRE 2009 (8 languages selected and 3s condition) show that LSTM RNN architecture achieves better performance than our best 4 layers DNN system using 20 times fewer parameters (∼1M vs ∼21M). In addition, we found LSTM RNN scores to be better calibrated than those produced by the i-vector or the DNN systems.

This work also shows that both LSTM RNN and DNN systems remarkably surpass the performance of the individual i-vector system. Furthermore, both neural network approaches can be combined leading to a improvement of >25% in terms of C_{avg} with respect to our best individual LSTM RNN system. Our best combined system also incorporates the scores from the i-vector system leading to a total improvement of 28%.

8. References

[1] Y. Muthusamy, E. Barnard, and R. Cole, "Reviewing automatic language identification," *Signal Processing Magazine, IEEE*, vol. 11, no. 4, pp. 33–41, 1994.

[2] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: A tutorial," *Circuits and Systems Magazine, IEEE*, vol. 11, no. 2, pp. 82–108, 2011.

[3] M. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 4, no. 1, pp. 31–44, 1996.

[4] L. Ferrer, N. Scheffer, and E. Shriberg, "A Comparison of Approaches for Modeling Prosodic Features in Speaker Recognition," in *International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 4414–4417.

[5] D. Martinez, E. Lleida, A. Ortega, and A. Miguel, "Prosodic features and formant modeling for an ivector-based language recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 6847–6851.

[6] P. Torres-Carrasquillo, E. Singer, T. Gleason, A. McCree, D. Reynolds, F. Richardson, and D. Sturim, "The

MITLL NIST LRE 2009 Language Recognition System," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 4994–4997.

[7] J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D. Toledano, and J. Gonzalez-Rodriguez, "Multilevel and Session Variability Compensated Language Recognition: ATVS-UAM Systems at NIST LRE 2009," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 6, pp. 1084–1093, 2010.

[8] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language Recognition via i-vectors and Dimensionality Reduction." in *INTERSPEECH*. ISCA, 2011, pp. 857–860.

[9] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language Recognition in iVectors Space." in *INTERSPEECH*. ISCA, 2011, pp. 861–864.

[10] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic Language Identification using Deep Neural Networks," *Acoustics, Speech, and Signal Processing, IEEE International Conference on, to appear.*, 2014.

[11] R. Cole, J. Inouye, Y. Muthusamy, and M. Gopalakrishnan, "Language identification with neural networks: a feasibility study," in *Communications, Computers and Signal Processing, 1989. Conference Proceeding., IEEE Pacific Rim Conference on*, 1989, pp. 525–529.

[12] M. Leena, K. Srinivasa Rao, and B. Yegnanarayana, "Neural network classifiers for language identification using phonotactic and prosodic features," in *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 404–408.

[13] G. Montavon, "Deep learning for spoken language identification," in *NIPS workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Mar. 2003.

[17] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[18] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," *ArXiv e-prints*, Feb. 2014.

[19] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of Pretrained Deep Neural Networks to Large Vocabulary speech recognition," in *Proceedings of Interspeech 2012*, 2012.

[20] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1232–1240.

[21] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for online training of recurrent network trajectories," *Neural Computation*, vol. 2, pp. 490–501, 1990.

[22] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *submitted to INTERSPEECH 2014*, 2014.

[23] N. Brümmer and D. van Leeuwen, "On Calibration of Language Recognition Scores," in *Proc. of Odyssey*, San Juan, Puerto Rico, 2006.

[24] N. Brümmer. Fusion and calibration toolkit [software package]. [Online]. Available: , http://sites.google.com/site/nikobrummer/focal.

[25] NIST, "The 2009 NIST SLR Evaluation Plan," www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf, 2009.

[26] N. Brümmer, "Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech," Ph.D. dissertation, Department of Electrical and Electronic Engineering, University of Stellenbosch,, 2010.

B.4 Improving DNN Speaker Independence with I-vector Inputs

TITLE	**Improving DNN Speaker Independence with I-vector Inputs.**	
AUTHORS	Andrew Senior	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
CONFERENCE	ICASSP 2014	Florence, Italy.
DOI	10.1109/ICASSP.2014.6853591	
DATE	2014/05	
PUBLISHER	IEEE	New York, NY, US

Summary: We propose providing additional utterance-level features as inputs to a neural network to facilitate speaker, channel and background normalization. Modifications of the basic algorithm are developed which result in significant reductions in word error rates (WERs).

Contributions: The candidate provided the necessary code to generate the i-vector systems and to compute partitions of the i-vector space using agglomerative clustering; generated i-vector models and clusters with optimized hyperparameters; and provided the augmented frequency features with i-vector components used to train the speech recognition models associated with each cluster.

IMPROVING DNN SPEAKER INDEPENDENCE WITH I-VECTOR INPUTS

Andrew Senior, Ignacio Lopez-Moreno

Google Inc.,
New York
{andrewsenior,elnota}@google.com

ABSTRACT

We propose providing additional utterance-level features as inputs to a deep neural network (DNN) to facilitate speaker, channel and background normalization. Modifications of the basic algorithm are developed which result in significant reductions in word error rates (WERs). The algorithms are shown to combine well with speaker adaptation by backpropagation, resulting in a 9% relative WER reduction. We address implementation of the algorithm for a streaming task.

Index Terms— Deep neural networks, large vocabulary speech recognition, Voice Search, i-vectors, speaker adaptation.

1. INTRODUCTION

Deep neural networks have come to prominence as acoustic models in recent years, surpassing the performance of the previous dominant paradigm, Gaussian Mixture Models (GMMs). One of the most powerful techniques for improving the accuracy of GMM speech models has been speaker adaptation wherein a speaker independent model is adapted on a small amount of data from a single speaker, with the resulting speaker-specific model performing better on test data from that speaker. Several studies [1, 2, 3] have shown that speaker adaptation is less effective with DNNs than with GMM acoustic models, partly because of the greater invariance of DNNs to speaker variations and their higher baseline accuracy.

Nevertheless, these studies do show that deep networks can be made more invariant to speaker variability. One of the problems with speaker adaptation is that it is hard to adapt a large number of parameters with only a small amount of data. Care must be taken to change the parameters sufficiently to have an effect without overfitting on the new data. Further, speaker adaptation results in a new model, or part-model, for each speaker which, in a cloud-based speech recognizer adds significant complexity and storage.

1.1. Deep networks

Recent results by many groups [4] have shown significant accuracy improvements over GMMs by using DNNs either to generate the GMM features or to directly estimate the acoustic model scores. Neural networks consist of many simple units which each compute a weighted sum of the activations of other units, and output an activation which is a nonlinear function of that sum. Typically these units are arranged in layers which receive input from the units in the previous layer, with the first layer computing a weighted sum of externally provided features, such as the filterbank energies of frames of speech. These networks can be trained to approximate a desired output function by the backpropagation of the error in the output compared to a target value provided for each training input example. We have previously applied hybrid DNNs for acoustic modelling in Google's VoiceSearch [5, 6] and YouTube [7] applications.

1.2. Speaker adaptation (of DNNs)

The classic techniques for speaker adaptation of Gaussian Mixture Models are (Constrained) Maximum Likelihood Linear Regression (CMLLR) [8, 9]) and Maximum A Posteriori modelling [10]. In the former, a linear transformation, computed to maximize the likelihood of the adaptation data, is applied to the features. This technique has been applied to the features input to a neural network, but has the limitation of requiring the transform to be computed with a GMM which also limits the dimensionality and types of features which can be used. We have found that the gains from using high dimensional, stacked mel scale log filterbank energies over using conventional low-dimensional speech features outweigh the gains from being able to do CMLLR adaptation. Bacchiani [11] has shown that GMMs can be speaker-adaptated using utterance i-vectors (Section 2).

Abrash *et al.* [2] showed that neural networks can be adapted by training an input transform or adapting the whole network with backpropagation, and Liao [3] has recently shown that these techniques can be applied to DNNs with millions of parameters, although the gains are smaller on larger networks which are inherently more speaker-independent than smaller networks.

Ström [12] showed that a neural network system trained with speaker identities could be used at inference time without knowing the speaker's identity, inferring a *speaker space vector* and reducing the WER by 2.5% relative. Abdel-Hamid and Jiang [13, 14] recently proposed providing speaker adaptation in a DNN by learning a similar speaker code which is used to compute speaker-normalized features. In experiments on the TIMIT dataset, they used backpropagation to learn a separate code for each speaker. This speaker code was then used as an input to the network for utterances by the same speaker. These experiments showed 5% relative phone error rate reductions with DNNs.

Seltzer *et al.* [15] have shown that augmenting the inputs of a neural network with an estimate of background noise level can improve the robustness of such a network to background noise. This "noise-aware" training gave a 4% relative improvement compared to a DNN baseline using the dropout technique.

While this paper was under review, Saon *et al.* published a study [16] in which they augment DNN inputs with *speaker* i-vector features, whereas we use *utterance* i-vectors in a similar manner. They demonstrate a 10% relative reduction in WER on the 300 hour Switchboard task.

2. I-VECTORS

In the speaker recognition community utterances are typically represented by a *supervector*, whose components are the Maximum A Posteriori (MAP) adaptation coefficients of a large Gaussian Mixture Model (GMM) known as the Universal Background Model (UBM).

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

A number of factors such as the speaker identity and so-called session factors can contribute to the variability in the parameters N and F. Session factors include undesired variation associated with the utterance length, phonetic dependency and environmental conditions. In the last few years Factor Analysis (FA) has proved to be successful in modelling these components of variability as low dimensional *latent* variables (*i.e.* manifolds).

Several alternative FA methods have been used for speaker recognition, namely Joint Factor Analysis (JFA) [17], Total Variability (TV) [18] and more recently, Probabilistic Linear Discriminant Analysis (PLDA) [19]. Unlike JFA, where the undesired session variability and the useful speaker variability are explicitly modelled as two non-overlapping manifolds, the TV model has shown superior performance by modelling all sources of variability in the supervector as a single manifold. A point in this space of latent variables is referred as an "identity vector", or i-vector. The PLDA model can be seen as a combination of the previous two techniques, focused on extracting the speaker variability from the utterance i-vector.

Since they provide a compact representation of speaker and session factors that we wish a speech recognition system to be invariant to, i-vectors and other FA-based factors have been used in the past for rapid speaker adaptation of speech recognition systems. However, most of these contributions were based on classical HMM-based acoustic models. The Eigenvoices model [20] uses short-term HMM-derived speaker factors (i.e. eigenvoices) to bring a general speech recognition model closer to a particular speaker, and Bacchiani [11] used i-vectors for a better modelling of session variability, demonstrating an 11% WER reduction..

2.1. Computing i-vectors

Utterance supervectors are typically represented by the accumulated and centered zero- and first-order Baum-Welch statistics, N and F respectively. N and F statistics are computed from a UBM, denoted by λ. For UBM mixture $m \in 1, \ldots, C$, with mean, μ_m, the corresponding zero- and centered first-order statistics are aggregated over all frames in the database:

$$N_m = \sum_t P(m|o_t, \lambda), \tag{1}$$

$$F_m = \sum_t P(m|o_t, \lambda)(o_t - \mu_m), \tag{2}$$

where $P(m|o_t, \lambda)$ is the Gaussian occupation probability for the mixture m given the spectral feature observation $o_t \in \Re^D$ at time t. The TV model can be seen as a classical FA generative model [21], with observed variables given by the vector of stacked statistics $F = \{F_1, F_2, \ldots, F_m\}$. The TV model defines a set of hidden variables $x \in \Re^L : P(x) = \mathcal{N}(0, I)$ and a Gaussian distribution $P(x|F)$ that represents the utterance. In order to formulate $P(x|F)$, the model imposes a Gaussian distribution over $P(F|x)$, which relates observed and hidden variables in terms of a the rectangular low rank matrix $T \in \Re^{CD \times L}$:

$$P(F \mid x) = \mathcal{N}(NTx, \Sigma), \tag{3}$$

being $\Sigma \in \Re^{CD \times CD}$ a diagonal covariance matrix in the space of F. Here, N denotes a diagonal matrix of size $CD \times CD$ formed by C diagonal blocks of size $D \times D$ where the m-th component block is given the matrix $N_m I_{(D \times D)}$.

The utterance i-vector is defined as the value of x that maximizes $P(x|F)$ -the mean value-. For the imposed values of $P(x)$ and $P(F|x)$ the i-vector is formulated as:

$$x = (I + T^t \Sigma^{-1} NT)^{-1} T^t \Sigma^{-1} F, \tag{4}$$

Size	Context		Layers	Units per layer	Output states	Params
	L	R				
Small	10	5	4	480	1000	1.5M
Medium	10	5	6	512	2000	2.7M
Large	16	5	6	2176	14247	70M

Table 1: Parameters for the fully-connected sigmoid neural networks with softmax outputs.

The TV model is thus a data driven model with parameters $\{\lambda, T, \Sigma\}$. In [18] the authors provide a more detailed explanation of deriving these parameters, using the EM algorithm.

3. ADAPTING DNNS WITH I-VECTORS

Here we propose the idea that i-vectors can be used as input features for neural networks, resulting in improved recognition. i-vectors encode precisely those effects to which we want our ASR system to be invariant: speaker, channel and background noise. While the targets to which we normally train are independent of these factors, providing the network with a characterisation of them at the input should enable it to normalise the signal with respect to them and thus better able to make its outputs invariant to them.

Consequently, we propose augmenting the traditional acoustic input features with the utterance i-vector. A network which takes a context window of c frames of d dimensional acoustic features is augmented with v i-vector dimensions resulting in a $cd + v$ dimensional input, as shown in Figure 1.

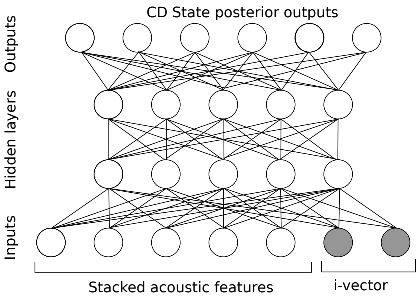


Fig. 1: Diagram of a 2-hidden layer neural network with inputs augmented with i-vectors.

As with traditional cross-entropy training, frames from the training data are randomly selected and stacked with the appropriate context window but all frames from a given utterance are augmented with the same v dimensional utterance i-vector.

3.1. Baseline Experiments

In our first experiments we trained three different sizes of network, with and without utterance i-vectors. The network configurations were chosen to suit both "cloud" speech recognition on a conventional server as well as two sizes of "embedded" speech recognizers designed to run on mobile phones of different processing power. Each network is fully connected with logistic sigmoid hidden layers and softmax outputs, receiving stacked 25ms frames of 40-dimensional Mel filterbank energy features as input. The number of parameters in the baseline networks are shown in Table 1, with the augmented networks having slightly more parameters in the initial layer because of the increased input dimension. All the networks are trained from random initialization with exponentially decaying learning rates.

The networks are trained on a corpus of 3 million utterances (about 1,750 hours) of US English Google voice search and dictation traffic, anonymized and hand-transcribed. This data is endpointed and aligned using a high accuracy server-sized neural network with 14247 context dependent (CD) states. For the smaller networks these state symbols are mapped through equivalence classes down to the smaller state inventories. During training, CD state frame accuracies are evaluated on the training data and on a held out development set of 200,000 frames. Word Error Rates (WERs) are measured on a test set of 23,000 hand-transcribed utterances sampled from live traffic. Training is by stochastic gradient descent with a minibatch size of 200 frames on a Graphics Processing Unit.

The parameters of the TV model, including the UBM, were also trained on this corpus. The UBM was trained with 1024 mixtures computed from 13 perceptual linear prediction coefficients with delta and delta-delta features appended. The matrix Σ was built by stacking the diagonal covariance matrices and never updated, while the matrix T was initialized using PCA and updated with 10 EM iterations for 300 latent variables.

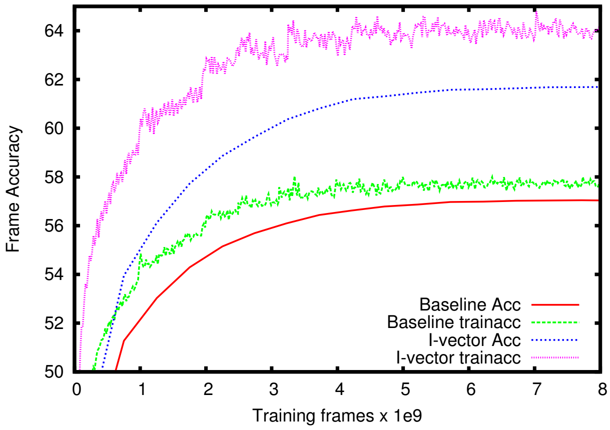


Fig. 2: Frame accuracies against billions of training samples on training and held-out dev sets during training for the larger network, with and without i-vector inputs.

Model size	Baseline		i-vector	
	WER	Frame acc.	WER	Frame acc.
Small	**17.8**	55.4	18.2	58.5
Medium	**15.0**	55.0	15.5	59.1
Large	**11.0**	57.1	12.3	59.1

Table 2: WERs and development set frame accuracy for baseline and with inputs augmented by 300 i-vector dimensions.

Figure 2 shows the progress of development set and training set frame accuracies during training of the small networks, and Table 2 shows the corresponding WERs. Early in training the i-vectors give a significant (3%) increase in development set frame accuracy and a larger (6%) increase in training set frame accuracy. As training progresses, the margin between training and dev-set accuracy diminishes, but the margin for the i-vector-augmented network remains much larger than for the baseline (2% vs 0.8%). From these graphs we infer that the network is able to use the i-vector to predict the frame classes, but is overfitting to the i-vector and is unable to use this information during decoding, resulting in higher WERs. Alternatively, it is possible that many voice search utterances are very short -only a few hundreds of voiced frames- and the 300-dimensional i-vectors estimate is not reliable. Both conjectures will explored in the next section.

4. REGULARIZATION

To avoid the overfitting we investigated two solutions:

1. Reduce the information content of the i-vectors.
2. Regularize the network parameters.

The first solution attempts to reduce the overfitting by limiting the amount of information presented to the network. To this end we truncate the i-vectors to a smaller dimension before augmenting the input vector. Arbitrarily the first k elements of the vector are preserved. Table 3 shows that reducing the dimensionality of the i-vector results in a lower WER, with the greatest gains being made with the smaller networks. Having validated the use of lower dimensional i-vectors, we trained low dimensional TV matricess with 20 and 50 dimensions and repeated the above experiment with these i-vectors. We found that training a medium network with a 20-dimensional i-vector led to a WER of 14.4%, outperforming the 300-dimensional i-vector truncated to 20 dimensions, but a 50-dimensional i-vector performing worse, at 14.9% WER.

Model size	i-vector dimensions (k)					
	0	20	50	100	200	300
Small	17.8	**17.0**	17.2	17.4	17.9	18.2
Medium	15.0	**14.5**	**14.5**	**14.5**	15.2	15.5
Large	11.0	**10.9**	**10.9**	11.2	11.8	12.3

Table 3: WERs when augmenting the network inputs with truncated i-vector inputs.

The second solution begins with a network trained without any i-vector information. This network's input layer's weight matrix is augmented with weights, initially set to zero, from 300 additional inputs. The network is then trained further with i-vector-augmented inputs, but with ℓ_2 regularization (weight decay) back to the original weights, *i.e.* adding a term to the loss function proportional to the sum-squared difference between the network's weights and those of the network before i-vector augmentation. Experimentation found good results with a weight decay parameter of 10^{-7} to 10^{-6}. The small, medium and large networks were augmented, before full convergence, at 12, 10 and 4 billion frames respectively. Table 4 shows the results of this regularization. We see that the regularized networks have a lower WER than the original, unaugmented features, and that the large regularized network outperforms the corresponding network trained with truncated i-vectors.

Training	Original	Regularization	
		10^{-7}	10^{-6}
Small	17.8	**17.2**	17.3
Medium	15.0	**14.5**	14.6
Large	11.0	**10.6**	10.8

Table 4: WERs when training an i-vector-augmented network while regularizing back to the original weights.

5. COMBINING WITH ADAPTATION

In this section we explore the interaction between using utterance i-vectors for invariance and the use of adaptation to provide speaker invariance. Liao [3] describes how training on an adaptation set for a particular speaker using backpropagation with ℓ_2 regularization back to the original, speaker-independent, model can reduce WERs on test data from the same speaker.

Here we compare our technique with this approach and show that the two can be used in combination to achieve even lower error rates. These experiments are conducted with the best "Medium" models only. We use the same personalization training and test sets used by Liao. These have 10 minutes of adaptation data for each of 80 speakers, and a total of 10,000 utterances (72,000 words) from the same 80 speakers in the test set. We report average word error rates across the entire test set. We use the "enrollment" protocol, i.e. the training data is manually transcribed and force-aligned with a large DNN model. The baseline model is adapted to each speaker's data with multiple passes. We continue to use the same, exponentially decaying learning rate, and an ℓ_2 regularization weight of 0.01. We find the best performance after 1 million frames of adaptation.

The results are shown in Table 5. As can be seen, the adapted baseline model achieves a lower word error rate than with the unadapted i-vector-augmented models, but when the latter are also adapted, their WER is also reduced, bringing a total of 9% relative WER reduction from the combined technique for the truncated i-vectors.

Model	Unadapted	Adapted
Baseline model	15.3	14.4
300 dim Regularized (10^{-7}) model	14.9	14.3
20-dim i-vector model	**14.7**	**14.0**

Table 5: WERs for medium DNNs on the personalization test set representing 80 speakers, using speaker independent models and models adapted on 10 minutes of data per speaker (with 1 million frames of adaptation).

6. STREAMING IMPLEMENTATION

One limitation of this approach is that the i-vector representation we are using is computed on an entire utterance, and thus can only be computed when all the data for an utterance is available. Our principal application is real-time transcription of utterances from mobile devices with minimum latency, which involves processing utterances as they are being spoken and streaming results back to the user even before the utterance is complete. This approach produces a very responsive speech interaction on the device, but means that whole-utterance approaches cannot be used in practice, although fast rescoring with the augmented models could be applied without introducing too much overall latency.

To address this incompatibility with our application we investigated using speaker-averaged i-vectors in place of utterance i-vectors. Here we compute an averaged speaker i-vector on the 10-minute adaptation set and used this for decoding the speaker's personalization data using the models trained above on per-utterance i-vectors. As can be seen in Table 6, the mismatched average i-vectors are not useful in decoding on a model trained with utterance i-vectors. In addition to matched training with speaker i-vectors, there are a number of alternative ways of training with ivectors for a streaming application which we will investigate in the future.

Model	Utterance	Speaker
Baseline model	15.3	
300 dim Regularized (10^{-7}) model	14.9	15.3
20-dim i-vector model	14.7	15.5

Table 6: WERs for medium DNNs on the 80 speaker personalization test set using speaker and utterance i-vectors in decoding.

- **On-line computation of the i-vectors:** We can compute the i-vectors based on the data so far, or use the *d-Vector* proposed by Variani et al. [22] computed, like our posteriors, based on a sliding window of frames.
- **Use of the i-vector from the speaker's previous utterance** Within a session, we expect variations in background noise, channel and speaker to be small, so the i-vector of the previous utterance may still be sufficient to provide invariance to these factors.

7. COMPARISON TO SIMILAR WORK

As noted earlier, Saon *et al.* published a similar work [16] while this paper was under review. They also augment the DNN input, but use the speaker i-vector for all utterances by the speaker, both in training and testing. Their DNNs are trained on LDA-projected PLP features from a narrow window which allow the use of conventional speaker adaptation. They show better performance with higher dimensional speaker i-vectors and obtain 10% relative WER reduction over speaker-independent features — their greater gains perhaps being due to the poorer baseline features used. They also demonstrated that i-vector augmentation combined well with a conventional speaker adaptation technique (CMLLR). They found that the i-vector dimension had to be at least 100, and in addition found that this technique was was beneficial in combination with sequence-discriminative training.

8. CONCLUSIONS

We have shown that using the utterance i-vectors as input features provides the neural networks with valuable information that, with the regularization we propose, bring about roughly a 4% relative reduction in word error rate for all model sizes. These techniques can be applied on any utterance, without requiring any speaker information or speaker adaptation or model storage. The technique has been shown to combine well with model adaptation, delivering an overall 9% WER reduction for models that are small enough to be run in real-time in a smart-phone, which are ideal candidates for speaker-adapted models.

These improvements are directly applicable to non-realtime applications, but are not well suited to a streaming scenario. We have proposed a variety of methods to address this in future work, but using the speaker i-vector in place of the utterance i-vector at test time did not help. It will be instructive to further investigate the relative benefit of using speaker i-vectors compared to utterance i-vectors which are far more noisy (particularly on short utterances) but offer independence to variations other than speaker identity.

9. ACKNOWLDEGEMENTS

Thanks to Hank Liao for providing the Personalization data sets and testing protocols.

10. REFERENCES

[1] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011.

[2] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *Proc. Eurospeech*, 1995.

[3] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. ICASSP*, 2013.

[4] G. Hinton, L. Deng, D. Yu, G.E. Dahl, Mohamed A., N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.

[5] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012.

[6] X. Lei, A. Senior, A. Gruenstein, and J. Sorenson, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proc. Interspeech*, 2013.

[7] H. Liao, E. McDermott, and A.W. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Proc. ASRU*, 2013.

[8] M.J.F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, 1998.

[9] C.J. Leggetter and P.C. Woodland, "Maximum likelihood linear regression speaker adaptation of contiuous density HMMs," in *Computer Speech and Languages*, 1997.

[10] J.L. Gauvain and C.H. Lee, "Bayesian learning of Gaussian mixture densities for hidden Markov models," in *Proc. DARPA Speech and Natural Language Workshop*, 1991.

[11] M. Bacchiani, "Rapid adaptation for mobile speech applications," in *Proc. ICASSP*, 2013, pp. 7903–7907.

[12] N. Ström, "Speaker adaptation by modeling the speaker variation in a continuous speech recognition system," in *Proc. ICSLP*, 1996.

[13] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. ICASSP*, 2013.

[14] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. Interspeech*, 2013.

[15] M.I. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP*, 2013.

[16] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using I-vectors," in *Proc. ASRU*, 2013.

[17] P. Kenny, "Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms," Available from: http://www.crim.ca/perso/patrick.kenny/FAtheory.pdf, in preparation.

[18] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 19, no. 4, pp. 788 – 798, February 2011.

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

[20] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 8, no. 6, pp. 695–707, 2000.

[21] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2007.

[22] E. Variani, X. Lei, E. McDermott, and I. Lopez-Moreno, "Text-dependent speaker verification using deep neural networks," in *Proc. ICASSP*, 2014.

B.5 Deep Neural Networks for Small Footprint Text-Dependent Speaker Verification

TITLE	**Deep Neural Networks for Small Footprint Text-Dependent Speaker Verification.**	
AUTHORS	Ehsan Variani	Google Inc.
	Xin Lei	Google Inc.
	Erik McDermott	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
	Javier González Domínguez	UAM
CONFERENCE	ICASSP 2014,	Florence, Italy.
DOI	10.1109/ICASSP.2014.6854363	
DATE	2014/05	
PUBLISHER	IEEE	New York, NY, US

Summary: In this paper, we investigate the use of deep neural networks (DNNs) for a small footprint text-dependent speaker verification task. At development stage, a DNN is trained to classify speakers at the frame-level. During the evaluation stage, we extract speaker specific features from the DNN, showing that simple speaker models built using such features can achieve comparable performance to other state-of-the-art approaches.

Contributions: The candidate provided the necessary code to generate the i-vector systems; generated baseline i-vector models with optimized hyperparameters; and collaborated in validating the performance with experimental results for the baseline system.

DEEP NEURAL NETWORKS FOR SMALL FOOTPRINT TEXT-DEPENDENT SPEAKER VERIFICATION

Ehsan Variani[1]*, *Xin Lei*[2], *Erik McDermott*[2], *Ignacio Lopez Moreno*[2], *Javier Gonzalez-Dominguez*[2,3]

[1]Johns Hopkins Univ., Baltimore, MD USA
[2]Google Inc., USA
[3]ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Spain
variani@jhu.edu {xinlei,erikmcd,elnota,jgd}@google.com

ABSTRACT

In this paper we investigate the use of deep neural networks (DNNs) for a small footprint text-dependent speaker verification task. At development stage, a DNN is trained to classify speakers at the frame-level. During speaker enrollment, the trained DNN is used to extract speaker specific features from the last hidden layer. The average of these speaker features, or d-vector, is taken as the speaker model. At evaluation stage, a d-vector is extracted for each utterance and compared to the enrolled speaker model to make a verification decision. Experimental results show the DNN based speaker verification system achieves good performance compared to a popular i-vector system on a small footprint text-dependent speaker verification task. In addition, the DNN based system is more robust to additive noise and outperforms the i-vector system at low False Rejection operating points. Finally the combined system outperforms the i-vector system by 14% and 25% relative in equal error rate (EER) for clean and noisy conditions respectively.

Index Terms— Deep neural networks, speaker verification.

1. INTRODUCTION

Speaker verification (SV) is the task of accepting or rejecting the identity claim of a speaker based on the information from his/her speech signal. Based on the text to be spoken, the SV systems can be classified into two categories, text-dependent and text-independent. Text-dependent SV systems require the speech to be produced from a fixed or prompted text phrase, while the text-independent SV systems operate on unconstrained speech. In this paper, we focus on a small footprint text-dependent SV task using fixed-text, although the proposed technique may be extended to text-independent tasks.

The SV process can be divided into three phases:

- Development: background models are trained from a large collection of data to define the speaker manifold. Background models vary from simple Gaussian mixture model (GMM) based Universal Background Models (UBMs) [1] to more sophisticated Joint Factor Analysis (JFA) based models [2, 3, 4].
- Enrollment: new speakers are enrolled by deriving speaker specific information to obtain speaker-dependent models. Speakers in the enrollment and development sets are not overlapped.
- Evaluation: each test utterance is evaluated using the enrolled speaker models and background models. A decision is made on the identity claim.

A wide variety of SV systems have been studied using different statistical tools for each of the three phases in verification. The state-of-the-art SV systems are based on i-vectors [5] and Probabilistic Linear Discriminant Analysis (PLDA). In these systems, JFA is used as a feature extractor to extract a low-dimensional i-vector as the compact representation of a speech utterance for SV.

Motivated by the powerful feature extraction capability and recent success of deep neural networks (DNNs) applied to speech recognition [6], we propose a SV technique based on DNN as the speaker feature extractor. A new type of DNN-based background model is used to directly model the speaker space. A DNN is trained to map frame-level features in a given context to the corresponding speaker identity target. During enrollment, the speaker model is computed as the average of activations derived from the last DNN hidden layer, which we refer to as a *deep vector* or “*d-vector*”. In the evaluation phase, we make decisions using the distance between the target d-vector and the test d-vector, similar to i-vector SV systems. One significant advantage of using DNNs for SV is that it is easy to integrate them into a state-of-the-art speech recognition system since they can share the same DNN inference engine and simple filterbank energies frontend.

The rest of this paper is organized as follows. In Section 2, previous related work on SV is described. In Section 3 we describe the proposed DNN-based SV system. Section 4 shows the experimental results for a small footprint text-dependent SV system. The DNN-based SV system is compared with an i-vector system in both clean and noisy conditions. We also evaluate the performance with different numbers of enrollment utterances and describe improvements from combination of two systems. Finally, Section 5 concludes the paper and discusses future work.

2. PREVIOUS WORK

The combination of i-vector and PLDA [5, 7] has become the dominant approach for text-independent speaker recognition. The i-vector represents an utterance in a low-dimensional space named total variability space. Given an utterance, the speaker- and session-dependent GMM supervector is defined as follows:

$$M = m + Tw \tag{1}$$

where m is the speaker- and session-independent supervector, usually taken to be the UBM supervector, T is a rectangular matrix of low rank, referred to as the total variability matrix (TVM), and w is a random vector with a standard normal distribution $N(0, I)$. The vector w contains the total factors and is referred to as the i-vector.

[*] Research conducted as an intern at Google.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

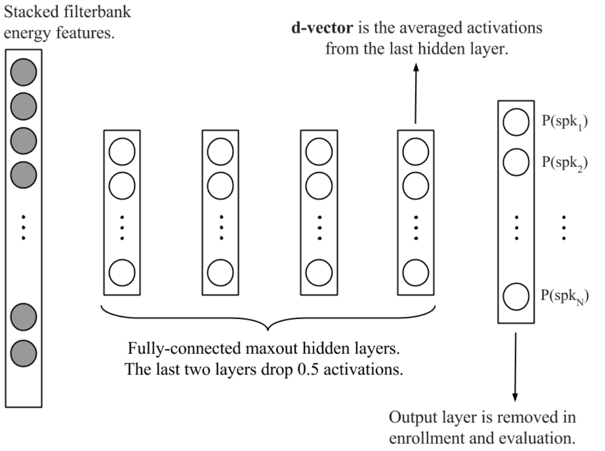

Stacked filterbank energy features.
d-vector is the averaged activations from the last hidden layer.
P(spk$_1$)
P(spk$_2$)
P(spk$_N$)
Fully-connected maxout hidden layers. The last two layers drop 0.5 activations.
Output layer is removed in enrollment and evaluation.

Fig. 1. The background DNN model for speaker verification.

Moreover, the PLDA on the i-vectors can decompose the total variability into speaker and session variability more effectively compared to JFA. The i-vector-PLDA technique and its variants have also been successfully used in text-dependent speaker recognition tasks [8, 9, 10].

In past studies, neural networks have been investigated for speaker recognition [11, 12]. Being nonlinear classifiers, neural networks can discriminate the characteristics of different speakers. The neural network was typically used as a binary classifier of target and non-target speakers, or multicategory classifiers for speaker identification purposes. Auto-associative neural networks (AANN) [13] were proposed to use the reconstruction error difference computed from the UBM-AANN and speaker specific AANN as the verification score. Multi-layer perceptrons (MLPs) with a bottleneck layer have been used to derive robust features for speaker recognition [14]. More recently, some preliminary studies have been conducted on using deep learning for speaker recognition, such as the use of convolutional deep belief networks [15] and Boltzmann machine classifiers [16].

3. DNN FOR SPEAKER VERIFICATION

The proposed background DNN model for SV is depicted in Figure 1. The idea is similar to [15] in the sense that neural networks are used to learn speaker specific features. The main differences are that here we perform supervised training, and use DNNs instead of convolutional neural networks. In addition, in this paper we evaluate on a SV task instead of the simpler speaker identification task.

3.1. DNN as a feature extractor

At the heart of the proposed approach in this work is the idea of using a DNN architecture as a speaker feature extractor. As in the i-vector approach, we look for a more abstract and compact representation of the speaker acoustic frames but using a DNN rather than a generative Factor Analysis model.

With this aim, we first built a supervised DNN, operating at the frame level, to classify the speakers in the development set. The input of this background network is formed by stacking each training frame with its left and right context frames. The number of outputs corresponds to the number of speakers in the development set, N. The target labels are formed as a 1-hot N-dimensional vector where the only non-zero component is the one corresponding to the speaker identity. Figure 1 illustrates the DNN topology.

Once the DNN has been trained successfully, we use the accumulated output activations of the last hidden layer as a new speaker representation. That is, for every frame of a given utterance belonging to a new speaker, we compute the output activations of the last hidden layer using standard feedforward propagation in the trained DNN, and then accumulate those activations to form a new compact representation of that speaker, the *d-vector*. We choose to use the output from the last hidden layer instead of the softmax output layer due to a couple of reasons. First, we can reduce the DNN model size for runtime by pruning away the output layer, and this also enables us to use a large number of development speakers without increasing DNN size at runtime. Second, we have observed better generalization to unseen speakers from the last hidden layer output.

The underlying hypothesis here is that the trained DNN, having learned compact representations of the development set speakers in the output of the last hidden layer, may also be able to represent unseen speakers.

3.2. Enrollment and evaluation

Given a set of utterances $X_s = \{O_{s_1}, O_{s_2}, \ldots, O_{s_n}\}$ from a speaker s, with observations $O_{s_i} = \{o_1, o_2, \ldots, o_m\}$, the process of enrollment can be described as follows. First, we use every observation o_j in utterance O_{s_i}, together with its context, to feed the supervised trained DNN. The output of the last hidden layer is then obtained, $L2$ normalized, and accumulated for all the observations o_j in O_{s_i}. We refer to the resulting accumulated vector as the *d-vector* associated with the utterance O_{s_i}. The final representation of the speaker s is derived by averaging all d-vectors corresponding for utterances in X_s.

During the evaluation phase, we first extract the normalized d-vector from the test utterance. Then we compute the cosine distance between the test d-vector and the claimed speaker's d-vector. A verification decision is made by comparing the distance to a threshold.

3.3. DNN training procedure

Given the low-resource conditions of the scenario explored in this study (see Section 4), we trained the background DNN as a *maxout* DNN using *dropout* [17][18].

Dropout is a useful strategy to prevent over-fitting in DNN finetuning when using a small training set [18][19]. In essence, the dropout training procedure consists of randomly omitting certain hidden units for each training token. Maxout DNNs [17] were conceived to properly exploit dropout properties. Maxout networks differ from the standard multi-layer perceptron (MLP) in that hidden units at each layer are divided into non-overlapping groups. Each group generates a single activation via the max pooling operation. Training of maxout networks can optimize the activation function for each unit.

Specifically, in this study, we trained a maxout DNN with four hidden layers and 256 nodes per layer, within the DistBelief framework [20]. A pool size of 2 is used per layer. The first two layers do not use dropout while the last two layers drop 50 percent of activations after dropout, as shown in Figure 1.

Regarding other configuration parameters, we used rectified linear units [21] as the non-linear activation function on hidden units and a learning rate of 0.001 with exponential decay (0.1 every

$5M$ steps). The input of the DNN is formed by stacking the 40-dimensional log filterbank energy features extracted from a given frame, together with its context, 30 frames to the left and 10 frames to the right. The dimension of the training target vectors is 496, which is the same as the number of speakers in the development set (see Section 4). The final maxout DNN model contains about $600K$ parameters, which is similar to the smallest baseline i-vector system.

4. EXPERIMENTAL RESULTS

The experiments are performed on a small footprint text-dependent SV task. The data set contains 646 speakers speaking the same phrase, "ok google", many times in multiple sessions. The gender distribution is balanced on the data set. 496 randomly selected speakers are used for training the background model and the remaining 150 speakers were used for enrollment and evaluation. The number of utterances per speaker for background model training varies from 60 to 130. For the enrollment speakers, the first 20 utterances are reserved for possible use in enrollment and the remaining utterances are used for evaluation. By default, we only use the first 4 utterances of the enrollment set for extracting speaker models. We used one out of 150 trials as a target trial and there are approximately 12750 trials in total.

4.1. Baseline system

In this small footprint text-dependent SV task, we aim to keep the model size small while achieving good performance. The baseline system is an i-vector based SV system similar to [5]. The GMM UBM is trained on 13-dimensional perceptual linear predictive (PLP) features with Δ and $\Delta\Delta$ features appended. We evaluate the equal error rate (EER) performance of the i-vector system with three different model sizes. The number of Gaussian components in the UBM, the dimension of the i-vectors and the dimension of Linear Discriminant Analysis (LDA) output are varied. The TVM is initialized using PCA and further refined using 10 EM iterations, while for UBM training we used 7 EM iterations. As shown in Table 1, the i-vector system performance degrades with reduced model size but not too significantly. The EER results with t-norm [22] for score normalization are consistently much better than with the raw scores. The smallest i-vector system contains about $540K$ parameters and is used as our baseline system.

Table 1. Comparison of EER results of i-vector systems with different number of UBM Gaussian components, i-vector and LDA output dimensions.

#Gaussians	i-vector Dim	LDA Dim	#Params	EER (raw)	EER (t-norm)
1024	300	200	12.2M	2.92%	2.29%
256	200	100	2.1M	3.11%	2.92%
128	100	100	540K	3.50%	2.83%

4.2. DNN verification system

The left plot in Figure 2 shows the detection error tradeoff (DET) curve comparison of the i-vector system and d-vector system. One interesting finding is that in the d-vector system the raw scores are slightly better than the t-norm scores, whereas in the i-vector system the t-norm scores are significantly better. The histogram analysis of the raw scores of the d-vector system indicates the distribution is heavy-tailed instead of a normal distribution. This suggests more sophisticated score normalization methods may be necessary for the d-vector SV system. Moreover, since t-norm requires extra storage and computation at runtime, we evaluate the d-vector systems using raw scores for the following experiments unless specified.

The overall performance of the i-vector system is better than the d-vector system: 2.83% EER using i-vector t-norm scores versus 4.54% with d-vector raw scores. However, in low False Rejection regions, as shown in right bottom part of the plots in Figure 2, the d-vector system outperforms the i-vector system.

We also experiment with different configurations for DNN training. Without maxout and dropout techniques, the EER of the trained DNN is about 2% absolute worse. Increasing the number of nodes to 512 in the hidden layers does not help significantly, while reducing the number of nodes to 128 gives much worse EER at 7.0%. Reducing the context window size to 10 frames on the left and 5 frames on the right also degrades the EER performance to 5.67%.

4.3. Effect of enrollment data

In d-vector SV system, there are no speaker adaptation statistics involved in the enrollment phase. Instead, the background DNN model is used to extract speaker-specific features for each utterance in both enrollment and evaluation phases. In this experiment we investigate how much the verification performance changes in the d-vector system with different numbers of enrollment utterances per speaker. We compare the performance results using 4, 8, 12 and 20 utterances for speaker enrollment.

Table 2. EER results of i-vector and d-vector verification systems using different number of utterances for enrollment.

	# utterances in enrollment			
	4	8	12	20
i-vector	2.83%	2.06%	1.64%	1.21%
d-vector	4.54%	3.21%	2.64%	2.00%

The EER results are listed in Table 2. It shows that both SV systems perform better with increasing numbers of enrollment utterances. The trend is similar for both systems.

4.4. Noise robustness

In practice there is usually a mismatch between development and runtime conditions. In this experiment, we examine the robustness of the d-vector SV system in noisy conditions and compare it with the i-vector system. The background models are trained with clean data. 10 dB cafeteria noise is added to the enrollment and evaluation data. The comparison of DET curves are shown in the right plot in Figure 2. As this figure illustrates, the performance of both systems is degraded by noise, but the performance loss of the d-vector system is smaller. Under 10 dB noisy environment, the overall performance of the d-vector system is very close to the i-vector system. At operating points of 2% or lower False Rejection probability, the d-vector system is in fact better than the i-vector system.

4.5. System combination

The results above show that the proposed d-vector system can be a viable SV approach when compared to the i-vector system. The assessment holds true mostly for noisy environments, or applications that require small footprint model and low False Rejection rates. Alternatively, here we aim to provide an analysis of a combined i/d-vector system. Although more sophisticated combinations can be

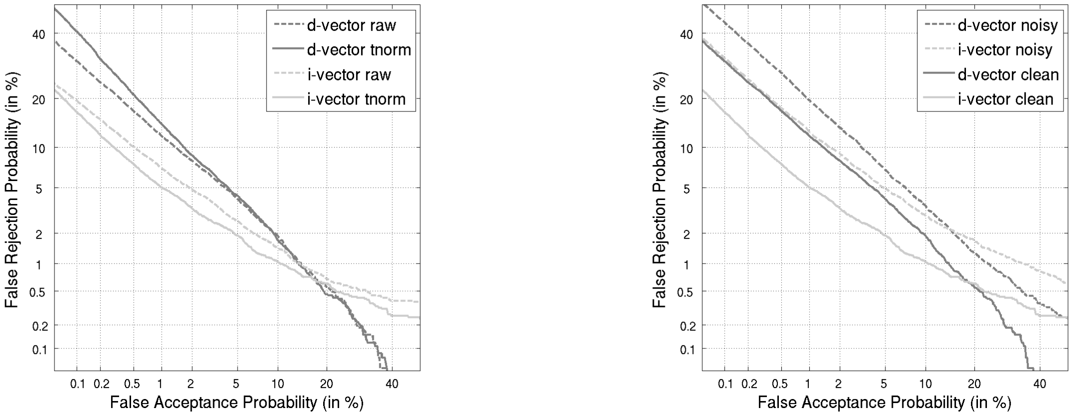


Fig. 2. Left: DET curve comparison between i-vector and d-vector speaker verification systems using raw and t-norm scores. Right: DET curve comparison of the two systems in clean and noisy conditions.

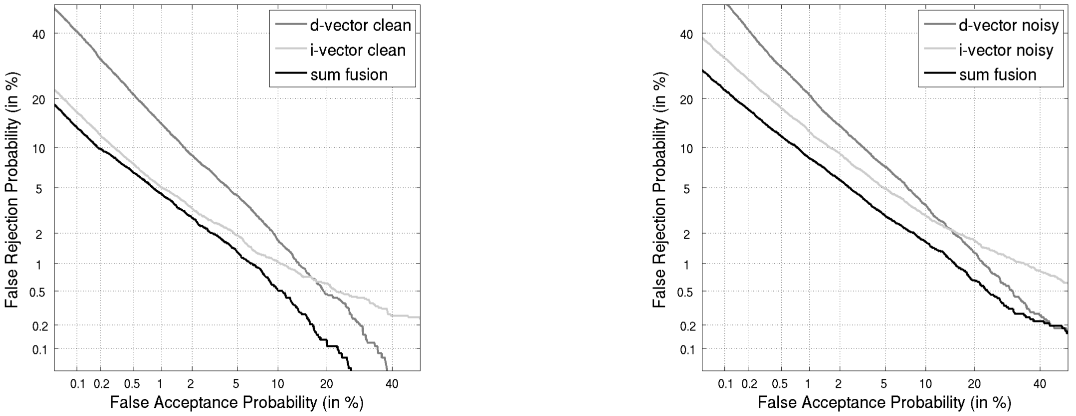


Fig. 3. DET curve for the sum fusion of the i-vector and d-vector systems in clean (left) and noisy (right) conditions.

devised at the feature level, our preliminary results in Figure 3 are obtained using a simple combination named as *sum fusion*, which sums the scores provided by each individual system for each trial. A prior t-norm stage was applied in both systems to facilitate the combination of scores. Results show that the combined system outperforms either component system in essentially all possible operating points and noise conditions. In terms of EER performance, the i/d-vector system beats the i-vector system by 14% and 25% relative, in clean and noisy conditions respectively.

5. CONCLUSIONS

In this paper we have proposed a new DNN based speaker verification method for a small footprint text-dependent speaker verification task. DNNs are trained to classify speakers with frame-level acoustic features. The trained DNN is used to extract speaker specific features. The average of these speaker features, or d-vector, is then used for speaker verification similarly to the popular i-vector. Experimental results show that the performance of the d-vector SV system is reasonably good compared to an i-vector system, and system fusion achieves much better results than the standalone i-vector system. A simple sum fusion of these two systems can improve the i-vector system performance in all operating points. The EER of the combined system is 14% and 25% better than our classical i-vector system in clean and noisy conditions respectively. Furthermore, the d-vector system is more robust to additive noise in enrollment and evaluation data. At low False Rejection operating points, the d-vector system outperforms the i-vector system.

Future work includes improving the current cosine distance scoring, as well as trying normalization schemes such as Gaussianization for the raw scores. We will explore different combination approaches, such as using a PLDA model over the the feature space of the i-vectors and d-vectors stacked. Finally, we aim to investigate the effect of increasing the number of development speakers and how speaker clustering affects performance.

Acknowledgments

The authors would like to thank our colleague Georg Heigold for help with training the DNN models.

6. REFERENCES

[1] D. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.

[2] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1435–1447, 2007.

[3] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker and session variability in GMM-based speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1448–1460, 2007.

[4] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of interspeaker variability in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 980–988, 2008.

[5] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, pp. 788–798, 2011.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.

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

[8] T. Stafylakis, P. Kenny, P. Ouellet, P. Perez, J. Kockmann, and P. Dumouchel, "Text-dependent speaker recogntion using PLDA with uncertainty propagation," in *Proc. Interspeech*, 2013.

[9] H. Aronowitz, "Text-dependent speaker verification using a small development set," in *Proc. Odyssey Speaker and Language Recognition Workshop*, 2012.

[10] A. Larcher, K.-A. Lee, B. Ma, and H. Li, "Phonetically-constrained PLDA modeling for text-dependent speaker verification with multiple short utterances," in *Proc. ICASSP*, 2013.

[11] J. Oglesby and J. S. Mason, "Optimisation of neural models for speaker identification," in *Proc. ICASSP*, 1990.

[12] Y. Bennani and P. Gallinari, "Connectionist approaches for automatic speaker recognition," in *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, 1994.

[13] B. Yegnanarayana and S.P. Kishore, "AANN: an alternative to GMM for pattern recognition," *Neural Networks*, vol. 15, no. 3, pp. 459–469, 2002.

[14] L.P. Heck, Y. Konig, M.K. Sönmez, and M. Weintraub, "Robustness to telephone handset distortion in speaker recognition by discriminative feature design," *Speech Communication*, vol. 31, no. 2, pp. 181–192, 2000.

[15] H. Lee, Y. Largman, P. Pham, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *NIPS*, 2009.

[16] T. Stafylakis, P. Kenny, M. Senoussaoui, and P. Dumouchel, "Preliminary investigation of Boltzmann machine classifiers for speaker recognitin," in *Proc. Odyssey Speaker and Language Recognition Workshop*, 2012.

[17] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. JMLR*, 2013, pp. 1319–1327.

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Susskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," in *arXive preprint*, 2012.

[19] G. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

[20] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

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

[22] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, pp. 42–54, 2000.

B.6 Large-Scale Speaker Identification

TITLE	**Large-Scale Speaker Identification.**	
CONFERENCE	ICASSP 2014	Florence, Italy.
AUTHORS	Ludwig Schmidt	Google Inc.
	Matthew Sharifi	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
DOI	10.1109/ICASSP.2014.6853878	
DATE	2014/05	
PUBLISHER	IEEE	New York, NY, US

Summary: In this paper, we propose a speaker recognition system based on i-vectors and locality sensitive hashing, an algorithm for fast nearest neighbor search in high dimensions. We show that locality sensitive hashing allows us to quickly approximate the cosine distance in our retrieval procedure based on i-vectors, thus enabling feasible large-scale speaker recognition applications.

Contributions: The candidate developed the necessary code to generate the i-vector systems; supervised the data collection from over one thousand YouTube videos; and supervised the generation and optimization of i-vector models.

LARGE-SCALE SPEAKER IDENTIFICATION

Ludwig Schmidt[*] *Matthew Sharifi and Ignacio Lopez Moreno*

MIT Google, Inc.

ABSTRACT

Speaker identification is one of the main tasks in speech processing. In addition to identification accuracy, large-scale applications of speaker identification give rise to another challenge: fast search in the database of speakers. In this paper, we propose a system based on i-vectors, a current approach for speaker identification, and locality sensitive hashing, an algorithm for fast nearest neighbor search in high dimensions. The connection between the two techniques is the cosine distance: on the one hand, we use the cosine distance to compare i-vectors, on the other hand, locality sensitive hashing allows us to quickly approximate the cosine distance in our retrieval procedure. We evaluate our approach on a realistic data set from YouTube with about 1,000 speakers. The results show that our algorithm is approximately one to two orders of magnitude faster than a linear search while maintaining the identification accuracy of an i-vector-based system.

Index Terms— speaker identification, i-vectors, locality sensitive hashing, kd-tree, indexing

1. INTRODUCTION

Speaker identification is one of the core problems in speech processing and acoustic modeling. Applications of speaker identification include authentication in security-critical systems, personalized speech recognition and searching for speakers in large corpora [1]. Due to the increasing amount of data – especially in web-scale applications – fast processing of speech data is becoming increasingly important. While the audio corpus can usually be pre-processed offline and in parallel, the retrieval procedure directly impacts user latency and needs to be executed as quickly as possible. In this paper, we study the problem of fast, text-independent speaker identification in large corpora. Our focus is on maintaining good identification performance while significantly increasing the speed of retrieval. In order to achieve this goal, we combine an i-vector-based speaker identification system with locality sensitive hashing (LSH) [2, 3], a powerful tool for approximate nearest neighbor search in high dimensions.

In this work, we are particularly interested in searching YouTube videos for a given speaker. YouTube is a prime example for the challenges of fast retrieval from a large data set: per day, about 16 years of video are currently being uploaded to YouTube [4]. Even if only a small fraction is human speech, the amount of data to be processed for a single query is still tremendous.

We show that our LSH-based retrieval approach is around 30× faster than a standard linear search on a realistic data set from YouTube with around 1,000 speakers. At the same time, the identification accuracy is still within 95% of the more expensive algorithm. Since we use LSH to approximate the cosine distance of i-vectors, our approach also has provable performance guarantees. Furthermore, there are implementations of LSH-based similarity search for data sets with more than one billion items [5]. Hence our approach promises excellent scalability for large-scale data.

2. BACKGROUND

2.1. Speaker identification with i-vectors

Robustly recognizing a speaker in spite of large *inter-session variability*, such as background noise or different communication channels, is one of the main limitations for speaker identification systems. In recent years, this challenge has been addressed with the Factor Analysis (FA) paradigm, which aims to express the main "factors" contributing to the observed variability in a compact way. Initial studies in this direction led to the Join Factor Analysis (JFA) formulation [6], where the acoustic space is divided into different subspaces. These subspaces independently model factors associated with the session variability and factors contributing to the *inter-speaker variability*, i.e., a speaker corresponds to a vector in a low-dimensional subspace.

The JFA model evolved into the Total Variability Model (TVM) [7], where all sources of variability (both speaker and session) are modeled together in a single low-dimensional space. In the TVM approach, the low-dimensional vector of latent factors for a given utterance is called the *i-vector*, and i-vectors are considered sufficient to represent the differences between various utterances. Now, speaker information and undesirable session effects are separated entirely in the i-vector domain. This separation step is typically carried out via classical Linear Discriminant Analysis (LDA) and / or Within Class Covariance Normalization (WCCN) [8]. The cosine distance is typically used for the final comparison of a speaker reference i-vector with an utterance i-vector [7]. Hereafter, we refer to the Total Variability system followed by the classical LDA and WCCN simply as Total Variability or TVM.

More recently, Probabilistic Linear Discriminant Analysis (PLDA) [9] has been proposed to independently model the speaker and session factors in the i-vector space with a probabilistic framework. However, this method performs a more complicated hypothesis test for i-vector matching, which impedes its use with LSH.

2.2. Locality sensitive hashing

The nearest neighbor problem is a core element in many search tasks: given a set of a points $\{x_1, \ldots, x_n\} \subseteq X$, a query point $q \in X$ and a distance function $d : X \times X \rightarrow \mathbb{R}^+$, find the point x_i minimizing $d(x_i, q)$. While efficient data structures for the exact problem in low-dimensional spaces are known, they have an exponential dependence on the dimension of X ("curse of dimensionality"). In order to circumvent this issue, LSH offers a trade-off between accuracy and running time. Instead of finding the exact nearest neighbor, the algorithm can return an approximate nearest neighbor, with the retrieval

[*] Research conducted as an intern at Google.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

time depending on the quality of the approximation. An approximation guarantee is still useful because the distance function d is often only an approximation of the ground truth. A particular strength of LSH is its provably sublinear running time, which also holds in practice and has led to many applications of the algorithm [3].

In order to use LSH with a given distance function d, the algorithm relies on a family of *locality sensitive hash functions*. Intuitively, a hash function is locality sensitive if two elements that are close under d are more likely to collide. There is a large body of research on locality sensitive hash functions for a wide range of distance metrics, including the Euclidean distance [10], Jaccard index [11] and the cosine distance [12].

Given a family of locality sensitive hash functions, the LSH algorithm builds a set of hash tables and hashes all points x_i into each hash table. For each hash table, we concatenate several locality sensitive hash functions to avoid unnecessary collisions (boosting precision). We maintain several hash tables to increase the probability of finding a close neighbor (boosting recall). Given a query point q, we look through all hash tables to find the x_i colliding with q and then return the best match.

2.3. Related work

There is a large body of work on using LSH for audio data. A common use case is efficiently finding remixed songs or near-duplicates in a large corpus [13, 14, 15, 16, 17, 18]. These systems are based on low-level acoustic features such as MFCCs and then use a variety of LSH-related techniques, e.g. shingles and min-hashing. While this approach works well for finding near-duplicate audio data, we are interested in text-*independent* speaker identification and hence require a more sophisticated acoustic model.

A recent paper [19] uses LSH for speaker identification with privacy guarantees. The authors mention efficiency gains due to LSH but use cryptographic hash functions and focus on the security aspects of their work. The speaker identification step is based on Gaussian mixture model (GMM) super-vectors without factor analysis while our work uses i-vectors. Moreover, the authors study the performance of their system on the YOHO data set [20], which consists of 138 speakers and is primarily intended for text-*dependent* speaker authentication.

The most closely related work is [21]. While the authors also employ factor analysis in their acoustic modeling step, their utterance comparison model [22] is different from i-vectors. Importantly, the authors use kernelized-LSH [23] in order to implement the distance function implied by their model. In contrast, we use the cosine distance, which is known to give good performance for i-vectors [7] and also provides provable guarantees in conjunction with LSH [12]. Furthermore, the authors of [21] explicitly state that the objective of their paper is to investigate the performance of their method on close-talk microphone recordings with *matched conditions*, leaving a more robust variant that is resistant to noise for further study. In our evaluation, we use a set of videos from YouTube that was not recorded for speaker identification. In particular, the channel effects in some videos are significantly different and noise is present in most recordings.

3. LOCALITY SENSITIVE HASHING FOR SPEAKER IDENTIFICATION

For clarity, we describe our proposed system as two separate components: the generation of i-vectors and the fast retrieval of similar i-vectors using LSH.

3.1. Generation of i-vectors

Given an utterance for which we want to generate an i-vector, we first represent the utterance in terms of a large GMM, the so-called Universal Background Model (UBM), which we parametrize with λ. Formally, let $\Theta = (o_1, \dots, o_a)$ with $o_i \in \mathbb{R}^D$ be a sequence of spectral observations extracted from the utterance. Then we compute the accumulated and centered first-order Baum-Welch statistics

$$N_m = \sum_t P(m|o_t, \lambda)$$

$$F_m = \sum_t P(m|o_t, \lambda)(o_t - \mu_m) \ ,$$

where μ_m is the mean vector of mixture component m, $m = 1, ..., C$ ranges over the mixture components of the UBM and $P(m|o, \lambda)$ is the Gaussian occupation probability for mixture m and observation o. Hereafter, we refer to $F \in \mathbb{R}^{CD}$ as the vector containing the stacked statistics $F = (F_1^T, \dots, F_C^T)^T$.

We now denote the i-vector associated with the sequence Θ as $x \in \mathbb{R}^d$. According to the TVM model, the vector F is related to x via the rectangular low-rank matrix $T \in \mathbb{R}^{CD \times d}$, known as the TVM subspace:

$$N^{-1}F = Tx \ ,$$

where $N \in \mathbb{R}^{CD \times CD}$ is a diagonal matrix with C blocks of size $D \times D$ along the diagonal. Block $m = 1, .., C$ is the matrix $N_m I_{(D \times D)}$.

The constraints imposed on the distributions of $P(x)$ and $P(F|x)$ lead to a closed-form solution for $P(x|F)$. The i-vector is the maximum a posteriori (MAP) point estimate of this distribution and is given by

$$x = (I + T^T \Sigma^{-1} N T)^{-1} T^T \Sigma^{-1} F \ ,$$

where $\Sigma \in \mathbb{R}^{CD \times CD}$ is the covariance matrix of F.

Therefore, our i-vector extraction procedure depends on the utterance data and the TVM model parameters λ, T and Σ. We refer to [24] for a more detailed explanation of how to obtain these parameters using the EM algorithm.

If the true speaker labels for each training i-vector are known, the final speaker i-vector is normally obtained by averaging all i-vectors belonging to the same speaker. Since we are interested in an unsupervised setting such as YouTube where speaker labels are not available for most of the utterances, we do not perform this i-vector averaging step in our system and instead keep the i-vectors of all utterances.

3.2. Locality sensitive hashing with i-vectors

As noted in the introduction, the main goal of our work is enabling *fast* retrieval of speakers. In the context of i-vector-based speaker identification, this means the following: for a given query i-vector, we efficiently want to find the best match in our previously computed set of i-vectors. Since this task is an instance of the nearest neighbor problem introduced above, we use LSH in order to enable fast retrieval.

A crucial point when using LSH is the right choice of distance function d. For i-vectors, it has been shown that the cosine distance $d(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|}$ gives very competitive performance for speaker identification [7]. Since the cosine distance can also be approximated well with locality sensitive hash functions [12], we use the cosine distance in our LSH algorithm. In particular, we use hash functions of the form

$$h_r(x) = \begin{cases} 1 & \text{if } x \cdot r \geq 0 \\ 0 & \text{if } x \cdot r < 0 \end{cases} ,$$

where we choose r as a random Gaussian vector. Geometrically, this hash function can be seen as hashing with a random hyperplane: r is perpendicular to the hyperplane and the result of the hash function indicates on which side of the hyperplane x lies. Since r has an isotropic distribution, we have $P[h_r(x) = h_r(y)] = 1 - \theta(x, y)/\pi$, where $\theta(x, y)$ is the angle between vectors x and y.

Our data structure has two main parameters: l, the number of hash tables, and k, the number of hyperplanes per hash table. Let $H_1, \ldots, H_l$ be the hash tables in our data structure. We use the construction from [25] in order to reduce the number of hash function evaluations: we maintain $m \approx \sqrt{l}$ hash functions of length $\frac{k}{2}$ and use the $\binom{m}{2} \approx l$ combinations as hash functions for the l hash tables. Formally, let $u_i(x) = (h_1^i(x), h_2^i(x), \ldots, h_{k/2}^i)$ for $i \in \{1, \ldots, m\}$ and h_j^i sampled as described above. Then the final hash functions for the l hash tables are $h_i(x) = (u_a(x), u_b(x))$ for all choices of a, b such that $1 \leq a < b \leq m$. Hence each h_i hashes an i-vector x to a string of k bits. Note that we do not need to store a full array with 2^k entries for each hash table but can instead resort to standard hashing for large k.

For a given database of i-vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, we initialize our LSH data structure as follows: each i-vector x_i is hashed with each hash function h_j and then inserted at position $h_j(x_i)$ in hash table H_j. The overall time complexity of the initialization step is $O(ndk\sqrt{l} + nl)$.

Algorithm 1 describes the fast retrieval procedure. The evaluation of the m hash functions u_i in lines 2 and 3 can be efficiently implemented with a vector-matrix multiplication as follows: we stack the normal vectors of the hyperplanes as rows into a matrix $U \in \mathbb{R}^{mk/2 \times d}$. The bits used in the hash functions are then given by $\frac{\mathrm{sgn}(Ux)+1}{2}$. The running time of the retrieval procedure is $O(dk\sqrt{l} + l + M)$, where M is the total number of matches found. M is typically small if the parameters k and l are properly chosen.

Algorithm 1 I-vector retrieval with LSH

```
1: function RetrieveIvector($q$)
2:     for $i \leftarrow 1, \ldots, m$ do
3:         Evaluate $u_i(q)$
4:     $C \leftarrow \{\}$                                   ▷ Set of candidates
5:     for $i \leftarrow 1, \ldots, l$ do
6:         $C \leftarrow C \cup H_i[h_i(q)]$                 ▷ Add candidates
7:     return $\arg\min_{x \in C} \frac{x \cdot q}{\|x\| \|q\|}$    ▷ Return best candidate
```

4. EXPERIMENTS

For our experiments, we focus on the main application outlined in the introduction: searching for speakers on YouTube.

4.1. Data set

We built a data set from the Google Tech Talk channel on YouTube [26], which contains about 1,803 videos of talks given at Google. Many of the videos have speaker labels and contain one main speaker. We decided on this data set because we wanted to use YouTube data with good ground truth information while avoiding manual labeling and speaker diarization issues.

After removing short videos and videos with more than one speaker, our data set contains 1,111 videos with 998 distinct speakers, with each video containing at least 30 minutes of the corresponding talk. 74 speakers appear in at least two videos. The recording quality of the videos varies with respect to audience noise, equipment, room acoustics and the distance from the speaker to the microphone. The list of videos with a unique ID for each speaker is available online at `http://people.csail.mit.edu/ludwigs/data/youtube_speakers_2013.txt`.

4.2. Results

We conducted two types of experiments. For each type of experiment, we studied the performance of our algorithm for utterances of length 10, 20 and 60 seconds. The utterances were selected randomly from the videos and correspond directly to segments of the talk. Hence, the duration of speech per utterance can be less than its nominal duration.

10tests In this setup, we divide each video into utterances of length t and then select 10 random utterances from each video to form the query set. The remaining utterances are used for i-vector training and as the retrieval database.

holdout10 For each speaker with at least two videos, we select one video randomly as the source of query i-vectors. All remaining videos are used for i-vector training and as the retrieval database. We then extract 10 random utterances of length t from the query videos and use the corresponding i-vectors as query points.

The second experiment setup is significantly more challenging as the speaker identification system has to ignore the channel mismatch between different videos. We include results for the first type of experiments to study the performance of our LSH data structure under matched conditions.

Since the goal of our work is fast retrieval, we focus on the trade-off between identification accuracy and computational efficiency. We use a linear search as comparison baseline and measure the following quantities:

Identification accuracy We count a query as identified correctly if the speaker corresponding to the query i-vector is the same as the speaker corresponding to the i-vector returned by the retrieval algorithm. We only consider the single top-scored speaker returned by the retrieval algorithm. The identification accuracy is then the empirical probablity of correct identification.

Retrieval time We measure the time the retrieval algorithm takes to return a candidate i-vector for a given query i-vector. Note that we exclude the time for converting a query utterance to a query i-vector. Since the time complexity of the conversion step is independent of the size of the database, the i-vector retrieval step will dominate the overall computational cost for large data sets.

Table 1 shows the key details of our experiments. In all experiments, we used 200-dimensional i-vectors. Figures 1 and 2 illustrate the trade-off between accuracy and performance we can achieve by varying the parameters of our LSH data structure. All experiments were conducted on a 3.2GHz CPU and each data point was averaged over all query points, with 10 trials per query point.

The results show that we can achieve speedups by one to two orders of magnitude while sacrificing only a small fraction of the identification accuracy. For an utterance length of 20s, our retrieval algorithm is roughly 150 times faster than the baseline on the 10tests experiments and about 35 times faster on the holdout experiments. In both cases, the relative accuracy is about 95% (we define relative accuracy as the ratio (LSH accuracy)/(baseline accuracy)). Moreover, we can achieve a wide range of trade-offs by varying the parameters of our LSH data structure.

An interesting phenomenon in our results is the performance gap between the matched and unmatched settings (10tests and holdout10, respectively). This discrepancy is probably due to the fact that the i-vectors are better clustered under well-matched recording conditions and consequently, the approximate guarantees of LSH have less impact. Therefore, considering only a small number of candidates in the hash tables is sufficient to find a correct match. In

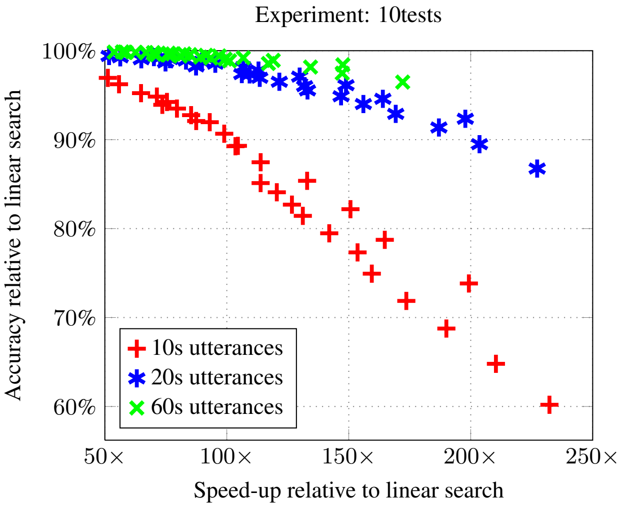


Fig. 1. Speedup vs. accuracy trade-off for the 10tests experiments. Each point corresponds to a choice of parameters for our LSH data structure. We vary k (the number of hyperplanes) from 6 to 20 in steps of 2 and l (the number of hash tables) from 80 to 300 in steps of 20.

contrast, the search for matches across videos is more challenging and hence requires iterating over a larger set of candidates.

We also compare our LSH-based retrieval algorithm with kd-trees, another popular method for nearest neighbor search in high dimensions [27]. In particular, we use the ANN library [28]. While kd-trees show superior performance on the simple 10tests data set, LSH is significantly faster on the more realistic holdout10 data set. Again, we suppose that this difference is due to the more challenging geometry of searching across videos.

Note that our results are not directly comparable with the speedups reported in [21]. In addition to using LSH, the authors also accelerate the computation of their utterance comparison model with a kernel function and include the resulting performance improvement in their LSH speedup. In particular, the authors report a speedup of 911× for a linear search using their kernel function, compared to a full evaluation of their utterance comparison model as baseline. Expressed in our metrics (i.e., after subtracting the time spent on computing the vector representation), their LSH scheme achieves a speedup of about 12× for *matched* conditions.

5. CONCLUSION

We have proposed a fast retrieval method for speaker identification in large data sets. Our work is based on combining two powerful approaches that interact via the cosine distance: locality sensitive hashing, which enables fast nearest neighbor search, and i-vectors, which provide good identification accuracy. Results on a realistic, large data set from YouTube show that we can achieve speedups of one to two orders of magnitude while sacrificing only a very small fraction of the identification accuracy. Hence our approach is a very promising candidate for large-scale speaker identification. Moreover, LSH could also be very useful for other large-scale applications of i-vectors, such as clustering.

6. ACKNOWLEDGEMENTS

L.S. would like to thank Piotr Indyk for helpful discussions.

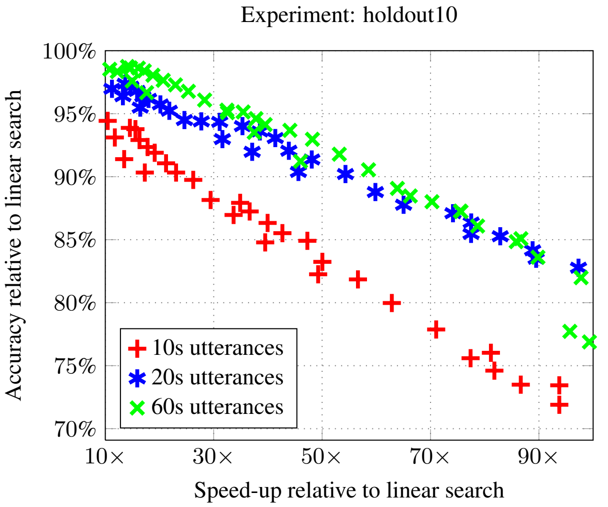


Fig. 2. Speedup vs. accuracy trade-off for the holdout10 experiments. Each point corresponds to a choice of parameters for our LSH data structure. We vary k (the number of hyperplanes) from 6 to 20 in steps of 2 and l (the number of hash tables) from 80 to 300 in steps of 20.

	Experiment					
	10tests			holdout10		
	10s	20s	60s	10s	20s	60s
Database size	44,348	44,394	22,213	51,736	51,803	31,104
Query i-vectors	1,200	1,200	1,200	740	739	740
Baseline accuracy	99.0%	99.5%	99.7%	53.4%	64.4%	74.2%
Baseline time (ms)	6.65	6.66	3.38	7.70	7.76	4.67
LSH accuracy	91.0%	95.6%	98.1%	50.1%	60.6%	70.6%
LSH time (ms)	0.071	0.045	0.023	0.495	0.220	0.132
Relative accuracy	**92.0%**	**96.1%**	**98.4%**	**93.8%**	**94.0%**	**95.1%**
Relative speedup	**93×**	**149×**	**148×**	**16×**	**35×**	**35×**
Hyper-planes (k)	18	18	18	14	16	16
Hash tables (l)	220	120	80	280	280	280
kd-tree rel. accuracy	95.2%	96.4%	98.4%	92.9%	93.7%	95.4%
kd-tree rel. speedup	568×	987×	773×	3.6×	2.7×	3.0×

Table 1. Summary of the data sets and results for the two types of experiments (10tests, holdout10) and three utterance lengths (10s, 20s, 60s). The relative accuracy is the ratio (LSH accuracy)/(baseline accuracy) and the relative speedup is the ratio (baseline time)/(LSH time). For each experiment, we present a choice of parameters for our LSH data structure so that we achieve a relative accuracy of at least 90%. The relative accuracy and speedup for kd-trees is also reported relative to the baseline.

7. REFERENCES

[1] D.A. Reynolds, “An overview of automatic speaker recognition technology,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[2] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *ACM Symposium on Theory of Computing (STOC)*, 1998.

[3] S. Har-Peled, P. Indyk, and R. Motwani, “Approximate nearest neighbor: Towards removing the curse of dimensionality,” *Theory of Computing*, vol. 8, no. 14, pp. 321–350, 2012.

[4] “Youtube statistics,” `http://www.youtube.com/yt/press/statistics.html`, accessed 3 November 2013.

[5] N. Sundaram, A. Turmukhametova, N. Satish, T. Mostak, P. Indyk, S. Madden, and P. Dubey, “Streaming similarity search over one billion tweets using parallel locality sensitive hashing,” in *International Conference on Very Large Data Bases (VLDB)*, 2014.

[6] P. Kenny, “Joint factor analysis of speaker and session variability: theory and algorithms,” Tech report CRIM-06/08-13, 2005.

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

[8] A.O. Hatch, S.S. Kajarekar, and A. Stolcke, “Within-class covariance normalization for SVM-based speaker recognition.,” in *International Conference on Spoken Language Processing (INTERSPEECH)*, 2006.

[9] P. Kenny, “Bayesian speaker verification with heavy-tailed priors,” in *Odyssey: The Speaker and Language Recognition Workshop*, 2010.

[10] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” in *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[11] A.Z. Broder, “On the resemblance and containment of documents,” in *Compression and Complexity of Sequences*, 1997.

[12] M. Charikar, “Similarity estimation techniques from rounding algorithms,” in *ACM Symposium on Theory of Computing (STOC)*, 2002.

[13] S. Baluja and M. Covell, “Content fingerprinting using wavelets,” in *European Conference on Visual Media Production (CVMP)*, 2006.

[14] M. Magas, M. Casey, and C. Rhodes, “mHashup: fast visual music discovery via locality sensitive hashing,” in *ACM SIGGRAPH new tech demos*, 2008.

[15] M. Casey, “AudioDB: Scalable approximate nearest-neighbor search with automatic radius-bounded indexing,” *The Journal of the Acoustical Society of America*, vol. 124, no. 4, pp. 2571–2571, 2008.

[16] M. Casey and M. Slaney, “Fast recognition of remixed music audio,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[17] M. Casey and M. Slaney, “Song intersection by approximate nearest neighbor search,” in *International Conference on Music Information Retrieval (ISMIR)*, 2006.

[18] M. Casey, C. Rhodes, and M. Slaney, “Analysis of minimum distances in high-dimensional musical spaces,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1015–1028, 2008.

[19] M.A. Pathak and B. Raj, “Privacy-preserving speaker verification as password matching,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[20] J.P. Campbell Jr., “Testing with the YOHO CD-ROM voice verification corpus,” in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1995.

[21] W. Jeon and Y. Cheng, “Efficient speaker search over large populations using kernelized locality-sensitive hashing,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[22] W. Jeon, C. Ma, and D. Macho, “An utterance comparison model for speaker clustering using factor analysis,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[23] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[24] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, “A study of interspeaker variability in speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[25] A. Andoni and P. Indyk, “Efficient algorithms for substring near neighbor problem,” in *ACM-SIAM symposium on Discrete algorithm (SODA)*, 2006.

[26] “Youtube search for google tech talks,” `https://www.youtube.com/results?q=GoogleTechTalks`, accessed 3 November 2013.

[27] J. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sept. 1975.

[28] D. Mount and S. Arya, “Ann: A library for approximate nearest neighbor searching,” `http://www.cs.umd.edu/~mount/ANN/`, accessed 3 November 2013.

B.7 Automatic Language Recognition Using Deep Neural Networks

TITLE	**Automatic Language Recognition Using Deep Neural Networks.**	
CONFERENCE	ICASSP 2014	Florence, Italy.
AUTHORS	Ignacio López Moreno	UAM / Google Inc.
	Javier González Domínguez	UAM
	Oldřich Plchot	Brno Univ. of Technologys
	David Martínez	Univ. of Zaragoza
	Joaquin González Rodríguez	UAM
	Pedro J. Moreno	Google Inc.
DOI	10.1109/ICASSP.2014.6854622	
DATE	2014/05	
PUBLISHER	IEEE	New York, NY, US

Summary: This conference article is a precursor to the journal article entitled Frame-by-Frame Language Identification in Short Utterances Using Deep Neural Networks. This article introduces Feed-forward Deep Neural Networks (DNN) for large scale Spoken Language Identification task. The proposed system focuses on recognizing over 30 languages form short Google voice queries. It explores the configurations under which DNN architectures can outperform alternative state-of-the-art approaches.

Contributions: The candidate proposed neural networks as a novel domain of application for language recognition; provided the necessary code to generate neural networks for language recognition; provided the necessary code to generate the baseline i-vector systems; collected approximately one year of labeled data (audio and parameters) from real voice queries; generated baseline i-vector and candidate DNN-based systems with optimized hyperparameters; and collaborated in validating the system with experimental results.

AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS

Ignacio Lopez-Moreno[1], *Javier Gonzalez-Dominguez*[1,2], *Oldrich Plchot*[3], *David Martinez*[4],
Joaquin Gonzalez-Rodriguez[2], *Pedro Moreno*[1]

[1]Google Inc., New York, USA
[2]ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Spain
[3]Brno University of Technology, Czech Republic
[4]Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain
{elnota,jgd}@google.com

ABSTRACT

This work studies the use of deep neural networks (DNNs) to address automatic language identification (LID). Motivated by their recent success in acoustic modelling, we adapt DNNs to the problem of identifying the language of a given spoken utterance from short-term acoustic features. The proposed approach is compared to state-of-the-art i-vector based acoustic systems on two different datasets: Google 5M LID corpus and NIST LRE 2009. Results show how LID can largely benefit from using DNNs, especially when a large amount of training data is available. We found relative improvements up to 70%, in C_{avg}, over the baseline system.

***Index Terms*—** Automatic Language Identification, i-vectors, DNNs

1. INTRODUCTION

The problem of automatic language identification (LID) can be defined as the process of automatically identifying the language of a given spoken utterance [1]. LID is daily used in several applications such as multilingual translation systems or emergency call routing, where the response time of a fluent native operator might be critical [1] [2].

Even though several high level approaches based on phonotactic and prosody are used as meaningful complementary sources of information [3][4][5], nowadays, many state-of-the-art LID systems still include or rely on acoustic modelling [6][7]. In particular, guided by the advances on speaker verification, the use of i-vector extractors as a front-end followed by diverse classification mechanisms has become the state-of-the-art in acoustic LID systems [8][9].

While previous works on neural networks applied to LID report results using shallow architectures [10][11] or convolutional neural networks [12], in this study, we propose the use of deep neural networks (DNNs) as a new method to perform LID at the acoustic level. Deep neural networks have recently proved to be successful in diverse and challenging machine learning applications, such as acoustic modelling [13] [14], visual object recognition [15] and many others [16]; especially when a large amount of training data is available.

Motivated by those results and also by the discriminative nature of DNNs, which could complement the i-vector generative approach, we adapt DNNs to work at the acoustic frame level to perform LID. Particularly, in this work, we build, explore and experiment with several DNNs configurations and compare the obtained results with several state-of-the-art i-vector based systems trained from exactly the same acoustic features.

To assess the proposed method's performance we experiment on two different and challenging LID datasets: 1. A dataset built from Google data, hereafter, Google 5M LID corpus and 2. The NIST Language Recognition Evaluation (LRE'09). Thus, first, we test the proposed approach in a real application; and second, we check if the same behaviour is observed in a familiar and standard evaluation framework for the LID community. In both cases, we focus on short test utterances (up to 3s).

The rest of this paper is organized as follows. Section 2 presents the i-vector based baseline systems, the proposed DNN architecture as well as the score calibration procedure. The experimental protocol and datasets used are then described in section 3. Results are discussed in section 4. Finally, section 5 is devoted to present conclusions and evaluate proposals for future work.

2. DEVELOPED SYSTEMS

2.1. i-vector Based LID Systems

To establish a baseline framework, we built different state-of-the-art LID acoustic systems based on i-vectors [9]. All those systems, while sharing i-vectors as the same starting point, differ in the type of back-end used to perform the final language classification.

From 39 PLP (13 + Δ + $\Delta\Delta$) feature vectors extracted

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

with a 10ms frame rate over 25ms long windows, we followed the standard recipe described in [17] to obtain i-vectors. We trained a Universal Background Model (UBM) with 1024 components and a 400-dimensional total variability subspace initialized by PCA and refined by 10 iterations of EM. Also, we filtered-out silence frames by using energy-based voice activity detector.

Once the i-vectors for every language were extracted, we used different strategies to perform classification. On the one hand, as a discriminative approach, we performed linear Logistic Regression (LR). On the other hand, two generative approaches were tested, LDA followed by cosine distance (LDA_CD), and a Gaussian modelling to fit the i-vectors of each language, with one (1G) or two components - with and without tied covariances - (2G_TC, 2G). We also explored the effect of using a single shared covariance across the languages (1G_SC) vs. per-language covariances. For further details about this approach, see [9].

2.2. DNN-based LID System

The DNN architecture used in this work is a fully connected feed-forward neural network [18]. The hidden layers contain units with rectified linear activation functions. The output is configured as a softmax layer with a cross-entropy cost function. Each hidden layer contains h (2560) units while the output layer dimension (s) corresponds to the number of target languages (N_L) plus one extra output for the out-of-set (oos) languages.

The DNN works at frame level, using the same features as the baseline systems described above (39 PLP). Specifically, the input layer is fed with 21 frames formed by stacking the current processed frame and its ±10 left-right context. Therefore, there are 819 (21 × 39) visible units, v. The number of total weights w, considering N_{hl} hidden layers, can be then easily computed as $w = (v \times h) + ((N_{hl} - 1) \times h \times h) + h \times s$. Figure 1 represents the complete topology of the network.

We trained all the DNN architectures presented in this work using asynchronous stochastic gradient descent within the DistBelief framework [19]. We also fixed the learning rate and minibatch size to 0.001 and 200 samples. Finally, we computed the output scores at utterance level by respectively averaging the log of the softmax output of all its frames (i.e.: log of the predicted posterior probabilities).

2.3. Logistic Regression Calibration

Our scores were calibrated using discriminatively trained, regularized multiclass logistic regression [20]. The calibration was trained in the ”cheating” way, that is, using the evaluation scores themselves. The reason, why we performed the cheating calibration, was to concentrate on the ability of the underlying models to discriminate between the given classes. We did not want to introduce other errors coming from over-training the systems on the training data-set and therefore producing miscalibrated scores for our evaluation set.

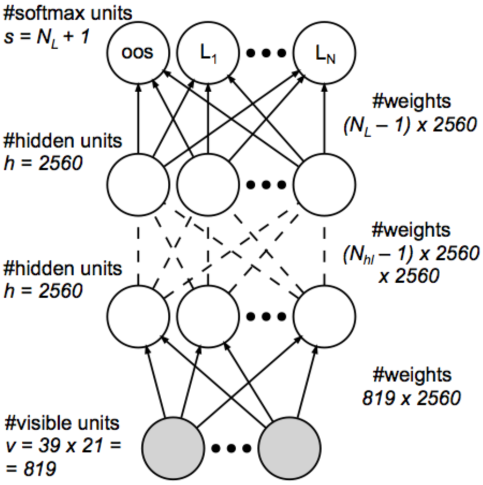


Fig. 1. DNN network topology

The L2 regularization penalty weight was chosen prior to training to be proportional to the mean magnitude of the conditioned input vectors (scores) [21].

The calibration uses an affine transform to convert the N_L-dimensional vector of input scores, $\mathbf{s}_t$, for trial t, into a N_L-dimensional calibrated score-vector, $\mathbf{r}_t$

$$\mathbf{r}_t = \mathbf{C}\mathbf{s}_t + \mathbf{d}, \tag{1}$$

The logistic regression parameters are given by $\mathbf{C}$, a full N_L-by-N_L matrix, and $\mathbf{d}$, a N_L-dimensional vector and they are trained by minimizing the multiclass cross-entropy with equalizing the amount of data for individual classes

$$F = \lambda\,\mathrm{tr}(\mathbf{C}^{\mathrm{T}}\mathbf{C}) - \sum_{i=1}^{N_L} \frac{1}{N_L N_i} \sum_{t \in \mathcal{R}_i} \log \frac{\exp(r_{it})}{\sum_{j=1}^{N} {}_L \exp(r_{jt})}, \tag{2}$$

where r_{it} is the ith component of $\mathbf{r}_t$ and $\mathcal{R}_i$ is the set of N_i training examples of language i.

3. EXPERIMENTAL PROTOCOL

3.1. Databases

Google 5M LID Corpus

We generated The Google 5M LID Corpus dataset by randomly picking queries from several Google speech recognition services such as Voice Search or the Speech API.

The Google ASR lattice posteriors were used to discard non-speech queries. Selected queries range from 1s up to 8s nominal duration, with average speech content of 2.1s.

Following the user's phone Voice Search language settings, we labelled a total of ∼5 million utterances, 150k per 34 different locales (25 languages + 9 dialects) yielding ∼87,5h of speech per language and a total of ∼2975h. A held-out test set of 1.5k utterances per language was created while the remainder was used for training and development.

Google queries are not linked to user identity information due to privacy concerns, and therefore, determining the exact number of speakers involved in this corpus is not possible. However, given the selection procedure, it is a reasonable assumption that the number of speakers is very large.

Language Recognition Evaluation 2009 Dataset.

The LRE evaluation in 2009 included, for the first time, data coming from two different audio sources. Besides Conversational Telephone Speech (CTS), used in the previous evaluations, telephone speech from broadcast news was used for both training and test purposes. Broadcast data were obtained via an automatic acquisition system from "Voice of America" news (VOA) where telephone and non-telephone speech is mixed. Up to 2TB of 8KHz raw data containing radio broadcast speech, with the corresponding language and audio source labels were distributed to participants; and a total of 40 languages (23 target and 17 out of set) were included.

Due to the large disparity on training material for every language (from ∼10 to ∼950 hours) and also, for the sake of clarity, we selected 8 representative languages for which at least 200 hours of audio are available: en (US English), es (Spanish), fa (Dari), fr (French), ps (Pashto), ru (Russian), ur (Urdu), zh (Chinese Mandarin). Further, to avoid misleading result interpretation due to the unbalanced mix of CTS and VOA, all the data considered in this dataset belong to VOA.

For evaluation, we used a subset of the official NIST LRE 2009 3s condition evaluation set (as for training, we also discarded CTS test segments), yielding a total of 2916 test segments of the 8 selected languages. That makes a total of 23328 trials.

3.2. Performance Metrics

In order to assess the performance, two different metrics were used. As the main error measure to evaluate the capabilities of one-vs.-all language detection, we use C_{avg} (average cost) as defined in the LRE 2009 [22][23] evaluation plan. C_{avg} is a measure of the cost of taking bad decisions, and therefore it considers not only discrimination, but also the ability of setting optimal thresholds (i. e., calibration). Further, well-known metric Equal Error Rate (EER) is used to show the performance, when considering only scores of each individual language. Detailed information can be found in the LRE'09 evaluation plan [22].

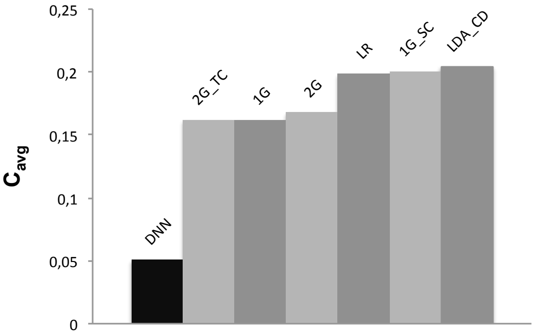


Fig. 2. C_{avg} results on Google 5M LID corpus. 8-hidden layer DNN vs. reference systems based on i-vectors.

4. RESULTS

4.1. Results on Google 5M LID Corpus

As a starting point for this study we compare the performance of the proposed DNN architecture and the reference systems on the large Google 5M LID dataset. Figure 2 shows this comparison. Considering i-vector systems, we found a similar performance for the discriminative back-end, Logistic Regression (LR) and the generative ones, Linear Discriminant Analysis (LDA_CD) and the one based on a single Gaussian with a shared covariance matrix across the languages (1G_SC). Interestingly, increasing to 2 Gaussians and allowing individual covariances matrices (systems 1G, 2G_TC, 2G) a relative improvement of ∼19% is obtained. respect to LR, LDA_CD and 1G systems. This fact suggests that within-class distribution can be different for the individual languages.

Nonetheless, the best performance is achieved by the DNN systems, where the 8-hidden layer DNN proposed architecture yields up to a ∼70% of relative improvement in C_{avg} terms with respect to the best reference system (2G_TC). This result demonstrates the ability of the DNN to exploit discriminative information in large datasets.

4.2. Results on LRE'09

Guided by the results presented above we moved to a more extensive analysis on LRE'09 evaluation data. Per-language results summarized in Table 1, show similar improvements on the LRE'09 dataset. A relative improvement of ∼43% in EER is obtained with the 8-hidden layer DNN, trained with 200h, with respect to the classical i-vector LDA_CD system.

The effect of using different numbers of layers is also highlighted in Table 1, where in addition to the 8-hidden layer DNN and the i-vector LDA_CD system, results with a 2-hidden layer DNN (DNN_2_200h) are also reported. Similarly, although the improvements are more modest than those

	Equal Error Rate (EER in %)								
	en	es	fa	fr	ps	ru	ur	zh	Average
Iv_200h	17.22	10.92	20.03	15.30	19.98	14.87	18.74	10.09	15.89
DNN_2_200h	12.66	5.04	19.67	8.60	17.84	8.75	14.78	5.54	11.61
DNN_8_200h	**8.65**	**3.74**	**17.22**	**7.53**	**16.01**	**5.59**	**13.10**	**4.82**	**9.58**

Table 1. Systems performance (ERR %) per language on LRE'09 (3s test segments)

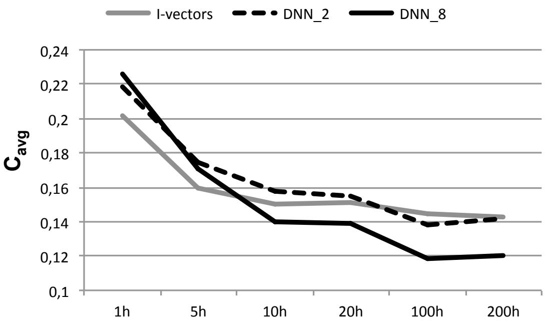


Fig. 3. DNNs vs i-vector performance in function of the per-language number of hours available. Results on LRE'09 dataset.

obtained with the 8-hidden layer network, the DNN_2_200h still outperforms the i-vector based system.

We then explore the effect of having different amounts of training data. Figure 3 shows the i-vector LDA_CD, DNN_2 and DNN_8 systems performance (C_{avg}) as a function of the number of hours per language used for training. With limited amount of data per language (<10h), the i-vector system yields the best performance. However, the more hours for training, the higher the improvement of DNN with respect to the i-vector systems. With the greatest amount of data, 200h, the relative improvement of the 8-hidden layer DNN with respect to the i-vector systems is ∼15% in C_{avg}.

This behaviour may be for several reasons. With <10h per language, the i-vector approach might be favoured by its subspace intrinsic nature. The UBM and the total variability matrix drive modelling to a constrained low-dimensional space. This fact facilitates i-vector approach to quickly retain most important language variations. Also, the number of free parameters to train in each system could play an important role (∼16M, ∼9M, ∼50M parameters for LDA_CD, DNN_2 and DNN_8 respectively). On the contrary, with abundant data (>20h) the i-vector based approach seems to saturate, while DNNs show a high ability to avoid local minima and overfitting even when containing a large number of free parameters (see performance differences between DNN_2 and DNN_8 in Figure 3).

5. DISCUSSION

In this work, we experimented with the use of deep neural networks (DNNs) to automatic language identification (LID). Guided by the success of DNNs for acoustic modelling, we explored their capability to learn discriminative language information from speech signals.

We compared the proposed DNNs architectures to several state-of-the-art acoustic systems based on i-vectors. Results on NIST LRE 2009 (8 languages selected) and Google 5M LID datasets (25 languages + 9 dialects), demonstrate that DNNs outperform, in most of the cases, current state-of-art approaches. This is especially true when large amount of data is available (> 20h), where unlike i-vectors approaches, which seem to saturate, DNNs still learn from data.

On the other hand, DNNs have several drawbacks, including the training time, or the number of parameters to store. Also, adjusting the proper number of hidden layers and units is an empirical exercise for every database. Fortunately, we found that (for the datasets used) moving from 8 to 2 hidden layers, did not have a dramatic impact on performance. Moreover, those adjustments could be done off-line, with testing time still reasonable.

As future work, we will focus on different lines such as establishing a more appropriate averaging of frame posteriors obtained in DNNs, exploring different fusions among DNNs and i-vector systems, or dealing with unbalanced training data.

6. REFERENCES

[1] Y.K. Muthusamy, E. Barnard, and R.A. Cole, "Reviewing automatic language identification," *Signal Processing Magazine, IEEE*, vol. 11, no. 4, pp. 33–41, 1994.

[2] E. Ambikairajah, Haizhou Li, Liang Wang, Bo Yin, and V. Sethu, "Language identification: A tutorial," *Circuits and Systems Magazine, IEEE*, vol. 11, no. 2, pp. 82–108, 2011.

[3] M. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 4, no. 1, pp. 31–44, 1996.

[4] L. Ferrer, N. Scheffer, and E. Shriberg, “A Comparison of Approaches for Modeling Prosodic Features in Speaker Recognition,” in *International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 4414–4417.

[5] D. Martinez, E. Lleida, A. Ortega, and A. Miguel, “Prosodic features and formant modeling for an ivector-based language recognition system,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 6847–6851.

[6] P.A. Torres-Carrasquillo, E. Singer, T. Gleason, Alan McCree, D.A. Reynolds, F. Richardson, and D. Sturim, “The MITLL NIST LRE 2009 Language Recognition System,” in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 4994–4997.

[7] J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D.T. Toledano, and J. Gonzalez-Rodriguez, “Multilevel and Session Variability Compensated Language Recognition: ATVS-UAM Systems at NIST LRE 2009,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 6, pp. 1084–1093, 2010.

[8] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and Reda Dehak, “Language Recognition via i-vectors and Dimensionality Reduction.,” in *INTERSPEECH*. 2011, pp. 857–860, ISCA.

[9] D. Martinez, O. Plchot, L. Burget, Ondrej Glembek, and Pavel Matejka, “Language Recognition in iVectors Space.,” in *INTERSPEECH*. 2011, pp. 861–864, ISCA.

[10] R.A. Cole, J.W.T. Inouye, Y.K. Muthusamy, and M. Gopalakrishnan, “Language identification with neural networks: a feasibility study,” in *Communications, Computers and Signal Processing, 1989. Conference Proceeding., IEEE Pacific Rim Conference on*, 1989, pp. 525–529.

[11] M. Leena, K. Srinivasa Rao, and B. Yegnanarayana, “Neural network classifiers for language identification using phonotactic and prosodic features,” in *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 404–408.

[12] G. Montavon, “Deep learning for spoken language identification,” in *NIPS workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[13] A. Mohamed, G.E. Dahl, and G. Hinton, “Acoustic Modeling using Deep Belief Networks,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012.

[14] G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups,” *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[15] D.C Ciresan, U. Meier, L.M. Gambardella, and J. Schmidhuber, “Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition,” *CoRR*, vol. abs/1003.0358, 2010.

[16] D. Yu and L. Deng, “Deep Learning and its Applications to Signal and Information Processing [exploratory dsp],” *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 145–154, 2011.

[17] N Dehak, P Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-End Factor Analysis for Speaker Verification,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 788 – 798, February 2011.

[18] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, “Application of Pretrained Deep Neural Networks to Large Vocabulary speech recognition,” in *Proceedings of Interspeech 2012*, 2012.

[19] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M.A Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, “Large Scale Distributed Deep Networks,” in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Eds., pp. 1232–1240. 2012.

[20] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2007.

[21] N. Brummer, S. Cumani, O. Glembek, M. Karafiat, P. Matejka, J. Pesan, O. Plchot, M. Soufifar, E. Villiers de, and J. Cernocky, “Description and Analysis of the BRNO276 system for LRE2011,” in *Proceedings of Odyssey 2012: The Speaker and Language Recognition Workshop*. 2012, pp. 216–223, International Speech Communication Association.

[22] NIST, “The 2009 NIST SLR Evaluation Plan,” www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf, 2009.

[23] N. Brummer, *Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech*, Ph.D. thesis, Department of Electrical and Electronic Engineering, University of Stellenbosch,, 2010.

Appendix C

Full Text of Patents

C.1 Identifying the Language of an Spoken Utterance

TITLE	**Identifying the Language of an Spoken Utterance.**	
INVENTORS	Javier González Domínguez	UAM
	Hasim Sak	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
DATE	2016/2/4	
NUMBER	US 20160035344 A1	
APPLICATION	14/817302	
ASSIGNEE	Google Inc.	

Summary: Methods, systems, and apparatus for identifying the language of a spoken utterance. The method includes receiving a plurality of audio frames and processing them using a long short term memory (LSTM) neural network to generate a respective language likelihood for each of a plurality of languages.

Contributions: Collaborated in the design of the training and inference stages of the system; collected approximately one year of labeled data (audio and parameters); collaborated in providing the necessary tools to generate and optimize topologies for the LSTM models.

US 20160035344A1

(19) **United States**

(12) **Patent Application Publication**

Gonzalez-Dominguez et al.

(10) **Pub. No.: US 2016/0035344 A1**

(43) **Pub. Date: Feb. 4, 2016**

(54) **IDENTIFYING THE LANGUAGE OF A SPOKEN UTTERANCE**

(71) Applicant: **Google Inc.**, Mountain View, CA (US)

(72) Inventors: **Javier Gonzalez-Dominguez**, Madrid (ES); **Hasim Sak**, New York, NY (US); **Ignacio Lopez Moreno**, New York, NY (US)

(21) Appl. No.: **14/817,302**

(22) Filed: **Aug. 4, 2015**

Related U.S. Application Data

(60) Provisional application No. 62/032,938, filed on Aug. 4, 2014.

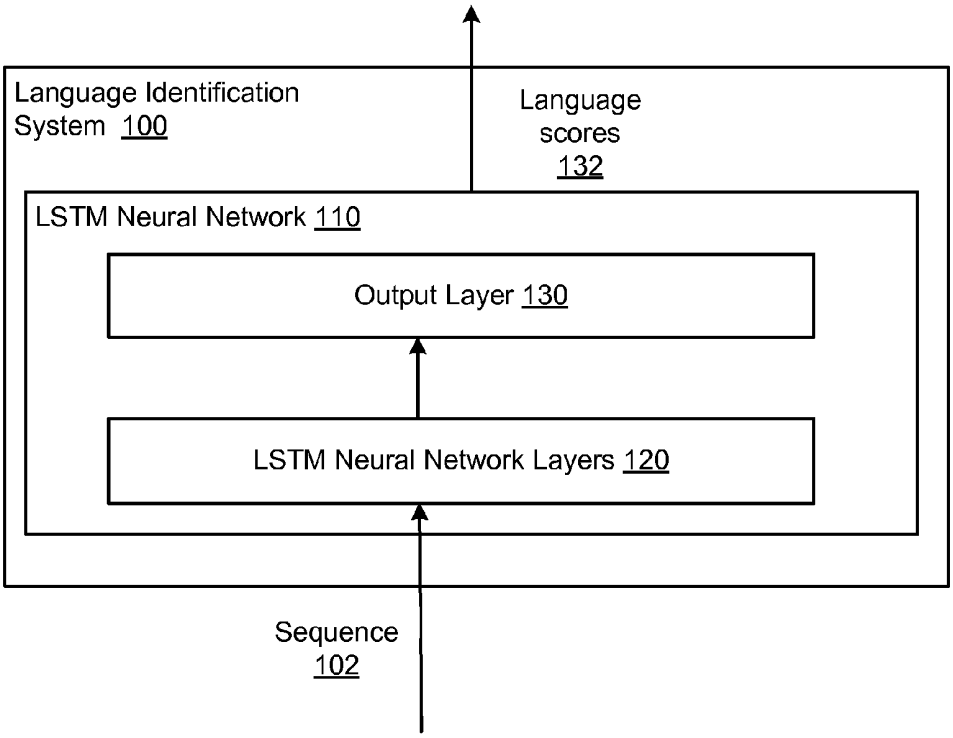
Publication Classification

(51) **Int. Cl.**
G10L 15/00 (2006.01)

(52) **U.S. Cl.**
CPC *G10L 15/005* (2013.01)

(57) **ABSTRACT**

Methods, systems, and apparatus, including computer programs encoded on computer storage media, for identifying the language of a spoken utterance. One of the methods includes receiving a plurality of audio frames that collectively represent at least a portion of a spoken utterance; processing the plurality of audio frames using a long short term memory (LSTM) neural network to generate a respective language score for each of a plurality of languages, wherein the respective language score for each of the plurality of languages represents a likelihood that the spoken utterance was spoken in the language; and classifying the spoken utterance as being spoken in one of the plurality of languages using the language scores.



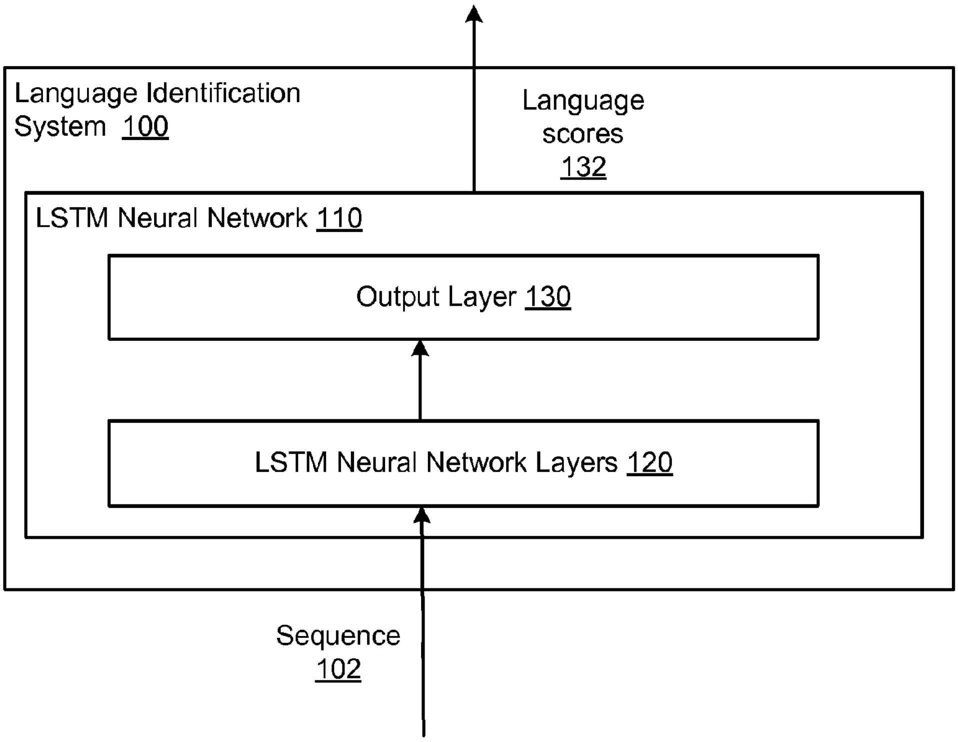

Language Identification System 100
Language scores 132
LSTM Neural Network 110
Output Layer 130
LSTM Neural Network Layers 120
Sequence 102

FIG. 1

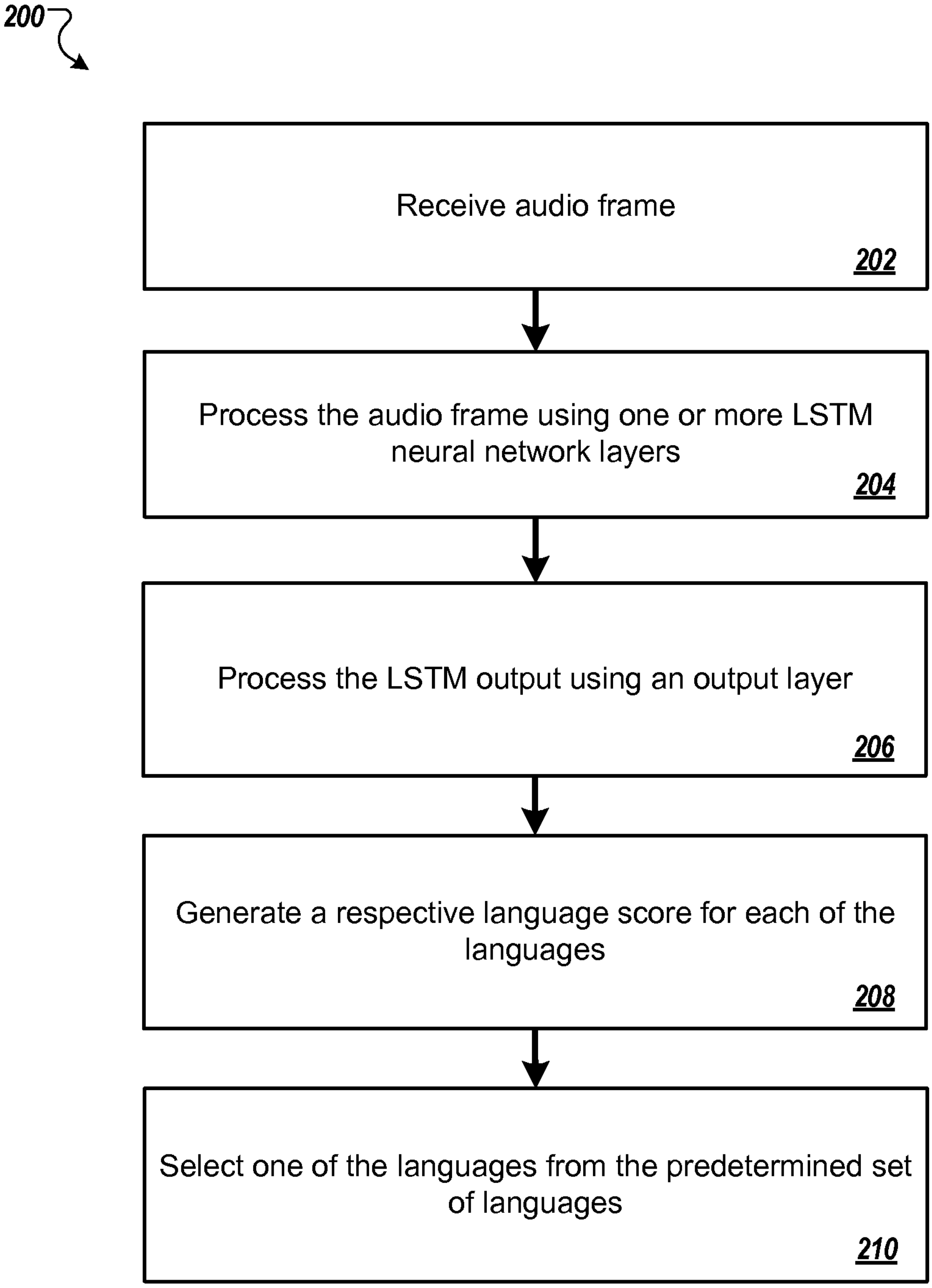

200
Receive audio frame
202
Process the audio frame using one or more LSTM neural network layers
204
Process the LSTM output using an output layer
206
Generate a respective language score for each of the languages
208
Select one of the languages from the predetermined set of languages
210

FIG. 2

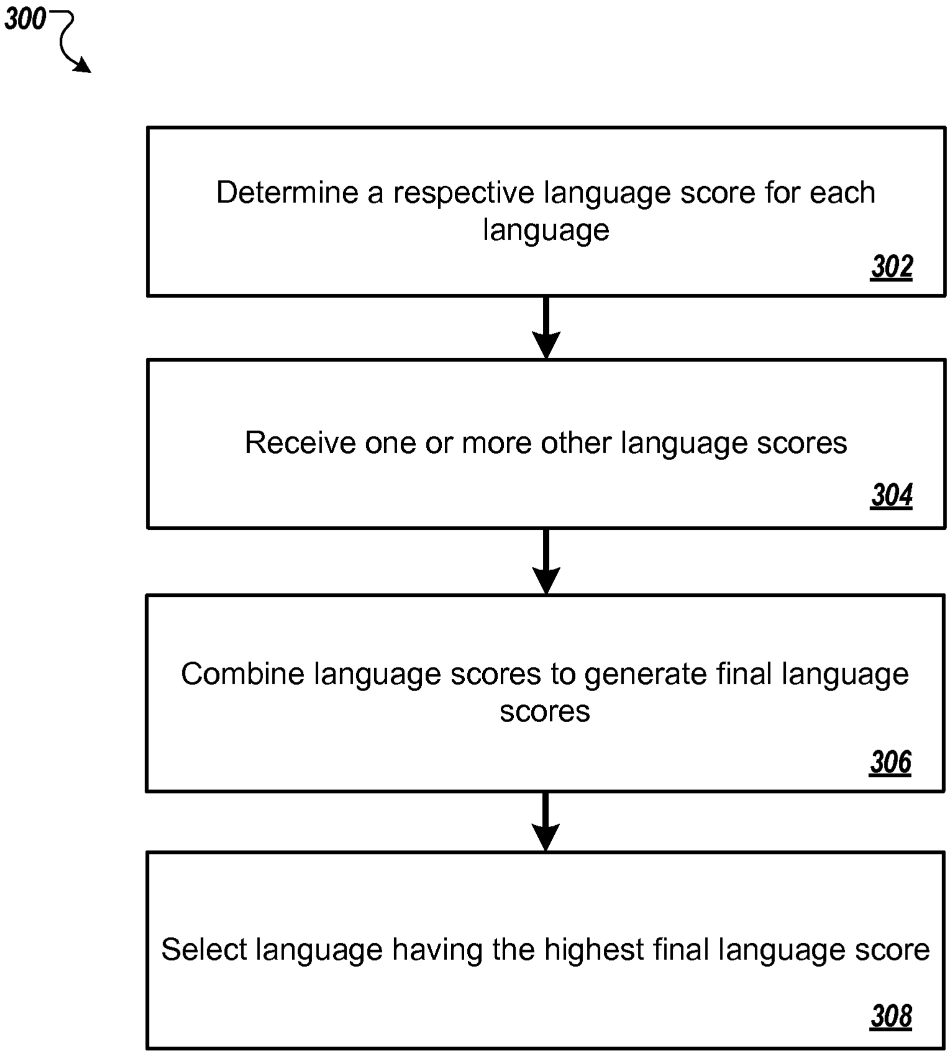

300
Determine a respective language score for each language
302
Receive one or more other language scores
304
Combine language scores to generate final language scores
306
Select language having the highest final language score
308

FIG. 3

IDENTIFYING THE LANGUAGE OF A SPOKEN UTTERANCE

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Application No. 62/032,938, filed Aug. 4, 2014, the contents of which are herein incorporated by reference.

BACKGROUND

[0002] This specification relates to identifying the language of a spoken utterance.

[0003] Speech-to-text systems can be used to generate a textual representation of a verbal utterance. Speech-to-text systems typically attempt to use various characteristics of human speech, such as the sounds produced, rhythm of speech, and intonation, to identify the words represented by such characteristics. Many speech-to-text systems are configured to only recognize speech in a single language or to require a user to manually designate which language the user is speaking.

SUMMARY

[0004] In general, one innovative aspect of the subject matter described in this specification can be embodied in methods that include the actions of receiving a plurality of audio frames that collectively represent at least a portion of a spoken utterance; processing the plurality of audio frames using a long short term memory (LSTM) neural network to generate a respective language score for each of a plurality of languages, wherein the respective language score for each of the plurality of languages represents a likelihood that the spoken utterance was spoken in the language; and classifying the spoken utterance as being spoken in one of the plurality of languages using the language scores.

[0005] Particular embodiments of the subject matter described in this specification can be implemented so as to realize one or more of the following advantages. The language in which an utterance was spoken can be accurately predicted by a language identification system. By using an LSTM neural network, the language identification system can be trained quicker and deployed more easily than other language identification systems. For example, by using an LSTM neural network, the language identification system can receive smaller inputs, e.g., audio frames without stacking, and have a smaller number of parameters while still generating accurate results. By using an LSTM neural network, the sequence of language scores for a given language generated by the language identification system while processing an utterance can be smoother, i.e., with less variation between scores. In some implementations, the language identification system can increase the accuracy of predictions by effectively combining the predictions generated by the language identification system with predictions generated by one or more other language identification systems.

[0006] The details of one or more embodiments of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIG. **1** shows an example language identification system.

[0008] FIG. **2** is a flow diagram of an example process for classifying an utterance as being spoken in a particular language.

[0009] FIG. **3** is a flow diagram of an example process for selecting a language using multiple language scores.

[0010] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0011] FIG. **1** shows an example language identification system **100**. The language identification system **100** is an example of a system implemented as computer programs on one or more computers in one or more locations, in which the systems, components, and techniques described below can be implemented.

[0012] The language identification system **100** receives a sequence of audio frames that collectively represent a spoken utterance and processes the audio frames in the sequence to classify the utterance as being spoken in one of a predetermined set of languages. In particular, the language identification system **100** processes each audio frame in the sequence using a long short term memory (LSTM) neural network **110** to generate a respective language score for each language in the set of languages. The language score for a given language represents a likelihood that the given language is the language in which the utterance represented by the sequence of audio frames was spoken. For example, the language identification system **100** can receive a sequence of audio frames **102** that represents an utterance and generate language scores **132** for the sequence **102**.

[0013] The LSTM neural network **110** includes one or more LSTM neural network layers **120** and an output layer **130**.

[0014] For each audio frame in an input sequence, the one or more LSTM neural network layers **120** are configured to process the audio frame to collectively generate an LSTM output for the audio frame. Each LSTM neural network layer includes one or more LSTM memory blocks. Each LSTM memory block can include one or more cells that each include an input gate, a forget gate, and an output gate that allow the cell to store previous activations generated by the cell, e.g., for use in generating a current activation or to be provided to other components of the LSTM neural network **110**. Example LSTM neural network layers are described in more detail in "Supervised Sequence Labelling with Recurrent Neural Networks," Alex Graves, Dissertation, Technische Universität München, München, 2008, available at http://www.cs.toronto.edu/~graves/phd.pdf.

[0015] The output layer **130** has been configured, e.g., through training, to, for each audio frame, receive the LSTM output generated by the one or more LSTM neural network layers **120** for the audio frame and to process the LSTM output to generate a set of frame scores that includes a respective frame score for each of the languages in the predetermined set of languages. The frame score for a given language represents the likelihood that the portion of the spoken utterance represented by the audio frame was spoken in the given language. In some implementations, the output layer **130** is a softmax output layer.

[0016] In some implementations, the one or more LSTM neural network layers **120** are bidirectional LSTM neural network layers, so that, when generating frame scores for an audio frame in position i in a sequence that includes L audio frames, the LSTM neural network layers **120** also process the audio frame at position L−i in the sequence. The output generated by processing both the audio frame i and the audio frame L−i is provided to the output layer **130** as the LSTM output for the audio frame in position i in the sequence. Example bidirectional LSTM neural network layers are described in more detail in “HYBRID SPEECH RECOGNITION WITH DEEP BIDIRECTIONAL LSTM,” Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed, available at http://www.cs.toronto.edu/~graves/asru_2013.pdf.

[0017] Once all of the language identification system **110** has processed all of the audio frames in the sequence using the LSTM neural network **110**, the language identification system **110** determines a language score for each of the languages in the set of languages from the frame scores for the language. In particular, for a given language in the set, the language identification system **110** combines the frame scores for the language across the audio frames in the sequence to generate the language score for the given language. Generating the language score for the language is described in more detail below with reference to FIG. **2**.

[0018] The language identification system **110** then uses the language scores to classify the utterance as being spoken in a particular language. In some implementations, the language identification system **110** selects the language having the highest language score as the language in which the utterance was spoken. In some other implementations, however, the language identification system **110** combines the language scores with one or more other language scores for each language generated by other language identification systems. That is, the language identification system **110** receives other language scores generated by other language identification systems, combines the other language scores with the language scores determined by the language identification system **100**, and then uses the combined language scores to classify the utterance as being spoken in a particular language. Combining language scores is described in more detail below with reference to FIG. **3**.

[0019] FIG. **2** is a flow diagram of an example process **200** for classifying an utterance as being spoken in a particular language. For convenience, the process **200** will be described as being performed by a system of one or more computers located in one or more locations. For example, a language identification system, e.g., the language identification system **100** of FIG. **1**, appropriately programmed, can perform the process **200**.

[0020] The system receives an audio frame from a sequence of audio frames that collectively represents an utterance (step **202**). Generally, each audio frame is generated from data calculated from a given time step in the spoken utterance. In particular, because the language identification is performed using an LSTM neural network that maintains an internal state, the audio frames can be generated without any stacking of audio frames, reducing the size of the inputs to the system. For example, the audio frames can each be 39-dimensional perceptual linear predictive (PLP) features calculated at respective time steps in the utterance.

[0021] The system processes the audio frame using one or more LSTM neural network layers to generate an LSTM output for the audio frame (step **204**). Each of the one or more LSTM neural network layers includes one or more LSTM memory blocks. Each LSTM memory block can include one or more cells that each include an input gate, a forget gate, and an output gate that allow the cell to store previous activations generated by the cell. The LSTM neural network layers collectively process the audio frame to generate the LSTM output in accordance with values of parameters of the LSTM neural network layers.

[0022] The system processes the LSTM output using an output layer to generate a respective frame score for each language in the predetermined set of languages (step **206**). The output layer is configured to process the LSTM output to generate the respective frame scores in accordance with values of parameters of the output layer.

[0023] After the system has processed each of the audio frames in the sequence, the system generates a respective language score for each of the languages in the set of languages (step **208**). In particular, the system generates the language score for a given language in the set of languages by combining the frame scores for the language. For example, the system can, for each language in the set, determine a logarithm of each of the frame scores for the language and then determine the language score for the language by averaging or otherwise combining the logarithms of the frame scores for the language. In some other implementations, the system combines the frame scores without first computing the logarithms.

[0024] The system selects one of the languages from the predetermined set of languages as the language in which the utterance was spoken using the language scores (step **210**). In some implementations, system selects the language having the highest language score as the language in which the utterance was spoken. In some other implementations, the system combines the language scores with one or more other language scores for each language as described in more detail below with reference to FIG. **3**.

[0025] In some implementations, rather than wait until after the system has processed each audio frame in the sequence, the system can generate the language scores as described above incrementally. For example, the system can generate the language scores after processing every i-th frame in the sequence, e.g., every frame, every other frame, or every tenth frame. In these implementations, the system can determine whether the language score for any of the language scores is high enough, i.e., exceeds a threshold score, and, if so, can select that language as the language in which the utterance was spoken. If none of the language scores is high enough, the system can continue processing the audio frames in the sequence. Thus, the system can classify the language of an utterance by processing frames that represent only a portion of the utterance.

[0026] The process **200** can be performed to predict a language for an utterance for which the desired output is not known, e.g., for a received sequence of audio frames that represent an utterance for which the spoken language has not yet been identified. The process **200** can also be performed on training sequences, i.e., sequences that represent utterance for which the spoken language has already been identified, as part of training the LSTM neural network to determine trained values of the parameters of the LSTM neural network, i.e., of parameters of the LSTM neural network layers and the output layer. In order to determine the trained values of the parameters of the LSTM neural network, the system can train the LSTM neural network on the training sequences using a conventional machine learning training technique, e.g., a truncated backpropagation through time training technique.

[0027] FIG. **3** is a flow diagram of an example process **300** for selecting a language using multiple language scores. For convenience, the process **300** will be described as being performed by a system of one or more computers located in one or more locations. For example, a language identification system, e.g., the language identification system **100** of FIG. **1**, appropriately programmed, can perform the process **300**.

[0028] The system determines a respective language score for each language in a predetermined set of languages, e.g., as described above with reference to FIG. **2** (step **302**).

[0029] The system receives, for each language, one or more other language scores (step **304**). Generally, each other language score for a given language is generated by a distinct language identification system. For example, the other language identification systems can include one or more systems that generate language scores using deep, feedforward neural networks. An example language identification system that uses a feedforward neural network is described in I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "*Automatic Language Identification using Deep Neural Networks*," Acoustics, Speech, and Signal Processing, IEEE International Conference 2014. As another example, the other language identification systems can include one or more i-vector language identification systems. An example i-vector language identification system is described in N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "*Language Recognition via i-vectors and Dimensionality Reduction*," in INTERSPEECH. ISCA, 2011, pp. 857-860.

[0030] The system combines, for each language, the language scores for the language to generate a final language score for the language (step **306**). For example, the system can combine the language scores in accordance with a set of combining parameters. In some implementations, the combining parameters include one or more parameters that are specific to each language and one or more parameters that are specific to each language identification system. For example, the final language score $\hat{s}_L(x_t)$ for a language L for an utterance x_t may satisfy:

$$\hat{s}_L(x_t) = \sum_{k=1}^{K} \alpha_k s_{kL}(x_t) + \beta_L,$$

where K is the total number of language identification systems, $s_{kL}(x_t)$ is the language score for the language L generated by the k-th language identification system, α_k is a trained value of a combining parameter specific to the k-th language identification system, and β_L is the trained value of a combining parameter for the language L. The system can determine the trained values of the combining parameters by training on training utterances using conventional training techniques.

[0031] The system selects the language having the highest final language score as the language in which the utterance was spoken (step **308**).

[0032] Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory program carrier for execution by, or to control the operation of, data processing apparatus. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them.

[0033] The term "data processing apparatus" encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can also include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[0034] A computer program (which may also be referred to or described as a program, software, a software application, a module, a software module, a script, or code) can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[0035] The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[0036] Computers suitable for the execution of a computer program include, by way of example, can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

[0037] Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0038] To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's client device in response to requests received from the web browser.

[0039] Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

[0040] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0041] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0042] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0043] Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous.

What is claimed is:

1. A method comprising:

receiving a plurality of audio frames that collectively represent at least a portion of a spoken utterance;

processing the plurality of audio frames using a long short term memory (LSTM) neural network to generate a respective language score for each of a plurality of languages, wherein the respective language score for each of the plurality of languages represents a likelihood that the spoken utterance was spoken in the language; and

classifying the spoken utterance as being spoken in one of the plurality of languages using the language scores.

2. The method of claim **1**, wherein the LSTM neural network comprises one or more LSTM neural network layers and an output layer, and wherein processing the plurality of audio frames using the LSTM neural network comprises, for each of the plurality of audio frames:

processing the audio frame through each of the one or more LSTM neural network layers to generate an LSTM output for the audio frame; and

processing the audio frame through the output layer to generate a respective frame score for each of the plurality of languages for the audio frame.

3. The method of claim **2**, wherein processing the plurality of audio frames further comprises:

determining the respective language score for each of the plurality of languages from the frame scores for the language for the plurality of audio frames.

4. The method of claim **3**, wherein determining the respective language score for each of the plurality of languages comprises:

determining a respective logarithm of each of the frame scores for the language; and

determining an average of the respective logarithms.

5. The method of claim **1**, wherein classifying the spoken utterance as having been spoken in one of the plurality of languages using the language scores comprises:

selecting a language having a highest language score as the language in which the spoken utterance was spoken.

6. The method of claim **1**, wherein classifying the spoken utterance as having been spoken in one of the plurality of languages using the language scores comprises:

obtaining, for each language, one or more other language scores, each other language score generated by another language identification system;

combining, for each language, the one or more other language scores for the language and the language score for the language to generate a final language score for the language; and

selecting a language having a highest final language score as the language in which the spoken utterance was spoken.

7. The method of claim **6**, wherein combining, for each language, the one or more other language scores for the language and the language score for the language comprises:

combining the other language scores and the language score in accordance with trained values of a set of combining parameters.

8. The method of claim **1**, wherein the LSTM neural network has been trained using a backpropagation through time training technique.

9. A system comprising one or more computers and one or more storage devices storing instructions that when executed by the one or more computers cause the one or more computers to perform operations comprising:

receiving a plurality of audio frames that collectively represent at least a portion of a spoken utterance;

processing the plurality of audio frames using a long short term memory (LSTM) neural network to generate a respective language score for each of a plurality of languages, wherein the respective language score for each of the plurality of languages represents a likelihood that the spoken utterance was spoken in the language; and

classifying the spoken utterance as being spoken in one of the plurality of languages using the language scores.

10. The system of claim **9**, wherein the LSTM neural network comprises one or more LSTM neural network layers and an output layer, and wherein processing the plurality of audio frames using the LSTM neural network comprises, for each of the plurality of audio frames:

processing the audio frame through each of the one or more LSTM neural network layers to generate an LSTM output for the audio frame; and

processing the audio frame through the output layer to generate a respective frame score for each of the plurality of languages for the audio frame.

11. The system of claim **10**, wherein processing the plurality of audio frames further comprises:

determining the respective language score for each of the plurality of languages from the frame scores for the language for the plurality of audio frames.

12. The system of claim **11**, wherein determining the respective language score for each of the plurality of languages comprises:

determining a respective logarithm of each of the frame scores for the language; and

determining an average of the respective logarithms.

13. The system of claim **9**, wherein classifying the spoken utterance as having been spoken in one of the plurality of languages using the language scores comprises:

selecting a language having a highest language score as the language in which the spoken utterance was spoken.

14. The system of claim **9**, wherein classifying the spoken utterance as having been spoken in one of the plurality of languages using the language scores comprises:

obtaining, for each language, one or more other language scores, each other language score generated by another language identification system;

combining, for each language, the one or more other language scores for the language and the language score for the language to generate a final language score for the language; and

selecting a language having a highest final language score as the language in which the spoken utterance was spoken.

15. The system of claim **14**, wherein combining, for each language, the one or more other language scores for the language and the language score for the language comprises:

combining the other language scores and the language score in accordance with trained values of a set of combining parameters.

16. The system of claim **9**, wherein the LSTM neural network has been trained using a backpropagation through time training technique.

17. A computer program product encoded on one or more non-transitory computer storage media, the computer program product comprising instructions that when executed by one or more computers cause the one or more computers to perform operations comprising:

receiving a plurality of audio frames that collectively represent at least a portion of a spoken utterance;

processing the plurality of audio frames using a long short term memory (LSTM) neural network to generate a respective language score for each of a plurality of languages, wherein the respective language score for each of the plurality of languages represents a likelihood that the spoken utterance was spoken in the language; and

classifying the spoken utterance as being spoken in one of the plurality of languages using the language scores.

18. The computer program product of claim **17**, wherein the LSTM neural network comprises one or more LSTM neural network layers and an output layer, and wherein processing the plurality of audio frames using the LSTM neural network comprises, for each of the plurality of audio frames:

processing the audio frame through each of the one or more LSTM neural network layers to generate an LSTM output for the audio frame; and

processing the audio frame through the output layer to generate a respective frame score for each of the plurality of languages for the audio frame.

19. The computer program product of claim **18**, wherein processing the plurality of audio frames further comprises:

determining the respective language score for each of the plurality of languages from the frame scores for the language for the plurality of audio frames.

20. The computer program product of claim **17**, wherein classifying the spoken utterance as having been spoken in one of the plurality of languages using the language scores comprises:

selecting a language having a highest language score as the language in which the spoken utterance was spoken.

* * * * *

C.2 Language Identification

TITLE	**Language Identification.**	
INVENTORS	Javier González Domínguez	UAM
	Ignacio López Moreno	UAM / Google Inc.
	David Eustis	Google Inc.
DATE	2015/12/17	
NUMBER	US 20150364129 A1	
APPLICATION	14/313490	
ASSIGNEE	Google Inc.	

Summary: Methods, systems, and apparatus for language identification. In some implementations, speech data for an utterance is received and provided to (i) a deep neural network (DNN) -based language identification module, and (ii) multiple speech recognizers that are each configured to recognize speech in a different language.
Contributions: Designed the training and inference stages of the system; collected about one year of data (audio and parameters), which was used to generate neural networks models for language recognition, and collaborated in the validation of the patent proposal with experimental results.

US 20150364129A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2015/0364129 A1**

Gonzalez-Dominguez et al. (43) **Pub. Date: Dec. 17, 2015**

(54) **LANGUAGE IDENTIFICATION**

(71) Applicant: **Google Inc.**, Mountain View, CA (US)

(72) Inventors: **Javier Gonzalez-Dominguez**, Madrid (ES); **Ignacio L. Moreno**, New York, NY (US); **David P. Eustis**, New York, NY (US)

(21) Appl. No.: **14/313,490**

(22) Filed: **Jun. 24, 2014**

Related U.S. Application Data

(60) Provisional application No. 62/013,383, filed on Jun. 17, 2014.

Publication Classification

(51) **Int. Cl.**

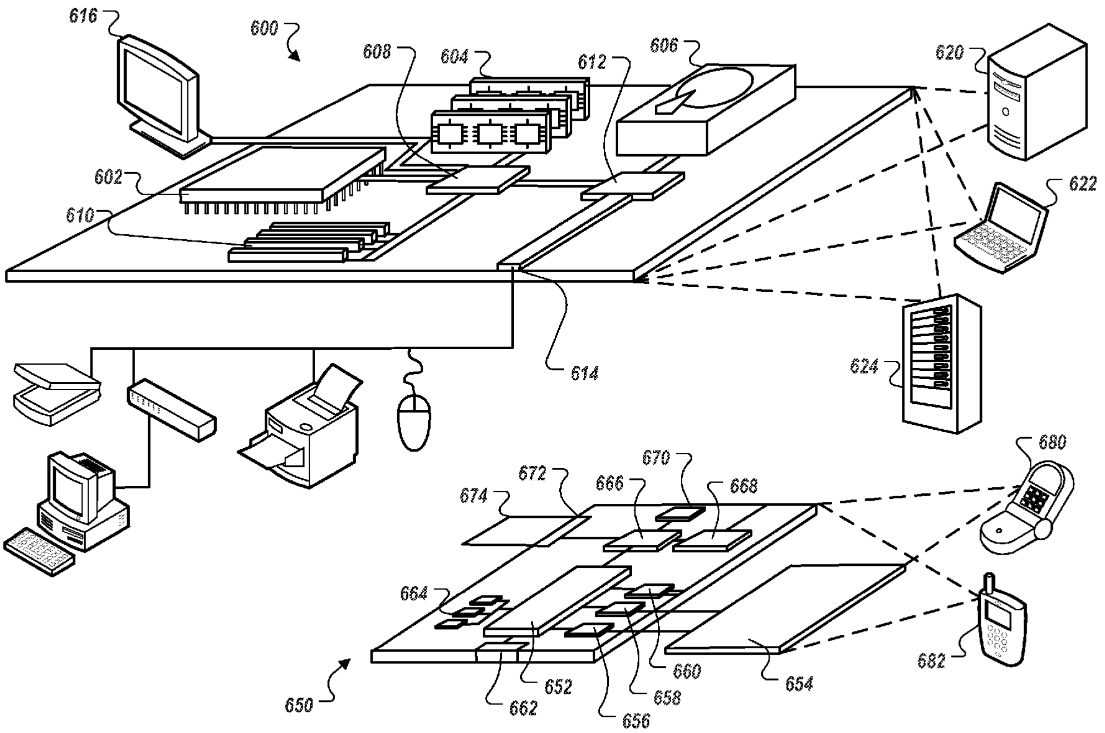
G10L 15/00	(2006.01)
G10L 15/02	(2006.01)

(52) **U.S. Cl.**

CPC ***G10L 15/005*** (2013.01); ***G10L 15/02*** (2013.01)

(57) **ABSTRACT**

Methods, systems, and apparatus, including computer programs encoded on a computer storage medium, for language identification. In some implementations, speech data for an utterance is received and provided to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language. From the language identification module, language identification scores corresponding to different languages are received, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language. A language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model is received. A language is selected based on the language identification scores and the language model confidence scores.



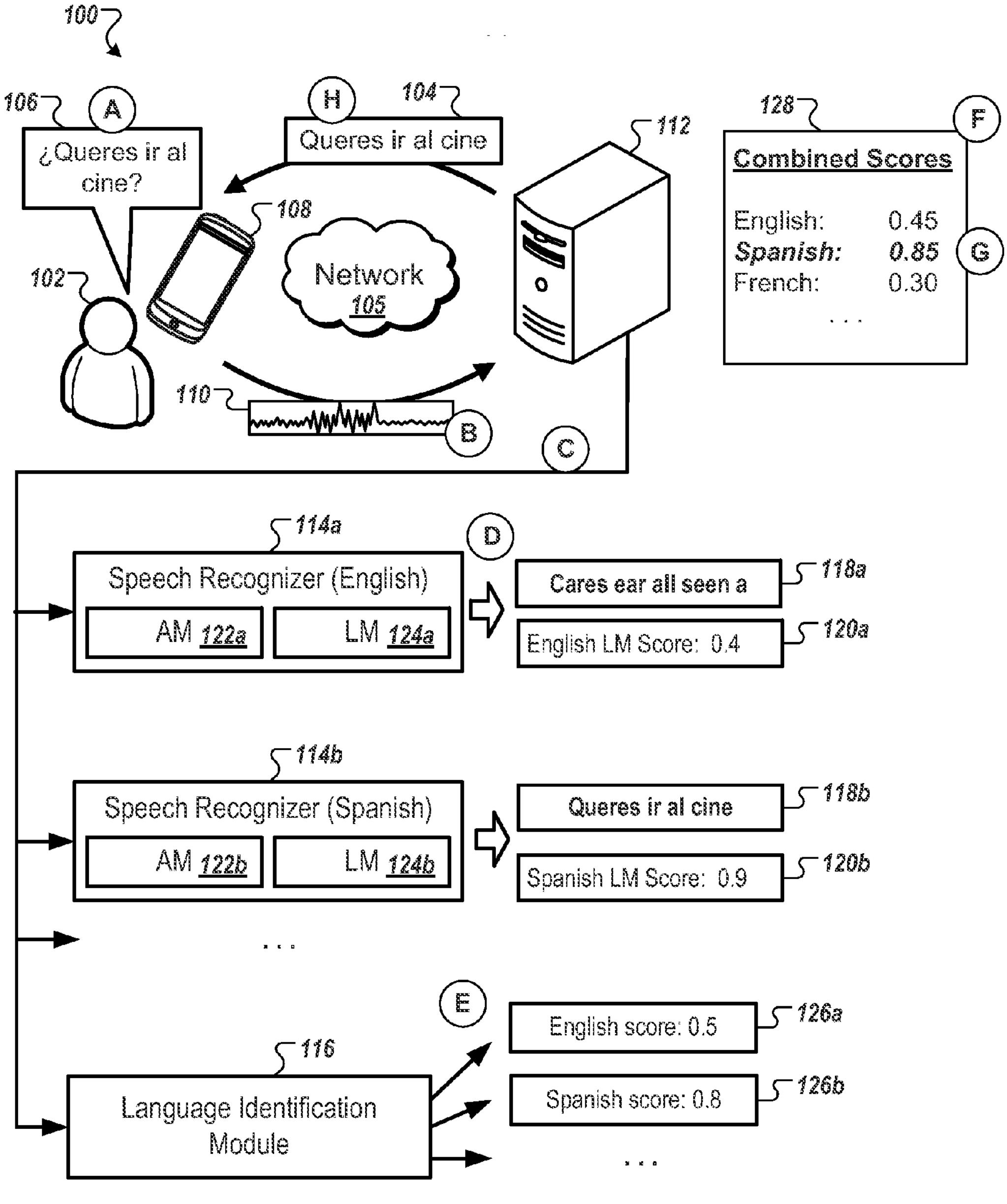

100
106
A
¿Queres ir al cine?
H
104
Queres ir al cine
112
128
F
Combined Scores
English: 0.45
Spanish: 0.85
French: 0.30
. . .
G
108
Network
105
102
110
B
C
114a
D
Speech Recognizer (English)
AM 122a
LM 124a
Cares ear all seen a
118a
English LM Score: 0.4
120a
114b
Speech Recognizer (Spanish)
AM 122b
LM 124b
Queres ir al cine
118b
Spanish LM Score: 0.9
120b
. . .
E
English score: 0.5
126a
116
Spanish score: 0.8
126b
Language Identification Module
. . .

FIG. 1

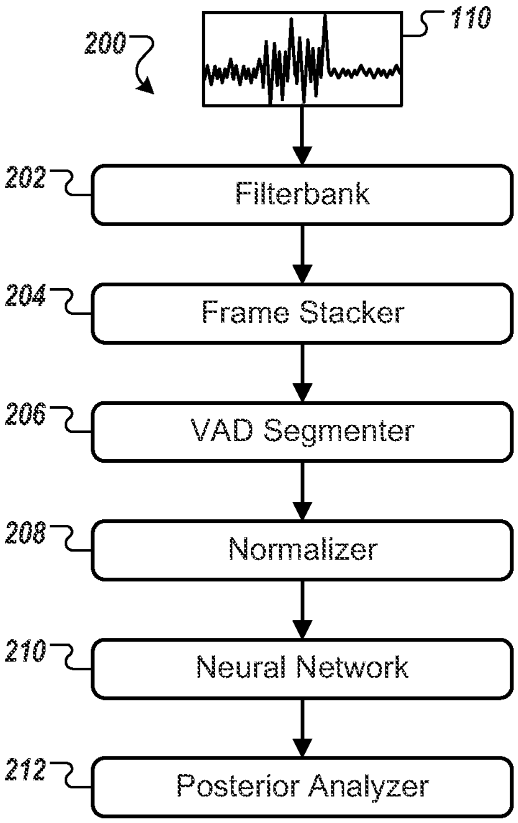

200
110
202
Filterbank
204
Frame Stacker
206
VAD Segmenter
208
Normalizer
210
Neural Network
212
Posterior Analyzer

FIG. 2

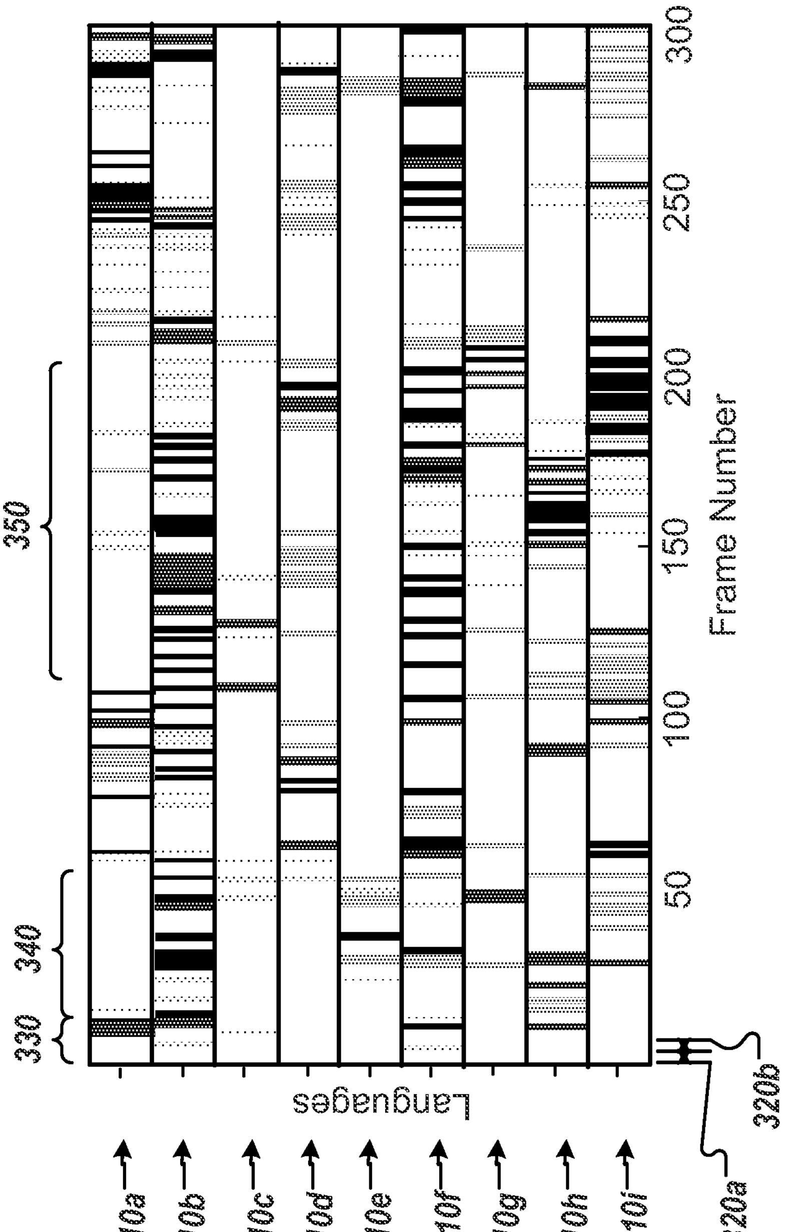

310a
310b
310c
310d
310e
310f
310g
310h
310i
Languages
320a
320b
330
340
350
0
50
100
150
200
250
300
Frame Number

FIG. 3

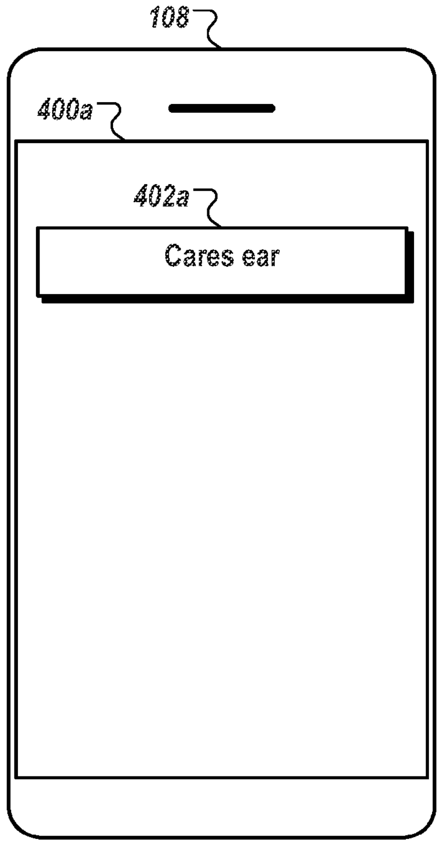


FIG. 4A

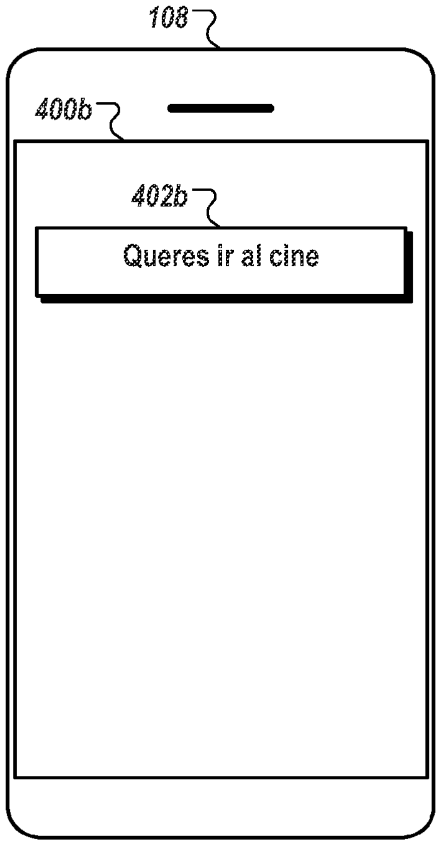


FIG. 4B

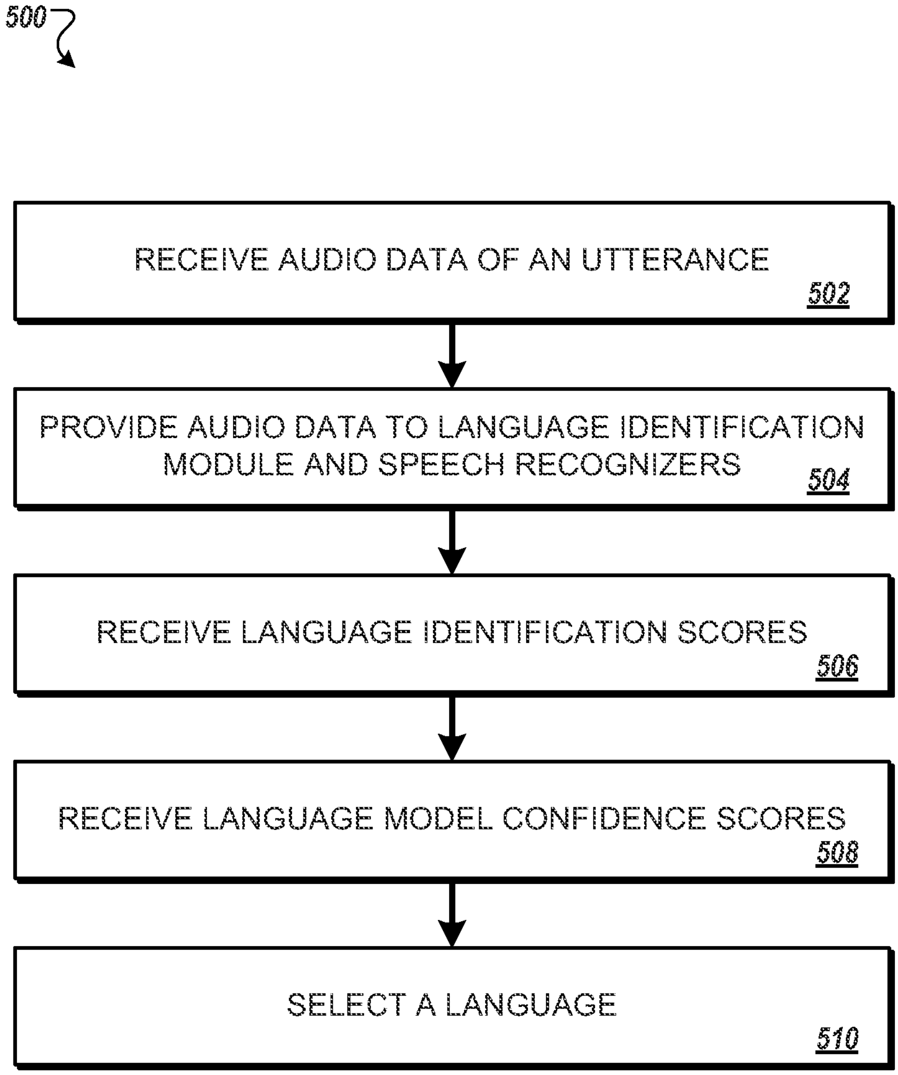

500
RECEIVE AUDIO DATA OF AN UTTERANCE
502
PROVIDE AUDIO DATA TO LANGUAGE IDENTIFICATION MODULE AND SPEECH RECOGNIZERS
504
RECEIVE LANGUAGE IDENTIFICATION SCORES
506
RECEIVE LANGUAGE MODEL CONFIDENCE SCORES
508
SELECT A LANGUAGE
510

FIG. 5

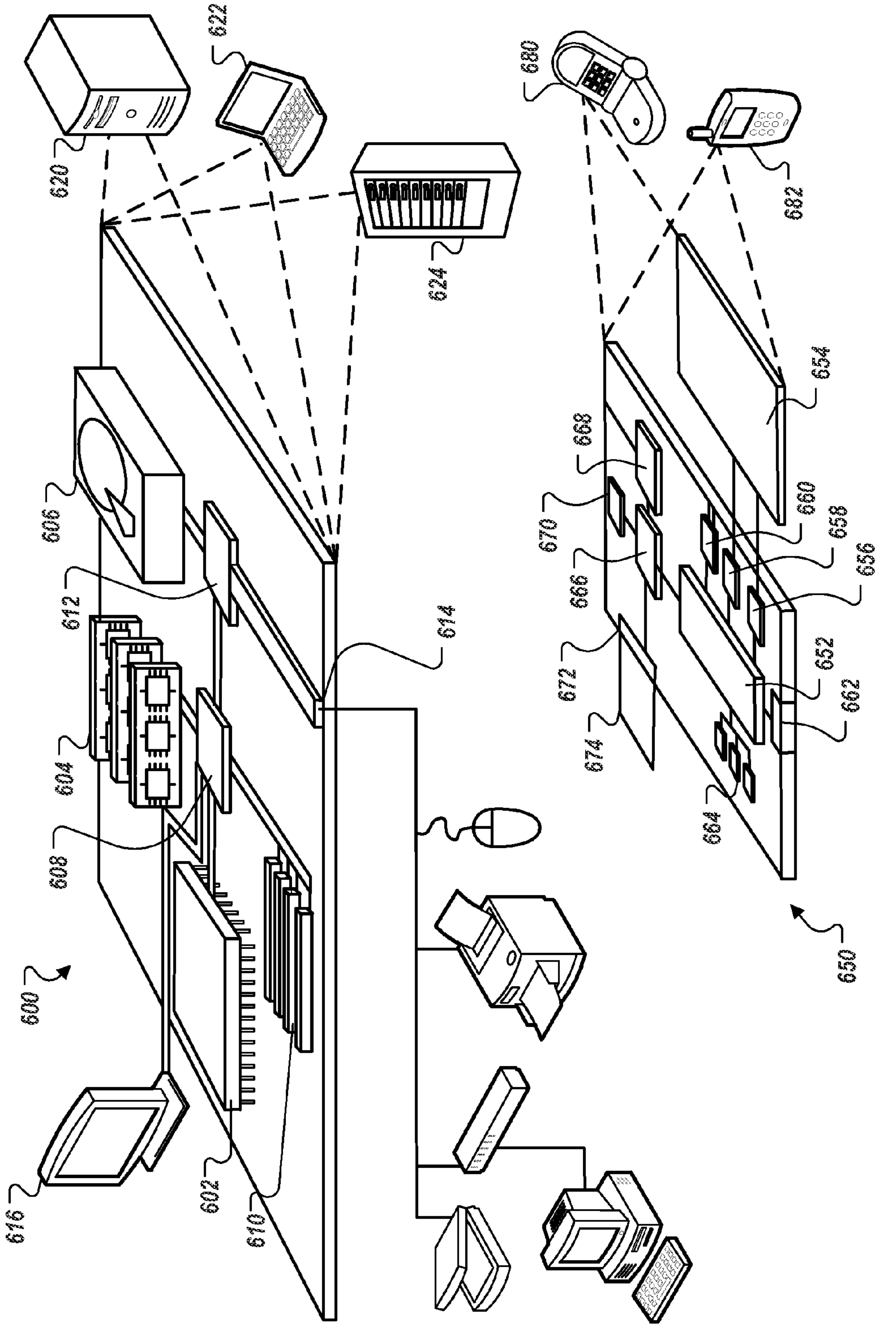

600
602
604
606
608
610
612
614
616
620
622
624
650
652
654
656
658
660
662
664
666
668
670
672
674
680
682

FIG. 6

LANGUAGE IDENTIFICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. Provisional Application Ser. No. 62/013,383, filed Jun. 17, 2014, the entire contents of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] The present document relates to automatic language identification.

BACKGROUND

[0003] Speech-to-text systems can be used to generate a textual representation of a verbal utterance. Speech-to-text systems typically attempt to use various characteristics of human speech, such as the sounds produced, rhythm of speech, and intonation, to identify the words represented by such characteristics. Many speech-to-text systems are configured to recognize speech in a single language, or require a user to manually designate which language the user is speaking.

SUMMARY

[0004] In some implementations, a computing system can automatically determine which language a user is speaking and transcribe speech in the appropriate language. For example, when a bilingual user alternates between speaking two different languages, the system may detect the change in language and transcribe speech in each language correctly. For example, if speech provided in a dictation session includes speech in different languages, the system may automatically detect which portions of the speech are in a first language, and which portions are in a second language. This may allow the system to transcribe the speech correctly, without requiring the user to manually indicate which language the user is speaking while dictating.

[0005] The system may identify the language that the user is speaking using a language identification module as well as speech recognizers for different languages. For example, each speech recognizer may attempt to recognize input speech in a single language. Each speech recognizer may provide a confidence score, such as a language model confidence score, indicating how likely its transcription is to be correct. The system may then use output of the language identification module and the speech recognizers to determine which language was most likely spoken. With the language identified, the system may provide the user a transcript of the user's speech in the identified language.

[0006] In one aspect, a method performed by one or more computers includes receiving speech data for an utterance. The method further includes providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language. The method further includes receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language. The method further includes receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model. The method further includes selecting a language based on the language identification scores and the language model confidence scores.

[0007] Other implementations of this and other aspects include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods, encoded on computer storage devices. A system of one or more computers can be so configured by virtue of software, firmware, hardware, or a combination of them installed on the system that in operation cause the system to perform the actions. One or more computer programs can be so configured by virtue of having instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

[0008] Implementations can include any, all, or none of the following features. For example, receiving the speech data for the utterance includes receiving the speech data from a user over a network; wherein the method further includes receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer; and providing the transcription in the selected language to the user over the network. The method including before receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer: receiving, from a particular one of the multiple speech recognizers, a preliminary transcription of the utterance in a language corresponding to the speech recognizer; providing the preliminary transcription to the user over the network before providing the transcription in the selected language to the user over the network. In some instances, the preliminary transcription is in the selected language. In some other instances, the preliminary transcription is a language different than the selected language. The preliminary transcription is provided over the network for display to the user; and wherein the transcription in the selected language is provided for display in place of the preliminary transcription, after the preliminary transcription has been provided over the network.

[0009] Implementations can include any, all, or none of the following features. For example, the method further includes receiving, from the particular one of the multiple speech recognizers, a preliminary language model confidence score that indicates a preliminary level of confidence that a language model has in the preliminary transcription of the utterance in a language corresponding to the language model; and determining that the preliminary language model confidence score is less than a language model confidence score received from the particular one of the multiple speech recognizers. Providing the speech data to a language identification module includes providing the speech data to a neural network that has been trained to provide likelihood scores for multiple languages. Selecting the language based on the language identification scores and the language model confidence scores includes determining a combined score for each of multiple languages, wherein the combined score for each language is based on at least the language identification score for the language and the language model confidence score for the language; and selecting the language based on the combined scores. Determining a combined score for each of multiple languages includes weighting the likelihood scores or the language model confidence scores using one or more weighting values. Receiving the speech data includes receiving speech data that includes an utterance of a user; further including before receiving the speech data, receiving data indicating multiple languages that the user speaks; storing data indicating the multiple languages that the user speaks; wherein providing the speech data to multiple speech recognizers that are each configured to recognize speech in a different language includes based on the stored data indicating the multiple languages that the user speaks, providing the speech data to a set of speech recognizers configured to recognize speech in a different one of the languages that the user speaks.

[0010] In one aspect, a non-transitory computer storage medium is tangibly encoded with computer program instructions that, when executed by one or more processors, cause a computer operations to perform operations including receiving speech data for an utterance. The operations further include providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language. The operations further include receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language. The operations further include receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model. The operations further include selecting a language based on the language identification scores and the language model confidence scores.

[0011] In one aspect, a system includes one or more processors and a non-transitory computer storage medium is tangibly encoded with computer program instructions that, when executed by one or more processors, cause a computer operations to perform operations. The operations include receiving speech data for an utterance. The operations further include providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language. The operations further include receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language. The operations further include receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model. The operations further include selecting a language based on the language identification scores and the language model confidence scores.

[0012] The systems and processes described here may be used to provide a number of potential advantages. A user able to speak in multiple languages may use a single system to transcribe utterances, without specifying which language the user wishes to speak. A speech recognition system may store user language preferences or history to aid in determining the language in which the user is speaking. Preliminary transcriptions may be provided quickly to a user while more accurate transcriptions are being generated. Once generated, a more accurate transcription can replace a preliminary transcription. The results of a language identification module and multiple speech recognizers may be combined to produce a result that is more accurate than results of an individual module alone.

[0013] Other features, aspects and potential advantages will be apparent from the accompanying description and figures.

DESCRIPTION OF DRAWINGS

[0014] FIG. **1** is a block diagram illustrating an example of a system for language identification and speech recognition.

[0015] FIG. **2** is a block diagram illustrating an example of a processing pipeline for a language identification module.

[0016] FIG. **3** is a diagram illustrating an example data related to speech recognition confidence scores.

[0017] FIGS. **4**A and **4**B are diagrams illustrating examples of user interfaces.

[0018] FIG. **5** is a flowchart illustrating an example of a process for language identification.

[0019] FIG. **6** is a schematic diagram that shows examples of a computing device and a mobile computing device.

[0020] Like reference symbols in the various drawings indicate like elements

DETAILED DESCRIPTION

[0021] A speech recognition system can be configured to receive an utterance and, as part of creating a transcription of the utterance, determine the language in which the user spoke. This can be very useful for multi-lingual users who may speak in different languages at different times, and may switch between languages in the middle of a dictation session.

[0022] In some implementations, a speech recognition system can use both a language identification module and a pool of language-specific speech recognizers to determine the language of an utterance. The language identification module may be configured to product a confidence score for each of a plurality of languages. The confidence scores for the language identification module may indicate likelihoods that the utterance was spoken in the respective languages. In addition, each of the language-specific speech recognizers can create a transcription in their specific language and can generate a confidence score for the transcription. The speech recognition system can use both the confidence scores from the language identification module and the speech recognizers to determine the most likely language uttered. The user may then be provided a text-based transcription in the determined language.

[0023] As such, the system may be used to dynamically determine the language that is spoken without receiving input that specifies in advance what language of speech will be provided. That is, the user is not required to tap a button to select a language, or speak the name of the language, or take any other action in advance to designate the language that will be spoken. Instead, the user may simply begin speaking the content that the user desires to enter, and the system determines the language automatically as the user speaks. The system may determine what language is spoken based on the sounds of the user's speech, as well as an analysis of the which words those sounds are likely to represent.

[0024] From the user's perspective, the system may be configured so that the user may speak in any of multiple languages, possibly changing languages mid-speech, and an appropriate transcription of the speech may be produced with no additional user inputs needed. The user may use the same interface regardless of which language is spoken, and the language may be detected without the user speaking a language-specific key-word before speaking their input or making any other user selection of a specific language in which dictation should occur.

[0025] Using the confidence scores for language models and language identification systems together can provide improved accuracy. Language identification scores may provide an estimate based primarily on acoustic properties, and accordingly indicate which language input audio sounds like. Language model scores are typically biased toward the coherence of a sentence or utterance as a whole. For example, language model scores may indicate how likely it is that a series of words is a valid sentence in a given language. Language model scores may also be based on a longer sequence of input than some language identification scores.

[0026] Scores based on acoustic signal characteristics are typically most accurate when a user speaks his or her native language. However, for a multi-lingual user or a user with an accent, speech may include acoustic markers or characteristics of multiple languages. Often, a multi-lingual user will have a non-native accent for at least one of the languages spoken. Language model confidence scores can be used to balance out the bias toward acoustic characteristics that frequently occurs in language identification scores. Using both types of confidence scores can provide robustness and accuracy that is better than can be achieved with either type of confidence score alone.

[0027] FIG. **1** is a block diagram illustrating an example of a system **100** for language identification and speech recognition. The system **100** includes a client device **108** and a computer system **112** that communicates with the client device **108** over a network **105**. The system **100** also includes speech recognizers **114** and a language identification module **116**. The figure illustrates a series of states (A) to (H) which illustrates a flow of data, and which may occur in the order shown or in a different order.

[0028] The client device **108** can be, for example, a desktop computer, a laptop computer, a cellular phone, a smart phone, a tablet computer, a music player, an e-book reader, a wearable computer, or a navigation system. The functions performed by the computing system **112** can be performed by individual computer systems or can be distributed across multiple computer systems. The network **105** can be wired or wireless or a combination of both, and may include private networks and/or public networks, such as the Internet. The speech recognizers **114** may be implemented on separate computing systems or processing modules, and may be accessed by the computing system **112** via remote procedure calls. In some implementations, functionality of the computing system **112**, the speech recognizers **114**, and/or the language identification module **116** may be implemented together using one or more computing systems.

[0029] In the example of FIG. **1**, the user **102** speaks an utterance **106** into the client device **108**, and data **110** representing the utterance **106** is transmitted to the computing system **112**. The computing system **112** identifies the language of the utterance **106** and provides a transcription **104** for the utterance **106**.

[0030] The user **102** in this example uses one or more services of the computing system **112**. For example, the user **102** may use the speech recognition service for dictation (e.g., speech-to-text transcription). As additional examples, the user **102** may use the speech recognition service that the computing system **112** provides as part of, for example, user authentication and authorization, data hosting, voice search, or cloud applications, such as web-based email, document authoring, web searching, or news reading.

[0031] The user **102** may be able to speak in more than one language, and may wish at times to submit spoken input to the client device **108** in different languages. For example, the user may be able to speak English and Spanish, and may dictate emails in either of these languages, depending on the intended recipient of the email. In some implementations, an account associated with the user **102** in the computing system **112** may store data indicating that the user's preferred languages are English and Spanish, or that the user **102** has a history of communicating with the computing system **112** in English and Spanish. This data may have been compiled, for example, based on setting selected by the user **102** and/or records indicative of historical communications with the computing system **112**.

[0032] In further detail, during stage (A), the user **102** speaks an utterance **106**. The client device **108** receives the utterance **106**, for example through a microphone built into or connected to the client device **108**. The client device **108** can create speech data **110** that represents the utterance **106** using any appropriate encoding technique, either commonly in use or custom-created for this application. The speech data **110** may be, for example, a waveform, a set of speech features, or other data derived from the utterance **106**.

[0033] During stage (B), the client device **108** sends the speech data **110** to the computing device **112** over the network **105**.

[0034] During stage (C), to begin the transcription process, the computing system **112** provides the speech data **110**, or data derived from the speech data **110**, to multiple speech recognizers **114** (e.g., **114***a*, **114***b*, and so on) and to a language identification module **116**. The computing system **112** requests transcriptions from the speech recognizers **114** and language identification outputs from the language identification module **116**. In some implementations, each of the speech recognizers **114** is configured to recognize speech in a single language. In such implementations, each speech recognizer **114** may be a language-specific speech recognizer, with each of the various speech recognizers **114** recognizing a different language.

[0035] In some implementations, the computing system **112** makes requests and provides the speech data **110** by making remote procedure calls to the speech recognizers **114** and to the language identification module **116**. These requests may be asynchronous and non-blocking. That is, the speech recognizers **114** and the language identification module **116** may each operate independently, and may operate in parallel. The speech recognizers **114** and the language identification module **116** may process requests from the computing system **112** that are at different times, and may complete their processing at different times. The initiation of a request or data transfer to one of the speech recognizers **114** or the language identification module **116** may not be contingent upon, and need not be stopped by, the initiation or completion of processing by any of the other speech recognizers **114** and to a language identification module **116**.

[0036] When the computing system **112** sends requests to the speech recognizers **114** and to the language identification module **116**, the computer system **112** may initiate a timeout clock that increments, for example, every millisecond. The timeout clock can measure a predetermined amount of time in which that the speech recognizers **114** and the language identification module **116** are given to provide responses to the requests from the computing system **112**.

[0037] In some implementations, information identifying multiple languages that the user **102** speaks may be known. For example, the user **102** may have previously indicated a set of multiple languages that the user **102** speaks. As another example, an email or text messaging account or a web browsing history may indicate languages that the user **102** speaks. The computing system **112** may use this information to limit the number of languages that are evaluated to those that the user **102** is likely to speak. For example, rather than request transcriptions and language identification scores for all languages, the computing system **112** may request transcriptions and scores for only the languages that are associated with or are determined likely to be spoken by the user **102**.

[0038] During stage (D), each speech recognizer **114** generates a proposed transcription **118** and a confidence score **120** for a particular language. In some implementations, each speech recognizer **114** may use similar processing systems that have access to different acoustic models and/or language models. For example, a speech recognizer **114*a*** generates an English transcription **118*a*** for the speech data **110**, using an English acoustic model **122*a*** and an English language model **124*a***. A speech recognizer **114*b*** generates a Spanish transcription **118*b*** for the speech data **110**, using a Spanish acoustic model **122*b*** and a Spanish language model **124*b***.

[0039] In general, acoustic models include data representing the sounds associated with a particular language and phonetic units that the sounds represent. Language models generally include data representing the words, syntax, and common usage patterns of a particular language. The speech recognizers **114*a***, **114*b*** each produce confidence scores, for example, values that indicate how confident a recognizer or model is in the transcription that was produced. In particular, the language models **124*a***, **124*b*** generate confidence scores **120*a***, **120*b*** that indicate how likely it is that the sequence of words in the associated transcription **118*a***, **118*b*** would occur in typical usage. The language model confidence score **120*a*** indicates a likelihood that the transcription **118*a*** is a valid English language sequence. Similarly, the language model confidence score **120*b*** indicates a likelihood that the transcription **118*b*** is a valid Spanish language sequence. In the example, the Spanish language model confidence score **120*b*** is larger than the English language model confidence score **120*a***, suggesting that the Spanish transcription **118*b*** is more likely to be correct than the English transcription **118*a***. This, in turn, indicates that it is more likely that the utterance **106** is a Spanish utterance than an English utterance, as discussed further below. The speech recognizers **114*a***, **114*b*** send the transcriptions **118*a***, **118*b*** as well as the language model confidence scores **120*a***, **120*b*** to the computing system **112**.

[0040] During stage (E), in response to the request and speech data **110** from the computing system **112**, the language identification module **116** generates a confidence score for each of a plurality of languages. The language identification module **116** may include one or more models configured to estimate, based on acoustic properties of an audio sample, the likelihood that audio represents speech of a particular language. As discussed further with respect to FIG. **2**, the language identification module **116** may include an artificial neural network or other model configured to receive speech features extracted from audio. The artificial neural network or other model may output, for each of several different languages, a confidence score indicating how well the speech features match the properties of a particular language.

[0041] In the example, the language identification module **116** provides a confidence score **126*a*** indicating the likelihood that the speech data **110** represents an English utterance. The language identification module **116** also provides a confidence score **126*b*** indicating the likelihood that the speech data **110** represents a Spanish utterance. The confidence score **126*b*** is higher than the confidence score **126*a***, indicating that the language identification module **116** estimates that there is a higher likelihood that the speech data **110** represents a Spanish utterance than an English utterance. Stage (E) may be designed to be initiated and/or completed at the same time as stage (D) and/or run concurrently with stage (D).

[0042] During stage (F), the computing system **112** combines the language model confidence scores **120** and the confidence scores **126** to generate combined scores **128**. Any appropriate combination techniques may be used to calculate the combined scores **128**. In this example, each combined score **128** represents an arithmetic average of a particular language's language model confidence score **120** and language identification module score **126**.

[0043] In some configurations, a weighting may be applied to each component confidence score **126** and **120**. This may be desirable, for example, if empirical testing (e.g., for a single user **102**, for a class of users, for all users) shows that a particular weighting gives more favorable results. For example, language model scores **120** may be slightly more or slightly less predictive of the language spoken than output of the language identification model, and may accordingly be given a slightly higher or lower weight.

[0044] In some implementations, additional data may be considered when calculating the combined score **128**. For example, if the user **102** is dictating an email addressed to a recipient with a .mx or .es top level domain, which are associated with Mexico and Spain, the combined score **128** for Spanish may be increased based on a likelihood that the recipient speaks that language.

[0045] During stage (G), the computing system **112** selects a language based on the combined scores **128**. For example, the computing system **112** may identify the combined score **128** that indicates the highest likelihood, and select the language corresponding to this score. In the example, the combined score for the Spanish language indicates a higher likelihood than the combined scores for other languages, so the computing system **112** determines that the utterance **106** is most likely a Spanish utterance.

[0046] During stage (H), the computing system **112** transmits transcription **118*b*** for the selected language to the client device **108** as the transcription **104** for the utterance **106**. Since the computing system **112** determined that the user **102** was most likely speaking Spanish rather than another language, the Spanish transcription **118*b*** is provided.

[0047] Once received, the client device **108** may use or process the transcription **104** as needed. For example, the client device **108** may use the transcription **104** as text input to the application and input field that currently has focus in the client device **108**. The computing system **112** may be configured to discard the other proposed transcriptions **118**, store them or later use, transmit one or more to the client device **108** in addition to the transcription **104**, or take any other appropriate action.

[0048] In some configurations, the user **102** may provide, in advance, an indication of multiple language that the user **102**

speaks. Using this information, scores and transcriptions may be generated for only the languages that the user **102** has indicated that he is likely to speak. For example, the language identification module **116** may generate confidence scores **126** only for languages associated with a speech recognizer **114**. In some configurations, these languages may be selected based on, for example, data associated with the user **102** in user profile data stored by the computing system **112**.

[0049] In some configurations, the computing system **112** may be configured to provide continuous or ongoing transcriptions **104** to the client device **108**. Results may be presented in or near real-time as the user dictates. In some cases, the speech recognizers **114** may process the data **110** at different speeds and/or may provide the computing system **112** with preliminary results before providing final results. For example, the English speech recognizer **114***a* may process the speech data **110** faster than the Spanish speech recognizer **114***b*. In such a case, the computing system **112** may provide the English proposed transcription **118***a* to the client device **108** while the Spanish speech recognizer **114***b*, and optionally the English speech recognizer **114***a*, execute to produce their final results.

[0050] As noted above, each of the speech recognizers **114** and the language identification module **116** may operate independently of each other and of the computing system **112**. Rather than wait for every speech recognizer **114** to provide output, the computing system **112** may wait until the end of a timeout period, and use the information from whichever of the modules that has responded within the timeout period. By setting an appropriate timeout period, a balance between responsiveness to the user **102** and accuracy may be achieved.

[0051] In such cases, the computing system **112** may create preliminary combined scores, similar to or different from the combined scores **128**. These preliminary combined scores may not include all languages, for example if some speech recognizers **114** have not produced preliminary results.

[0052] In cases in which a preliminary transcription is provided to the client device **108** and when a final result has a higher combined score **128** than the combined score for the preliminary results, the associated final transcription **118** may be provided to the client device **108** as an update. The client device **108** may be configured to, for example, replace the preliminary transcription with the updated transcription. In cases in which the preliminary combined score is the same or greater than the greatest combined score **128**, the computing system **112** may transmit no update, transmit a transcription **118** as an update, or take any other appropriate action.

[0053] Although one particular example is shown here, other systems may be used to accomplish similar results. For example, the speech recognition process may all be performed locally on the client device **108** or another device.

[0054] While one speech recognizer **114** per language, and one language per speech recognizer **114**, is shown, other configurations are possible. For example, another system may have a speech recognizer **114** per dialect of one or more languages, and/or may have a speech recognizer **114** configured to recognize two or more languages.

[0055] Although not shown here, the computing system **112** may include or have access to speech recognizers **114** that are specific to other languages and not used for a particular user's **102** utterance **106**. For example, data recording the user's **102** preferences or historic activity may indicate that the user **102** is fluent in English and Spanish, but not French or Chinese. The computing system **112**, while including and/or having access to speech recognizers **114** specific to French and/or Chinese, may choose not to send the data **110** to those recognizers **114**. In some configurations, the computing system **112** may send the data **110** to one, some, or all other speech recognizers **114**.

[0056] In some implementations, the computing system **112** uses the combined scores **128**, or the confidence scores **120** and/or **126**, to determine boundaries where speech of one language ends and speech of another language begins. For example, output of the language models **114** may be used to identify likely boundaries between words. The computing system **112** can define the speech between each of the boundaries as a different speech segment, and may select the most likely language for each speech segment. The computing system **112** may then splice together the transcriptions of various speech recognizers to determine the final transcription, with each speech segment being represented by the transcription corresponding to its selected language.

[0057] FIG. **2** is a block diagram of an example of a processing pipeline **200** for a speech recognition module. The processing pipeline **200** may be used, for example, by the language identification module **116**, as described with reference to FIG. **1**. Although a particular number, type, and order of pipeline elements are shown here, different numbers, types, or orders of pipeline elements may be used in other configurations to achieve the same or similar results.

[0058] A filterbank **202** may receive data representing an utterance, for example data **110** or other data. The filterbank **202** may be configured as one or more filters (e.g., band-pass, discrete-time, continuous-time) that can separate the received data into frames. These frames may represent a time-based partitioning of the received data, and thus the utterance. In some configurations, each frame may represent a period of time on the order of milliseconds (e.g., 1 ms, 10 ms, 100 ms, etc.). A frame stacker **204** may, for each particular frame generated by the filterbank **202**, stack surrounding frames with the particular frame. For example, the previous 20 and following 5 frames may be stacked with a current frame. In this case, each stack can represent 26 frames of the data **110**. If each frame represents, for example, 10 ms of speech, then the stack represents 260 ms of an utterance.

[0059] A Voice Activity Detection (or VAD) segmenter **206** may, for each stack of frames, segment out portions of that represent no voice activity. For example, when an utterance includes a pause between utterances, portions of or all of a stack of frames may represent some or all of that pause. These stacks or portions of the stacks may be segmented out so that, for example, they are not examined further down the pipeline.

[0060] A normalizer **208** may normalize the segmented stacks of frames. For example, the normalizer **208** may be configured to normalize the stacks such that each stack contains the same mean and standard deviation of a parameter or parameters of the stacks. This normalization may be useful, for example, to normalize stacks related to utterances with variable volume, static interference, or other audio artifacts.

[0061] A neural network **210** may receive the normalized, segmented stacks of frames and may generate confidence scores for each of a plurality of languages. In some configurations, the neural network **210** may be an artificial neural network, such as a deep neural network. A deep neural network may be, for example, a neural network that contains a large number of hidden nodes compared to the number of edge nodes in the network.

[0062] A posterior analyzer **212** can use the output of the neural network **210** to perform an analysis and to log confidence scores for a plurality of languages. This analysis may be, for example, a Bayesian posterior probability analysis that assigns, for each language, a confidence or probability that the original utterance was in the associated language. These confidence values may be, for example, the confidence scores **126**, described with reference to FIG. **1**.

[0063] For each input stack of frames, the language identification module may produce multiple outputs, one for each of the languages that the language identification module is trained to identify. As additional input is provided, additional confidence values are produced. For example, a new input stack may be input for each 10 ms increment of speech data, and each input can have its own corresponding set of outputs. A computing system or the language identification module itself may average these outputs together to produce an average score. For example, ten different English language confidence scores may be averaged together, each representing estimates for a different 10 ms region of speech, may be averaged together to generate a single confidence score that represents a likelihood for the entire 100 ms period represented by ten different input stacks. Averaging the individual outputs of a neural network or other model can improve the overall accuracy of the speech recognition system.

[0064] FIG. **3** is a schematic diagram of example data **300** related to speech recognition confidence scores. The data **300** may be a visualization of, for example, confidence scores generated by the language identification module **116** of FIG. **1**. In some configurations, the computing system **112** may never generate the visualization as shown, instead operating on the data in non-visual form.

[0065] The data **300** as shown is organized into a two-dimensional table. The table includes rows **310***a*-**310***i* that each correspond to a different language. For example, the row **310***a* may indicate confidence score values for English, the row **310***b* may indicate confidence score values for Spanish, the row **310***c* may indicate confidence score values for French, and so on. The languages may be, for example, each of the languages that a language identification module is trained to evaluate.

[0066] Each row **310***a*-**310***i* indicates a sequence of language identification module confidence scores for a corresponding language, where the scores may be, for example, the output of a trained neural network. The different values correspond to estimates based on different speech frames of an utterance, with the values from left to right showing a progression from beginning to end of the utterance. For example, scores are shown for a first analysis period **320***a*, which may indicate a first 10 ms frame of an utterance, other scores are shown for a second analysis period **320***b* that may indicate a subsequent 10 ms frame of the utterance, and so on. For each frame, a different score may be determined for each of language.

[0067] For each frame, the table includes a cell that is shaded based on the confidence value for the associated speech frame and language. In the example, values range from zero to 1, with darker regions representing higher probability estimates, and lighter regions representing lower probability estimates. The data **300** shows that the estimates of which language is being spoken may vary from frame to frame, even for an utterance in a single language. For example, the scores in the region **330** suggest that the utterance is in English, but the scores for the region **340** suggest that the utterance is in Spanish. Accordingly averaging values across multiple frames may help to provide consistency in estimating a language.

[0068] Further, the data **300** shows that, in the region **350**, the scores may not clearly indicate which language is being spoken. In the example, the estimates for multiple languages suggest that several languages are equally likely. Since confidence scores based on acoustic features may not always indicate the correct language, or may not identify the correct language with high confidence, confidence scores from language models may be used to improve accuracy, as discussed with respect to FIG. **1**.

[0069] FIGS. **4**A and **4**B show an example user interface **400** showing a preliminary transcription replaced by another transcription. In this example, the client device **108**, as described with reference to FIG. **1**, is shown as generating the user interface **400**. However, in other examples, other computing hardware may be used to create the user interface **400** or another user interface for displaying transcriptions.

[0070] As shown, the user interface **400** includes an input field **402**. The user interface **400** may provide a user with one or more ways to submit text to the input field **402**, including but not limited to the speech-to-text as described, for example, with respect to FIG. **1**. In this example, the user interface **400***a* displays a preliminary transcription in the input field **402***a*. For example, the user may have entered an utterance in an unspecified language. A preliminary transcription of "Cares ear," which includes English words, is generated by the client device **108** and/or networked computing resources communicably coupled to the client device **108**.

[0071] After displaying the preliminary transcription, and as additional speech is analyzed, the speech recognition system may determine that the correct language is different from the language of the preliminary transcription. As a result, the speech recognition system may provide a final transcription that replaces some or all of the preliminary transcription. For example, the client device **108** may receive input that replaces the preliminary transcription with a final transcription of "Queres it al cine", as shown in the input field **402***b*. In this example, the preliminary transcription shown in the input field **402***a* is an English language transcription, and the final transcription shown in the input field **402***b* is in a different language—Spanish. In other examples, the preliminary and final transcriptions may be in the same language having the same or different text.

[0072] FIG. **5** is a flowchart of an example process **500** for speech recognition. The example process **500** will be described here with reference to the elements of the system **100** described with reference to FIG. **1**. However, the same, similar, or different elements may be used to perform the process **500** or a different process that may produce the same or similar results.

[0073] Speech data for an utterance is received (**502**). For example, the user **102** can navigate to an input field and press an interface button indicating a desire to submit speech-to-text input. The client device **108** may provide the user **102** with a prompt, and the user **102** can speak utterance **106** into the user's **102** client device **108**. In response, the client device **108** may generate data **110** to represent the utterance **106** and transmit that data **110** to the computing system **112**.

[0074] Speech data is provided to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language (**504**). For example, the computing system **112** may identify one or more candidate languages that the utterance **106** may be in. For example, the computing system **112** may use the mobile computing device's **108** geolocation information and/or data about the user **102** to identify candidate languages. For each candidate language, the computing system **112** may make a request or provide the data **110** to a corresponding language-specific speech recognizer **114** via, for example, a remote procedure call. Additionally, the computing system **112** may make a request or provide the data **110** to the language identification module **116** via, for example, a remote procedure call.

[0075] Language identification scores corresponding to different languages are received from the language identification module (**506**). The language identification scores each indicate a likelihood that the utterance is speech in the corresponding language. For example, the language identification module **116** may use a processing pipeline, such as the processing pipeline **200** as described with reference to FIG. **2**, or another processing pipeline or other structure to generate confidence scores **126**. The language identification module **116** may return these confidence scores **126** to the computing system **112**, for example by return of a remote procedure call.

[0076] A language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model is received from each of the multiple speech recognizers (**508**). For example, the speech recognizers **114** may generate confidence scores **120** using, for example, the acoustic models **122** and language models **124**. The speech recognizers **114** may return these confidence scores **120** to the computing system **112**, for example by return of remote procedure calls.

[0077] A language is selected based on the language identification scores and the language model confidence scores (**510**). For example, the computing system **112** may use the confidence scores **126**, the confidence scores **120**, and optionally other data to determine the most likely language of the utterance **106**. Once the most likely language is selected, the corresponding transcription **118** may be transmitted by the computing system **112** to the mobile computing device **104** such that the transcription **118** is displayed to the user **102**.

[0078] FIG. **6** shows an example of a computing device **600** and an example of a mobile computing device that can be used to implement the techniques described here. The computing device **600** is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. The mobile computing device is intended to represent various forms of mobile devices, such as personal digital assistants, cellular telephones, smart-phones, and other similar computing devices. The components shown here, their connections and relationships, and their functions, are meant to be exemplary only, and are not meant to limit implementations of the inventions described and/or claimed in this document.

[0079] The computing device **600** includes a processor **602**, a memory **604**, a storage device **606**, a high-speed interface **608** connecting to the memory **604** and multiple high-speed expansion ports **610**, and a low-speed interface **612** connecting to a low-speed expansion port **614** and the storage device **606**. Each of the processor **602**, the memory **604**, the storage device **606**, the high-speed interface **608**, the high-speed expansion ports **610**, and the low-speed interface **612**, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor **602** can process instructions for execution within the computing device **600**, including instructions stored in the memory **604** or on the storage device **606** to display graphical information for a GUI on an external input/output device, such as a display **616** coupled to the high-speed interface **608**. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0080] The memory **604** stores information within the computing device **600**. In some implementations, the memory **604** is a volatile memory unit or units. In some implementations, the memory **604** is a non-volatile memory unit or units. The memory **604** may also be another form of computer-readable medium, such as a magnetic or optical disk.

[0081] The storage device **606** is capable of providing mass storage for the computing device **600**. In some implementations, the storage device **606** may be or contain a computer-readable medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. A computer program product can be tangibly embodied in an information carrier. The computer program product may also contain instructions that, when executed, perform one or more methods, such as those described above. The computer program product can also be tangibly embodied in a computer- or machine-readable medium, such as the memory **604**, the storage device **606**, or memory on the processor **602**.

[0082] The high-speed interface **608** manages bandwidth-intensive operations for the computing device **600**, while the low-speed interface **612** manages lower bandwidth-intensive operations. Such allocation of functions is exemplary only. In some implementations, the high-speed interface **608** is coupled to the memory **604**, the display **616** (e.g., through a graphics processor or accelerator), and to the high-speed expansion ports **610**, which may accept various expansion cards (not shown). In the implementation, the low-speed interface **612** is coupled to the storage device **606** and the low-speed expansion port **614**. The low-speed expansion port **614**, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet) may be coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

[0083] The computing device **600** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server **620**, or multiple times in a group of such servers. In addition, it may be implemented in a personal computer such as a laptop computer **622**. It may also be implemented as part of a rack server system **624**. Alternatively, components from the computing device **600** may be combined with other components in a mobile device (not shown), such as a mobile computing device **650**. Each of such devices may contain one or more of the computing device **600** and the mobile computing device **650**, and an entire system may be made up of multiple computing devices communicating with each other.

[0084] The mobile computing device **650** includes a processor **652**, a memory **664**, an input/output device such as a display **654**, a communication interface **666**, and a transceiver **668**, among other components. The mobile computing device **650** may also be provided with a storage device, such as a micro-drive or other device, to provide additional storage. Each of the processor **652**, the memory **664**, the display **654**, the communication interface **666**, and the transceiver **668**, are interconnected using various buses, and several of the components may be mounted on a common motherboard or in other manners as appropriate.

[0085] The processor **652** can execute instructions within the mobile computing device **650**, including instructions stored in the memory **664**. The processor **652** may be implemented as a chipset of chips that include separate and multiple analog and digital processors. The processor **652** may provide, for example, for coordination of the other components of the mobile computing device **650**, such as control of user interfaces, applications run by the mobile computing device **650**, and wireless communication by the mobile computing device **650**.

[0086] The processor **652** may communicate with a user through a control interface **658** and a display interface **656** coupled to the display **654**. The display **654** may be, for example, a TFT (Thin-Film-Transistor Liquid Crystal Display) display or an OLED (Organic Light Emitting Diode) display, or other appropriate display technology. The display interface **656** may comprise appropriate circuitry for driving the display **654** to present graphical and other information to a user. The control interface **658** may receive commands from a user and convert them for submission to the processor **652**. In addition, an external interface **662** may provide communication with the processor **652**, so as to enable near area communication of the mobile computing device **650** with other devices. The external interface **662** may provide, for example, for wired communication in some implementations, or for wireless communication in other implementations, and multiple interfaces may also be used.

[0087] The memory **664** stores information within the mobile computing device **650**. The memory **664** can be implemented as one or more of a computer-readable medium or media, a volatile memory unit or units, or a non-volatile memory unit or units. An expansion memory **674** may also be provided and connected to the mobile computing device **650** through an expansion interface **672**, which may include, for example, a SIMM (Single In Line Memory Module) card interface. The expansion memory **674** may provide extra storage space for the mobile computing device **650**, or may also store applications or other information for the mobile computing device **650**. Specifically, the expansion memory **674** may include instructions to carry out or supplement the processes described above, and may include secure information also. Thus, for example, the expansion memory **674** may be provide as a security module for the mobile computing device **650**, and may be programmed with instructions that permit secure use of the mobile computing device **650**. In addition, secure applications may be provided via the SIMM cards, along with additional information, such as placing identifying information on the SIMM card in a non-hackable manner.

[0088] The memory may include, for example, flash memory and/or NVRAM memory (non-volatile random access memory), as discussed below. In some implementations, a computer program product is tangibly embodied in an information carrier. The computer program product contains instructions that, when executed, perform one or more methods, such as those described above. The computer program product can be a computer- or machine-readable medium, such as the memory **664**, the expansion memory **674**, or memory on the processor **652**. In some implementations, the computer program product can be received in a propagated signal, for example, over the transceiver **668** or the external interface **662**.

[0089] The mobile computing device **650** may communicate wirelessly through the communication interface **666**, which may include digital signal processing circuitry where necessary. The communication interface **666** may provide for communications under various modes or protocols, such as GSM voice calls (Global System for Mobile communications), SMS (Short Message Service), EMS (Enhanced Messaging Service), or MMS messaging (Multimedia Messaging Service), CDMA (code division multiple access), TDMA (time division multiple access), PDC (Personal Digital Cellular), WCDMA (Wideband Code Division Multiple Access), CDMA2000, or GPRS (General Packet Radio Service), among others. Such communication may occur, for example, through the transceiver **668** using a radio-frequency. In addition, short-range communication may occur, such as using a Bluetooth, WiFi, or other such transceiver (not shown). In addition, a GPS (Global Positioning System) receiver module **670** may provide additional navigation- and location-related wireless data to the mobile computing device **650**, which may be used as appropriate by applications running on the mobile computing device **650**.

[0090] The mobile computing device **650** may also communicate audibly using an audio codec **660**, which may receive spoken information from a user and convert it to usable digital information. The audio codec **660** may likewise generate audible sound for a user, such as through a speaker, e.g., in a handset of the mobile computing device **650**. Such sound may include sound from voice telephone calls, may include recorded sound (e.g., voice messages, music files, etc.) and may also include sound generated by applications operating on the mobile computing device **650**.

[0091] The mobile computing device **650** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a cellular telephone **680**. It may also be implemented as part of a smart-phone **682**, personal digital assistant, or other similar mobile device.

[0092] Various implementations of the systems and techniques described here can be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations can include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

[0093] These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and can be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms machine-readable medium and computer-readable medium refer to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic

Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term machine-readable signal refers to any signal used to provide machine instructions and/or data to a programmable processor.

[0094] To provide for interaction with a user, the systems and techniques described here can be implemented on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user can be received in any form, including acoustic, speech, or tactile input.

[0095] The systems and techniques described here can be implemented in a computing system that includes a back end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front end component (e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the systems and techniques described here), or any combination of such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network (LAN), a wide area network (WAN), and the Internet.

[0096] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

What is claimed is:

1. A method performed by one or more computers, the method comprising:

receiving speech data for an utterance;

providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language;

receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language;

receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model; and

selecting a language based on the language identification scores and the language model confidence scores.

2. The method of claim **1**, wherein receiving the speech data for the utterance comprises receiving the speech data from a user over a network;

wherein the method further comprises:

receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer; and

providing the transcription in the selected language to the user over the network.

3. The method of claim **2**, further comprising, before receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer:

receiving, from a particular one of the multiple speech recognizers, a preliminary transcription of the utterance in a language corresponding to the speech recognizer;

providing the preliminary transcription to the user over the network before providing the transcription in the selected language to the user over the network.

4. The method of claim **3** wherein the preliminary transcription is in the selected language.

5. The method of claim **3** wherein the preliminary transcription is a language different than the selected language.

6. The method of claim **3** wherein the preliminary transcription is provided over the network for display to the user; and

wherein the transcription in the selected language is provided for display in place of the preliminary transcription, after the preliminary transcription has been provided over the network.

7. The method of claim **3** wherein the method further comprises:

receiving, from the particular one of the multiple speech recognizers, a preliminary language model confidence score that indicates a preliminary level of confidence that a language model has in the preliminary transcription of the utterance in a language corresponding to the language model; and

determining that the preliminary language model confidence score is less than a language model confidence score received from the particular one of the multiple speech recognizers.

8. The method of claim **1**, wherein providing the speech data to a language identification module comprises providing the speech data to a neural network that has been trained to provide likelihood scores for multiple languages.

9. The method of claim **1**, wherein selecting the language based on the language identification scores and the language model confidence scores comprises:

determining a combined score for each of multiple languages, wherein the combined score for each language is based on at least the language identification score for the language and the language model confidence score for the language; and

selecting the language based on the combined scores.

10. The method of claim **9**, wherein determining a combined score for each of multiple languages comprises weighting the likelihood scores or the language model confidence scores using one or more weighting values.

11. The method of claim **1**, wherein receiving the speech data comprises receiving speech data that includes an utterance of a user;

further comprising:

before receiving the speech data, receiving data indicating multiple languages that the user speaks;

storing data indicating the multiple languages that the user speaks;

wherein providing the speech data to multiple speech recognizers that are each configured to recognize speech in a different language comprises, based on the stored data indicating the multiple languages that the user speaks, providing the speech data to a set of speech recognizers configured to recognize speech in a different one of the languages that the user speaks.

12. A non-transitory computer storage medium tangibly encoded with computer program instructions that, when executed by one or more processors, cause a computer device to perform operations comprising:

receiving speech data for an utterance;

providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language;

receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language;

receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model; and

selecting a language based on the language identification scores and the language model confidence scores.

13. The medium of claim **12**, wherein receiving the speech data for the utterance comprises receiving the speech data from a user over a network;

wherein the operations further comprise:

receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer; and

providing the transcription in the selected language to the user over the network.

14. The medium of claim **13**, the operations comprising, before receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer:

receiving, from a particular one of the multiple speech recognizers, a preliminary transcription of the utterance in a language corresponding to the speech recognizer;

providing the preliminary transcription to the user over the network before providing the transcription in the selected language to the user over the network.

15. The medium of claim **12**, wherein receiving the speech data comprises receiving speech data that includes an utterance of a user;

the operations further comprising:

before receiving the speech data, receiving data indicating multiple languages that the user speaks;

storing data indicating the multiple languages that the user speaks;

wherein providing the speech data to multiple speech recognizers that are each configured to recognize speech in a different language comprises, based on the stored data indicating the multiple languages that the user speaks, providing the speech data to a set of speech recognizers configured to recognize speech in a different one of the languages that the user speaks.

16. A system comprising:

one or more processors; and

a non-transitory computer storage medium tangibly encoded with computer program instructions that, when executed by the one or more processors, cause a computer device to perform operations comprising:

receiving speech data for an utterance;

providing the speech data to (i) a language identification module and (ii) multiple speech recognizers that are each configured to recognize speech in a different language;

receiving, from the language identification module, language identification scores corresponding to different languages, the language identification scores each indicating a likelihood that the utterance is speech in the corresponding language;

receiving, from each of the multiple speech recognizers, a language model confidence score that indicates a level of confidence that a language model has in a transcription of the utterance in a language corresponding to the language model; and

selecting a language based on the language identification scores and the language model confidence scores.

17. The system of claim **16**, wherein receiving the speech data for the utterance comprises receiving the speech data from a user over a network;

wherein the operations further comprise:

receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer; and

providing the transcription in the selected language to the user over the network.

18. The system of claim **17**, the operations comprising, before receiving, from each of the speech recognizers, a transcription of the utterance in a language corresponding to the speech recognizer:

receiving, from a particular one of the multiple speech recognizers, a preliminary transcription of the utterance in a language corresponding to the speech recognizer;

providing the preliminary transcription to the user over the network before providing the transcription in the selected language to the user over the network.

19. The system of claim **18**, wherein the preliminary transcription is provided over the network for display to the user; and

wherein the transcription in the selected language is provided for display in place of the preliminary transcription, after the preliminary transcription has been provided over the network.

20. The system of claim **18**, wherein the operations further comprises:

receiving, from the particular one of the multiple speech recognizers, a preliminary language model confidence score that indicates a preliminary level of confidence that a language model has in the preliminary transcription of the utterance in a language corresponding to the language model; and

determining that the preliminary language model confidence score is less than a language model confidence score received from the particular one of the multiple speech recognizers.

* * * * *

C.3 Cluster Specific Speech Model

TITLE	**Cluster Specific Speech Model.**	
INVENTORS	Andrew Senior	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
DATE	2015/3/20	
NUMBER	US 20150269931 A1	
APPLICATION	14/663,610	
ASSIGNEE	Google Inc.	

Summary: Methods, systems, and apparatus for receiving data representing acoustic characteristics of a user’s voice; selecting a cluster for the data from among a plurality of clusters, where each cluster is associated with a speech model trained by a neural network using data in the respective cluster.

Contributions: The candidate provided the necessary code to generate the i-vector systems and to compute partitions of the i-vector space using agglomerative clustering; generated i-vector models and clusters with optimized hyperparameters; and provided the augmented frequency features with i-vector components used to train the speech recognition models associated with each cluster.

US 20150269931A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2015/0269931 A1**

Senior et al. (43) **Pub. Date: Sep. 24, 2015**

(54) **CLUSTER SPECIFIC SPEECH MODEL**

(71) Applicant: **Google Inc.**, Mountain View, CA (US)

(72) Inventors: **Andrew W. Senior**, New York, NY (US); **Ignacio Lopez Moreno**, New York, NY (US)

(21) Appl. No.: **14/663,610**

(22) Filed: **Mar. 20, 2015**

Related U.S. Application Data

(60) Provisional application No. 61/969,426, filed on Mar. 24, 2014.

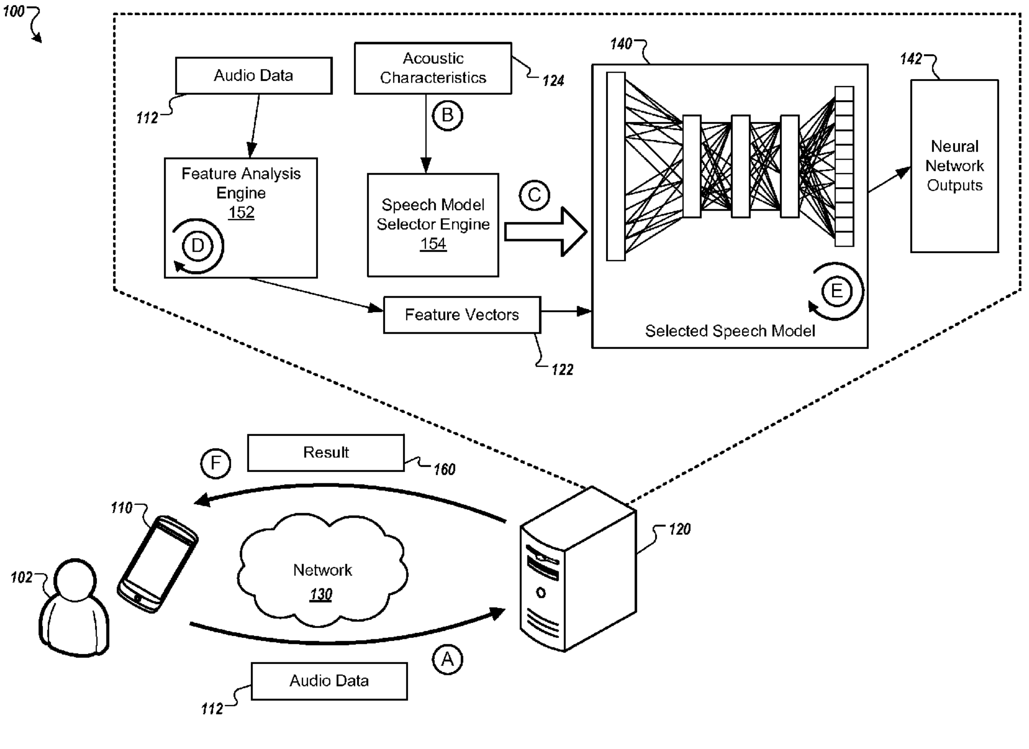
Publication Classification

(51) **Int. Cl.**
G10L 15/06 (2006.01)

(52) **U.S. Cl.**
CPC ***G10L 15/063*** (2013.01); *G10L 2015/0631* (2013.01)

(57) **ABSTRACT**

Methods, systems, and apparatus, including computer programs encoded on a computer storage medium, for receiving data representing acoustic characteristics of a user's voice; selecting a cluster for the data from among a plurality of clusters, where each cluster includes a plurality of vectors, and where each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster; and in response to receiving one or more utterances of the user, providing the speech model associated with the cluster for transcribing the one or more utterances



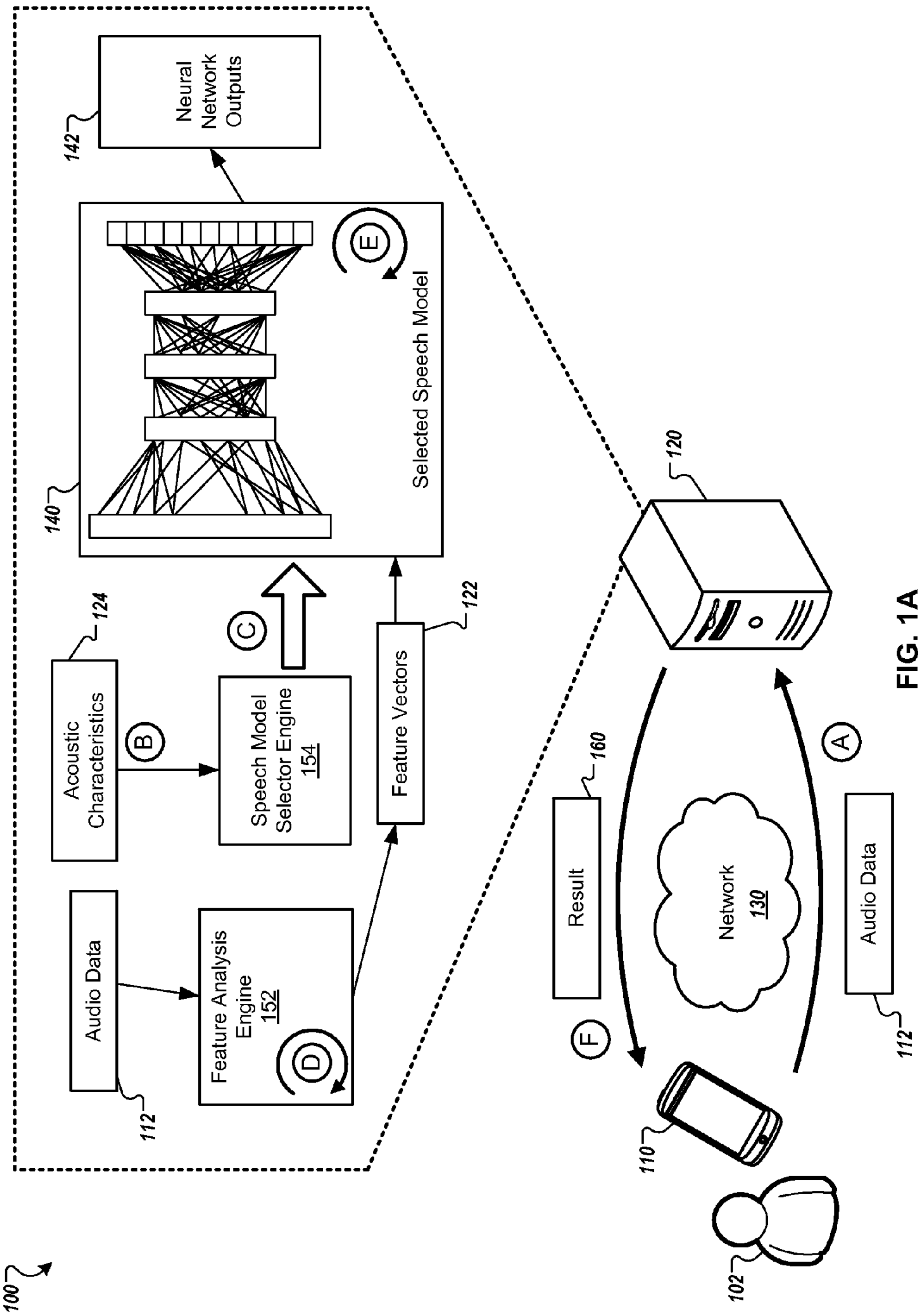

100
Audio Data
112
Feature Analysis Engine 152
D
Acoustic Characteristics
124
B
Speech Model Selector Engine 154
C
Feature Vectors
122
140
Selected Speech Model
E
142
Neural Network Outputs
120
Result
160
F
Network 130
Audio Data
112
A
110
102

FIG. 1A

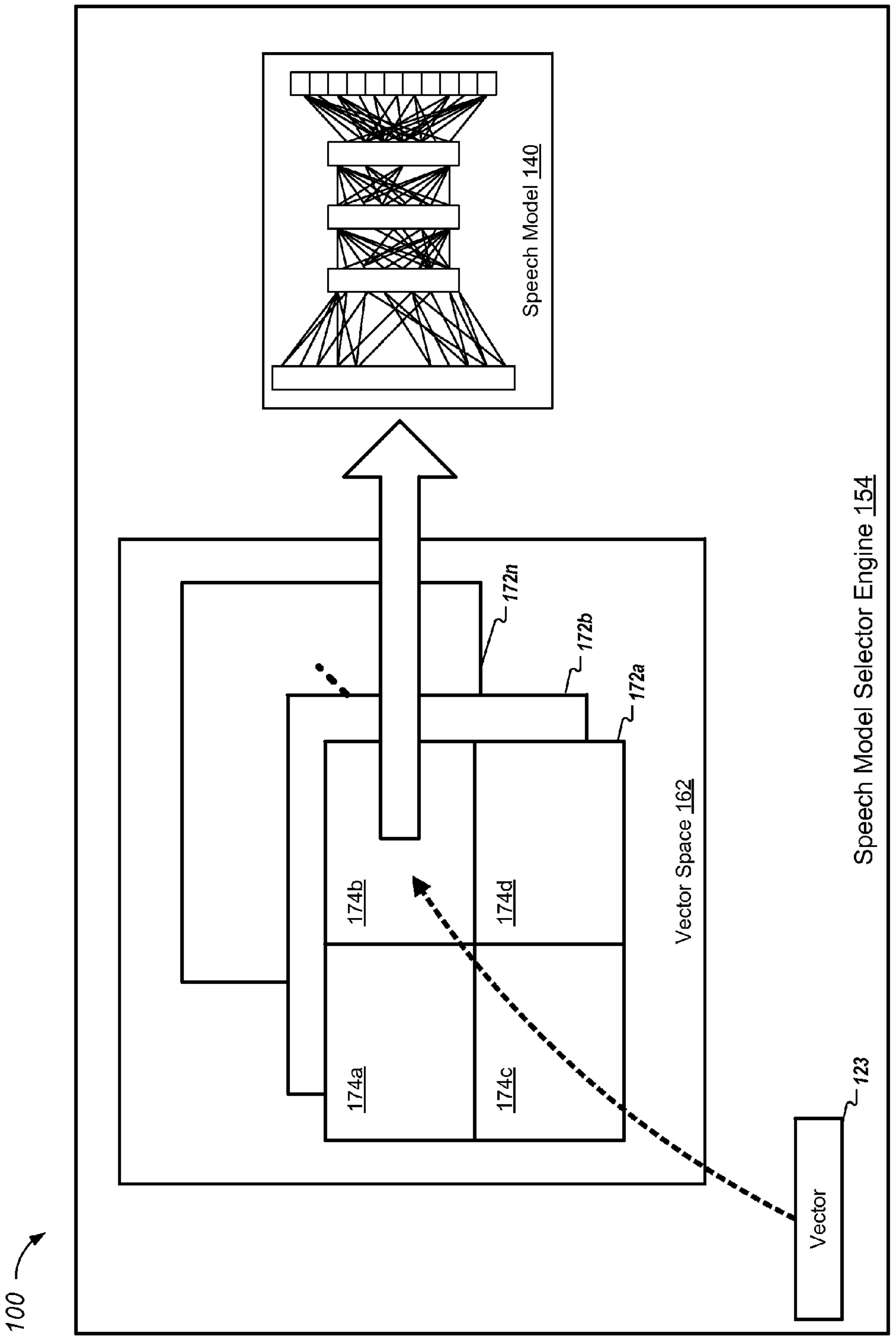

100
Speech Model 140
Speech Model Selector Engine 154
Vector Space 162
172n
172b
172a
174a
174b
174c
174d
Vector
123

FIG. 1B

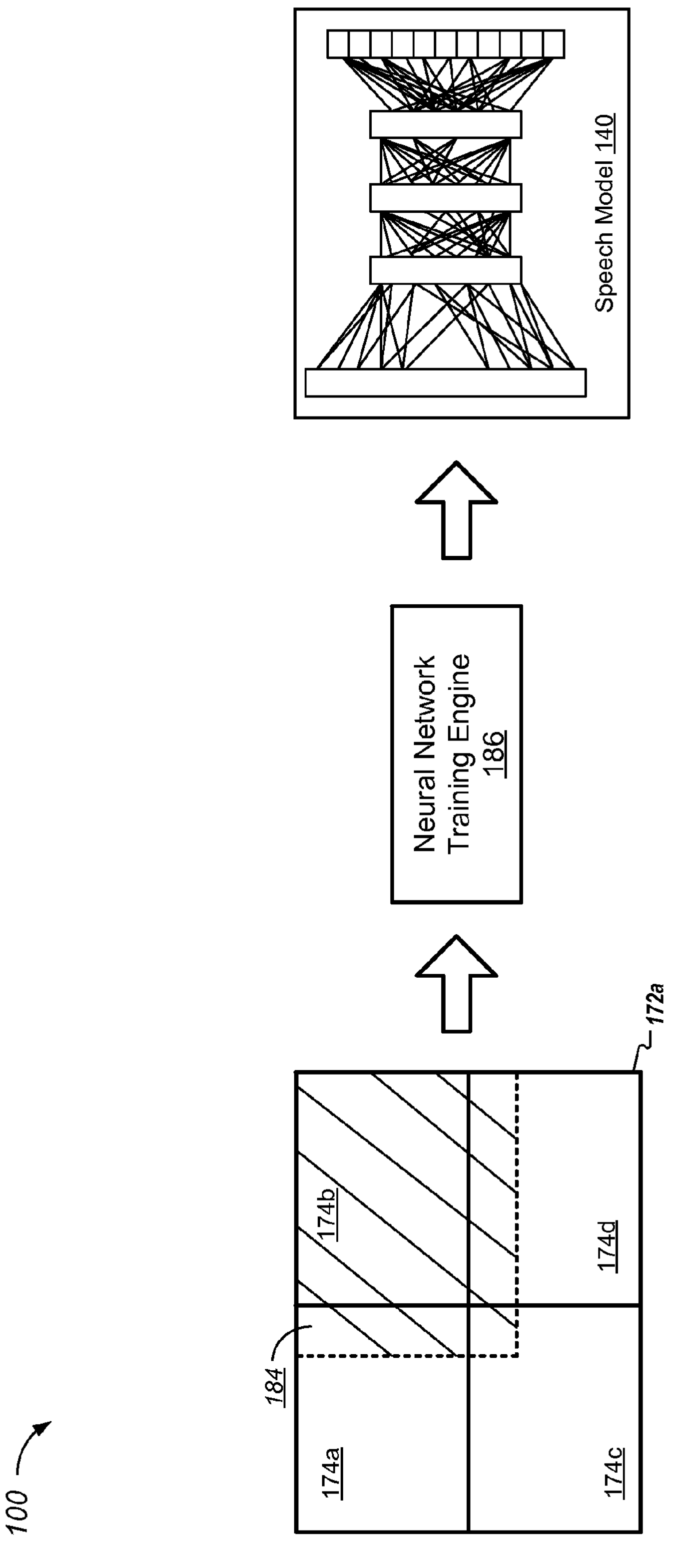

100
172a
174a
174b
174c
174d
184
Neural Network Training Engine 186
Speech Model 140

FIG. 1C

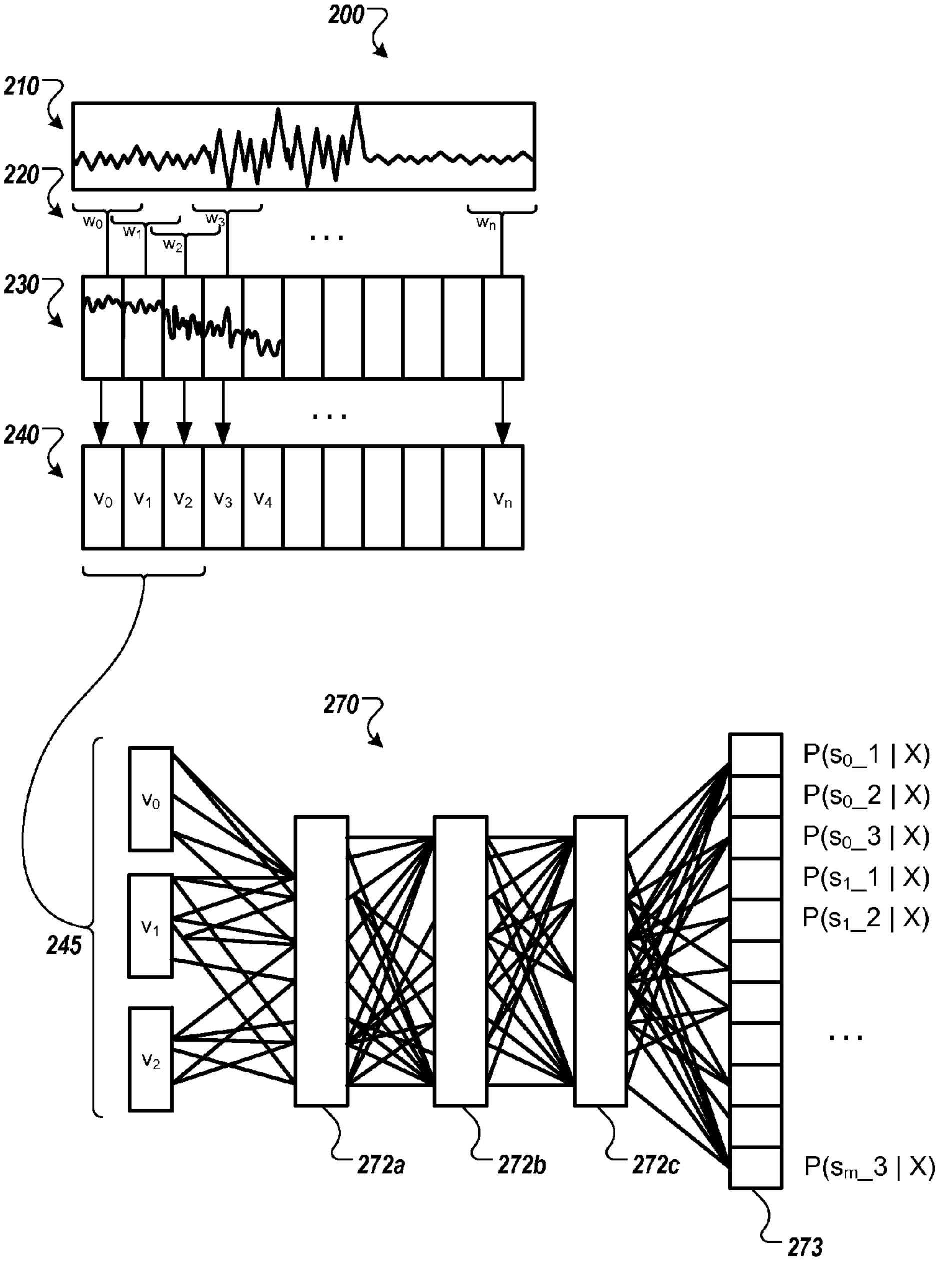

200
210
220
w_0
w_1
w_2
w_3
...
w_n
230
...
240
v_0
v_1
v_2
v_3
v_4
v_n
270
245
v_0
v_1
v_2
P(s₀_1 | X)
P(s₀_2 | X)
P(s₀_3 | X)
P(s₁_1 | X)
P(s₁_2 | X)
...
P(sₘ_3 | X)
272a
272b
272c
273

FIG. 2

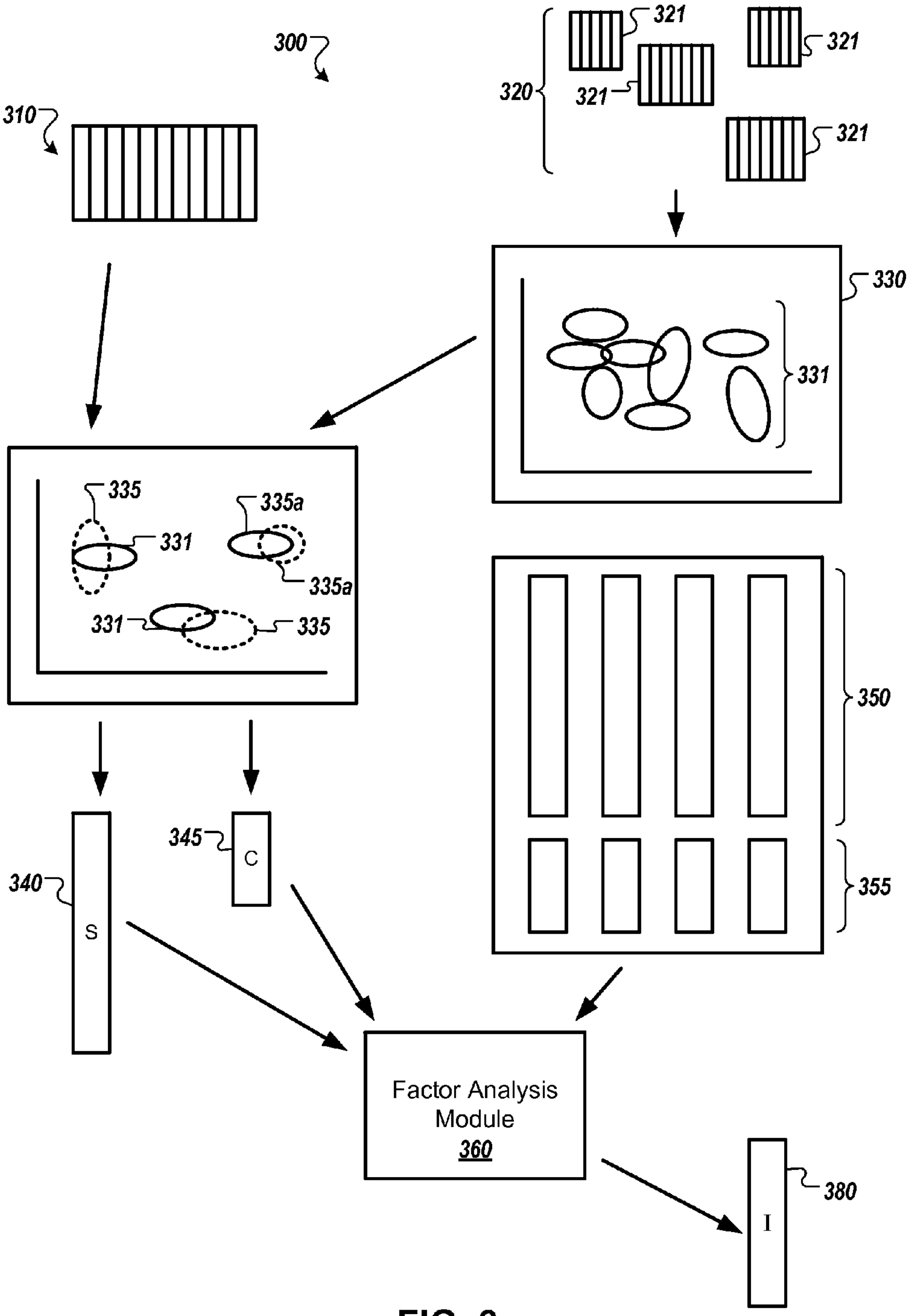

300
310
320
321
321
321
321
330
331
335
331
335a
335a
331
335
350
355
340
S
345
C
Factor Analysis
Module
360
380
I

FIG. 3

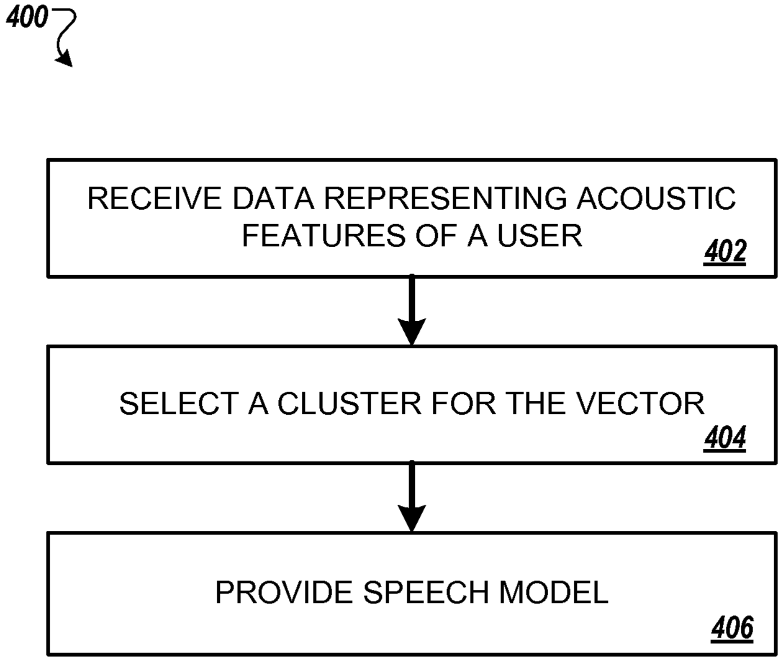

400
RECEIVE DATA REPRESENTING ACOUSTIC FEATURES OF A USER
402
SELECT A CLUSTER FOR THE VECTOR
404
PROVIDE SPEECH MODEL
406

FIG. 4

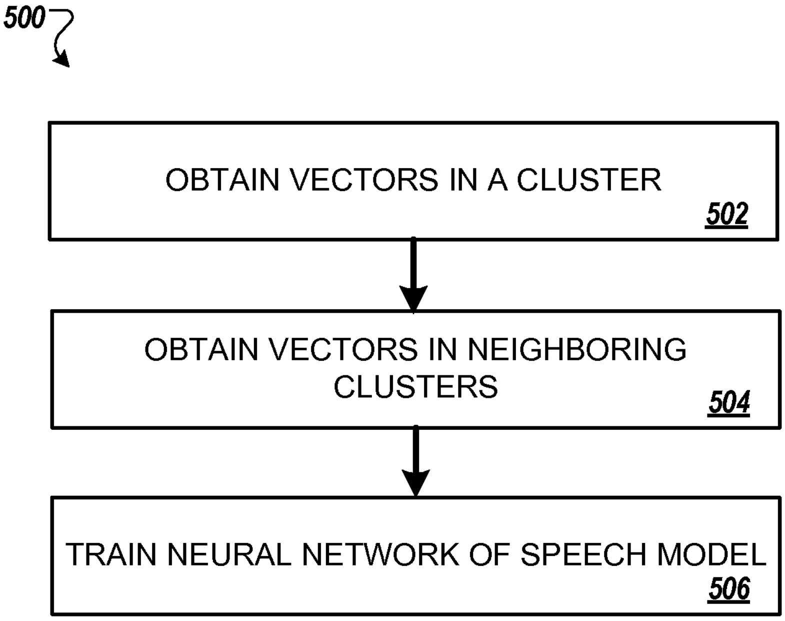

500
OBTAIN VECTORS IN A CLUSTER
502
OBTAIN VECTORS IN NEIGHBORING CLUSTERS
504
TRAIN NEURAL NETWORK OF SPEECH MODEL
506

FIG. 5

CLUSTER SPECIFIC SPEECH MODEL

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Application Ser. No. 61/969,426, filed on Mar. 24, 2014, which is incorporated by reference.

TECHNICAL FIELD

[0002] This specification describes technologies related to speech recognition.

BACKGROUND

[0003] Automatic speech recognition is an important technology that is used in mobile devices and other devices. In general, automatic speech recognition attempts to provide accurate transcriptions of what a person has said.

SUMMARY

[0004] According to one innovative aspect of the subject matter described in this specification, acoustic characteristics of users' utterances may be represented as vectors in a vector space. The vector space may be segmented into multiple clusters, where a speech model based on a neural network may be trained for each cluster using vectors in the respective cluster. Acoustic characteristics of a new user's utterance may be represented as a vector in the vector space, and a corresponding cluster may be selected. A speech model associated with the selected cluster may be provided to the user for speech recognition.

[0005] In general, one innovative aspect of the subject matter described in this specification can be embodied in methods that include the actions of receiving data representing acoustic characteristics of a user's voice, selecting a cluster for the data from among a plurality of clusters, where each cluster includes a plurality of vectors, and where each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster, and in response to receiving one or more utterances of the user, providing the speech model associated with the cluster for transcribing the one or more utterances.

[0006] These and other embodiments may each optionally include one or more of the following features. For instance, the plurality of clusters may be segmented based on vector distances to centroids of the clusters. Selecting a cluster for the data may include determining a vector based on the data, determining that a vector distance between the vector and the cluster is a shortest distance compared to vector distances between the vector and other clusters of the plurality of clusters, and based on determining that the vector distance between the vector and the cluster is the shortest distance, selecting the cluster for the vector.

[0007] Selecting a cluster for the data may include receiving data indicative of latent variables of multivariate factor analysis of an audio signal of the user, and selecting an updated cluster using the latent variables.

[0008] The process may include receiving a feature vector that models audio characteristics of a portion of an utterance of the user, and determining, using the feature vector as an input, a candidate transcription for the utterance based on an output of the neural network of the speech model. Providing the speech model for transcribing the one or more utterances may include providing the speech model to a computing device of the user.

[0009] The acoustic characteristics of the user may include a gender of the user, an accent of the user, a pitch of an utterance of the user, background noises around the user, or age group of the user. The data may be an i-vector, and where the neural network may be trained using the i-vectors in the cluster and one or more i-vectors in one or more neighboring clusters. Each cluster may include a distinct plurality of vectors, and each cluster may be associated with a distinct speech model.

[0010] Other implementations of this and other aspects include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods, encoded on computer storage devices. A system of one or more computers can be so configured by virtue of software, firmware, hardware, or a combination of them installed on the system that in operation cause the system to perform the actions. One or more computer programs can be so configured by virtue of having instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

[0011] Advantageous implementations may include one or more of the following features. Vectors derived from utterances represent a combination of acoustic characteristics of the utterances, and the optimization of a speech model based on these vectors may provide a better recognition of the user's speech than a speech model optimized using a specific acoustic characteristic. If acoustic characteristics of a user are previously known, a speech model may be preselected and provided to the user before the user speaks. Speech models derived from clusters may be more compact in size because the training vectors have been segmented by clusters. A compact speech model may be loaded to a mobile computing device for performing speech recognition directly on the mobile computing device.

[0012] The details of one or more implementations are set forth in the accompanying drawings and the description below. Other potential features and advantages will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. **1**A is a block diagram that illustrates an example of a system for speech recognition using neural networks.

[0014] FIG. **1**B is a block diagram that illustrates an example of a system for selecting a speech model for a user.

[0015] FIG. **1**C is a block diagram that illustrates an example of a system for training a speech model based on neural network.

[0016] FIG. **2** is a diagram that illustrates an example of processing for speech recognition using neural networks.

[0017] FIG. **3** is a diagram that illustrates an example of processing to generate an i-vector.

[0018] FIG. **4** is a flow diagram that illustrates an example of a process for providing a speech model based on acoustic characteristics of a user.

[0019] FIG. **5** is a flow diagram that illustrates an example of a process for training a neural network of a speech model associated with a cluster.

[0020] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0021] FIG. **1**A is a block diagram that illustrates an example of a system **100** for speech recognition using neural networks. Briefly, based on acoustic characteristics of a user, a speech model from among multiple speech models is selected to recognize a user's spoken utterances. The system **100** includes a client device **110**, a computing system **120**, and a network **130**.

[0022] In the system **100**, the client device **110** can be, for example, a desktop computer, laptop computer, a tablet computer, a wearable computer, a cellular phone, a smart phone, a music player, an e-book reader, a navigation system, or any other appropriate computing device. The functions performed by the computing system **120** can be performed by individual computer systems or can be distributed across multiple computer systems. The network **130** can be wired or wireless or a combination of both and can include the Internet.

[0023] In some implementations, the computing system **120** may select a speech model **140** for a user **102** based on data **124** that represents acoustic characteristics, of the user **102**. The user **102** may use the client device **110** to communicate with the computing system **120** through the network **130**, and use the speech model **140** to recognize one or more utterances spoken by the user **102**. The computing system **120** may process the recognized utterances, and send the results back to the client device **110**.

[0024] Although not shown in FIG. **1**, in some other implementations, the computing system **120** may provide the selected speech model **140** to the user device **110**. The user **102** may then use the speech model **140** stored at the client device **110** to recognize one or more utterances spoken by the user **102** without reaching out to a remote server, (e.g., the computing system **120**), for the speech recognition, and therefore may save communications bandwidth and time.

[0025] It is desirable that the size of the speech model **140** be compact because the memory space on the client device **110** may be limited. As described below, the speech model **140** is based on a trained neural network. The neural network may be trained using a subset of training data that have been segmented into multiple clusters based on acoustic characteristics, and may result a more compact model for the client device **110**.

[0026] FIG. **1** also illustrates an example flow of data, shown in stages (A) to (F). Stages (A) to (F) may occur in the illustrated sequence, or they may occur in a sequence that is different than in the illustrated sequence. In some implementations, one or more of the stages (A) to (F) may occur offline, where the computing system **120** may perform computations when the client device **110** is not connected to the network **130**.

[0027] During stage (A), the user **102** speaks utterances to the client device **110**. The client device **110** records the utterances, and transmits the utterances as audio data **112** to the computing system **120** via the network **130**. In some implementations, the utterances may include one or more phrases that are known to the computing system **120**. For example, the user **102** may speak a phrase "Hello, phone" that is known to the computing system **120** before speaking the rest of the speech that the user **102** wishes the speech model to process.

[0028] During stage (B), the computing system **120** obtains data **124** representing acoustic characteristics of the user **102**, and inputs the data **124** to the speech model selector engine **154**. In some implementations, the data **124** may be a vector. In some implementations, the data **124** may represent time-independent characteristics of the utterances of the user. For example, the data **124** may be an i-vector that is described further in descriptions of FIG. **3** below.

[0029] In some implementations, an i-vector may be a current utterance i-vector derived from the current utterance (e.g., the audio data **112**) being recognized. In some implementations, the i-vector may be derived from audio signals other than the audio data **112** containing the utterances being recognized. For example, the i-vector may be derived from a prior utterance of the same speaker whose utterances are included in the audio data **112**.

[0030] In some implementations, the i-vector may be a speaker i-vector that is pre-computed for a particular speaker using multiple utterances of the speaker (e.g., utterances from multiple different recording sessions, such as recordings on different days). To generate a speaker i-vector, an i-vector can be determined for each utterance in a set of multiple utterances of the speaker. The i-vectors can be averaged together to obtain generate the speaker i-vector. In some implementations, where a speaker i-vector is used rather than an utterance i-vector derived from the utterance being recognized, post processing may discriminative training, such as linear discriminant analysis, to identify attributes that are indicative of speaker characteristics. For example, various techniques can be used to isolate speaker characteristics, independent of noise, room characteristics, and other non-speaker-dependent characteristics.

[0031] Unlike an i-vector computed using the audio data **112** being recognized, i-vectors derived from prior utterances may not reflect the particular background noise characteristics of the audio data **112**. These i-vectors will indicate characteristics of the speaker's voice and speaking style and are thus useful in recognition. In addition, the noise in prior utterances may be similar to the noise in the current utterance. The speaker i-vector may be calculated from a set recent utterances, such as a predetermined number of most recent utterances or audio acquired within a threshold time period, which may approximate the noise conditions of the current utterance if the recent utterances were recorded in a similar setting.

[0032] In some implementations, the computing system **120** may identify the speaker and select an i-vector based on the speaker's identity. An i-vector may be calculated for each of multiple users, and the i-vectors may be stored for later use in recognizing speech of the corresponding users. The computing system **120** may receive a device identifier for a device, such as a mobile phone, that the speaker is using to record speech. In addition, or as an alternative, the computing system **120** may receive a user identifier that identifies the user, such as a name or user account login. The computing system **120** may identify the speaker as a user that owns the device or a user is logged into a user account on the device. In some implementations, the computing system **120** may identify the speaker before recognition begins, or before audio is received during the current session. The computing system **120** may then look up the i-vector that corresponds to the identified user and use that i-vector to recognize received speech.

[0033] In some implementations, a successive approximation technique may be used to approximate and re-estimate the i-vector **250** while audio is received. The i-vector **250** may be re-estimated at a predetermined interval, for example, each time a threshold amount of new audio has been received. For example, a first i-vector may be estimated using the initial three seconds of audio received. Then, after another three seconds of audio has been received, a second i-vector may be estimated using the six seconds of audio received so far. After another three seconds, a third i-vector may be estimated using all nine seconds of audio received, and so on. The re-estimation period may occur at longer intervals, such as 10 seconds or 30 seconds, to reduce the amount of computation required. In some implementations, i-vectors are re-estimated at pauses in speech (e.g., as detected by a speech energy or voice activity detection algorithm), rather than at predetermined intervals.

[0034] As another example, the data **124** may be a supervector that is described further in descriptions of FIG. **3** below. In some implementations supervector may be a current utterance supervector derived from the current utterance (e.g., the audio data **112**) being recognized. In some implementations, the supervector may be derived from audio signals other than the audio data **112** containing the utterances being recognized. In some implementations, the supervector may be a speaker supervector that is pre-computed for a particular speaker using multiple utterances of the speaker. In some implementations, the computing system **120** may identify the speaker and select a supervector based on the speaker's identity.

[0035] As another example, the data **124** may be a feature vector that is an output of a hidden layer of a neural network. In some implementations, the system may obtain access to a neural network that has been trained to provide a distinct 1×N feature vector for each of N training speakers. In some implementations, the feature vectors for the different training speakers may be orthogonal to each other. The computing system **120** may input speech features corresponding to an utterance to the neural network, and then obtain a feature vector corresponding to the utterance based on output of a hidden layer of the neural network. For example, a 1×N feature vector may be obtained based on output of a last hidden layer of the neural network. In some implementations, the feature vector may be an average of the multiple feature vectors for multiple utterances.

[0036] In some implementations, the data **124** may be obtained using a portion of the audio data **112**. For example, the computing system **120** may generate a vector using the known phrase "Hello, phone" spoken by the user **102**. In some other implementations, the data may be obtained using a phrase that was previously spoken by the user **102**. For example, the computing system **120** may learn the identity of the user **102** by the identity of the client device **110**, and may access a phrase that was previously spoken by the user **102**, where the phrase was recorded and stored in the computing system **120** or another computing system. As another example, the computing system **120** may learn the identity of the user **102** because the user **102** has logged in the computing system **120** using an identity that is associated with a profile accessible by the computing system **120**.

[0037] During stage (C), the speech model selector engine **154** selects a speech model **140** for the user **102** based on the data **124**. In general, the system **100** includes multiple speech models implemented by neural networks, where each neural network is trained using a different set of vectors representing audio characteristics of training utterances. Briefly, a vector corresponding to the data **124** is projected to a vector space that includes the vectors of training utterances for the multiple speech models. The vector space is segmented into clusters. Depending on which cluster the vector is projected to, a speech model that is associated with the cluster is selected for the user **102**. The selection of the speech models is described further in descriptions of FIG. **1**B, and the training of the speech models is described further in descriptions of FIG. **1**C.

[0038] As used in this specification, an "engine" (or "software engine") refers to a software implemented input/output system that provides an output that is different from the input. An engine can be an encoded block of functionality, such as a library, a platform, a Software Development Kit ("SDK"), a software module, or an object.

[0039] During stage (D), the audio data **112** is input to a feature analysis engine **152** to determine one or more feature vectors **122** that correspond to the audio data **112**. In general, a feature vector **122** indicates audio characteristics during a different portion or window of the audio signal **112**. Each feature vector **122** may indicate acoustic properties of, for example, a 10 millisecond (ms), 25 ms, or 50 ms portion of the audio signal **112**.

[0040] During stage (E), the feature vectors **122** are input in the selected speech model **140**. In some implementations, the selected speech model **140** may be implemented using a neural network trained to act as an acoustic model. For example, the speech model **140** indicates likelihoods that feature vectors correspond to different speech units when the feature vectors and certain types of additional information are provided.

[0041] The speech model **140** produces neural network outputs **142**, which the computing system **120** uses to identify a transcription for the audio signal **112**. For example, the computing system **120** may provide the neural network outputs **142** to weighted finite state transducers that approximate a hidden Markov model (HMM), which may include information about a lexicon indicating the phonetic units of words, a grammar, and a language model that indicates likely sequences of words. The output of the HMM can be a word lattice from which the transcription may be derived.

[0042] During stage (F), a result **160** is transmitted from the computing system **120** to the client device **110** over the network **130**. In some implementations, the computing system **120** may provide the transcription to the client device **110** as the result **160**. In some other implementations, the computing system **120** may provide the transcription to another computing system for additional process, and provide the output of the additional process as the result **160**. For example, the computing system **120** may provide a transcription to a search engine to perform a search, and return the search results to the user **102**.

[0043] FIG. **1**B is a block diagram that illustrates an example of a system **100** for selecting a speech model for a user. In general, training vectors derived from training utterances are used to train neural networks for speech recognition. A training vector has a predetermined dimension, and represents acoustic characteristics of a particular training speaker. Training vectors of training utterances may be mapped to a vector space **162** accessible by the speech model selector engine **154**, where the vector space **162** may be segmented into multiple clusters. Because training vectors represent a combination of acoustic characteristics, and are not limited to one specific acoustic or demographic characteristic (e.g., pitch, gender, accent, etc.), the clustering of the training vectors enables grouping of users with similar speech patterns across multiple acoustic characteristics. The training of a neural network for a speech model of a cluster may be performed by the computing system **120**, or another computing system not shown here. The optimization of a neural network based on these clustered training vectors may provide a better recognition of the user's speech than a speech model optimized using one specific acoustic characteristic.

[0044] In some implementations, the clustering of training vectors may use hierarchical divisive clustering or k-Means. For example, given a predetermined number of cluster centroids in the vector space **162**, the vector space **162** may be segmented into the predetermined number of clusters, where each training vector is mapped to a respective cluster according to which centroid is the closest. For example, if the predetermined number is two, the system may segment the training utterances by gender. Here, the vector space **162** is segmented into clusters **172***a*, **172***b*, . . . , and **172***n*. In some implementations, a cluster may be further segmented. For example, the cluster **172***a* may be segmented into clusters **174***a*, **174***b*, **174***c*, and **174***d*.

[0045] Other known techniques for clustering training vectors may be used. For example, a Gaussian mixture model (GMM) may be used to cluster the training vectors. Given a predetermined number of clusters, the corresponding cluster centroids may be input to a GMM, where a mean value may be generated for each cluster centroid. Each training vector may then be input to the GMM to generate a respective output value. Each training vector may be mapped to a respective cluster according to which mean value associated with a cluster provides the smallest difference to the respective output value of the training vector.

[0046] In some implementations, for each cluster, a neural network is trained as a speech model using the training vectors in the cluster. The trained speech model is therefore optimized for the acoustic characteristics represented by the training vectors in the cluster. The speech model trained by training vectors provides an advantage over a speech model trained over a specific acoustic characteristic because training vectors typically represents a combination of acoustic characteristics. The training of a neural network may be implemented by a training algorithm that is described in more details in descriptions of FIG. **2**, or by another training algorithm.

[0047] The speech model selector engine **154** maps a vector **123** to a cluster in the vector space **162**. In some implementations, the vector **123** may be derived from the data **124**. In some other implementations, the vector **123** may be the data **124**. Here, the vector **123** is mapped to the cluster **174***b*, and the speech model selector engine **154** selects the speech model **140** corresponding to the cluster **174***b* for the user **102**. In some implementations, the speech model **140** is then used by the computing system **120** to transcribe the user's subsequent utterances. In some other implementations, the speech model **140** may be provided to the client device **110** to transcribe the user's utterances directly on the client device **110**.

[0048] FIG. **1**C is a block diagram that illustrates an example of a system **100** for training a speech model based on neural network. As described in FIG. **1**B, for each cluster, a neural network is trained as a speech model using the training vectors in the cluster. However, in the event that a vector **123** is mapped near a boundary between two or more clusters in the vector space **162**, the trained speech model **140** may not perform well because there may not be sufficient number of training vectors near and within the boundaries of one cluster to provide good training results for vectors that are near the boundaries.

[0049] In some implementations, a spilling technique may be used to address this issue. In a spilling technique, in addition to the training vectors in a particular cluster, one or more training vectors in the neighboring clusters of the particular cluster may also be used to train the speech model of the particular cluster. For example, a neural network training engine **186** may train a neural network for a speech model **140** of the cluster **174***b* by including training vectors that are mapped inside the shaded areas in clusters **174***a*, **174***c*, and **174***d*.

[0050] In some implementations, the boundary of the spilling technique may be a predetermined distance to the centroid of the particular cluster. In some other implementations, the boundary of the spilling technique may be determined based on a number of training vectors in the particular cluster. For example, the neural network training engine **186** may determine the boundary of the spilling technique based on a minimum number of training vectors needed to train the speech model. In some implementations, the boundary of the spilling technique may be determined based on a number of training vectors in the neighboring clusters. For example, the neural network training engine **186** may determine the boundary of the spilling technique based on a minimum number of training vectors in the neighboring clusters needed to train the speech model. In some implementations, the boundary of the spilling technique may be determined based on a number of training vectors near the boundary of the particular cluster. For example, the neural network training engine **186** may determine to enable the spilling technique if a number of training vectors within and near the boundary are below a minimum number.

[0051] FIG. **2** is a diagram **200** that illustrates an example of processing for speech recognition using neural networks. The operations discussed are described as being performed by the computing system **120**, but may be performed by other systems, including combinations of multiple computing systems.

[0052] The computing system **120** receives data about an audio signal **210** that includes speech to be recognized. The computing system **120** or another system then performs feature extraction on the audio signal **210**. For example, the computing system **120** analyzes different segments or analysis windows **220** of the audio signal **210**. The windows **220** are labeled $w_0 \ldots w_n$, and as illustrated, the windows **220** can overlap. For example, each window **220** may include 25 ms of the audio signal **210**, and a new window **220** may begin every 10 ms. For example, the window **220** labeled w_0 may represent the portion of audio signal **210** from a start time of 0 ms to an end time of 25 ms, and the next window **220**, labeled w_1, may represent the portion of audio signal **120** from a start time of 10 ms to an end time of 35 ms. In this manner, each window **220** includes 15 ms of the audio signal **210** that is included in the previous window **220**.

[0053] The computing system **120** performs a Fast Fourier Transform (FFT) on the audio in each window **220**. The results of the FFT are shown as time-frequency representations **230** of the audio in each window **220**. From the FFT data for a window **220**, the computing system **120** extracts features that are represented as an acoustic feature vector **240** for the window **220**. The acoustic features may be determined by binning according to filterbank energy coefficients, using a mel-frequency ceptral component (MFCC) transform, using a perceptual linear prediction (PLP) transform, or using other techniques. In some implementations, the logarithm of the energy in each of various bands of the FFT may be used to determine acoustic features.

[0054] The acoustic feature vectors **240**, labeled $v_1 \ldots v_n$, include values corresponding to each of multiple dimensions. As an example, each acoustic feature vector **240** may include a value for a PLP feature, a value for a first order temporal difference, and a value for a second order temporal difference, for each of 13 dimensions, for a total of 39 dimensions per acoustic feature vector **240**. Each acoustic feature vector **240** represents characteristics of the portion of the audio signal **210** within its corresponding window **220**.

[0055] The computing system **120** uses a neural network **270** that can serve as a speech model and indicate likelihoods that acoustic feature vectors **240** represent different phonetic units. The neural network **270** includes an input layer **271**, a number of hidden layers **272***a*-**272***c*, and an output layer **273**. The neural network **270** may receive acoustic feature vectors as input.

[0056] The neural network **270** has been trained to estimate likelihoods that the feature vectors represent particular phonetic units. For example, during training, input to the neural network **270** may be a combination of acoustic feature vectors corresponding to the utterance from which the acoustic feature vectors were derived. Many inputs combining acoustic feature vectors can be used to train the neural network **270**, and the various training data sets can include acoustic feature vectors derived from utterances from multiple speakers.

[0057] To recognize speech in the audio signal **210** using the neural network **270**, the computing system **120** inputs the different sets of acoustic feature vectors **240** at the input layer **271** of the neural network **270**. In the example, the neural network **270** receives a set **245** of acoustic feature vectors **240** that includes (i) an acoustic feature vector **240** for a window **220** of speech to be recognized and (ii) one or more acoustic feature vectors **240** that serve as context. The set **245** can include acoustic feature vectors **240** corresponding to a predefined number of consecutive windows **220**. In the example, the set **245** includes the acoustic feature vector **240** labeled v_1, which indicates features of audio in the window **220** labeled w_1. As context for this feature vector, the set **245** also includes the acoustic feature vectors **240** labeled v_0 and v_2, which respectively indicate features of audio in the windows **220** immediately preceding and immediately following the window **220** labeled w_1. The set **245** of acoustic feature vectors **240** are concatenated or stacked together to form the complete input to the neural network **270**.

[0058] At the output layer **273**, the neural network **270** indicates likelihoods that the speech in the window **220** under analysis (e.g., the window w_1 corresponding to acoustic feature vector v_1) corresponds to specific phonetic units. In some implementations, the phonetic units used may be phones or components of phones. In the example, the potential phones are referred to as $s_0 \ldots s_m$. The phones may be any of the various phones in speech, such as an "ah" phone, an "ae" phone, a "zh" phone, and so on. The phones $s_0 \ldots s_m$ may include all of the possible phones that may occur in the audio signal **210**, or fewer than all of the phones that may occur. Each phone can be divided into three acoustic states.

[0059] The output layer **273** provides predictions or probabilities of acoustic states given the data at the input layer **271**. The output layer **273** can provide a value, for each state of each phone, that indicates the probability that the acoustic feature vector v_1 represents the particular state of the particular phone. For example, for a first phone, s_0, the output layer **273** can provide a first value that indicates a probability $P(s_{0\_}1|X)$, which indicates a probability that the window w_1 includes the first acoustic state of the s_0 phone, given the set of input, X, provided at the input layer **271**. For a first phone, s_1, the output layer **273** can provide a second value indicating a probability $P(s_{0\_}2|X)$, indicating a probability that the window w_1 includes the second acoustic state of the s_0 phone, given the set of input, X, provided at the input layer **271**. Similar outputs can be provided for all states of all of the phones $s_0 \ldots s_m$.

[0060] The computing system **120** provides different sets of acoustic feature vectors **240** to the neural network **270** to receive predictions or probabilities of the acoustic states in the different windows **220**. The computing system **120** may apply a sliding window to the acoustic feature vectors **240** to select different sets. In the example, the sliding window has a size of three acoustic feature vectors **240**. For example, the computing system **120** may provide acoustic feature vectors **240** v_1, v_2, and v_3 as input to the neural network **270** to obtain output values regarding the speech in window w_2. The computing system **120** may provide acoustic feature vectors **240** v_2, v_3, and v_4 as input to the neural network **270** to obtain output values regarding the speech in the window w_3. In this manner, the computing system **120** may obtain outputs corresponding to each position of the sliding window across the acoustic feature vectors **240**.

[0061] The output of the neural network **270** is provided to a set of weighted finite state transducers that represents a language model composed with context information, a lexicon, and a grammar. The set of weighted finite state transducers can approximate an HMM. The weighted finite state transducers output a word lattice that the computing system **120** can use to determine a transcription for the audio signal.

[0062] As indicated above, each output from the neural network **270** can include a posterior probability P(state|X), representing a likelihood of a particular acoustic state given the current set of input data, X. In some implementations, the computing system **120** divides the posterior, P(state|X) by the prior, P(state), to generate a scaled posterior probability for each output. The resulting scaled posterior probabilities are then input to the weighted finite state transducers for further processing.

[0063] In the example of FIG. **2**, the sliding window of acoustic feature vectors **240** includes three acoustic feature vectors **240**. More or fewer acoustic feature vectors **240** may be provided in each set of input to the neural network **270**. For example, 2, 3, 5, 10, or another number of feature vectors for windows **220** before and after a central vector may be input simultaneously to the neural network **270**.

[0064] FIG. **3** is a diagram **300** that illustrates an example of processing to generate an i-vector. I-vectors are time-independent components that represent overall characteristics of an audio signal rather than characteristics at a specific segment of time within an utterance. I-vectors can summarize a variety of characteristics of audio that are independent of the phonetic units spoken, for example, information indicative of the identity of the speaker, the language spoken, recording channel properties, and noise characteristics.

[0065] The example of FIG. **3** illustrates processing to calculate an i-vector **380** for a sample utterance **310**. The computing system **120** accesses training data **320** that includes a number of utterances **321**. The training data **320** may include utterances **321** including speech from different speakers, utterances **321** having different background noise conditions, and utterances **321** having other differences. Each of the utterances **321** is represented as a set of acoustic feature vectors. Each of the acoustic feature vectors can be, for example, a 39-dimensional vector determined in the same manner that the acoustic feature vectors **240** are determined in the example of FIG. **2**.

[0066] The computing system **120** uses the utterances **321** to train a Gaussian mixture model (GMM) **330**. For example, the GMM **330** may include 1000 39-dimensional Gaussians **331**. The GMM **330** is trained using the acoustic feature vectors of the utterances **321** regardless of the phones or acoustic states that the acoustic feature vectors represent. As a result, acoustic feature vectors corresponding to different phones and acoustic states are used to train the GMM **330**. For example, all of the acoustic feature vectors from all of the utterances **321** in the training data **320** can be used to train the GMM **330**. In this respect, the GMM **330** is different from GMMs that are trained with only the acoustic feature vectors for a single phone or a single acoustic state.

[0067] When the sample utterance **310** is received, the computing system **120** determines acoustic feature vectors that describe the utterance **310**. The computing system **120** classifies the acoustic feature vectors of the utterance **310** using the GMM **330**. For example, the Gaussian **331** that corresponds to each acoustic feature vector of the sample utterance **310** may be identified. The computing system **120** then re-estimates the Gaussians **331** that are observed in the sample utterance **310**, illustrated as re-estimated Gaussians **335** shown in dashed lines. As an example, a set of one or more acoustic feature vectors of the sample utterance **310** may be classified as matching a particular Gaussian **331***a* from the GMM **330**. Based on this set of acoustic feature vectors, the computing system **120** calculates a re-estimated Gaussian **335***a* having a mean and/or variance different from the Gaussian **331***a*. Typically, only some of the Gaussians **331** in the GMM **330** are observed in the sample utterance **310** and re-estimated.

[0068] The computing system **120** then identifies differences between the Gaussians **331** and the corresponding re-estimated Gaussians **335**. For example, the computing system **120** may generate difference vectors that each indicate changes in parameters between a Gaussian **331** and its corresponding re-estimated Gaussian **335**. Since each of the Gaussians is 39-dimensional, each difference vector can have 39 values, where each value indicates a change in one of the 39 dimensions.

[0069] The computing system **120** concatenates or stacks the difference vectors to generate a supervector **340**. Because only some of the Gaussians **331** were observed and re-estimated, a value of zero (e.g., indicating no change from the original Gaussian **331**) is included in the supervector **340** for each the 39 dimensions of each Gaussian **331** that was not observed in the sample utterance **310**. For a GMM **330** having 1000 Gaussians that are each 39-dimensional, the supervector **340** would include 39,000 elements. In many instances, Gaussians **331** and the corresponding re-estimated Gaussians **335** differ only in their mean values. The supervector **340** can represent the differences between the mean values of the Gaussians **331** and the mean values of the corresponding re-estimated Gaussians **335**.

[0070] In addition to generating the supervector **340**, the computing system **120** also generates a count vector **345** for the utterance **310**. The values in the count vector **345** can represent 0^{th} order Baum-Welch statistics, referred to as counts or accumulated posteriors. The count vector **345** can indicate the relative importance of the Gaussians **331** in the GMM **330**. The count vector **345** includes a value for each Gaussian **331** in the GMM **330**. As a result, for a GMM **330** having 1000 Gaussians, the count vector **345** for the utterance **310** would include 1,000 elements. Each value in the vector **345** can be the sum of the posterior probabilities of the feature vectors of the utterance **310** with respect to a particular Gaussian **331**. For example, for a first Gaussian **331***a*, the posterior probability of each feature vector in the utterance **310** is computed (e.g., the probability of occurrence of the feature vector as indicated by the first Gaussian **331***a*). The sum of the posterior probabilities for the feature vectors in the utterance **310** is used as the value for the first Gaussian **331***a* in the count vector **345**. Posterior probabilities for the each feature vector in the utterance **310** can be calculated and summed for each of the other Gaussians **331** to complete the count vector **345**.

[0071] In the same manner that the supervector **340** and count vector **345** was generated for the sample utterance **310**, the computing system **120** generates a supervector **350** and a count vector **355** for each of the utterances **321** in the training data **320**. The GMM **330**, the supervectors **350**, and the count vectors **355** may be generated and stored before receiving the sample utterance **310**. Then, when the sample utterance **310** is received, the previously generated GMM **330**, supervectors **350**, and count vectors can be accessed from storage, which limits the amount of computation necessary to generate an i-vector for the sample utterance **310**.

[0072] The computing system **120** uses the supervectors **350** to create a factor analysis module **360**. The factor analysis module **360**, like the GMM **330** and the supervectors **350**, may be generated in advance of receiving the sample utterance **310**. The factor analysis module **360** can perform multivariate factor analysis to project a supervector to a lower-dimensional vector that represents particular factors of interest. For example, the factor analysis module may project a supervector of 39,000 elements to a vector of only a few thousand elements or only a few hundred elements.

[0073] The factor analysis module **360**, like the GMM **330**, is trained using a collection of utterances, which may be the utterances in the same training data **320** used to generate the GMM **330**. An adapted or re-estimated GMM may be determined for each of the i utterances [U_1, U_2, . . . , U_i] in the training data **320**, in the same manner that the re-estimated Gaussians **335** are determined for the utterance **310**. A supervector **350** [S_1, S_2, . . . , S_i] and count vector **355** [C_1, C_2, . . . , C_i] for each utterance [U_1, U_2, . . . , U_i] is also determined. Using the vector pairs [S_i, C_i] for each utterance, the factor analysis module **360** is trained to learn the common range of movement of the adapted or re-estimated GMMs for the utterances [U_1, U_2, . . . , U_i] relative to the general GMM **330**. Difference parameters between re-estimated GMMs and the GMM **330** are then constrained to move only over the identified common directions of movement in the space of the supervectors. Movement is limited to a manifold, and the variables that describe the position of the difference parameters over the manifold are denoted as i-vectors. As a result, the factor analysis module **360** learns a correspondence [S_i, C_i]-->i-vector, such that S_i/C_i=f(i-vector$_i$), where f( ) is a linear function f(x)=T*x and T is a matrix.

[0074] The computing system **120** inputs the supervector **340** and count vector **345** for the sample utterance **310** to the trained factor analysis module **360**. The output of the factor analysis module **360** is the i-vector **380**, which includes latent variables of multivariate factor analysis. The i-vector **380** represents time-independent characteristics of the sample utterance **310** rather than characteristics of a particular window or subset of windows within the sample utterance **310**. In some implementations, the i-vector **380** may include, for example, approximately 300 elements.

[0075] FIG. **4** is a flow diagram that illustrates an example of a process **400** for speech recognition using neural networks. The process **400** may be performed by data processing apparatus, such as the computing system **120** described above or another data processing apparatus.

[0076] The system receives data representing acoustic characteristics of a user's voice (**402**). The acoustic characteristics of the user may include a gender of the user, an accent of the user, a pitch of an utterance of the user, background noises around the user, or age group of the user. In some implementations, the data may be a vector.

[0077] The system selects a cluster for the data among a plurality of clusters (**404**). In some implementations, each cluster includes a plurality of vectors, where each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster. In some implementations, the system may determine a vector based on the data. In some other implementations, the data may be a vector. In some implementations, the vector may be an i-vector, where the neural network may be trained using the i-vectors in the cluster. In addition, the neural network may also be trained using one or more i-vectors in one or more neighboring clusters.

[0078] In some implementations, the plurality of clusters may be segmented based on vector distances to centroids of the clusters. The system may determine that a vector distance between the vector and the cluster is a shortest distance compared to vector distances between the vector and other clusters of the plurality of clusters. The system may select the cluster for the vector based on determining that the vector distance between the vector and the cluster is the shortest distance.

[0079] In some implementations, the system may receive data indicative of latent variables of multivariate factor analysis of an audio signal of the user. The system may select an updated cluster using the latent variables. For example, an i-vector indicates latent variables of multivariate factor analysis.

[0080] In some implementations, each cluster may include a distinct plurality of vectors, where each cluster is associated with a distinct speech model. In some other implementations, one or more clusters may include overlapping vectors.

[0081] In response to receiving one or more utterances of the user, the system provides the speech model associated with the cluster for transcribing the one or more utterances (**406**). In general, the selected speech model is implemented using a neural network trained to act as an acoustic model. For example, the speech model indicates likelihoods that feature vectors correspond to different speech units when the feature vectors and certain types of additional information are provided.

[0082] The speech model produces neural network outputs, which the system uses to identify a transcription for the audio signal. For example, the system may provide the neural network outputs to, for example, weighted finite state transducers that approximate a hidden Markov model (HMM), which may include information about a lexicon indicating the phonetic units of words, a grammar, and a language model that indicates likely sequences of words. The output of the HMM can be a word lattice from which the transcription may be derived.

[0083] In some implementations, the system may receive a feature vector that models audio characteristics of a portion of an utterance of the user. The system may determine, using the feature vector as an input, a candidate transcription for the utterance based on an output of the neural network of the speech model.

[0084] In some implementations, the system may provide the speech model to a computing device of the user. In some implementations, the system may provide the transcription to the client device. In some other implementations, the system may provide the transcription to another computing system for additional process, and provide the output of the additional process as the result. For example, the system may provide a transcription to a search engine to perform a search, and return the search results to the user.

[0085] FIG. **5** is a flow diagram that illustrates an example of a process **500** for training a neural network. The process **500** may be performed by data processing apparatus, such as the computing system **120** described above or another data processing apparatus.

[0086] The system obtains vectors in a cluster (**502**). In some implementations, the clustering of vectors may use hierarchical divisive clustering or k-Means. For example, given a predetermined number of cluster centroids in a vector space, the vector space may be segmented into the predetermined number of clusters, where each training vector is mapped to a respective cluster according to which centroid is the closest. In some implementations, the clustering of vectors may use GMM or other clustering techniques.

[0087] The system obtains vectors in neighboring clusters (**504**). In some implementations, in addition to the training vectors in a particular cluster, one or more training vectors in the neighboring clusters of the particular cluster may also be used to train the speech model of the particular cluster.

[0088] In some implementations, the system may determine the boundary of the spilling technique using a predetermined distance to the centroid of the particular cluster. In some implementations, the system may determine the boundary of the spilling technique based on a number of training vectors in the particular cluster. In some implementations, the system may determine the boundary of the spilling technique based on a number of training vectors in the neighboring clusters. In some implementations, the system may determine the boundary of the spilling technique based on a number of training vectors near the boundary of the particular cluster.

[0089] The system trains a neural network of a speech model of the cluster (**506**). The trained speech model is therefore optimized for the acoustic characteristics represented by the vectors in the cluster and the one or more vectors in the neighboring clusters.

[0090] FIG. **6** shows an example of a computing device **600** and a mobile computing device **650** that can be used to implement the techniques described here. The computing device **600** is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. The mobile computing device **650** is intended to represent various forms of mobile devices, such as personal digital assistants, cellular telephones, smart-phones, and other similar computing devices. The components shown here, their connections and relationships, and their functions, are meant to be examples only, and are not meant to be limiting.

[0091] The computing device **600** includes a processor **602**, a memory **604**, a storage device **606**, a high-speed interface **608** connecting to the memory **604** and multiple high-speed expansion ports **610**, and a low-speed interface **612** connecting to a low-speed expansion port **614** and the storage device **606**. Each of the processor **602**, the memory **604**, the storage device **506**, the high-speed interface **608**, the high-speed expansion ports **610**, and the low-speed interface **612**, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor **602** can process instructions for execution within the computing device **600**, including instructions stored in the memory **604** or on the storage device **606** to display graphical information for a GUI on an external input/output device, such as a display **616** coupled to the high-speed interface **608**. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0092] The memory **604** stores information within the computing device **600**. In some implementations, the memory **604** is a volatile memory unit or units. In some implementations, the memory **604** is a non-volatile memory unit or units. The memory **604** may also be another form of computer-readable medium, such as a magnetic or optical disk.

[0093] The storage device **606** is capable of providing mass storage for the computing device **600**. In some implementations, the storage device **606** may be or contain a computer-readable medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. Instructions can be stored in an information carrier. The instructions, when executed by one or more processing devices (for example, processor **602**), perform one or more methods, such as those described above. The instructions can also be stored by one or more storage devices such as computer- or machine-readable mediums (for example, the memory **604**, the storage device **606**, or memory on the processor **602**).

[0094] The high-speed interface **608** manages bandwidth-intensive operations for the computing device **600**, while the low-speed interface **612** manages lower bandwidth-intensive operations. Such allocation of functions is an example only. In some implementations, the high-speed interface **608** is coupled to the memory **604**, the display **616** (e.g., through a graphics processor or accelerator), and to the high-speed expansion ports **610**, which may accept various expansion cards (not shown). In the implementation, the low-speed interface **612** is coupled to the storage device **606** and the low-speed expansion port **614**. The low-speed expansion port **614**, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet) may be coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

[0095] The computing device **600** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server **620**, or multiple times in a group of such servers. In addition, it may be implemented in a personal computer such as a laptop computer **622**. It may also be implemented as part of a rack server system **624**. Alternatively, components from the computing device **600** may be combined with other components in a mobile device (not shown), such as a mobile computing device **650**. Each of such devices may contain one or more of the computing device **600** and the mobile computing device **650**, and an entire system may be made up of multiple computing devices communicating with each other.

[0096] The mobile computing device **650** includes a processor **652**, a memory **664**, an input/output device such as a display **654**, a communication interface **666**, and a transceiver **668**, among other components. The mobile computing device **650** may also be provided with a storage device, such as a micro-drive or other device, to provide additional storage. Each of the processor **652**, the memory **664**, the display **654**, the communication interface **666**, and the transceiver **668**, are interconnected using various buses, and several of the components may be mounted on a common motherboard or in other manners as appropriate.

[0097] The processor **652** can execute instructions within the mobile computing device **650**, including instructions stored in the memory **664**. The processor **652** may be implemented as a chipset of chips that include separate and multiple analog and digital processors. The processor **652** may provide, for example, for coordination of the other components of the mobile computing device **650**, such as control of user interfaces, applications run by the mobile computing device **650**, and wireless communication by the mobile computing device **650**.

[0098] The processor **652** may communicate with a user through a control interface **658** and a display interface **656** coupled to the display **654**. The display **654** may be, for example, a TFT (Thin-Film-Transistor Liquid Crystal Display) display or an OLED (Organic Light Emitting Diode) display, or other appropriate display technology. The display interface **656** may comprise appropriate circuitry for driving the display **654** to present graphical and other information to a user. The control interface **658** may receive commands from a user and convert them for submission to the processor **652**. In addition, an external interface **662** may provide communication with the processor **652**, so as to enable near area communication of the mobile computing device **650** with other devices. The external interface **662** may provide, for example, for wired communication in some implementations, or for wireless communication in other implementations, and multiple interfaces may also be used.

[0099] The memory **664** stores information within the mobile computing device **650**. The memory **664** can be implemented as one or more of a computer-readable medium or media, a volatile memory unit or units, or a non-volatile memory unit or units. An expansion memory **674** may also be provided and connected to the mobile computing device **650** through an expansion interface **672**, which may include, for example, a SIMM (Single In Line Memory Module) card interface. The expansion memory **674** may provide extra storage space for the mobile computing device **650**, or may also store applications or other information for the mobile computing device **650**. Specifically, the expansion memory **674** may include instructions to carry out or supplement the processes described above, and may include secure information also. Thus, for example, the expansion memory **674** may be provide as a security module for the mobile computing device **650**, and may be programmed with instructions that permit secure use of the mobile computing device **650**. In addition, secure applications may be provided via the SIMM cards, along with additional information, such as placing identifying information on the SIMM card in a non-hackable manner.

[0100] The memory may include, for example, flash memory and/or NVRAM memory (non-volatile random access memory), as discussed below. In some implementations, instructions are stored in an information carrier. that the instructions, when executed by one or more processing devices (for example, processor **652**), perform one or more methods, such as those described above. The instructions can also be stored by one or more storage devices, such as one or more computer- or machine-readable mediums (for example, the memory **664**, the expansion memory **674**, or memory on the processor **652**). In some implementations, the instructions can be received in a propagated signal, for example, over the transceiver **668** or the external interface **662**.

[0101] The mobile computing device **650** may communicate wirelessly through the communication interface **666**, which may include digital signal processing circuitry where necessary. The communication interface **666** may provide for communications under various modes or protocols, such as GSM voice calls (Global System for Mobile communications), SMS (Short Message Service), EMS (Enhanced Messaging Service), or MMS messaging (Multimedia Messaging Service), CDMA (code division multiple access), TDMA (time division multiple access), PDC (Personal Digital Cellular), WCDMA (Wideband Code Division Multiple Access), CDMA2000, or GPRS (General Packet Radio Service), among others. Such communication may occur, for example, through the transceiver **568** using a radio-frequency. In addition, short-range communication may occur, such as using a Bluetooth, WiFi, or other such transceiver (not shown). In addition, a GPS (Global Positioning System) receiver module **670** may provide additional navigation- and location-related wireless data to the mobile computing device **650**, which may be used as appropriate by applications running on the mobile computing device **650**.

[0102] The mobile computing device **650** may also communicate audibly using an audio codec **660**, which may receive spoken information from a user and convert it to usable digital information. The audio codec **660** may likewise generate audible sound for a user, such as through a speaker, e.g., in a handset of the mobile computing device **650**. Such sound may include sound from voice telephone calls, may include recorded sound (e.g., voice messages, music files, etc.) and may also include sound generated by applications operating on the mobile computing device **650**.

[0103] The mobile computing device **650** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a cellular telephone **680**. It may also be implemented as part of a smart-phone **682**, personal digital assistant, or other similar mobile device.

[0104] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the disclosure. For example, various forms of the flows shown above may be used, with steps re-ordered, added, or removed.

[0105] Embodiments and all of the functional operations described in this specification may be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments may be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer-readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable-medium may be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter affecting a machine-readable propagated signal, or a combination of one or more of them. The computer-readable medium may be a non-transitory computer-readable medium. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus may include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal that is generated to encode information for transmission to suitable receiver apparatus.

[0106] A computer program (also known as a program, software, software application, script, or code) may be written in any form of programming language, including compiled or interpreted languages, and it may be deployed in any form, including as a standalone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program may be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program may be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[0107] The processes and logic flows described in this specification may be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows may also be performed by, and apparatus may also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[0108] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer may be embedded in another device, e.g., a tablet computer, a mobile telephone, a personal digital assistant (PDA), a mobile audio player, a Global Positioning System (GPS) receiver, to name just a few. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory may be supplemented by, or incorporated in, special purpose logic circuitry.

[0109] To provide for interaction with a user, embodiments may be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user may provide input to the computer. Other kinds of devices may be used to provide for interaction with a user as well; for example, feedback provided to the user may be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user may be received in any form, including acoustic, speech, or tactile input.

[0110] Embodiments may be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface or a Web browser through which a user may interact with an implementation of the techniques disclosed, or any combination of one or more such back end, middleware, or front end components. The components of the system may be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

[0111] The computing system may include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0112] While this specification contains many specifics, these should not be construed as limitations, but rather as descriptions of features specific to particular embodiments. Certain features that are described in this specification in the context of separate embodiments may also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment may also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination may in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0113] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems may generally be integrated together in a single software product or packaged into multiple software products.

[0114] Thus, particular embodiments have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims may be performed in a different order and still achieve desirable results.

What is claimed is:

1. A method comprising:

receiving data representing acoustic characteristics of a user's voice;

selecting a cluster for the data from among a plurality of clusters, wherein each cluster includes a plurality of vectors, and wherein each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster; and

in response to receiving one or more utterances of the user, providing the speech model associated with the cluster for transcribing the one or more utterances.

2. The method of claim **1**,

wherein the plurality of clusters are segmented based on vector distances to centroids of the clusters, and

wherein selecting a cluster for the data comprises:

determining a vector based on the data;

determining that a vector distance between the vector and the cluster is a shortest distance compared to vector distances between the vector and other clusters of the plurality of clusters; and

based on determining that the vector distance between the vector and the cluster is the shortest distance, selecting the cluster for the vector.

3. The method of claim **1**, wherein selecting a cluster for the data further comprises:

receiving data indicative of latent variables of multivariate factor analysis of an audio signal of the user; and

selecting an updated cluster using the latent variables.

4. The method of claim **1**, comprising:

receiving a feature vector that models audio characteristics of a portion of an utterance of the user; and

determining, using the feature vector as an input, a candidate transcription for the utterance based on an output of the neural network of the speech model.

5. The method of claim **1**, wherein providing the speech model for transcribing the one or more utterances comprises providing the speech model to a computing device of the user.

6. The method of claim **1**, wherein the acoustic characteristics of the user includes a gender of the user, an accent of the user, a pitch of an utterance of the user, background noises around the user, or age group of the user.

7. The method of claim **1**, wherein the data is an i-vector, and wherein the neural network is trained using the i-vectors in the cluster and one or more i-vectors in one or more neighboring clusters.

8. The method of claim **1**, wherein each cluster includes a distinct plurality of vectors, and wherein each cluster is associated with a distinct speech model.

9. A computer-readable medium storing software having stored thereon instructions, which, when executed by one or more computers, cause the one or more computers to perform operations of:

receiving data representing acoustic characteristics of a user's voice;

selecting a cluster for the data from among a plurality of clusters, wherein each cluster includes a plurality of vectors, and wherein each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster; and

in response to receiving one or more utterances of the user, providing the speech model associated with the cluster for transcribing the one or more utterances.

10. The computer-readable medium of claim **9**,

wherein the plurality of clusters are segmented based on vector distances to centroids of the clusters, and

wherein selecting a cluster for the data comprises:

determining a vector based on the data;

determining that a vector distance between the vector and the cluster is a shortest distance compared to vector distances between the vector and other clusters of the plurality of clusters; and

based on determining that the vector distance between the vector and the cluster is the shortest distance, selecting the cluster for the vector.

11. The computer-readable medium of claim **9**, wherein selecting a cluster for the data further comprises:

receiving data indicative of latent variables of multivariate factor analysis of an audio signal of the user; and

selecting an updated cluster using the latent variables.

12. The computer-readable medium of claim **9**, wherein the operations comprise:

receiving a feature vector that models audio characteristics of a portion of an utterance of the user; and

determining, using the feature vector as an input, a candidate transcription for the utterance based on an output of the neural network of the speech model.

13. The computer-readable medium of claim **9**, wherein providing the speech model for transcribing the one or more utterances comprises providing the speech model to a computing device of the user.

14. The computer-readable medium of claim **9**, wherein the data is an i-vector, and wherein the neural network is trained using the i-vectors in the cluster and one or more i-vectors in one or more neighboring clusters.

15. A system comprising:

one or more processors and one or more computer storage media storing instructions that are operable, when executed by the one or more processors, to cause the one or more processors to perform operations comprising:

receiving data representing acoustic characteristics of a user's voice;

selecting a cluster for the data from among a plurality of clusters, wherein each cluster includes a plurality of vectors, and wherein each cluster is associated with a speech model trained by a neural network using at least one or more vectors of the plurality of vectors in the respective cluster; and

in response to receiving one or more utterances of the user, providing the speech model associated with the cluster for transcribing the one or more utterances.

16. The system of claim **15**,

wherein the plurality of clusters are segmented based on vector distances to centroids of the clusters, and

wherein selecting a cluster for the data comprises:

determining a vector based on the data;

determining that a vector distance between the vector and the cluster is a shortest distance compared to vector distances between the vector and other clusters of the plurality of clusters; and

based on determining that the vector distance between the vector and the cluster is the shortest distance, selecting the cluster for the vector.

17. The system of claim **15**, wherein selecting a cluster for the data further comprises:

receiving data indicative of latent variables of multivariate factor analysis of an audio signal of the user; and

selecting an updated cluster using the latent variables.

18. The system of claim **15**, wherein the operations comprise:

receiving a feature vector that models audio characteristics of a portion of an utterance of the user; and

determining, using the feature vector as an input, a candidate transcription for the utterance based on an output of the neural network of the speech model.

19. The system of claim **15**, wherein providing the speech model for transcribing the one or more utterances comprises providing the speech model to a computing device of the user.

20. The system of claim **15**, wherein the data is an i-vector, and wherein the neural network is trained using the i-vectors in the cluster and one or more i-vectors in one or more neighboring clusters.

* * * * *

C.4 Speaker Identification

TITLE	**Speaker Identification.**	
INVENTORS	Matthew Sharifi	Google Inc.
	Ignacio López Moreno	UAM / Google Inc.
	Ludwig Schmidt	Google Inc.
DATE	2014/10/24	
NUMBER	US 20150127342 A1	
APPLICATION	14/523,198	
ASSIGNEE	Google Inc.	

Summary: Methods, systems, and apparatus for performing speaker identification based on an fingerprint vector obtained from an spoken utterance. Hash values are determined for the fingerprint vector according to multiple different hash functions, and allowing for fast and accurate speaker detection.

Contributions: The candidate provided the necessary code to generate the i-vector systems; supervised the data collection from over one thousand YouTube videos; and supervised the generation and optimization of i-vector models.

US 20150127342A1

(19) **United States**

(12) **Patent Application Publication**

Sharifi et al.

(10) **Pub. No.: US 2015/0127342 A1**

(43) **Pub. Date: May 7, 2015**

(54) **SPEAKER IDENTIFICATION**

(71) Applicant: **Google Inc.**, Mountain View, CA (US)

(72) Inventors: **Matthew Sharifi**, Palo Alto, CA (US); **Ignacio Lopez Moreno**, New York, NY (US); **Ludwig Schmidt**, Cambridge, MA (US)

(21) Appl. No.: **14/523,198**

(22) Filed: **Oct. 24, 2014**

Related U.S. Application Data

(60) Provisional application No. 61/899,434, filed on Nov. 4, 2013.

Publication Classification

(51) **Int. Cl.**

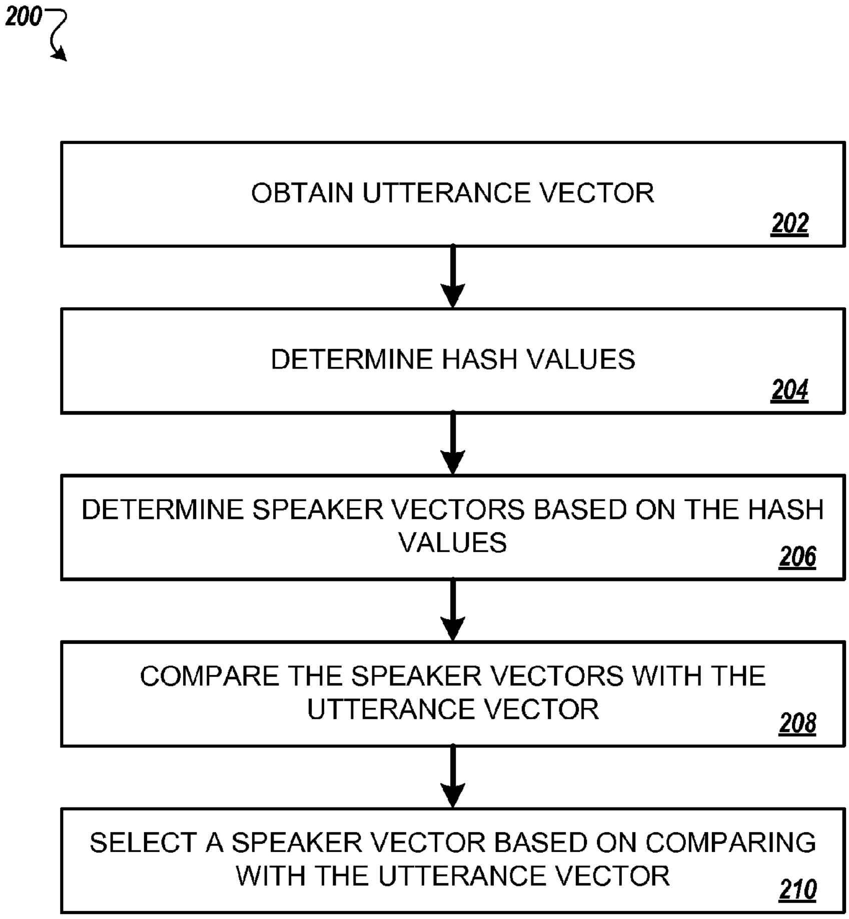
G10L 17/06 (2006.01)
G10L 17/02 (2006.01)

(52) **U.S. Cl.**

CPC *G10L 17/06* (2013.01); *G10L 17/02* (2013.01)

(57) **ABSTRACT**

Methods, systems, and apparatus, including computer programs encoded on a computer storage medium, for performing speaker identification. In some implementations, an utterance vector that is derived from an utterance is obtained. Hash values are determined for the utterance vector according to multiple different hash functions. A set of speaker vectors from a plurality of hash tables is determined using the hash values, where each speaker vector was derived from one or more utterances of a respective speaker. The speaker vectors in the set are compared with the utterance vector. A speaker vector is selected based on comparing the speaker vectors in the set with the utterance vector.


200
OBTAIN UTTERANCE VECTOR
202
DETERMINE HASH VALUES
204
DETERMINE SPEAKER VECTORS BASED ON THE HASH VALUES
206
COMPARE THE SPEAKER VECTORS WITH THE UTTERANCE VECTOR
208
SELECT A SPEAKER VECTOR BASED ON COMPARING WITH THE UTTERANCE VECTOR
210

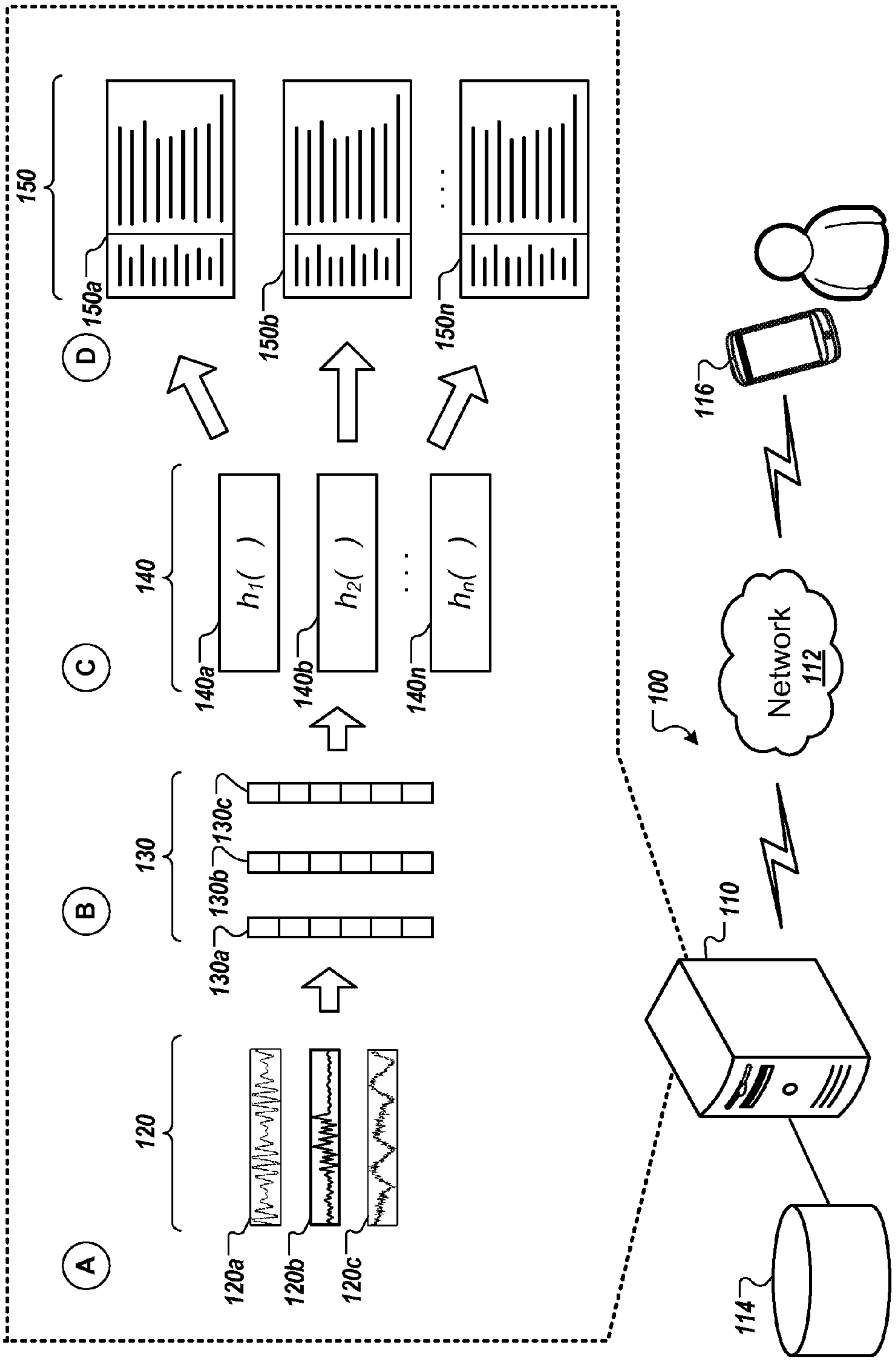
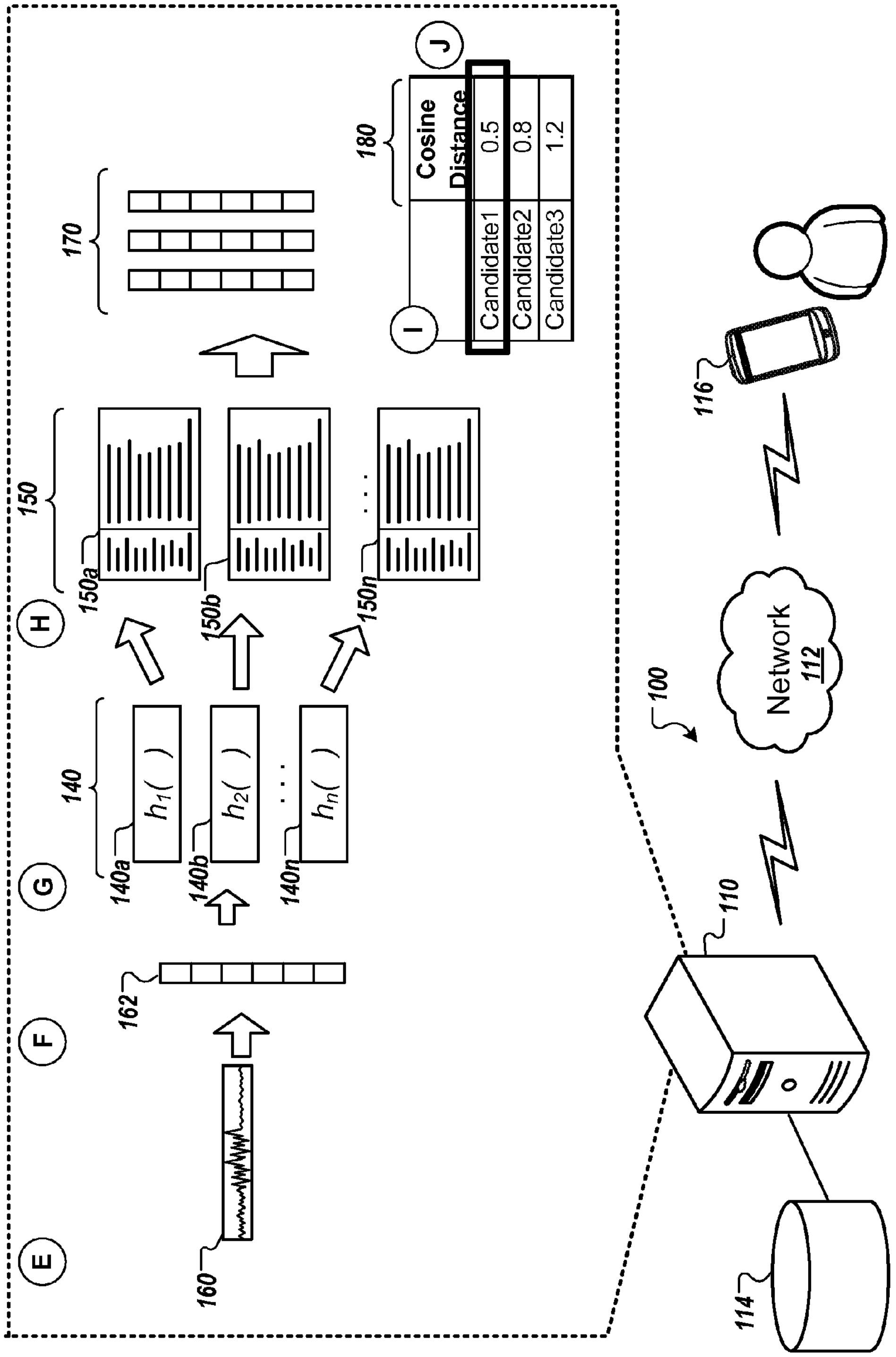

A
120
120a
120b
120c
B
130
130a
130b
130c
C
140
140a
$h_1(\ )$
140b
$h_2(\ )$
140n
$h_n(\ )$
D
150
150a
150b
150n
100
110
Network
112
114
116

FIG. 1A



	Cosine Distance
Candidate1	0.5
Candidate2	0.8
Candidate3	1.2

FIG. 1B

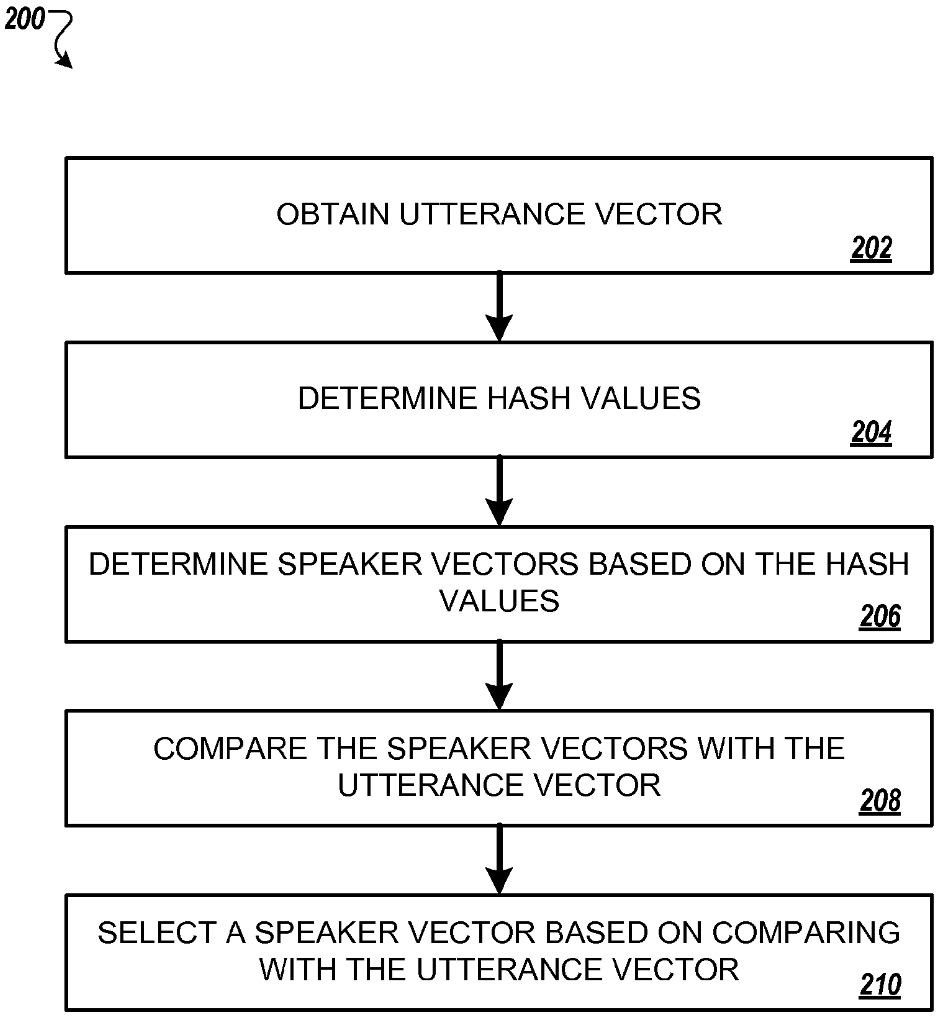

200
OBTAIN UTTERANCE VECTOR
202
DETERMINE HASH VALUES
204
DETERMINE SPEAKER VECTORS BASED ON THE HASH VALUES
206
COMPARE THE SPEAKER VECTORS WITH THE UTTERANCE VECTOR
208
SELECT A SPEAKER VECTOR BASED ON COMPARING WITH THE UTTERANCE VECTOR
210

FIG. 2

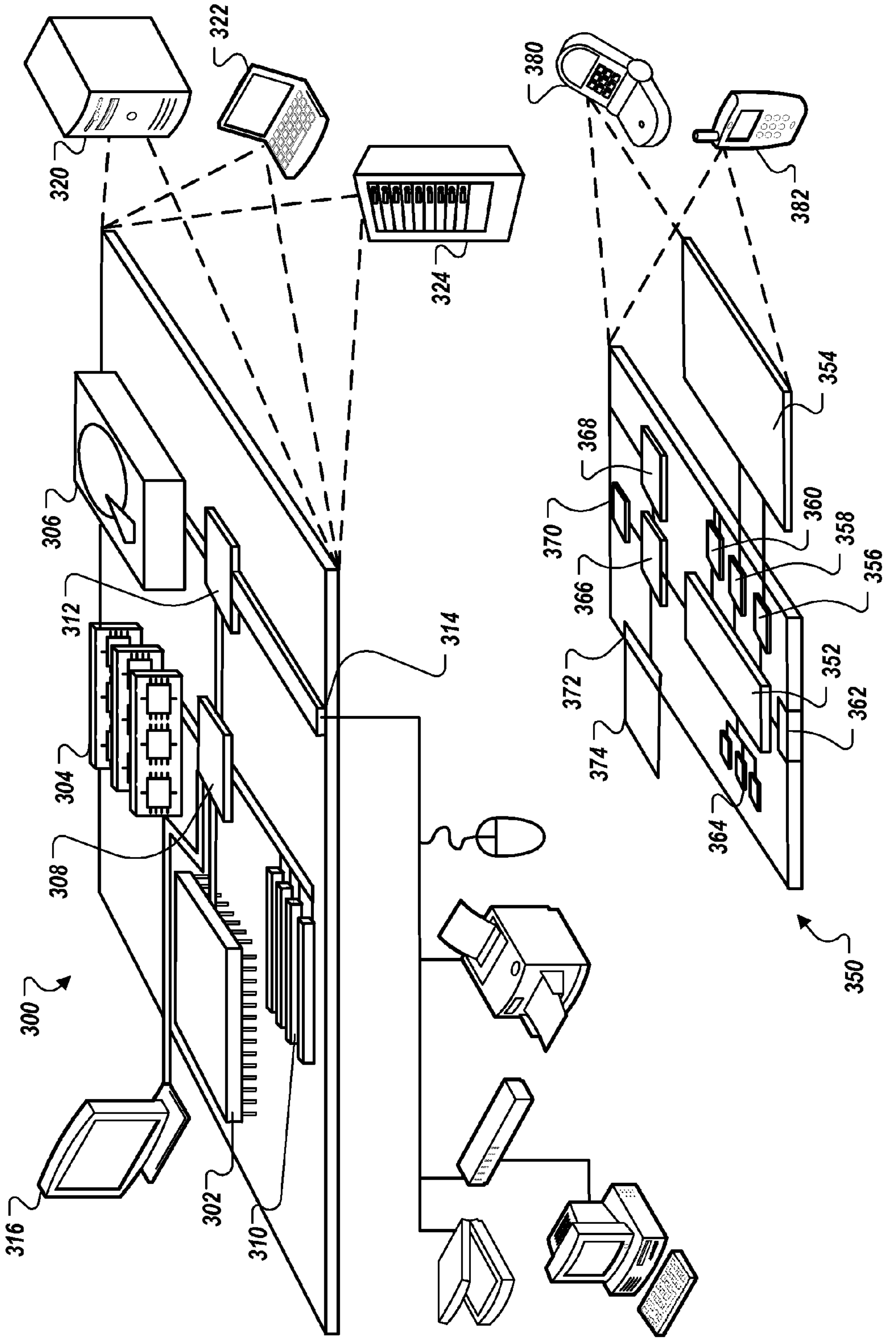

300
302
304
306
308
310
312
314
316
320
322
324
350
352
354
356
358
360
362
364
366
368
370
372
374
380
382

FIG. 3

SPEAKER IDENTIFICATION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Patent Application Ser. No. 61/899,434, filed Nov. 4, 2013. The entire contents of U.S. Provisional Patent Application Ser. No. 61/899,434 are hereby incorporated by reference.

TECHNICAL FIELD

[0002] This specification generally relates to speech recognition systems, and some implementations relate to determining or verifying an identity of a speaker.

BACKGROUND

[0003] Speaker identification generally involves determining a likely identity of a speaker based on speech samples from the speaker. Often, the more potential speaker identities a system has to select from among, the more computation and time is required to identify the correct speaker from among the potential speaker identities.

SUMMARY

[0004] Speaker identification is an important area of speech processing. In addition to identification accuracy, large-scale applications of speaker identification give rise to another challenge: fast search in the database of speakers. As discussed below, a system for speaker identification can use speaker vectors such as identity vectors ("i-vectors") and locality sensitive hashing, an algorithm for fast nearest-neighbor search in high dimensions. The connection between the two techniques is the cosine distance: the cosine distance may be used to compare vectors, and locality sensitive hashing allows approximation of the cosine distance in the retrieval procedure. In some implementations, an approach that combines the use of speaker vector techniques and locality sensitive hashing can be faster than a linear search while maintaining high identification accuracy.

[0005] In one general aspect, a method includes: obtaining an utterance vector that is derived from an utterance; determining hash values for the utterance vector according to multiple different hash functions; determining a set of speaker vectors from a plurality of hash tables using the hash values, each speaker vector being derived from one or more utterances of a respective speaker; comparing the speaker vectors in the set with the utterance vector; and selecting a speaker vector based on comparing the speaker vectors in the set with the utterance vector.

[0006] Implementations may include one or more of the following features. For example, the utterance vector includes obtaining an utterance i-vector for the utterance, the utterance i-vector comprising parameters determined using multivariate factor analysis of the utterance; and determining the set of speaker vectors from the plurality of hash tables using the hash values includes determining a set of speaker i-vectors from the plurality of hash tables, each speaker i-vector comprising parameters determined using multivariate factor analysis of one or more utterances of a respective speaker. Obtaining the utterance vector includes obtaining an utterance vector comprising parameters determined based on deep neural network activations that occur in response to information about the utterance being provided to the deep neural network; and determining the set of speaker vectors from the plurality of hash tables using the hash values includes determining a set of speaker vectors in which each speaker vector includes parameters determined based on deep neural network activations that occur in response to information about one or more utterances of a respective speaker being provided to the deep neural network.

[0007] Implementations may include one or more of the following features. For example, accessing data indicating associations between the speaker vectors and respective speakers; determining, based on the data indicating the associations between the speaker vectors and the respective speakers, a speaker identity corresponding to the selected speaker vector; and outputting data indicating the speaker identity. The method may include: identifying one or more media items that include utterances of a speaker corresponding to the selected speaker vector; and outputting data indicating the identified one or more media items. The method may include: determining that the selected speaker vector corresponds to a particular user; and based at least in part on the determining that the selected speaker vector corresponds to a particular user identity, authenticating the particular user.

[0008] Implementations may include one or more of the following features. For example, determining the hash values includes determining the hash values using one or more locality sensitive hash functions. Determining the hash values includes determining the hash values based on a position of the utterance vector with respect to different hyperplanes. Determining the hash values includes: determining first hash values for the utterance vector based on a first set of hash functions; and determining second hash values as different combinations of two or more of the first hash values; where determining the set of speaker vectors from the plurality of hash tables using the hash values includes determining the set of speaker vectors from the plurality of hash tables using the second hash values. Comparing the speaker vectors in the set with the utterance vector includes determining similarity scores that each indicate a degree of similarity of the utterance vector and one of the speaker vectors in the set; and selecting a speaker vector includes selecting the speaker vector that the similarity scores indicate is most similar to the utterance vector.

[0009] Implementations may include one or more of the following features. For example, determining similarity scores that each indicate a degree of similarity of the utterance vector and one of the speaker vectors in the set includes determining a cosine distance between the utterance vector and each of the speaker vectors in the set. Selecting the speaker vector that the similarity scores indicate is most similar to the utterance vector includes: identifying the smallest cosine distance from among the determined cosine distances; determining that the smallest cosine distance is less than a maximum distance threshold value; and based on determining that the smallest cosine distance is less than the maximum distance threshold value, selecting the speaker vector corresponding to the smallest cosine distance. Each of the speaker vectors corresponds to a different speaker; and the method further includes providing data indicating that the speaker corresponding to the selected speaker vector is the speaker of the utterance. The method may include obtaining multiple speaker vectors that each indicate characteristics of speech of a respective speaker; and, for each particular speaker vector of the multiple speaker vectors: determining hash values for the particular speaker vector according to each of the multiple different hash functions; and inserting the particular speaker vector into each of the plurality of hash tables based on the hash values. Obtaining multiple speaker vectors that each indicate characteristics of speech of a respective speaker includes: accessing a set of multiple video resources; and generating a speaker vector for each of the multiple video resources.

[0010] In another general aspect, a method includes: obtaining an utterance i-vector for an utterance; determining hash values for the utterance i-vector according to multiple different hash functions; determining a set of speaker i-vectors from a plurality of hash tables using the hash values; comparing the speaker i-vectors in the set with the utterance i-vector; and selecting a speaker i-vector based on comparing the speaker i-vectors in the set with the utterance i-vector.

[0011] In another general aspect, a method includes: obtaining multiple speaker i-vectors that each correspond to a different speaker; and for each of the multiple speaker i-vectors: (i) determining hash values for the speaker i-vector according to multiple different hash functions; and (ii) inserting the speaker i-vector into a plurality of hash tables based on the hash values.

[0012] Implementations may include one or more of the following features. For example, determining the hash values includes determining the hash values using locality sensitive hash functions. Determining the hash values includes determining the hash values based on a position of the speaker i-vector with respect to different hyperplanes. Determining hash values for the speaker i-vector according to multiple different hash functions includes: determining first hash values for the speaker i-vector based on a first set of hash functions; and determining second hash values as different combinations of two or more of the first hash values. Inserting the speaker i-vector into a plurality of hash tables based on the hash values includes inserting the speaker i-vector into each of the plurality of hash tables based on the second hash values.

[0013] Other implementations of these and other aspects include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods, encoded on computer storage devices. A system of one or more computers can be so configured by virtue of software, firmware, hardware, or a combination of them installed on the system that in operation cause the system to perform the actions. One or more computer programs can be so configured by virtue of having instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

[0014] The details of one or more implementations of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other potential features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIGS. **1**A and **1**B are diagrams illustrating a system for speaker identification.

[0016] FIG. **2** is a flow diagram illustrating a process for speaker identification.

[0017] FIG. **3** is a schematic diagram that shows an example of a computing device and a mobile computing device.

[0018] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0019] Speaker identification is one of the core areas of speech processing and acoustic modeling. Applications of speaker identification include authentication in security-critical systems, personalized speech recognition, and searching for speakers in large corpora. Due to the increasing amount of data—especially in web-scale applications—fast processing of speech data is becoming increasingly important. While the audio corpus can usually be pre-processed offline and in parallel, the retrieval procedure directly impacts user latency and needs to be executed as quickly as possible. The techniques discussed below can be used to perform fast, text-independent speaker identification in large corpora. Good identification performance (e.g., accuracy) can be maintained while significantly increasing the speed of retrieval. These techniques can include an i-vector-based speaker identification system combined with locality sensitive hashing (LSH), a powerful tool for approximate nearest neighbor search in high dimensions.

[0020] One application of the speaker identification system discussed below is searching web videos for a given speaker. Web-based videos are an example of the challenges of fast retrieval from a large data set. Each day, several years' worth of video are being uploaded to the web. Even if only a small fraction of this video is human speech, the amount of data to be processed for a single query is still very large.

[0021] The LSH-based retrieval approach discussed below can be faster than a linear search. In some instances, LSH-based search is several times faster, or even one or more orders of magnitude faster. At the same time, the identification accuracy may be close to or roughly equivalent to the more expensive algorithm. When LSH is used to approximate the cosine-distance of i-vectors, the techniques can be implemented with provable performance guarantees. Implementations of LSH-based similarity search may be used with large data sets, such as data sets of hundreds of thousands of items, millions of items, tens of millions of items, hundreds of millions of items, or a billion items or more. Thus, some implementations can have excellent scalability for large-scale data.

[0022] In some implementations, i-vector-based speaker identification techniques are used to identify a speaker. Robustly recognizing a speaker in spite of large inter-session variability such as background noise or different communication channels is one of the main limitations for speaker identification systems. Most techniques can be framed into the Factor Analysis (FA) paradigm, which aims to express the main "factors" contributing to the observed variability in a compact way. Another technique is the Joint Factor Analysis (JFA) formulation, where the acoustic space is divided into different subspaces. These subspaces independently model factors associated with the session variability and factors contributing to the interspeaker variability, e.g., a speaker corresponds to a vector in a low-dimensional subspace.

[0023] Another technique is the Total Variability Model (TVM), where all sources of variability (both speaker and session) are modeled together in a single low-dimensional space. In the TVM approach, the low-dimensional vector of latent factors for a given utterance is called an i-vector, and i-vectors are considered sufficient to represent the differences between various utterances. Now, speaker information and undesirable session effects are separated entirely in the i-vector domain. This separation step is typically carried out via classical Linear Discriminant Analysis (LDA) and/or Within

Class Covariance Normalization (WCCN). The cosine distance is typically used for the final comparison of a speaker reference i-vector with an utterance i-vector. Hereafter, the Total Variability system followed by the classical LDA and WCCN is referred to simply as Total Variability or TVM.

[0024] Probabilistic Linear Discriminant Analysis (PLDA) may be used to independently model the speaker and session factors in the i-vector space with a probabilistic framework. This method uses a hypothesis test for i-vector matching.

[0025] In some implementations, locality sensitive hashing may be used to facilitate data retrieval. The nearest neighbor problem is a core element in many search tasks: given a set of a points $\{x_1, \ldots, x_n\} \subseteq X$, a query point $q \in X$ and a distance function $d: X \times X \rightarrow \mathbb{R}^+$, find the point x_i minimizing $d(x_i, q)$. While efficient data structures for the exact problem in low-dimensional spaces are known, they have an exponential dependence on the dimension of X ("curse of dimensionality"). In order to circumvent this issue, LSH solves the c-approximate nearest-neighbor problem: instead of finding the nearest neighbor, it suffices to return a point x_i with $d(x_i, q) \leq c \min_{xj \in X} d(x_j, q)$. An approximate guarantee is still useful because the distance function d is often only an approximation of the ground truth. A particular strength of LSH is its provably sublinear running time, which also holds in practice.

[0026] In order to use LSH with a given distance function d, the algorithm relies on a family of locality sensitive hash functions. Intuitively, a hash function is locality sensitive if two elements that are close under d are more likely to collide. Locality sensitive hash functions can include any of various distance metrics, including, for example, the Euclidean distance, the Jaccard index, and the cosine similarity.

[0027] Given a family of locality sensitive hash functions, the LSH algorithm builds a set of hash tables and hashes all points x_i into each hash table. For each hash table, several locality sensitive hash functions may be concatenated to avoid unnecessary collisions, which can increase precision. Several hash tables can be maintained to increase the probability of finding a close neighbor, which may improve recall. Given a query point q, the system can look through all hash tables to find the x_i colliding with q and then return the best match.

[0028] In some implementations, LSH techniques may be used for speaker identification. For example, LSH may be used for storage and retrieval of i-vectors and data associated with speakers. In some implementations, LSH indexing techniques may be used to select one or more speaker

[0029] In some implementations, the speaker identification system can generate i-vectors and can retrieve similar i-vectors. In some implementations, given an utterance for which an i-vector should be generated, the utterance is first represented in terms of a large Gaussian mixture model (GMM), the so-called Universal Background Model (UBM), which can be parameterized with λ. Formally, let $\Theta = (o_1, \ldots, o_O)$ with $o_t \in \mathbb{R}^D$ be a sequence of spectral observations extracted from the utterance. Then the following accumulated and centered first order Baum-Welch statistics are computed as shown below in Equation 1:

$$N_m = \sum_{t=1}^{O} P(m \mid o_t, \lambda)$$

-continued

$$F_m = \sum_{t=1}^{O} P(m \mid o_t, \lambda)(o_t - \mu_m),$$

[0030] In Equation 1, m is the mean vector of mixture component m, m=1, . . . , C ranges over the mixture components of the UBM and P(m|o, λ) is the Gaussian occupation probability for mixture m and observation o. Hereafter, $F \in \mathbb{R}^{CD}$ is referred to as the vector containing the stacked statistics $F = (F_1^T, \ldots, F_C^T)^T$.

[0031] The i-vector associated with the sequence Θ is denoted as $x \in \mathbb{R}^d$. According to the TV model, the vector F is related to x via the rectangular low-rank matrix $T \in \mathbb{R}^{CD \times d}$ known as the TV subspace, as shown in Equation 2:

$$N^{-1}F = Tx,$$

[0032] In Equation 2, $T \in \mathbb{R}^{CD \times CD}$ is a diagonal matrix with C blocks of size D×D along the diagonal. Block m=1, . . . , C is the matrix $N_m I_{(D \times D)}$.

[0033] The constraints imposed on the distributions of P(x) and P(F|x) lead to a closed-form solution for P(x|F). The i-vector is the mean of this distribution and is given as shown below in Equation 3:

$$x = (I + T^T \Sigma^{-1} N T)^{-1} T^T \Sigma^{-1} F,$$

[0034] In Equation 3, $Z \in \mathbb{R}^{CD \times CD}$ is the covariance matrix of F. Therefore, in some implementations, i-vector extraction depends on the utterance data and the TV model parameters λ, T, and Σ.

[0035] If the true speaker labels for each training i-vector are known, the final speaker i-vector is normally obtained by averaging all i-vectors belonging to the same speaker. In an unsupervised setting, such as using web videos where speaker labels are not available for most of the utterances, the i-vector averaging step can be omitted, and instead the i-vectors of all utterances are kept.

[0036] In some implementations, locality-sensitive hashing may be used with i-vectors to identify speakers. One application of the system is enabling fast retrieval of speaker information. In the context of i-vector-based speaker identification, this may include, for example, for a given query i-vector, it may be desirable to find a similar vector in a previously computed set of i-vectors. In some instances, it may be desirable to find the best match, or a set of the closest matches. Since this task is an instance of the nearest neighbor problem introduced above, LSH may be used to enable fast retrieval.

[0037] One aspect of using LSH is the choice of distance function d. For i-vectors, the cosine distance may be expressed as shown below in Equation 4:

$$d(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

[0038] Cosine distance measurements, as shown in Equation 4, gives competitive performance for speaker identification. Since the cosine distance can also be approximated well with locality sensitive hash functions, the cosine distance may be used in the LSH algorithm. In particular, hash functions of the form shown below in Equation 5:

$$h_r(x) = \begin{cases} 1 & \text{if } x \cdot r \geq 0 \\ 0 & \text{if } x \cdot r < 0 \end{cases},$$

[0039] In Equation 5, r can be chosen as a random Gaussian vector. Geometrically, this hash function can be seen as hashing with a random hyperplane: r is perpendicular to the hyperplane and the result of the hash function indicates on which side of the hyperplane x lies. Since r has an isotropic distribution, the result is P[hr(x)=hr(y)]=1−Θ(x, y)/π, where (x, y) is the angle between vectors x and y.

[0040] The data structure has two main parameters: l, the number of hash tables, and k, the number of hyperplanes per hash table. Let $H_1, \ldots, H_l$ be the hash tables in the data structure. To reduce the number of hash function evaluations, the system can maintain $m \approx \sqrt{l}$ hash functions of length k/2 and use the

$$\binom{m}{2} \approx l$$

combinations as hash functions for the l hash tables. Formally, let $u_i(x)=(h_1^i(x), h_2^i(x), \ldots, h_{k/2}^i(x))$ for $i \in \{1, \ldots, m\}$ and $h_j^i(x)$ sampled as described above. Then the hash functions for the hash tables are $h_i(x)=(u_a(x), u_b(x))$ with $1 \leq a < b \leq m$, i.e., each h_i hashes an i-vector x to a string of k bits. Note that it is not necessary to store a full array with 2,000 entries for each hash table. Instead, the system can use standard hashing for large values of k.

[0041] For a given database of i-vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^D$, the LSH data structure may be initialized as follows: each i-vector x_i is hashed with each hash function h_j and then inserted at position $h_j(x_i)$ in hash table H_j. The overall time complexity of the initialization step is $O(ndk\sqrt{l}+nl)$.

[0042] Process 1, shown below, describes an example of a retrieval procedure.

Process 1 I-vector retrieval with LSH

```
1: function RETRIEVEIVECTOR(q)
2:     for i ← 1, . . . , m do
3:         Evaluate $u_i(q)$
4:     C ← { }                          ▷ Set of candidates
5:     for i ← 1, . . . , l do
6:         $C \leftarrow C \cup H_i[h_i(q)]$         ▷ Add candidates
7:     return $\arg\min_{x \in C} \frac{x \cdot q}{\|x\|\,\|q\|}$   ▷ Return best candidate
```

[0043] In Process 1, the evaluation of the m hash functions u_i in lines 2 and 3 can be efficiently implemented with a vector-matrix multiplication as follows. The normal vectors of the hyperplanes are stacked as rows into a matrix $U \in \mathbb{R}^{mk/2 \times d}$. The bits used in the hash functions are then given by

$$\frac{\mathrm{sgn}(Ux)+1}{2}.$$

The running time of the retrieval procedure is $O(dk\sqrt{l}+l+M)$, where M is the total number of matches found.

[0044] In some instances, speaker vectors are well clustered under well-matched recording conditions and so a small number of candidates in the hash tables may be sufficient to find a correct match. In other instances, the search for matches across data sets having widely varying recording conditions is more challenging and may require iterating over a larger set of candidates.

[0045] The techniques discussed herein can be used to allow fast retrieval method for speaker identification in large data sets. The system combines aspects of two approaches that interact via the cosine distance: locality sensitive hashing, which enables fast nearest-neighbor search, and i-vectors, which provide good identification accuracy. Moreover, LSH could also be very useful for other large-scale applications of i-vectors, such as clustering.

[0046] FIGS. **1**A and **1**B are diagrams illustrating examples of a system **100** for speaker identification. The system **100** includes a computing system **110**, a network **112**, data storage **114**, and a client device **116**. In FIG. **1**A, the computing system **110** processes audio information and uses LSH to store information about various different speakers. In FIG. **1**B, the computing system **110** uses the stored information to identify a speaker of an utterance.

[0047] In the system **100**, the functions performed by the computing system **110** can be performed by individual computer systems or can be distributed across multiple computer systems. The network **112** can be wired or wireless or a combination of both and can include the Internet. The client device **114** can be, for example, a desktop computer, laptop computer, a tablet computer, a wearable computer, a cellular phone, a smart phone, a music player, an e-book reader, a navigation system, or any other appropriate device.

[0048] Referring to FIG. **1**A, during stage (A), the speaker identification system **110** accesses audio data **120** that includes utterances of multiple speakers. For example, the audio data **120** may include various audio recordings or videos. In some implementations, the audio data **120** may include data from public data sources, such as videos available on the Internet. Other collections of audio data **120** may additionally or alternatively be used. In the illustrated example, the audio data **120** includes three different audio segments **120***a*-**120***c*, which each include an utterance of a different speaker. In some implementations, the speaker identification system can use audio data **120** including utterances of hundreds, thousands, or millions of different speakers.

[0049] During stage (B), the speaker identification system **110** obtains speaker vectors **130** for the various speakers whose utterances are included in the audio data **112**. A speaker vector **130** may be data that indicates characteristics of a speaker's voice. In some implementations, the speaker vectors are i-vectors. In other implementations, the speaker vectors are deep vectors ("d-vectors") determined using a deep neural network. Other data that indicates characteristics of a speaker's voice may be additionally or alternatively be used as a speaker vector.

[0050] As discussed above, an i-vector can be a set of parameters extracted or determined using factor analysis of one or more utterances. The i-vector may be data indicative of latent variables of multivariate factor analysis. For example, the i-vector may represent a projection into a low-dimensional, total variability subspace that was trained using factor analysis. The i-vector may indicate audio characteristics that are independent of the words spoken by the speaker. As a result, the i-vector may represent any of various factors indicative of the identity of the speaker, including, for example, characteristics of the speaker's speaking style, the speaker's gender, the speaker's age, the speaker's language, and/or the speaker's accent. The i-vector may be derived from only a portion of an utterance, an entire utterance, or multiple utterances of a speaker.

[0051] A d-vector can be as set of speaker-specific features extracted from a layer of a neural network. Various types of neural networks may be sued, including deep neural networks and recurrent neural networks. For example, the d-vector can be a set of activations at an output layer of a neural network, or a set of activations at a hidden layer of a neural network. In some implementations, individual d-vectors for an utterance or for different portions of an utterance may be used as speaker vectors. In some implementations, multiple vectors, e.g., for different portions of an utterance, or for multiple utterances of the same speaker, may be averaged together to form a d-vector.

[0052] To generate a d-vector, features extracted from an utterance can be provided as input to a trained neural network. In some implementations, the neural network has been trained to classify inputs of a defined set of speakers, but the speaker that spoke the utterance for which the d-vector is being generated is not required to be in the defined set of speakers. Speech features may be determined for individual speech frames, for example, segments of 10 ms to 50 ms of the utterance. The set of speech features for each frame are input to the neural network, sequentially, with each set of speech features producing a corresponding set of activations at predetermined layer of the neural network. For each input speech frame, a vector may be determined based on the activations at the predetermined output layer. These vectors may each be used as different d-vectors corresponding to the same speaker. In addition or as an alternative, the vectors for multiple speech frames may be averaged together. The averaged vector, like the d-vectors for individual frames, may be saved as a d-vector that indicates qualities characteristic of the speaker's voice.

[0053] A speaker vector **130***a*-**130***c* may be generated from each audio segment **120***a*-**120***c*. For example, when the audio data **120** includes various audio files or video files, a different speaker vector may be generated for each file, to indicate the characteristics of the utterances in that file. As a result, the speaker vector **130***a* indicates characteristics of the voice of the speaker of the utterances in the audio segment **120***a*, the speaker vector **130***b* indicates characteristics of the voice of the speaker of the utterances in the audio segment **120***b*, and so on. If a particular audio segment includes speech of multiple speakers, a separate speaker vector may be generated for each speaker. For example, a recording with speech of multiple speakers may be split into separate segments each including utterances of a single speaker. As a result, processing a particular recording may result in multiple speaker vectors that each correspond to a different speaker.

[0054] In some implementations, a speaker vector may be determined using multiple utterances or multiple recordings for a single speaker. For example, multiple recordings of a particular speaker may be identified, for example, based on labels assigned to the recordings. A different i-vector may be generated from each recording, and the i-vectors may be averaged or otherwise combined to form an average or composite speaker vector that represents characteristics of the particular speaker's voice.

[0055] In some implementations, a speaker whose utterances are recorded in an audio segment **120***a*-**120***c* may be unknown. When a name or other identifying information of the speaker is not known, the speaker vector may be associated with information indicating the source of the audio segment, such as a URL or other identifier for the audio recording or video that included the utterances from which the speaker vector was derived.

[0056] During stage (C), the speaker identification system **110** determines keys or index values for each speaker vector **130***a*-**130***c*. The keys or indexes can be hash values, for example, hash values determined using one or more LSH algorithms. For each speaker vector **130***a*-**130***c*, a hash value may be determined for each of multiple hash functions **140**. In the illustrated example, a set of n hash functions **140***a*-**140***n* are used. As a result, for each speaker vector **130***a*-**130***c, n* different hash values may be determined.

[0057] In some implementations, hash values for a speaker may be determined based on a position of a speaker vector with respect to different hyperplanes, such as random or pseudo-random hyperplanes. For example, a set of k-dimensional hyperplanes may be defined. A hash value may be determined by projecting the speaker vector onto the k-dimensional space and making a binary determination, for each hyperplane, which side of the hyperplane the projection falls. The comparison of the projection with each hyperplane may produce one bit of the hash value.

[0058] Hash values may also be determined by combining or concatenating hash values or other values derived from speaker vectors. Calculating shorter hash values may require less computation, but longer hash values may be desirable to allow for greater precision. In addition, longer hash values often lead to fewer hash collisions in a hash table. If three hash values A, B, and C are determined for a particular speaker vector, a set of longer hash values may be determined by concatenating different combinations of the hash values, for example, generating hash values AB, AC, and BC. In this example, the length of the hash values doubles and the only computation required is the concatenation operation.

[0059] During stage (D), the speaker identification system **100** stores associations between the keys or index values and corresponding speaker vectors **130***a*-**130***c*. Data identifying the speaker corresponding to a speaker vector, or a source of the audio data was used to determine the speaker vector, may be associated with the keys or index values. The associations may be stored in the data storage **114**, for example, in an associative array such as a hash table, for later use in identifying speakers.

[0060] In the illustrated example, speaker vectors **130***a*-**130***c* are stored in a plurality of hash tables **150**. A set of n hash tables **150***a*-**150***n* are used, with one hash table **150***a*-**150***n* corresponding to each hash function **140***a*-**140***n*. An entry for each speaker vector **130***a*-**130***c* is included in each hash table **150***a*-**150**. For example, in hash table **150***a*, the speaker vector **130***a* is associated with the hash value determined by applying the hash function **140***a* to the speaker vector **130***a*. In hash table **150***b*, the same speaker vector **130***a* is associated with the hash value determined by applying the hash function **140***b* to the speaker vector **130***a*. The pattern continues for all of the n hash tables, so that in hash table **150***n*, the speaker vector **130***a* is associated with the hash value determined by applying the hash function **140***n* to the speaker vector **130***a*. The other speaker vectors **130***b*, **130***c* and their associated speaker information are indexed in the hash tables **150**_a_-**150**_n_ in the same manner.

[0061] Associating the speaker vectors **130**_a_-**130**_c_ with different hash values in different hash tables can increase the likelihood of identifying appropriate speaker vectors during speaker identification. Some hash functions may indicate that certain items match or are similar, while other hash functions may not. As a result, it may be easier to identify speaker vectors as similar to a particular vector using multiple hash functions and multiple hash tables than when using a single hash function and a single hash table.

[0062] Referring to FIG. **1**B, the speaker identification system **110** uses the hash tables to identify a speaker of an utterance. During stage (E), the speaker identification system **110** receives data that identifies an utterance **160** of a speaker to be identified. The utterance **160** may be identified in or included in a request to identify the speaker of the utterance **160**, or a request for other content containing utterances of the same speaker. For example, the client device **116** may provide an audio recording that includes an utterance or a video that includes an utterance. As another example, the speaker identification system **110** may receive an identifier, such as a Uniform Resource Locator (URL) for a resource that includes a recording of an utterance. Having found a particular video of interest in a large video collection, a user may desire to find other videos in the collection that involve the same speaker or speakers as the particular video.

[0063] During stage (F), the speaker identification system **110** obtains an utterance vector **162** for the utterance **160**. The utterance vector **162** can indicate characteristics of the voice of the speaker of the utterance **160**. In particular, the utterance vector **162** may indicate characteristics of the utterance **160** independent of the specific words or sounds spoken. The utterance vector **162** can be determined in the same manner as the speaker vectors **130**_a_-**130**_c_ discussed above. In some implementations, when the speaker vectors **130**_a_-**130**_c_ are i-vectors, the utterance vector **162** may also be an i-vector. In other implementations, when the speaker vectors **130**_a_-**130**_c_ are d-vectors, the utterance vector may also be a d-vector. Other data that indicates characteristics of a speaker's voice may be additionally or alternatively be used as an utterance vector **162**.

[0064] During stage (G), the speaker identification system **110** determines keys or index values using the utterance vector **162**. For example, multiple hash values may be determined. Each of the hash functions **140**_a_-**140**_n_ may be applied to the utterance vector **162** to generate n hash values. As discussed above, the hash functions **140**_a_-**140**_n_ may use locality-sensitive algorithms so that similar inputs to the hash functions **140**_a_-**140**_n_ produce similar outputs. For example, the outputs may be clustered according to similarity of the inputs, or the differences between outputs may generally correspond to a cosine distance or other distance between the inputs.

[0065] During stage (H), the speaker identification system **110** uses the keys or index values to identify a set of speaker vectors **170**. The hash values that are generated based on the utterance vector **162** can be used to identify similar speaker vectors in the hash tables **150**_a_-**150**_n_. For example, the hash value produced by applying the hash function **140**_a_ to the utterance vector **162** is used to identify speaker vectors from the hash table **150**_a_. The hash value produced by applying the hash function **140**_b_ to the utterance vector **162** is used to identify speaker vectors from the hash table **150**_b_, and so on. Speaker vectors may be retrieved from any or all of the hash tables **150**_a_-**150**_n_, and different speaker vectors may be obtained from different hash tables **150**_a_-**150**_n_.

[0066] In some implementations, the set of candidate speaker vectors **170** may be selected using all of the hash tables **150**_a_-**150**_n_. For example, a lookup may be performed for each of the hash tables **150**_a_-**150**_n_, and in some instances, one or more speaker vectors may be selected from each of the hash tables **150**_a_-**150**_n_. In some implementations, each speaker vector that is identified from any of the hash tables **150**_a_-**150**_n_ is included in the set of candidate speaker vectors **170**. In some implementations, a speaker vector is included in the set of candidate speaker vectors **170** only if the same speaker vector is selected from multiple hash tables **150**_a_-**150**_n_. A minimum threshold number can be set. A speaker vector may be included in the set of candidate speaker vectors **170** only when the same speaker vector is selected from a number of the hash tables **150**_a_-**150**_n_ that is equal to or greater than the minimum threshold. For example, if the threshold is set to 2, speaker vectors retrieved from only a single hash table **150**_a_-**150**_n_ would not be included, but speaker vectors that are retrieved from each of two or more different hash tables **150**_a_-**150**_n_ would be included in the set of candidate speaker vectors **170**.

[0067] The set **170** is a group of candidate speaker vectors that have a degree of similarity to the utterance vector **162**. Since locality-sensitive hashing is used to select the speaker vectors in the set **170**, the set **170** may include the vectors that are most similar to the utterance vector **162**. For example, using LSH hash functions that approximate cosine distance measures, the speaker vectors may be those that have the lowest cosine distance relative to the utterance vector **162**. As a result, a very large set of speaker vectors can be quickly narrowed down to a subset that is most likely to be similar to the utterance vector **162**. Rather than comparing the utterance vector **162** to every speaker vector one by one, the utterance vector **162** may be compared with the speaker vectors included in the set **170** of candidate speaker vectors.

[0068] During stage (I), the speaker identification system **110** compares the utterance vector **162** with the speaker vectors in the set **170** of candidate speaker vectors. In some implementations, the speaker identification system **110** determines similarity scores **180** that indicate how similar each speaker vector in the set **170** is to the utterance vector **162**. For example, the speaker identification system **110** may determine a cosine distance between each speaker vector in the set **170** and the utterance vector.

[0069] During stage (J) the speaker identification system **110** selects a speaker vector from the set **170** based on the similarity scores **180**. For example, the speaker vector having the highest degree of similarity to the utterance vector **162** may be selected. In the example of FIG. **1**A, the speaker vector having the smallest cosine distance from the utterance vector **162** is selected. The speaker whose utterance was used to generate the selected speaker vector may be considered to be the most likely speaker of the utterance **160**. As a result, the name or other information about the speaker associated with the selected speaker vector may be accessed and provided, for example, to the client device **116** or another device. In some implementations, audio or video that was used to generate the selected speaker vector may be provided. As a result, data identifying media items including utterances of the same speaker that spoke the utterance **160** may be provided to the client device **116**.

[0070] In some implementations, a similarity threshold is set, and a speaker vector is selected only if the most similar speaker vector satisfies the similarity threshold. For example, a maximum cosine distance threshold may be set. Cosine distance scores that exceed the threshold may be considered too different to represent the same speaker as the utterance **162**, and may not be selected as likely being the same speaker. In some instances, the most similar speaker vector in the set **170** may not satisfy the similarity threshold, and so the speaker identification system **110** may determine that no speaker can be identified with a high degree of confidence. In some instances, multiple speaker vectors from the set **170** may satisfy the similarity threshold, indicating that multiple good candidate speakers have been identified. When each speaker vector represents a different media item, each speaker vector having a similarity score that satisfies the threshold may be likely to include speech of the same speaker. In some implementations, the speaker identification system **110** indicates multiple of or each of the speakers or media items corresponding to speaker vectors satisfying the similarity threshold.

[0071] FIG. **2** is a flow diagram illustrating an example of a process **200** for speaker identification. The process may be performed by one or more computing systems, such as the computing system **110** of FIGS. **1**A and **1**B.

[0072] An utterance vector for an utterance is obtained (**202**). In some implementations, the utterance vector is an utterance i-vector comprising parameters determined using multivariate factor analysis of the utterance. In some implementations, the utterance vector includes parameters determined based on deep neural network activations that occur in response to information about the utterance being provided to a trained deep neural network.

[0073] Hash values are determined for the utterance vector according to multiple different hash functions (**204**). In some implementations, the hash values are determined using one or more locality sensitive hash functions. For example, the hash values may be determined based on a position of the utterance i-vector with respect to different hyperplanes, such as random or pseudo-random hyperplanes.

[0074] In some implementations, hash values are determined as a combination of other hash values. For example, to determine the hash values, first hash values for the utterance vector can be determined based on a first set of hash functions. Second hash values may also be determined as different combinations of two or more of the first hash values. For example, the second hash values can each be a different permutation of the first hash values.

[0075] A set of speaker vectors is determined using the hash values (**206**). For example, a candidate set of speaker vectors can be selected from a plurality of hash tables. The set of speaker vectors determined using the hash values is a proper subset of the speaker vectors in the hash tables. For example, a set of candidate speaker vectors may be selected, where the number of candidate speaker vectors is an order of magnitude smaller, or several orders of magnitude smaller, than the total number of speaker vectors referenced by the hash tables. In some instances, the set includes only a few speaker vectors, for example, one or two of the speaker vectors selected from each hash table.

[0076] The hash values for the utterance vector may each correspond to a different one of the hash tables. The speaker vectors that correspond to the hash values of the utterance vector, if any exist, may be identified and included in the candidate set. In some implementations, one or more other speaker vectors may also be selected. For example, a system may select speaker vectors that are not associated with the exact hash values determined for the utterance vector but are nonetheless associated with similar or nearby hash values, e.g., hash values within a range or cluster about the hash values for the utterance.

[0077] In some implementations the set of speaker vectors is a set of speaker i-vectors where each speaker i-vector includes parameters determined using multivariate factor analysis of one or more utterances of a respective speaker. In some implementations the set of speaker vectors is a set of speaker vectors in which each speaker vector includes parameters determined based on deep neural network activations that occur in response to information about one or more utterances of a respective speaker being provided to the deep neural network.

[0078] The speaker vectors in the set are compared with the utterance vector (**208**). For example, similarity scores can be generated to indicate the degree of similarity of the utterance vector and each of the speaker vectors. In some implementations, a cosine distance is computed between the utterance vector and each of the speaker vectors in the candidate set.

[0079] A speaker vector is selected from the set based on comparing the speaker vectors in the set with the utterance vector (**210**). For example, the speaker vector that has the smallest cosine distance with respect to the utterance vector can be selected. The speaker corresponding to the selected speaker vector can be determined to be the likely speaker of the utterance.

[0080] In some implementations, data is accessed that indicates associations between speaker vectors and the respective speakers. Based on the data, a speaker identity corresponding to the selected speaker vector can be identified. Data indicating the speaker identity can then be output.

[0081] In some implementations, once the speaker of the utterance is identified, one or media items that include utterances of the identified speaker are also identified. Data indicating the media items that include utterances of the identified speaker may then be provided to a user. The system may indicate that the media includes speech or other content relating to the same person, and the system may indicate the identity of the person. For example, a user watching a particular video on the internet may request other videos involving the same person or people. A system may analyze the utterances in the particular video, identify the speaker, and provide links or other information about additional videos that also include speech of the same person or people.

[0082] In some implementations, the speaker identification techniques may be used for authentication. For example, it may be determined that the selected speaker vector corresponds to a particular user, and based on the determination, the particular user may be authenticated to, for example, a device, application, website, or service.

[0083] FIG. **3** shows an example of a computing device **300** and an example of a mobile computing device **350** that can be used to implement the techniques described above. The computing device **300** is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. The mobile computing device is intended to represent various forms of mobile devices, such as personal digital assistants, cellular telephones, smart-phones, and other similar computing devices.

The components shown here, their connections and relationships, and their functions, are meant to be exemplary only, and are not meant to limit implementations of the inventions described and/or claimed in this document.

[0084] The computing device **300** includes a processor **302**, a memory **304**, a storage device **306**, a high-speed interface **308** connecting to the memory **304** and multiple high-speed expansion ports **310**, and a low-speed interface **312** connecting to a low-speed expansion port **314** and the storage device **306**. Each of the processor **302**, the memory **304**, the storage device **306**, the high-speed interface **308**, the high-speed expansion ports **310**, and the low-speed interface **312**, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor **302** can process instructions for execution within the computing device **300**, including instructions stored in the memory **304** or on the storage device **306** to display graphical information for a GUI on an external input/output device, such as a display **316** coupled to the high-speed interface **308**. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0085] The memory **304** stores information within the computing device **300**. In some implementations, the memory **304** is a volatile memory unit or units. In some implementations, the memory **304** is a non-volatile memory unit or units. The memory **304** may also be another form of computer-readable medium, such as a magnetic or optical disk.

[0086] The storage device **306** is capable of providing mass storage for the computing device **300**. In some implementations, the storage device **306** may be or contain a computer-readable medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. A computer program product can be tangibly embodied in an information carrier. The computer program product may also contain instructions that, when executed, perform one or more methods, such as those described above. The computer program product can also be tangibly embodied in a computer- or machine-readable medium, such as the memory **304**, the storage device **306**, or memory on the processor **302**.

[0087] The high-speed interface **308** manages bandwidth-intensive operations for the computing device **300**, while the low-speed interface **312** manages lower bandwidth-intensive operations. Such allocation of functions is exemplary only. In some implementations, the high-speed interface **308** is coupled to the memory **304**, the display **316** (e.g., through a graphics processor or accelerator), and to the high-speed expansion ports **310**, which may accept various expansion cards (not shown). In the implementation, the low-speed interface **312** is coupled to the storage device **306** and the low-speed expansion port **314**. The low-speed expansion port **314**, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet) may be coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

[0088] The computing device **300** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server **320**, or multiple times in a group of such servers. In addition, it may be implemented in a personal computer such as a laptop computer **322**. It may also be implemented as part of a rack server system **324**. Alternatively, components from the computing device **300** may be combined with other components in a mobile device (not shown), such as a mobile computing device **350**. Each of such devices may contain one or more of the computing device **300** and the mobile computing device **350**, and an entire system may be made up of multiple computing devices communicating with each other.

[0089] The mobile computing device **350** includes a processor **352**, a memory **364**, an input/output device such as a display **354**, a communication interface **366**, and a transceiver **368**, among other components. The mobile computing device **350** may also be provided with a storage device, such as a micro-drive or other device, to provide additional storage. Each of the processor **352**, the memory **364**, the display **354**, the communication interface **366**, and the transceiver **368**, are interconnected using various buses, and several of the components may be mounted on a common motherboard or in other manners as appropriate.

[0090] The processor **352** can execute instructions within the mobile computing device **350**, including instructions stored in the memory **364**. The processor **352** may be implemented as a chipset of chips that include separate and multiple analog and digital processors. The processor **352** may provide, for example, for coordination of the other components of the mobile computing device **350**, such as control of user interfaces, applications run by the mobile computing device **350**, and wireless communication by the mobile computing device **350**.

[0091] The processor **352** may communicate with a user through a control interface **358** and a display interface **356** coupled to the display **354**. The display **354** may be, for example, a TFT (Thin-Film-Transistor Liquid Crystal Display) display or an OLED (Organic Light Emitting Diode) display, or other appropriate display technology. The display interface **356** may comprise appropriate circuitry for driving the display **354** to present graphical and other information to a user. The control interface **358** may receive commands from a user and convert them for submission to the processor **352**. In addition, an external interface **362** may provide communication with the processor **352**, so as to enable near area communication of the mobile computing device **350** with other devices. The external interface **362** may provide, for example, for wired communication in some implementations, or for wireless communication in other implementations, and multiple interfaces may also be used.

[0092] The memory **364** stores information within the mobile computing device **350**. The memory **364** can be implemented as one or more of a computer-readable medium or media, a volatile memory unit or units, or a non-volatile memory unit or units. An expansion memory **374** may also be provided and connected to the mobile computing device **350** through an expansion interface **372**, which may include, for example, a SIMM (Single In Line Memory Module) card interface. The expansion memory **374** may provide extra storage space for the mobile computing device **350**, or may also store applications or other information for the mobile computing device **350**. Specifically, the expansion memory **374** may include instructions to carry out or supplement the processes described above, and may include secure information also. Thus, for example, the expansion memory **374** may be provide as a security module for the mobile computing device **350**, and may be programmed with instructions that permit secure use of the mobile computing device **350**. In addition, secure applications may be provided via the SIMM cards, along with additional information, such as placing identifying information on the SIMM card in a non-hackable manner.

[0093] The memory may include, for example, flash memory and/or NVRAM memory (non-volatile random access memory), as discussed below. In some implementations, a computer program product is tangibly embodied in an information carrier. The computer program product contains instructions that, when executed, perform one or more methods, such as those described above. The computer program product can be a computer- or machine-readable medium, such as the memory **364**, the expansion memory **374**, or memory on the processor **352**. In some implementations, the computer program product can be received in a propagated signal, for example, over the transceiver **368** or the external interface **362**.

[0094] The mobile computing device **350** may communicate wirelessly through the communication interface **366**, which may include digital signal processing circuitry where necessary. The communication interface **366** may provide for communications under various modes or protocols, such as GSM voice calls (Global System for Mobile communications), SMS (Short Message Service), EMS (Enhanced Messaging Service), or MMS messaging (Multimedia Messaging Service), CDMA (code division multiple access), TDMA (time division multiple access), PDC (Personal Digital Cellular), WCDMA (Wideband Code Division Multiple Access), CDMA2000, or GPRS (General Packet Radio Service), among others. Such communication may occur, for example, through the transceiver **368** using a radio-frequency. In addition, short-range communication may occur, such as using a Bluetooth, Wi-Fi, or other such transceiver (not shown). In addition, a GPS (Global Positioning System) receiver module **370** may provide additional navigation- and location-related wireless data to the mobile computing device **350**, which may be used as appropriate by applications running on the mobile computing device **350**.

[0095] The mobile computing device **350** may also communicate audibly using an audio codec **360**, which may receive spoken information from a user and convert it to usable digital information. The audio codec **360** may likewise generate audible sound for a user, such as through a speaker, e.g., in a handset of the mobile computing device **350**. Such sound may include sound from voice telephone calls, may include recorded sound (e.g., voice messages, music files, etc.) and may also include sound generated by applications operating on the mobile computing device **350**.

[0096] The mobile computing device **350** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a cellular telephone **380**. It may also be implemented as part of a smart-phone **382**, personal digital assistant, tablet computer, wearable computer, or other similar mobile device.

[0097] Various implementations of the systems and techniques described here may be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations may include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

[0098] These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and may be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms "machine-readable medium" and "computer-readable medium" refer to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term "machine-readable signal" refers to any signal used to provide machine instructions and/or data to a programmable processor.

[0099] To provide for interaction with a user, the systems and techniques described here may be implemented on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user may provide input to the computer. Other kinds of devices may be used to provide for interaction with a user as well; for example, feedback provided to the user may be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user may be received in any form, including acoustic, speech, or tactile input.

[0100] The systems and techniques described here may be implemented in a computing system that includes a back end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front end component (e.g., a client computer having a graphical user interface or a Web browser through which a user may interact with an implementation of the systems and techniques described here), or any combination of such back end, middleware, or front end components. The components of the system may be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

[0101] The computing system may include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0102] A number of embodiments have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. In addition, the logic flows depicted in the figures do not require the particular order shown, or sequential order, to achieve desirable results. In addition, other steps may be provided, or steps may be eliminated, from the described flows, and other components may be added to, or removed from, the described systems. Accordingly, other embodiments are within the scope of the following claims.

What is claimed is:

1. A method comprising:

obtaining an utterance vector that is derived from an utterance;

determining hash values for the utterance vector according to multiple different hash functions;

determining a set of speaker vectors from a plurality of hash tables using the hash values, each speaker vector being derived from one or more utterances of a respective speaker;

comparing the speaker vectors in the set with the utterance vector;

selecting a speaker vector based on comparing the speaker vectors in the set with the utterance vector;

determining a speaker identity corresponding to the selected speaker vector; and

outputting data indicating the speaker identity.

2. The method of claim **1**, wherein obtaining the utterance vector comprises obtaining an utterance i-vector for the utterance, the utterance i-vector comprising parameters determined using multivariate factor analysis of the utterance; and

wherein determining the set of speaker vectors from the plurality of hash tables using the hash values comprises determining a set of speaker i-vectors from the plurality of hash tables, each speaker i-vector comprising parameters determined using multivariate factor analysis of one or more utterances of a respective speaker.

3. The method of claim **1**, wherein obtaining the utterance vector comprises obtaining an utterance vector comprising parameters determined based on deep neural network activations that occur in response to information about the utterance being provided to the deep neural network; and

wherein determining the set of speaker vectors from the plurality of hash tables using the hash values comprises determining a set of speaker vectors in which each speaker vector comprises parameters determined based on deep neural network activations that occur in response to information about one or more utterances of a respective speaker being provided to the deep neural network.

4. The method of claim **1**, further comprising:

identifying one or more media items that include utterances of a speaker corresponding to the selected speaker vector; and

outputting data indicating the identified one or more media items.

5. The method of claim **1**, further comprising:

determining that the selected speaker vector corresponds to a particular user; and

based at least in part on the determining that the selected speaker vector corresponds to a particular user identity, authenticating the particular user.

6. The method of claim **1**, wherein determining the hash values comprises determining the hash values using one or more locality sensitive hash functions.

7. The method of claim **1**, wherein determining the hash values comprises determining the hash values based on a position of the utterance vector with respect to different hyperplanes.

8. The method of claim **1**, wherein determining the hash values comprises:

determining first hash values for the utterance vector based on a first set of hash functions; and

determining second hash values as different combinations of two or more of the first hash values;

wherein determining the set of speaker vectors from the plurality of hash tables using the hash values comprises determining the set of speaker vectors from the plurality of hash tables using the second hash values.

9. The method of claim **1**, wherein comparing the speaker vectors in the set with the utterance vector comprises determining similarity scores that each indicate a degree of similarity of the utterance vector and one of the speaker vectors in the set; and

wherein selecting a speaker vector comprises selecting the speaker vector that the similarity scores indicate is most similar to the utterance vector.

10. The method of claim **9**, wherein determining similarity scores that each indicate a degree of similarity of the utterance vector and one of the speaker vectors in the set comprises determining a cosine distance between the utterance vector and each of the speaker vectors in the set.

11. The method of claim **10**, wherein selecting the speaker vector that the similarity scores indicate is most similar to the utterance vector comprises:

identifying the smallest cosine distance from among the determined cosine distances;

determining that the smallest cosine distance is less than a maximum distance threshold value; and

based on determining that the smallest cosine distance is less than the maximum distance threshold value, selecting the speaker vector corresponding to the smallest cosine distance.

12. The method of claim **1**, wherein each of the speaker vectors corresponds to a different speaker; and

wherein the method further comprises providing data indicating that the speaker corresponding to the selected speaker vector is the speaker of the utterance.

13. The method of claim **1**, further comprising:

obtaining multiple speaker vectors that each indicate characteristics of speech of a respective speaker; and

for each particular speaker vector of the multiple speaker vectors:

determining hash values for the particular speaker vector according to each of the multiple different hash functions; and

inserting the particular speaker vector into each of the plurality of hash tables based on the hash values.

14. The method of claim **1**, wherein obtaining multiple speaker vectors that each indicate characteristics of speech of a respective speaker comprises:

accessing a set of multiple video resources; and

generating a speaker vector for each of the multiple video resources.

15. A computer-readable storage device storing instructions that, when executed by one or more computers, cause the one or more computers to perform operations comprising:

obtaining an utterance vector that is derived from an utterance;

determining hash values for the utterance vector according to multiple different hash functions;

determining a set of speaker vectors from a plurality of hash tables using the hash values, each speaker vector being derived from one or more utterances of a respective speaker;

comparing the speaker vectors in the set with the utterance vector; and selecting a speaker vector based on comparing the speaker vectors in the set with the utterance vector.

16. The storage device of claim **15**, wherein obtaining the utterance vector comprises obtaining an utterance i-vector for the utterance, the utterance i-vector comprising parameters determined using multivariate factor analysis of the utterance; and

wherein determining the set of speaker vectors from the plurality of hash tables using the hash values comprises determining a set of speaker i-vectors from the plurality of hash tables, each speaker i-vector comprising parameters determined using multivariate factor analysis of one or more utterances of a respective speaker.

17. The storage device of claim **15**, wherein the operations further comprise:

accessing data indicating associations between the speaker vectors and respective speakers;

determining, based on the data indicating the associations between the speaker vectors and the respective speakers, a speaker identity corresponding to the selected speaker vector; and

outputting data indicating the speaker identity.

18. A system comprising:

one or more computers; and

a computer-readable storage device storing instructions that, when executed by the one or more computers, cause the one or more computers to perform operations comprising:

obtaining an utterance vector that is derived from an utterance;

determining hash values for the utterance vector according to multiple different hash functions;

determining a set of speaker vectors from a plurality of hash tables using the hash values, each speaker vector being derived from one or more utterances of a respective speaker;

comparing the speaker vectors in the set with the utterance vector; and

selecting a speaker vector based on comparing the speaker vectors in the set with the utterance vector.

19. The system of claim **18**, wherein obtaining the utterance vector comprises obtaining an utterance i-vector for the utterance, the utterance i-vector comprising parameters determined using multivariate factor analysis of the utterance; and

wherein determining the set of speaker vectors from the plurality of hash tables using the hash values comprises determining a set of speaker i-vectors from the plurality of hash tables, each speaker i-vector comprising parameters determined using multivariate factor analysis of one or more utterances of a respective speaker.

20. The system of claim **18**, wherein the operations further comprise:

accessing data indicating associations between the speaker vectors and respective speakers;

determining, based on the data indicating the associations between the speaker vectors and the respective speakers, a speaker identity corresponding to the selected speaker vector; and

outputting data indicating the speaker identity.

* * * * *

Colophon

This book was typeset by the author using LaTeX2e. The main body of the text was set using a 12-points Computer Modern Roman font. All graphics and images were included formatted as Encapsuled Postscript ($^{\text{TM}}$ Adobe Systems Incorporated). The final postscript output was converted to Portable Document Format (PDF) and printed.