LID-Senones and Their Statistics for Language Identification

Ma Jin, Yan Song, Ian McLoughlin, *Senior Member, IEEE*, and Li-Rong Dai

***Abstract*—Recent research on end-to-end training structures for language identification has raised the possibility that intermediate language-sensitive feature units exist which are analogous to phonetically sensitive senones in automatic speech recognition systems. Termed language identification (LID)-senones, the statistics derived from these feature units have been shown to be beneficial in discriminating between languages, particularly for short utterances. This paper examines the evidence for the existence of LID-senones before designing and evaluating LID systems based on low- and high-level statistics of LID-senones with both generative and discriminative models. For the standard NIST LRE 2009 task on 23 languages, LID-senone-based systems are shown to outperform state-of-the-art deep neural network/i-vector methods both when LID-senones are used directly for classification and when LID-senone statistics are used for i-vector formation.**

***Index Terms*—Language identification deep neural network i-vector LID-senones.**

I. Introduction

LANGUAGE identity is an inherent attribute of speech utterances, but not one that is easy to conceptualise in either acoustic or phonotactic terms. While the phonetic content of utterances is relatively easy to model using end-to-end schemes labelled at a frame level, it is difficult to map language identity at a frame level, and therefore difficult to label training data for supervised learning. Working backwards from a language identification (LID) label, it becomes necessary to find an effective utterance representation which is sensitive to language information, but is in turn derived from frame-level features extracted from a section of input speech.

i-vector based approaches [1], [2] currently achieve state-of-the-art performance for LID. An i-vector utterance representation is both compact and representative of the underlying utterance. However, i-vectors are learned in an unsupervised fashion without using language labels. They therefore rely upon techniques such as linear discriminant analysis (LDA) and within-class covariance normalization (WCCN) to build backend models for LID.

In the search for LID-sensitive features, we note that deep learning techniques such as deep neural networks (DNNs) [3], [4], have demonstrated their capabilities in several related fields to infer powerful feature extraction layers. DNNs have also been shown to improve i-vector performance in terms of acoustic modelling.

Song *et al.* [5], Richardson *et al.* [6] and Jiang *et al.* [7] each used deep bottleneck features (DBFs) for LID. These features were extracted by deep bottleneck networks (DBNs) that had been extensively pre-trained for automatic speech recognition (ASR) [8]. DBFs, used in this way, were shown to be inherently robust for different speakers, channels and background noise. Meanwhile Lei *et al.* [9], Kenny *et al.* [10] and Ferrer *et al.* [11] proposed systems which collected sufficient statistics using structured DNNs to form effective representations of the underlying phonemes or their states. These together provide very good evidence that DNNs are effective for both front-end frame-level feature extraction as well as back-end utterance-level modelling, assuming that sufficient good quality training data is available.

It is therefore clear that DBFs or senones, both derived from DBN acoustic modelling, are effective at representing language-based content, although hybrid combinations may be better still. However these feature extractors must be trained using phoneme or phoneme state labels rather than LID labels. This means that languages with similar phonetic content and statistics will be encoded with similar features, complicating the task of the back-end LID classifier. Instead, we believe it would be better if the feature extractors themselves produced features which are more language-discriminative.

In an aim to construct more task-aware features, recent LID research has tended towards building end-to-end schemes that are trained with LID labels. For example, Jiang *et al.* [12] showed that fine tuning pre-trained DNN parameters using an LID-specific corpus can improve performance. However his scheme used lattice-based optimisation to adjust parameters of final layers, and this fine tuning did not back propagate to earlier acoustic layers.

We also note that convolutional neural networks (CNNs) have demonstrated impressive front-end feature representation capabilities for large-scale speech and visual object recognition tasks [13], [14]. Multi-layer CNNs can be decomposed into a front-end stack of convolutional and pooling layers, followed

Manuscript received April 24, 2017; revised August 8, 2017 and October 14, 2017; accepted October 16, 2017. Date of publication October 23, 2017; date of current version December 5, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tomi Kinnunen. *(Corresponding author: Ian Vince McLoughlin.)*

M. Jin, Y. Song, and L-R. Dai are with the National Engineering Laboratory of Speech and Language Information Processing, University of Science and Technology of China, Hefei 230000, China (e-mail: jinma525@mail.ustc.edu.cn; songy@ustc.edu.cn; lrdai@ustc.edu.cn).

I. McLoughlin is with the School of Computing, University of Kent, Chatham ME4 4AG, U.K. (e-mail: ivm@kent.ac.uk).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

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/TASLP.2017.2766023

by a back end stack of fully-connected layers and then a classifier. The convolution-pooling layer pairs in the front-end stack can be thought of as feature extractors, whereas the final stack maps frame-level features into an utterance representation that is amenable to linear classification. For LID tasks, Lozano-Diez *et al.* [15] evaluated different CNN structures and demonstrated early results for end-to-end methods that performed comparably to shifted delta cepstra (SDC) i-vector systems.

Lopez-Moreno *et al.* [16] and Gonzalez-Dominguez *et al.* [17] also presented end-to-end schemes that performed well using large scale DNNs and long short-term memory (LSTM) recurrent neural networks (RNN). Interestingly, the evidence from these papers is that CNNs can perform well, but appear to have different strengths to DNNs. Both machine learning methods are able to learn useful and related, but quite different, inferences.

With this background, the authors set out to combine the strengths of DNN and CNN. Convinced by the need to incorporate end-to-end training, this could not be based around the current state-of-the-art i-vector methods. Instead, a DNN-CNN hybrid [18] was introduced, that combined powerful DBN-based feature extraction followed by CNN-based language modelling. The structure used a stack of convolutional layers to form language-discriminative units from DNN deep bottleneck feature inputs. These units, named LID-senones [18], were then classified by being averaged over a context time window using spatial pyramid pooling (SPP) [19] to form an utterance representation. The entire system was trained end-to-end with language labels. When tested on the six most highly confused language pairs from the NIST LRE 2009 corpus, this language identification network (*LID-net*) outperformed state-of-the-art methods for short duration utterances, and matched them on longer duration utterances [18]. While performance was good, *LID-net* had several weaknesses. First it did not make use of higher level statistics, basing its classification only on averaging the LID-senones. Secondly, it named the units LID-senones but did not explore this interesting idea further. Thirdly, *LID-net* was evaluated with only the 6 most confused LRE 2009 languages, only in terms of equal error rate (EER).

A. Contribution

Both the promise of the end-to-end DNN-CNN hybrid approach and the interesting concept of LID-senones, raised a number of questions that this paper sets out to explore. In particular we design and evaluate two new task aware structures that evaluate discriminative and generative versions of the LID-senone based classifier. Results will be presented below that show substantial performance improvement over other published systems on NIST LRE 2009 for all utterance durations. The specific contributions that this paper makes in each area are shown below;

1) The assumption that lower layers of a CNN, trained from bottleneck features with language labels, can extract LID-senones is examined further in Section IV to provide a basis to build classifiers for these features and their statistics.
2) The spatial pyramid pooling (SPP) methods of *LID-net* made use of averaging of LID-senones. Since i-vector based systems benefit from higher order statistics, we experiment with bilinear pooling to collect first and second order LID-senone posterior statistics in Section V-C.
3) Recent research has shown that outputs from different network layers can contain complementary information which is useful for classification in LID [20]. In Section II-A, we design a new structure called *LID-bnet* which explores the effectiveness of combining statistics across different layers.
4) Given that this make use of higher level statistics for classification, we explore the possibility of using the same information to form i-vectors in *LID-net-i* in Section V-D.

Our evaluations make use of all 23 languages of NIST LRE 2009 and report performance in standard terms using both equal error rate (EER) and $\mathrm{C}_{\mathrm{avg}}$, with separate results for 30s, 10s and 3s utterances. Performance evaluation of each system will be presented in Section V and discussed in Section VI. Section VII will conclude this paper and present suggestions for further work.

II. End-to-End LID-Net Structures

A. LID-Net System

LID-net is a task-aware neural network that spans from frame to utterance level, as shown in Fig. 1 [18]. The whole system includes a DNN front end followed by a deep CNN comprising multiple convolutional layer blocks, then a spatial pyramid pooling [19] (SPP) layer followed by a fully connected classification layer. The DNN front end acts as a feature transformer, trained to output task-specific language-sensitive features (we call these LID-features) from general acoustic input features, in this cases PLPs. Since DNN systems are known to be effective at a frame level [16],[17], these layers process acoustic features frame by frame.

Given that the lower DNN layers act as a feature extractor, the subsequent layers can be considered an utterance representation extractor. We know that the statistics of senones can be discriminative in languages [11], [21], but we aim for a similar feature that is task-aware, which we name LID-senones. These are derived by the CNN from LID-features and have statistics that are even more discriminative for LID – we will explore LID-senones further in Section IV-A.

The input to the DNN layers is current frame PLP, ΔPLP and $\Delta\Delta$PLP features over a $10 - 1 - 10$ context window. The DNN output is a compact vector representation (LID-features) which is stacked across a further sliding time window to form a two dimensional input image for the CNN. The CNN contains a deep stack of convolutional blocks (each consisting of a convolutional layer followed by a batch normalization ‘bnorm’ layer [22]), with SPP used to gather statistics of output predictions from across each variable length utterance into a vector which is then classified into a language identity by the final fully connected layer.

As mentioned previously, the input data to the SPP block (i.e., the output data from the end of the CNN convolutional layer stack) are termed LID-senones and will be explored further in Section IV. Unlike a normal pooling layer with a fixed

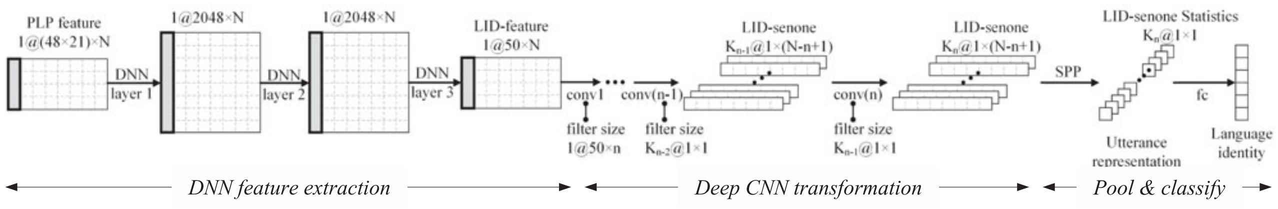

Fig. 1. End-to-end *LID-net* system spanning DNN feature extractor (left), through deep CNN layers (centre), pooling and fully connected classifier (right). The notation $K@1{\times}N$ means that there are K channels of 1 by N sized features.

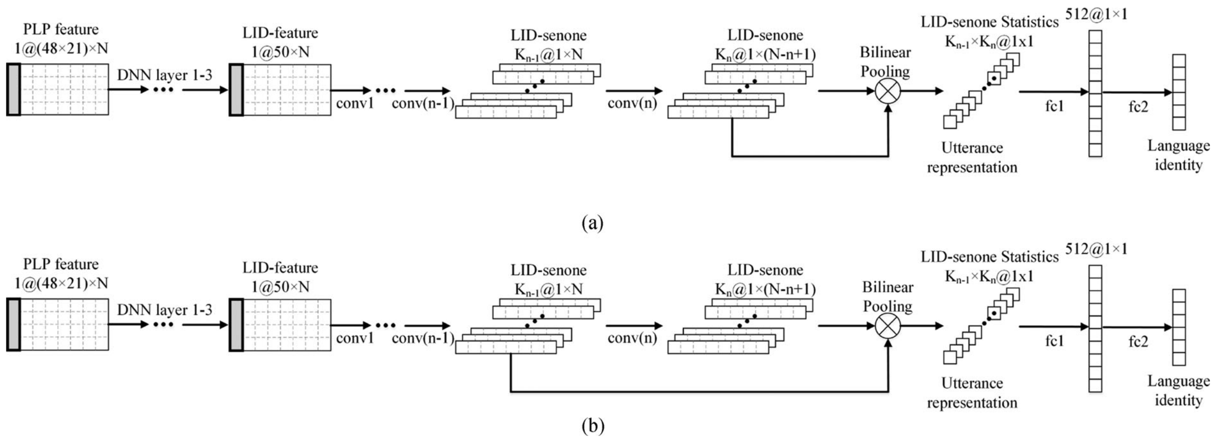

Fig. 2. *LID-bnet* showing second order pooling of statistics from (a) within the same convolutional block and (b) across different blocks. The notation $K@1 \times N$ means that there are K channels of 1 by N sized features.

pooling size yielding variable output dimension for variable input length, SPP maintains a fixed number of pools irrespective of the length of input. If input feature vector size is C channels of $M \times M$ images (i.e., $C@M \times M$), and we implement $[L, L]$ SPP, it means that the feature vector will be segmented into $L \times L$ parts with a pooling size of $\lceil \frac{M}{L} \rceil$ and stride is $\lfloor \frac{M}{L} \rfloor$, with every sub-spatial region executing max/average pooling. By this mechanism, C channels would give an $L \times L \times C$ feature matrix, which is reshaped to an $(L \times L \times C)@1 \times 1$ multi-dimensional matrix. The important fact is that the output feature size is independent of M.

LID-net evaluated in this paper uses a SPP pooling size of $[1, 1]$ since this was found to give consistently better performance than $[1, 2]$ SPP. The fixed size SPP output vector is then mapped to output language identity labels through a single fully connected (fc) layer. The detailed design and evaluation of *LID-net* will be given in Section V-B.

B. *LID-bnet System*

In *LID-net*, the SPP layer collects the mean of LID-senones for classification. As we will see in Section V, this slightly outperforms current state-of-the-art methods. But we note that it has been demonstrated that other LID techniques gain a substantial performance improvement by using higher order senone statistics [7]. It is thus reasonable to expect that higher order statistics from LID-senones will likewise be beneficial. Since SPP cannot collect higher order statistics, we adopted another pooling technique from the image processing domain, where two dimensional feature maps are common. The bilinear method of Lin *et al.* [23] works well on classification tasks, which is essentially the machine learning task we are performing in *LID-net*. Bilinear Pooling is computed in a spatial, rather than temporal domain, although our image is formed across a time context so does inherently incorporate temporal information. The input image to be pooled is the utterance-length stacked set of LID-senones that are output from the final convolutional layer. The fixed size output from bilinear pooling are LID-senone statistics over each utterance, and these are then classified by two fully connected neural network layers, into a language identity target. We name this *LID-bnet*, and present the block diagram in Fig. 2

We will show later in Section V-C how bilinear pooling can be adapted to yield equivalents of both first and second order Baum-Welch statistics based upon the choice of pooling inputs. We also note that bilinear pooling is where the pooling inputs can be taken from two CNN layers within the same convolutional block, or from layers in different convolutional blocks. These two options are shown in Fig. 2 with the top diagram showing bilinear pooling both from the final convolutional block, and the lower diagram performing cross-layer pooling from the final two convolutional blocks.

We will separately evaluate each pooling method and input arrangement in Section V-B.

III. RELATED WORK

A. *DNN/i-Vector System*

The baseline LID system used for comparison in this paper is the DNN/i-vector method [5], [9], [10] using statistics from bottleneck [7] features to form an i-vector [6], [11].

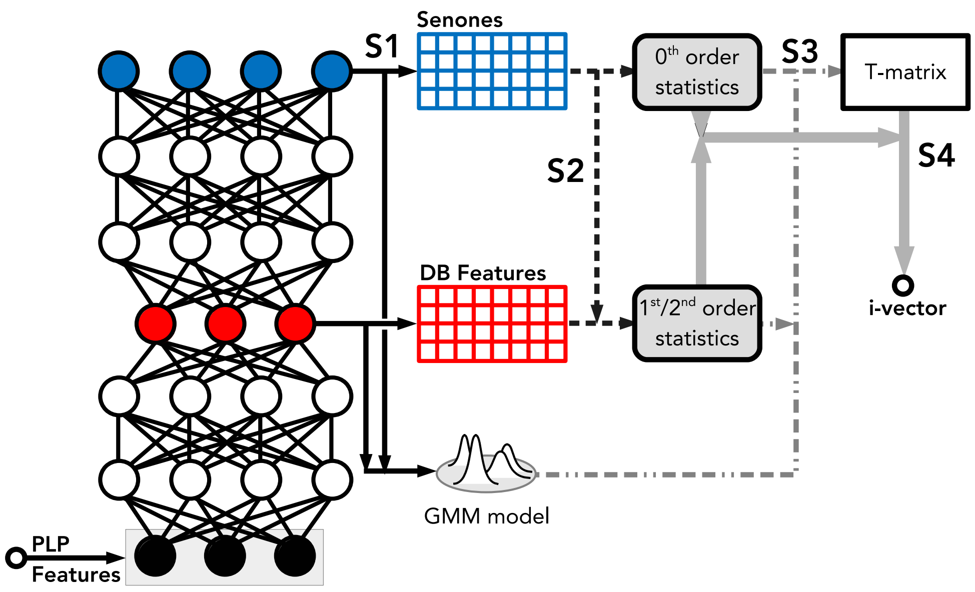

Fig. 3. Structure of the DNN/i-vector system based on senone statistics.

This well studied state-of-the-art technique requires a good quality related dataset, usually from the ASR domain, to train a DNN to extract phoneme states from either MFCC or PLP input representations, generally with context (e.g., SDC). This DNN, and its front-end layers before a bottleneck constriction, are then used directly in the LID system, typically after fine tuning with LID training data. For LID classification, the trained network is used for front-end feature extraction, with those features plus their statistics used for i-vector formation. This process is shown in Fig. 3. The DNN network input consists of PLP features (concatenated with ΔPLP and ΔΔPLP features over a $10 - 1 - 10$ context window), and the output are senones. Deep bottleneck features (DBF) are activations from an intermediate bottleneck layer which are used along with senone posteriors for i-vector formation as shown in the right half of Fig. 3, in four processing steps (S1 to S4). The full detail on the baseline system can be found in [7]. The front end DB feature extractor is trained using the SwitchBoard corpus.

In Fig. 3 the DNN stack is shown on the left, trained with senone labels. The lower half of that stack is essentially a feature extractor, producing DBFs from the constricted bottleneck layer. In DNN/i-vector these DBF features are used, along with the senone posteriors, for classification. However each of the new systems proposed in this paper, such as *LID-net*, use the same LID-features as input to their CNN stacks. For fair comparison, these extractors are identical: The weights of the trained DNN layers up to the bottleneck are transferred to the new systems, and used as fixed front-end LID-feature extractors.

IV. LID-Senone Analysis

A. What are LID-Senones?

Senones are known as the individual and distinct repeated units that make up a sequence of phones in spoken utterances. By breaking words into triphones or senones, it is hoped that the smaller units better encode how phones are affected by context. The important characteristics of senones is that they should be speaker, channel, language and noise insensitive but should be sensitive to phone state. For ASR, senone based techniques predominate in state-of-the-art systems [24] which make use of between about 3000 to 9000 senones for English. Senones have also performed well for speaker identification (SID) [9], and for LID tasks, even though both language and speaker information are unlikely to be completely encoded phonetically.

For LID systems, the use of better machine learning techniques has improved performance, but the rate of performance improvement may be slowing. This prompts the question as to whether more task-aware alternatives to ASR senones can be found for LID. Such representation units would need to be speaker, channel and noise insensitive, but language-sensitive. Thus they would need to be trained with LID data using language labels, rather than triphone state labels (ruling out the existing DNN/i-vector methods which cannot be trained end-to-end using language labels).

In the same way that DNN/i-vector systems build a feature transformer from input PLP features, trained with senone labels, to yield DBF features, *LID-net* trains a feature transformer from input DBF features (called LID-features), with LID labels, to yield intermediate output features. We have termed these LID-senones [18] and will now explore their characteristics further.

B. LID-Senone Statistics

To explore the statistics from LID-senones, the activations from the CNN convolutional block prior to the fully connected layer in *LID-net* were captured and analysed. We did the same for senones from the DNN/i-vector system for comparison, extracted directly from the trained DNN classifier with senone output (the left side of Fig. 3). In total, a selection of about 20 different short utterances were selected from the LID training corpus for Dari and for Farsi, two highly confused languages. The utterances had different phonetic content and were by

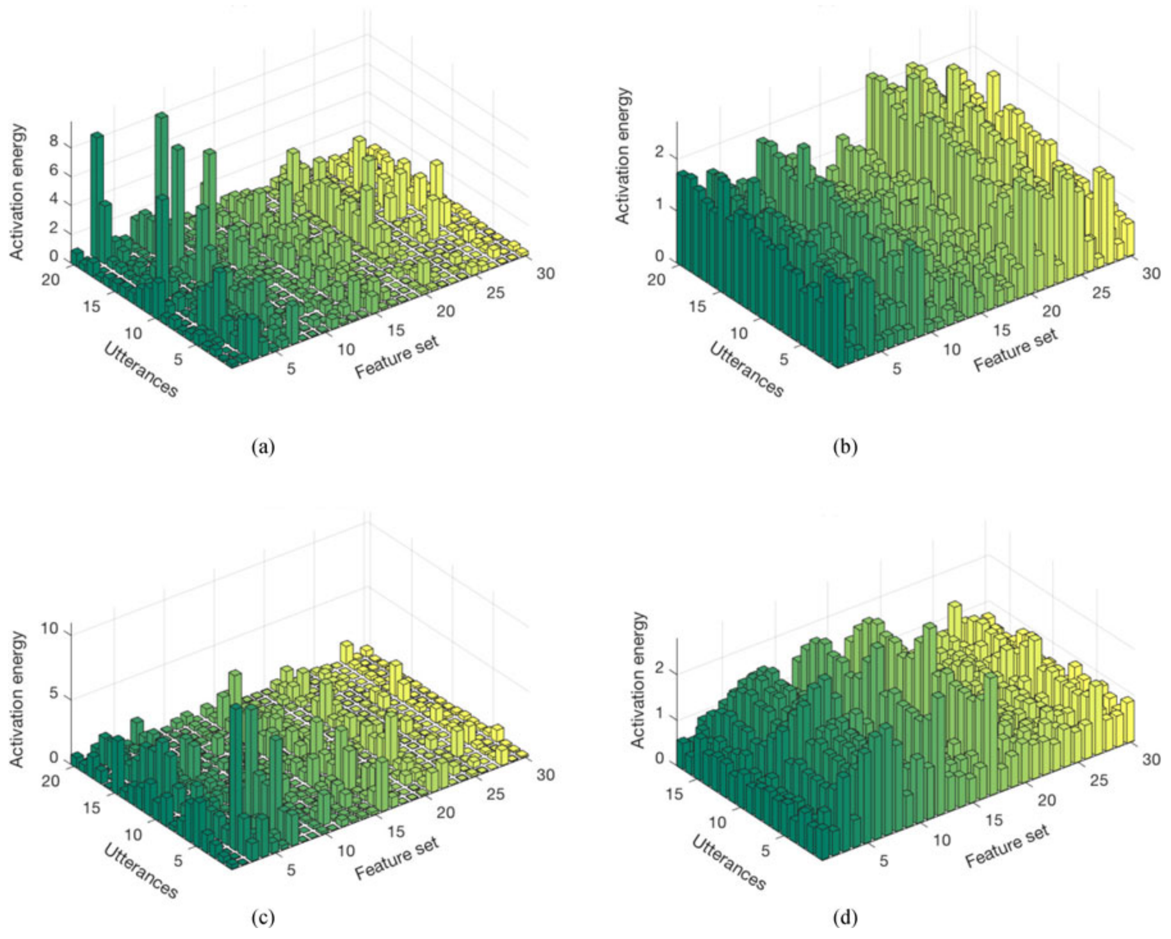

Fig. 4. 0th order activation statistics of 30 features for consecutive utterances for (a) Farsi senones, (b) Farsi LID-senones, (c) Dari senones and (d) Dari LID-senones.

different speakers. The 20 Dari utterances were applied to the DNN/i-vector system and to *LID-net*. Utterance-level features were captured from each system – these were the senones and LID-senones respectively. The process was repeated for the 20 Farsi utterances. The entire senone dimensionality is too large to conveniently visualise, so we selected a random subset of 30 senone and LID-senone activation statistics for display (using the same features in each case). These are plotted in Fig. 4. The Farsi utterances are plotted on top and the Dari utterances below, with senone statistics on the left and LID-senone statistics on the right. In each plot, statistics for the same sequence of 30 features is presented along the x-axis (left-to-right), for 20 difference utterances along the y-axis (front-to-back), The plot firstly shows that senone statistics are more sparse than those for LID-senones, but clearly shows strong repeated structural features in the LID-senone plots (running front-to-back). These indicate feature distribution patterns that are quite distinctly unique for each language. In fact, it should be noted that the plots were not normalized – if the LID-senone plots are normalized to their peak activation value, the intra-class similarity and inter-class difference is even more apparent.

Together, this provides some evidence that LID-senone statistics derived from *LID-net* are more discriminative for language than senones derived from an ASR acoustic model.

C. *LID-Senone Extraction*

Although the same type of LID-features are used in all systems, LID-senones are extracted from the output of the stack of CNN convolutional blocks in Figs. 1 and 2, and are subject to a number of parameter choices within the CNN.

For two dimensional input features, CNNs analyse small blocks across two axes in a stepwise fashion, usually with dimensions such as 3×3 or 5×5. This is sensible because pictures generally have very high spatial correlation across those scales. However this is not true in practice for LID-features, which tend to have very low spatial correlation.

To visualise this, the correlation matrix of a set of LID-features is plotted in Fig. 5. Diagonal values are high, whereas off-diagonal correlations are generally small or even negative, giving little evidence for the use of small convolutional kernels. We thus adopt a kernel size that covers the entire dimension of LID-features, namely $1@50 \times n$ where the dimension of LID-features is 50, and the size of the features after convolutional layer 1 is $K_1@1 \times (N - n + 1)$.

Given this input feature map, to ascertain how many frames of LID-features are suitable for constructing a LID-senone, we explored different LID-feature contexts, with a filter length of $50 \times n$ in the first convolutional layer, beginning with $n = 1$ and

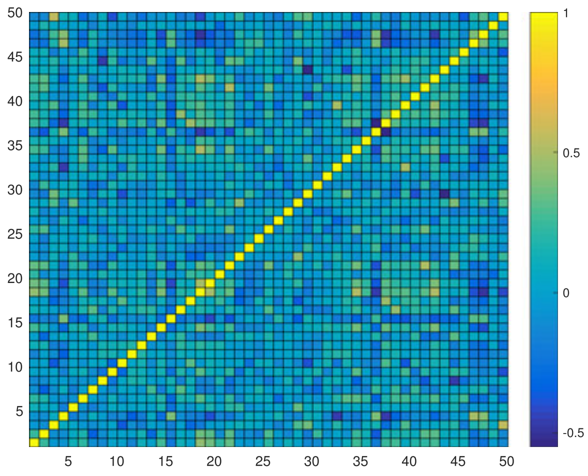

Fig. 5. Autocorrelation matrix for typical 50 dimensional LID-features.

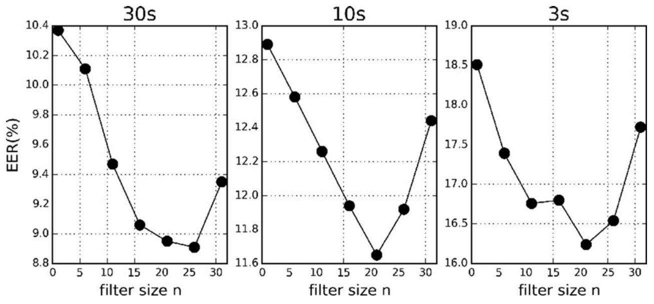

Fig. 6. Evaluations with different convolutional filter lengths.

testing with a step size of 5 up to 31. For efficient evaluation, we only used the 6 most confusable languages from the NIST LRE 2009 corpus (namely the language-pairs of Russian - Ukrainian, Hindi - Urdu and Dari - Farsi) to obtain raw EER performance for each filter size and utterance length.

Results are presented in Fig. 6 and show that the trend in all durations is that the best performance is obtained with a context size of around 21. We therefore adopted a $1@50 \times 21$ kernel size, which means a $10 - 1 - 10$ sliding window of LID-features is used to form the image input to the first convolutional layer.

Since, from Section III-A, we note that LID-features in the DNN/i-vector system are also derived from a fixed $10 - 1 - 10$ context window of PLP features (i.e., 21 acoustic frames), it means that a total span of 41 frames of acoustic features contribute to each LID-senone. This contrasts to evidence in [7] showing that just 21 frames of acoustic features are the optimal contribution for each senone for LID tasks (hence the use of $10 - 1 - 10$ context in DNN/i-vector).

Having sized the feature dimensionality and context of LID-senones, these features will form the basic input for all CNN systems evaluated in this paper.

D. Using LID-Senones to Form i-Vectors

If LID-senones can indeed represent basic units that are discriminative for languages in a similar way to senones for ASR tasks, this raises the question as to whether LID-senones can replace senones in other systems. For example, it may be possible to use total variability (TV) modelling to obtain a language discriminant utterance representation i-vector. This is effectively replacing the role of DBFs and senones in the DNN/i-vector system with LID-senones. In other words, LID-senones could replace the DBFs while LID-senones posteriors replace the senone posteriors. If such a hypothesis is reasonable, then the resulting system should be more language sensitive than the DNN/i-vector baseline. We therefore construct and evaluate just such a system, naming it *LID-net-i*, with a structure as shown in Fig. 7.

The detailed structure and evaluation of the *LID-net-i* system will be presented in Section V-D.

V. Implementation and Evaluation

A. Evaluation Methodology and Experimental Data

To evaluate the effectiveness of the proposed systems and compare against the baseline, we used the NIST LRE 2009 dataset and testing protocol. The 2009 language recognition evaluation (LRE) comprised 23 target languages, namely: Amharic, Bosnian, Cantonese, Creole, Croatian, Dari, English-American, English-Indian, Farsi, French, Georgian, Hausa, Hindi, Korean, Mandarin, Pashto, Portuguese, Russian, Spanish, Turkish, Ukrainian, Urdu and Vietnamese.

1) Training and Testing Data: Training utterances for each language were from Conversational Telephone Speech (CTS) and narrow band Voice of America (VOA) radio broadcasts. The CTS data used for training incorporated material from previous evaluations conducted by NIST (LRE 1996, LRE 2003, LRE 2005 and LRE 2007). The utterances were mainly collected from CallFriend, CallHome and Mixer databases The VOA partition data was from the NIST-provided datasets: VOA2 and VOA3.

The training data for each language is quite imbalanced, with recordings for languages like Mandarin and English-American exceeding 100 hours. By contrast there was less than 5 hours of English-Indian data. It should also be noted that some language data was collected from only one of the sources. Up to about 15 hours of each target language was selected for training and approximately 80 separate 30s duration utterances set aside for use as the development dataset (the remainder was used for training).

The test utterances are also divided into three duration groups, i.e., 30s, 10s and 3s, comprising 10,376, 10,427 and 10,375 speech utterances respectively.

2) Testing and Performance Evaluation: The language detection task in LRE 2009 is to determine whether a hypothesised language is spoken within a test segment or not [25]. Since test utterances vary in length, performance was evaluated separately for approximate utterance lengths of 30s, 10s and 3s, with the latter task being particularly difficult since short utterances could contain very little language-specific information and were often too short to yield meaningful statistics.

Several metrics can be used to assess LID performance (in terms of one-versus-all language detection). Classical equal error rate (EER) gives the performance when false acceptance and false rejection rates are equal. Average decision cost (C_{avg}) [25] is a measure of the cost of taking incorrect decisions over all

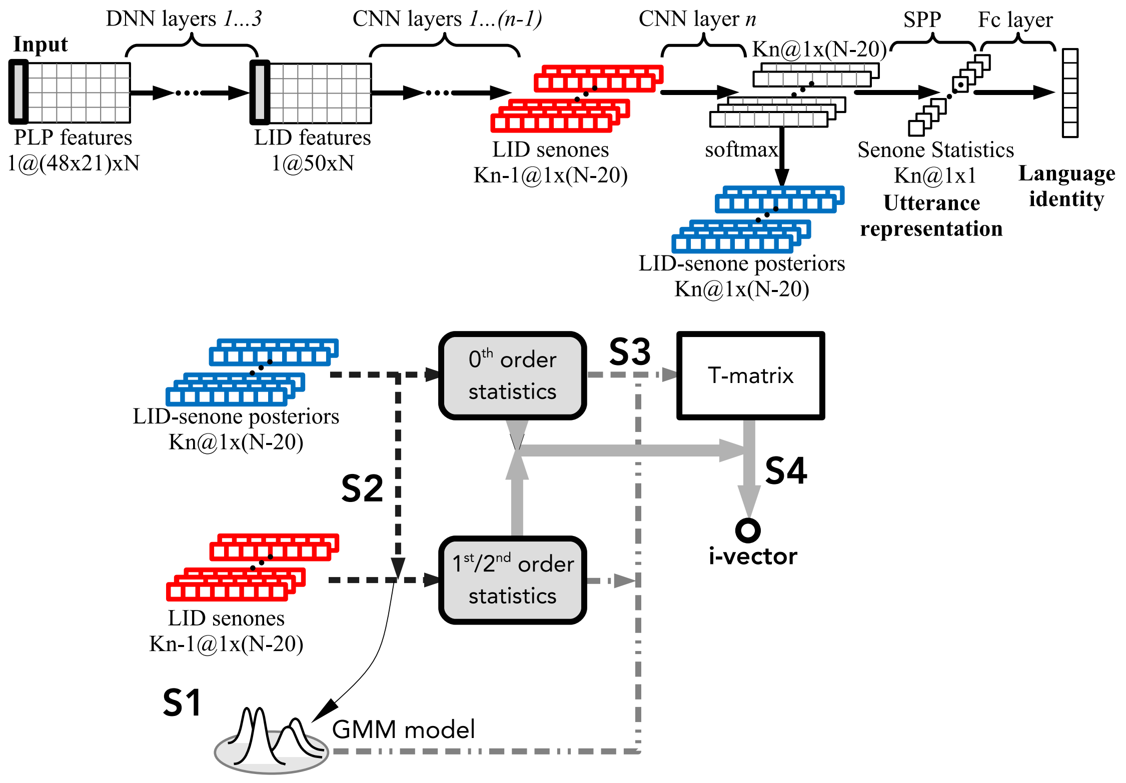

Fig. 7. Structure of the CNN-DNN *LID-net-i* i-vector based system using LID-senone statistics gathered from the upper end-to-end system to form a background model (steps S1 and S2) to create a total variability (T) matrix (S3), needed to obtain an i-vector from each utterance statistic (S4). Notation $K@1 \times N$ means that there are K channels of 1 by N sized features.

languages;

$$C_{\text{avg}} = \frac{1}{2N_T} \sum_{l \in L_T} \left\{ P_M(l) + \frac{1}{N_T - 1} \sum_{l' \in L_N} P_F(l, l') \right\}$$

where P_M and P_F are the miss (rejection) and false miss error rates respectively and L_T and L_N represent the target and non-target languages (N_T is the number of target languages). We also made use of DET curves [26], to visualise the range of possible system operating points.

For our evaluations, we trained and tested separate networks for the 30s, 10s and 3s tasks. Although they had the same structure, they contained different weights after training. As mentioned above, the 3s task was particularly difficult, whereas the nature of the 30s task made it much easier. However we made use of training data augmentation techniques to divide the longer utterances into shorter segments to help train the 3s and 10s networks. The consequence is that those networks enjoyed the benefits of far more training data than the 30s network, meaning that the performance of each network is constrained in different ways.

2) DNN Front End Layers Configuration and Training

All *LID-net*, *LID-bnet*, *LID-net-i* and DNN/i-vector systems utilised the same front end DNN layers. These began as a 6 layer DNN with central 50 dimensional bottleneck layer (i.e., $48 \times 21 - 2048 - 2048 - 50 - 2048 - 2048 - 3020$). We used Switchboard for training the DNN to give DBFs. After training, the layers up to the bottleneck layer were transferred to each new system and then trained end-to-end using LRE 2009 training data. The posteriors at the output of the CNN layers are denoted LID-senones. These layers are all listed in Table I for each system.

Given that the DNN layers were pre-trained and fixed to act as a feature extractor, the following CNN layers were then trained directly on LID data to learn a mapping from input LID-features to utterance-level LID labels. The configuration settings used for each neural network stack are given in Table I, with layers 10 and 11 for *LID-net* and *LID-bnet* shown separately, since the former network ended at layer 11 rather than layer 12. Each CNN block actually consisted of a convolutional layer followed by batch normalisation (‘bnorm’). A dropout of 0.5 was also applied after the first two convolutional blocks, and each system was trained with a learning rate that began at 0.05, and was then multiplied by 0.1 after every 5 epochs, terminating after 15 epochs. Batch size was variable (it was maximised to fit into GPU memory of 12GBytes and therefore was dependant upon the size of the training data set), ranging from 256 for the 3s *LID-net* system, 128 for 3s *LID-bnet*, 64 and 32 respectively for the 10s systems, 16 and 8 for the 30s systems. All pooling layers in these systems used average rather than max pooling.

B. Evaluation of LID-net System

As shown in Table I, *LID-net* had six convolutional layers, taking LID-features derived from the DNN plus context as input.[1]

[1]The notation used in this paper is that a size of $K@H \times W$ means there are K channels of height H features with width W.

TABLE I
NEURAL NETWORK CONFIGURATIONS FOR LID-NET AND LID-BNET SYSTEMS

Layer	Stage	Input Size	Configuration
1	DNN layer1	$(48 \times 21) \times N$	connections: $(48 \times 21) \times 2048$
2	DNN layer2	$2048 \times N$	connections: 2048×2048
3	DNN layer3	$2048 \times N$	connections: 2048×50
4	CNN block1	$1@50 \times N$	conv1 filter $1@50 \times 21$ / bnorm
5	CNN block2	$512@1 \times (N-20)$	conv2 filter $512@1 \times 1$ / bnorm
6	CNN block3	$512@1 \times (N-20)$	conv3 filter $512@1 \times 1$ / bnorm
7	CNN block4	$512@1 \times (N-20)$	conv4 filter $512@1 \times 1$ / bnorm
8	CNN block5	$512@1 \times (N-20)$	conv5 filter $512@1 \times 1$ / bnorm
9	CNN block6	$512@1 \times (N-20)$	conv6 filter $512@1 \times 1$ / bnorm
10 (LID-net)	SPP	$K@1 \times (N-20)$	SPP pooling size: $[1, N-20]$
10 (LID-bnet)	bilinear O2P	$K@1 \times (N-20)$	input from convs 5,6 and 6,6
11 (LID-net)	full connection	$K@1 \times 1$	connections: $K \times 23$ outputs
11 (LID-bnet)	full connection1	$(K \times K)@1 \times 1$	connections: $(K \times K) \times 512$
12 (LID-bnet)	full connection2	$512@1 \times 1$	connections: 512×23 outputs

TABLE II
COMPARISON BETWEEN LID-NET, DBF/I-VECTOR AND GMM/I-VECTOR FROM [11]

System	30s		10s		3s	
	EER	C_{avg}	EER	C_{avg}	EER	C_{avg}
GMM/i-vector [11]	-	1.15	-	1.98	-	6.82
DBF/i-vector	1.48	1.10	3.05	2.14	10.79	7.48
LID-net-32	1.49	1.05	2.74	1.54	7.67	6.02
LID-net-64	1.54	0.75	2.92	1.64	7.76	5.99
LID-net-128	1.55	0.91	2.89	2.00	7.58	6.15
LID-net-256	**1.46**	1.21	**2.66**	**1.46**	**7.57**	**5.05**
LID-net-512	1.50	**0.74**	2.81	1.49	7.79	6.64

Performance is given in EER (%) and C_{avg} (%).

It was trained end-to-end with LID labels, using SPP to collate utterance level statistics for classification by the single fully connected layer 11. We evaluated *LID-net* with a number of different pooling lengths from 32 to 512 in the feature map after CNN block 6 for comparison. Its performance is compared, in Table II, to the current state-of-the-art DBF/i-vector system. A notation such as LID-net-32 means that the feature map after CNN block 6 pooled across 32 features.

The benefits of end-to-end training are evident from the performance: it outperformed the baseline DBF/i-vector system over all scales, particularly for the most difficult 3s task. These results lend some confidence to the idea of greater language discriminating ability implicit in LID-senones. From these results, pooling over 256 features appears to be a reasonable choice, among tested configurations, for all scales.

C. Evaluation of LID-bnet System

The structure of *LID-bnet* is shown in Fig. 2 and its detailed configuration given in Table I. It shared the same three lowest layer structures and weights with *LID-net*, since DNN layers 1 to 3 were pre-trained in a bottleneck network on good and extensive ASR material. The next six CNN convolutional blocks had the same configuration as in *LID-net*, but different weights and thus extracted different LID-senones. This due to CNN layers being trained in a backwards direction from the output end of the network, and that end of the network differed substantially between *LID-net* and *LID-bnet*.

In *LID-net*, the frame-level LID-senones were pooled into an utterance representation by SPP after CNN block 6, however this only made use of simple averaging. As mentioned in Section II-A, in order to obtain higher order statistics it was necessary to replace SPP.

C. LID-bnet Configuration

The bilinear pooling function $\mathcal{B}$ can be written as $\boldsymbol{f_{A,B}} = \mathcal{B}(\boldsymbol{f_A}, \boldsymbol{f_B})$ [23]. Let f_A and f_B be the A and B feature maps derived from structured CNN layers A and B. These could be from the same CNN block or from different CNN blocks (shown in Fig. 2(a) and (b) respectively).

If $f_{A,B}$ is the output of bilinear pooling, the size of f_A and f_B are $(H \times W) \times K_A$ and $(H \times W) \times K_B$ respectively (reshaped from $K_A@H \times W$ and $K_B@H \times W$ respectively), implying both f_A and f_B must have the same feature dimension W and H to be compatible, but could subsequently pool over different length sets of features.

The bilinear pooling operation can be developed to $\boldsymbol{f_{A,B}} = \mathcal{B}(\boldsymbol{f_A}, \boldsymbol{f_B}) = \mathcal{P}(\boldsymbol{f_A}^T \cdot \boldsymbol{f_B})$. The feature map outputs are combined at each location using the matrix outer product, thus the shape of $(\boldsymbol{f_A}^T \cdot \boldsymbol{f_B})$ is simply $K_A \times K_B$. To obtain an utterance representation descriptor, the pooling function $\mathcal{P}$ aggregates the bilinear feature across the entire spatial domain of one combination at different scales.

Elements in feature map f_A are defined as $f_{Ad}(t)$ ($d = 1 \dots K_A$, $t = 1 \dots N$) and f_B, after the softmax operation, becomes γ, which can be viewed as the posterior of corresponding LID-senones at a frame level, with elements defined as $\gamma_k(t)$ ($k = 1 \dots K_B$, $t = 1 \dots N$). Using the feature map f_A and γ, the first order LID-senone statistics are,

$$\boldsymbol{f_{AB}(k)} = \frac{1}{N} \sum_{t=1}^{N} \gamma_k(t) \cdot \boldsymbol{f_A(t)} \tag{1}$$

If we instead use feature maps f_A and f_B directly, the bilinear pooling would model the second order LID-senone statistics,

$$\boldsymbol{f_{AB}} = \frac{1}{N} \boldsymbol{f_A}^T \cdot \boldsymbol{f_B} \tag{2}$$

TABLE III
PERFORMANCE OF CROSS- AND SAME-LAYER LID-BNET ON 3S UTTERANCES FOR DIFFERENT POOLING LENGTHS K

K	Cross-layer network				Same-layer		LID-net	
	2nd order		1st order		2nd order		0th order	
	EER	ACC	EER	ACC	EER	ACC	EER	ACC
32	6.97	72.89	7.08	72.99	7.19	72.80	7.67	71.63
64	6.94	73.11	7.15	72.99	7.16	73.04	7.76	71.52
128	7.05	72.86	7.08	73.03	7.25	73.23	7.58	71.79
256	7.09	73.43	7.37	72.56	7.13	73.40	**7.57**	**71.87**
512	**6.86**	**72.77**	7.11	71.43	7.17	73.05	7.79	71.53

If f_A and f_B are from the same layer in the CNN, this would be the formula to compute O2P in [27, eq. (1)]. In [27] it is a second order statistic computed over image spatial dimensions, but in *LID-bnet*, features are derived across acoustic analysis frame contexts. This means that while it differs from a traditional Baum-Welch second order statistic, it is effectively still a second order time domain statistic.

C. Same- or Cross-Layer Pooling

After transferring trained LID-net parameters to the corresponding *LID-bnet*, we re-trained using the same training data to verify whether bilinear pooling improves performance further. Focusing on 3s utterances, we conducted many experiments to explore the mechanism for computing first/second order statistics through same- or cross-layer pooling.

Table III lists the performance for various systems on 3s utterances. The number K in each row of the table indicates a test that involved either *LID-net-K* or an *LID-bnet* initialised from *LID-net-K*. Performance is given for both cross-layer and same-layer pooling and results for second and first-order statistics are listed separately (except for same-layer pooling). *LID-net* results are listed on the right for comparison, and we see that all *LID-bnet* configurations outperformed the original method. This is thanks to the robustness that is gained by using high-order LID-senone statistics. Cross-layer bilinear pooling performed better than same-layer pooling, which indicates that additional useful information was being gained by incorporating statistics from an earlier layer. Using the second order statistics was more robust in every case than that just first order statistics. Therefore the following evaluations only list the performance of the best configuration – second order LID-senone statistics obtained from cross-layer bilinear pooling.

C. Performance of LID-bnet

Like *LID-net*, *LID-bnet* has six convolutional layers, and again we evaluated with pooling between 32 and 512 feature maps after CNN layer 6 for comparison. The performance is shown in Table IV for each tested K, where an K of 32 means the corresponding *LID-net-32* pooled 32 feature maps after CNN layer 6. Comparing this to the *LID-net* performance in Table II, improvement is noted, especially at shorter utterance scales in both EER and C_{avg} scores. Although the optimal pooling length in *LID-bnet* was found to be different for each utterance duration, the best performance tended to be achieved with a shorter length than in *LID-net*. This highlights the compactness of the the bilinear pooling method: pooling over just 64 features in *LID-bnet* can outperform both the DBF/i-vector and the LID-net EER for shorter utterances. While the difficult 3s task benefits from improvements in system architecture, the 30s task appears to be data-limited rather than architecture-limited, and the newer architectures do not contribute additional data to the task, hence yield only minimal improvements for that scale.

TABLE IV
PERFORMANCE OF CROSS LAYER BILINEAR NETWORK FOR ALL SCALES WITH DIFFERENT POOLING LENGTHS K

K	30s		10s		3s	
	EER	C_{avg}	EER	C_{avg}	EER	C_{avg}
32	1.52	0.77	2.39	**1.20**	6.97	6.20
64	**1.48**	0.95	2.40	1.38	6.94	5.32
128	1.59	**0.66**	2.33	1.50	7.05	5.52
256	1.58	0.87	**2.32**	1.74	7.09	5.26
512	1.51	0.87	2.43	1.46	**6.86**	**4.38**

TABLE V
COMPARISON BETWEEN LID-NET-I POOLING OVER 32, 64 AND 128 FEATURES AND DNN/I-VECTOR FOR ALL UTTERANCE LENGTH CATEGORIES

System	30s		10s		3s	
	EER	C_{avg}	EER	C_{avg}	EER	C_{avg}
GMM/i-vector [11]	-	1.15	-	1.98	-	6.82
DBF/i-vector	1.48	1.10	3.05	2.14	10.79	7.48
LID-net-i-32	1.46	1.21	**2.39**	**1.79**	**8.10**	**6.77**
LID-net-i-64	1.43	1.10	2.44	1.87	8.30	7.38
LID-net-i-128	**1.41**	**0.96**	2.49	1.88	8.53	7.87

GMM/i-vector is from [11].

D. Design and Evaluation of the LID-net-i System

The *LID-net-i* structure hypothesised in Section IV-D, replaces the use of DBFs and senones of the DNN/i-vector system with LID-senones (as derived from *LID-net*).

In practice, after training a correspondingly sized *LID-net* system, we extracted LID-senones and their posteriors, in order to train a language independent GMM model. From this we obtained zeroth, first and second order Baum-Welch statistics to train a T matrix from which we could extract an i-vector. The i-vectors derived from LID-senones should, if our hypothesis is correct, incorporate a greater degree of language discriminant information than those derived from ASR senones.

To implement and evaluate this, we trained several *LID-net* systems, each with six convolutional layers. The first five layers had 512 channels while the sixth layer had a number of channels which varied from 32 to 128 as required to evaluate different lengths over which the statistics were pooled. The T matrices were trained over five epochs and the performance of the final system obtained in terms of EER (%) and C_{avg} for all testing conditions. Results are presented in Table V.

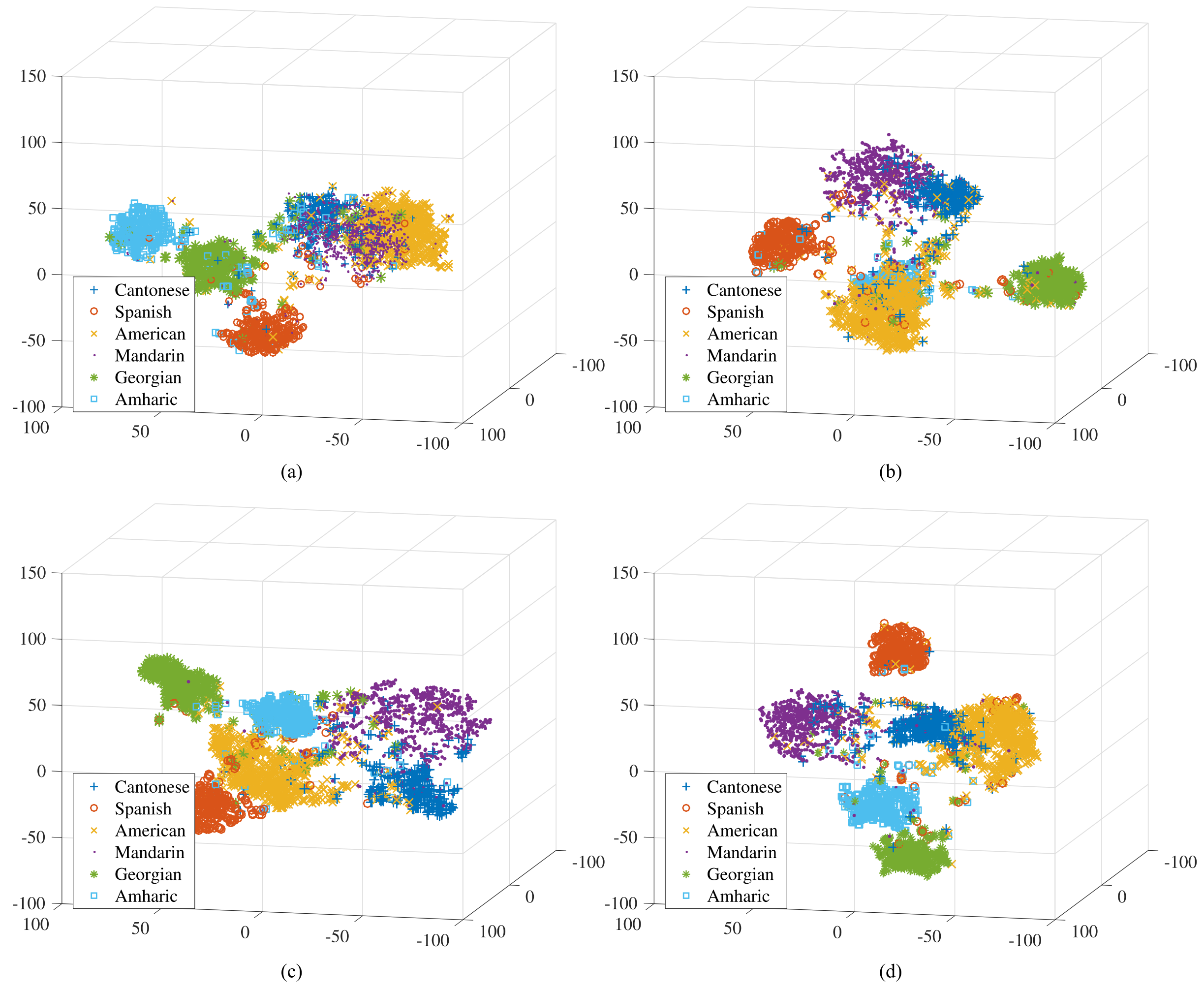

Fig. 8. Manifold plots [28] for (a) DNN/i-vector, (b) LID-net-i, (c) LID-net and (d) LID-bnet.

The results clearly show the advantages of *LID-net-i* over the baseline DNN/i-vector system, as well as the GMM/i-vector system using DBFs from Ferrer *et al.* [11]. Since the lower four systems are identical apart from the source of the utterance level statistics (from senones and LID-senones respectively), the results corroborate the hypothesis that LID-senones are better able to represent language information than senones.

To explore this in a different way, we have used t-SNE [28] to obtain cluster manifolds for a set of six languages from each system presented in this paper (four less confused languages plus one relatively confused language pair were chosen in an effort to highlight separability). t-SNE is a variant of stochastic neighbour embedding (SNE) that is conveniently able to visualise high-dimensional data by mapping it into a lower dimensionality. In this case we reduce the trained 3s statistics to three-dimensions and plot views of each system in Fig. 8 that highlight features that include cluster sizes, overlap and separability.[2] For i-vector systems the data is shown after LDA and WCCN.

[2]Since it is difficult to visualize three dimensional information from a two dimensional representation, we have made the original plots available for download from http://www.lintech.org/LID/ in MATLAB .fig format.

The manifold plots show much better class separability for the new systems compared to the baseline – for example the clusters in the *LID-bnet* plot are almost all linearly separable, whereas the Mandarin, Cantonese and even American English clusters in the DNN/i-vector system are highly overlapped. This again highlights the advantages of LID-senones – whether used in an end-to-end system (*LID-net* and *LID-bnet*) or to provide the statistical utterance level information in an otherwise traditional i-vector system (*LID-net-i*).

VI. Discussion of Results

A. System Performance Summary

To summarise the new systems presented in this paper, the final performance is given in Table VI, alongside the state-of-the-art baseline, with the best performing EER and C_{avg} scores shown in bold.

Starting with the 30s utterances, which we believe is data-limited, we see that *LID-net-i* achieves a 4.7% relative EER improvement over the baseline. This is due to the advantages of LID-senones and their statistics, which are designed to be language discriminant, over the traditional senones. However, the relative improvement is small because phoneme or tri-phone

TABLE VI
EVALUATIONS ON DIFFERENT SYSTEMS INCLUDING DNN/I-VECTOR, LID-NET, LID-BNET AND LID-NET-I

System name	30s		10s		3s	
	EER	C_{avg}	EER	C_{avg}	EER	C_{avg}
DBF/i-vector	1.48	1.10	3.05	2.14	10.79	7.48
LID-net	1.46	0.74	2.66	1.46	7.57	5.05
LID-bnet	1.48	**0.66**	**2.32**	**1.20**	**6.86**	**4.38**
LID-net-i	**1.41**	0.96	2.39	1.79	8.10	6.77

Performance is given in EER (%) and C_{avg} (%) for all test conditions.

state statistics do contain language specific information; but this is only useful when their statistics are abundant enough – and this is precisely the case for 30s utterances. As expected, the 10s and 3s tasks show a greater relative gain in using LID-senones, because the utterances are shorter and hence less able to contribute discriminative senone statistics.

For 10s utterances, *LID-bnet* EER performance is better by 23.9%, not only because it takes advantage of the discriminative capability of the end-to-end structure, but also because it extracts the high order statistics of the LID-senones. *LID-net-i* achieves second best performance with a 21.6% relative EER improvement over baseline. This is due to the LID-senone statistics being more discriminative on languages, as well as the utterance length being sufficient to collect meaningful statistics. *LID-net* also achieves a relative 10% EER improvement over baseline, even though it only simply averages the LID-senones to get their statistics (again, we note that effectively this means that LID-senone statistics are better than higher order senone statistics).

The most difficult 3s utterance task is the most interesting, with the strong modelling capability of the end-to-end approaches dominating so that both *LID-net* and *LID-bnet* outperform the i-vector systems. We explain this by arguing that when shorter utterances are modelled by a generative model, since there are insufficient statistics available in the speech, the accuracy of the model is compromised. The generative method concentrates on the distribution of different languages rather than pushing the same language into compact clusters, and hence different languages have a larger variance and greater overlap. End-to-end training allows *LID-net* to achieve 29.8% relative EER improvement over baseline while the bilinear higher order pooling of *LID-bnet* achieves 36.4% relative EER improvement.

For all utterance durations, we note that *LID-bnet* achieves the best C_{avg} performance – which we again attribute to the power of LID-senones. The discriminative method looks for boundaries between languages, so even when languages are confusable, it is easier to find an appropriate threshold, and hence have lower C_{avg} score than the generative modelling techniques.

B. DET Curve

DET curves [26] are plotted for all systems in Fig. 9, showing the trade-offs possible between false positive and false negative detections. The EER results of Table VI are derived from DET curves (from the transect of a 45° line from the origin with each curve), but the overall shape of the curves yields additional useful information. From Fig. 9 we note firstly that the *LID-bnet* system is almost always the highest performing method at all trade-off positions, whereas the baseline DNN/i-vector system is almost always the worst among tested systems, with the exception of low rates of false negatives at the 30s utterance scale. Secondly, we note that separation between curves at their centre positions, which represent the span of EER results, are scale dependent. This means that the 30s scale results are far closer in the centre than are the 10s results while the 3s results are more separated. The interpretation of this is that the 30s task is training data limited, and hence improvements in architecture of statistics quality have less effect than in the 10s or 3s tasks. For the 3s task, meanwhile, improvements in architecture lead to a much greater relative gain in performance. Both of these characteristics have been noted in the results presented up to now.

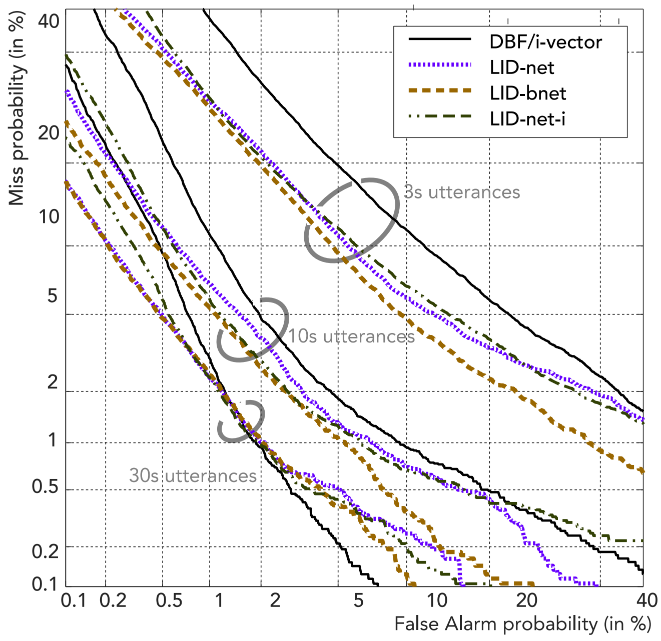

Fig. 9. DET curve plots for all systems and all utterance scales.

C. Confusion Matrices

Confusion matrices are plotted, normalized to the size of test data, for each system over a 3s utterance scale in Fig. 10. The highly confused language pairs are clearly identifiable from these plots, but apart from those language pairs which we already know to be inherently confusable, we note some other interesting features. Firstly, while *LID-net* and *LID-bnet* understandably share similar confusion tendencies, the other systems are much less similar. In fact *LID-net-i* arguably resembles DNN/i-vector confusions more closely. Secondly, again apart from those six pairs, i-vector and SPP or bilinear pooling methods – while they use identical information – draw different conclusions. In fact there is sufficient dissimilarity in the confusion matrices to raise the intriguing possibility that a performance gain might be achievable through a fusion of methods. This idea is left for future exploration.

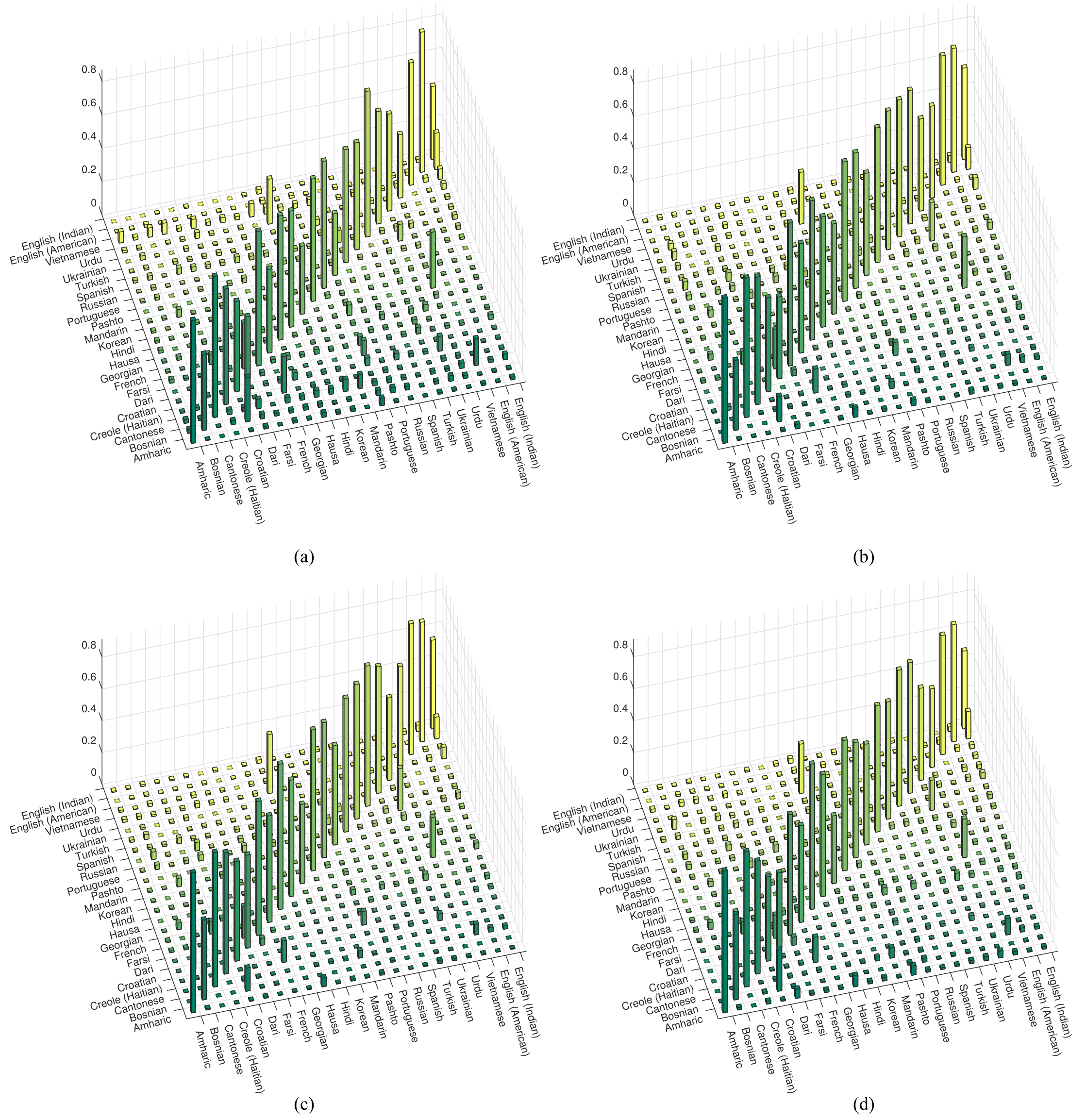

Fig. 10. Normalized confusion matrices for 3s utterances for (a) DNN/i-vector, (b) *LID-net*, (c) *LID-bnet* and (d) *LID-net-i*.

VII. Conclusion

This paper has presented new language identification structures that utilise a well trained DNN front end for feature extraction, followed by a deep stack of convolutional layers, pooling methods and fully connected output classifiers. There are several levels of time domain context implicit in the structures, starting at frame level and culminating in pooling of features or statistics across an utterance. The key features of these methods are firstly that they can be trained end-to-end for a language identification task and secondly that they form an intermediate feature representation which we have called LID-senones. The end-to-end training capability of these systems allows them to be trained at an utterance level to be more discriminative in terms of language identity, compared to current state-of-the art systems such as the DNN/i-vector used as a baseline in this paper, which is generative in nature. The increased level of discrimination is borne out in the excellent results that these methods achieve, particularly for the most difficult 3s utterance LID task; in which they improve on current state-of-the-art system performance by over 36% in terms of EER and 41% in terms of $\mathrm{C}_{\mathrm{avg}}$.

The intermediate LID-senone representation derived within the end-to-end structures is analogous to senones in an ASR system, and we have explored in some detail how they should be language sensitive but relatively speaker, channel and noise insensitive. Evidence presented in this paper shows that LID-senones encode language identity; their distribution is relatively unchanging in intra-language comparisons, but significantly different in inter-language comparisons. LID-senone systems are shown to perform well, even when a DNN/i-vector system is constructed to use LID-senones instead of senones, yielding significant performance improvements at all utterance scales.

As future work it would be interesting to explore the noise robustness of SPP operating over different scales or when pooling a greater number of shorter statistics, perhaps at a syllabic or word scale.

References

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

[2] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proc. InterSpeech*, 2011, pp. 857–860.

[3] G. Hinton *et al.*, "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.

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

[5] Y. Song, X. Hong, B. Jiang, R. Cui, I. V. McLoughlin, and L. Dai, "Deep bottleneck network based i-vector representation for language identification," in *Proc. InterSpeech*, 2015, pp. 398–402.

[6] F. Richardson, D. Reynolds, and N. Dehak, "A unified deep neural network for speaker and language recognition," arXiv:1504.00923, 2015.

[7] B. Jiang, Y. Song, S. Wei, J.-H. Liu, I. V. McLoughlin, and L.-R. Dai, "Deep bottleneck features for spoken language identification," *PLoS One*, vol. 9, no. 7, 2014, Art. no. e100795.

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

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

[10] P. Kenny, V. Gupta, T. stafylakis, P. Quellet, and J. Alam, "Deep neural networks for extracting Baum-Welch statistics for speaker recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 1695–1699.

[11] L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer, "Study of senone-based deep neural network approaches for spoken language recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 1, pp. 105–116, Jan. 2016.

[12] B. Jiang, Y. Song, S. Wei, I. Mcloughlin, and L.-R. Dai, "Task-aware deep bottleneck features for spoken language identification," in *Proc. InterSpeech*, 2014, pp. 3012–3016.

[13] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

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

[16] 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. Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 5337–5341.

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

[18] M. Jin, Y. Song, and I. McLoughlin, "LID-senone extraction via deep neural networks for end-to-end language identification," in *Proc. Odyssey 2016*, Jun. 2016, pp. 210–216.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.

[20] Y. Song, R. Cui, X. Hong, I. Mcloughlin, J. Shi, and L. Dai, "Improved language identification using deep bottleneck network," in *Proc. 2015 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 4200–4204.

[21] S. M. Siniscalchi, J. Reed, T. Svendsen, and C.-H. Lee, "Universal attribute characterization of spoken language for automatic spoken language recognition," *Comput. Speech Lang.*, vol. 27, no. 1, pp. 209–227, 2013.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proc. 32nd Int. Conf. Mach. Lear.*, in PMLR, vol. 37, pp. 448–456, 2015.

[23] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1449–1457.

[24] I. V. McLoughlin, *Speech and Audio Processing: A MATLAB-Based Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[25] A. Martin and C. Greenberg, "The 2009 NIST language recognition evaluation," in *Proc. Odyssey 2009, Speaker Lang. Recognit. Workshop*, 2010, pp. 165–171.

[26] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," in *Proc. Eur. Conf. Speech Commun. Technol.*, 1997, pp. 1895–1898.

[27] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 430–443.

[28] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Ma Jin received the Master's degree from the National Engineering Laboratory of Speech and Language Information Processing, University of Science and Technology of China, Hefei, China, for his research on language identification. He is currently working at Microsoft as a Software Engineer, focusing on language model rescoring and language identification.

Yan Song received the B.Sc. degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1994, and the M.Sc. and Ph.D. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, in 1997 and 2006, respectively. He is currently an Associate Professor in the National Engineering Laboratory for Speech and Language Information Processing, and has been a Faculty Member in the Department of Electronic Engineering and Information Science, University of Science and Technology of China, since 2000. His research interests include multimedia information processing, automatic language identification, speaker diarization, and image classification.

Ian McLoughlin (M'94–SM'04) received the Ph.D. degree in electronic and electrical engineering from the University of Birmingham, Birmingham, U.K., in 1997. He worked for more than 10 years in the R&D industry and about 15 years in academia, on three continents. He is currently a Professor at the University of Kent, Medway, U.K. He has written many papers and several patents on speech analysis and communications, and is an author of four books on speech processing and embedded computation. He is a Fellow of the IET and a Chartered Engineer. He received the Chinese Academy of Sciences President's International Fellowship Award and the Hundred Talent Program Funding from Anhui Province, China.

Li-Rong Dai received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 1983, the M.S. degree from Hefei University of Technology, Hefei, China, in 1986, and the Ph.D. degree in signal and information processing from the University of Science and Technology of China, Hefei, China, in 1997. In 1993, he joined the University of Science and Technology of China where he is currently a Professor in the School of Information Science and Technology. His current research interests include speech synthesis, speaker and language recognition, speech recognition, digital signal processing, voice search technology, machine learning, and pattern recognition. He has published more than 50 papers in these areas.