Odyssey 2016
21-24 Jun 2016, Bilbao, Spain

LID-senone Extraction via Deep Neural Networks for End-to-End Language Identification

Ma Jin[1], *Yan Song*[1], *Ian Mcloughlin*[2], *Li-Rong Dai*[1], *Zhong-Fu Ye*[1 3]

[1]National Engineering Laboratory of Speech and Language Information Processing
University of Science and Technology of China, China
[2]School of Computing, University of Kent, Medway, UK
[3]State Key Laboratory of Mathematical Engineering and Advanced Computing, China
jinma525@mail.ustc.edu.cn, {songy, lrdai, zfye}@ustc.edu.cn, ivm@kent.ac.uk

Abstract

A key problem in spoken language identification (LID) is how to effectively model features from a given speech utterance. Recent techniques such as end-to-end schemes and deep neural networks (DNNs) utilising transfer learning such as bottleneck (BN) features, have demonstrated good overall performance, but have not addressed the extraction of LID-specific features. We thus propose a novel end-to-end neural network which aims to obtain effective LID-senone representations, which we define as being analogous to senones in speech recognition. We show that LID-senones combine a compact representation of the original acoustic feature space with a powerful descriptive and discriminative capability. Furthermore, a novel incremental training method is proposed to extract the weak language information buried in the acoustic features of insufficient language resources. Results on the six most confused languages in NIST LRE 2009 show good performance compared to state-of-the-art BN-GMM/i-vector and BN-DNN/i-vector systems. The proposed end-to-end network, coupled with an incremental training method which mitigates against over-fitting, has potential not just for LID, but also for other resource constrained tasks.

Index Terms: language identification, utterance representation, end-to-end neural network, LID-senone, incremental training method

1. Introduction

Language identity is a key attribute of spoken utterances. Unlike phonetic content which can be easily modelled using end-to-end schemes at a frame level, it is difficult to map from frame level features to a language target. Working backwards from an LID target, it is necessary to have an effective utterance representation which is in turn derived from frame-level features, that are extracted from a section of input speech. At present, the i-vector representation [1, 2] is the state-of-the-art for utterance representation, due to its compactness and good performance. However, i-vectors are extracted in an unsupervised fashion without using language labels, and thus require linear discriminant analysis (LDA) or similar methods to build backend models.

In terms of feature representation, deep learning techniques such as DNNs [3], have demonstrated their learning capabilities in several related fields, and various extensions and additions have been studied to improve i-vector performance for acoustic modelling. Song *et.al*, Richardson *et.al* and Bing *et.al* [4, 5, 6] proposed using deep bottleneck features (DBFs) from a DNN trained for automatic speech recognition (ASR) [7]. DBFs are inherently robust for different speakers, channels and background noises. Lei *et.al*, Kenny *et.al* and Ferrer *et.al* [8, 9, 10] proposed collecting sufficient statistics also using structured DNNs to form an effective representation of underlying phonemes or phoneme states. It seems that DNNs are effective for either front-end frame-level feature extraction or back-end utterance-level modelling, where sufficient good quality and quantity training data is available. DBFs or senones, both derived from ASR training, are therefore clearly able to represent language-based content. However they are trained to assign to phonemes or phoneme states. While this is useful for LID, it does not specifically encode language-discriminative information, particularly for highly confusable languages that may have similar phoneme-level statistics.

Recent LID research has gradually moving towards task-aware or end-to-end schemes, which have demonstrated impressive performance. Jiang *et.al* showed [11] that tuning the pre-trained DNN parameters using an LID-specific corpus can improve performance. However this lattice-based optimisation scheme adjusts final layers and does not propagate to earlier acoustic layers.

Convolutional neural networks (CNNs) have also demonstrated impressive front-end feature representation results on large-scale speech and visual object recognition tasks [12, 13]. In current multi-layer CNNs, convolution-pooling-layer pairs can be thought of as front-end feature extractors, followed by final pooling layers to map frame-level features into an utterance representation that is amenable to linear classification. Lozano-Diez *et.al* [14] evaluated different CNNs for LID, demonstrating comparable results for short utterances to end-to-end methods. Lopez-Moreno *et.al* and Gonzalez-Dominguez *et.al* [15, 16] also proposed end-to-end schemes using large scale DNNs and long short-term memory (LSTM) recurrent neural networks (RNN), which both perform well. Output scores at utterance level are the log-average of the softmax outputs across all frames.

1.1. Contribution

We propose a new end-to-end approach, named **LID-net** that combines the proven frame-level feature extraction capabilities of the DNN with the effective utterance level mapping abilities

Supported by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (grant, 2015A15).

doi: 10.21437/Odyssey.2016-30

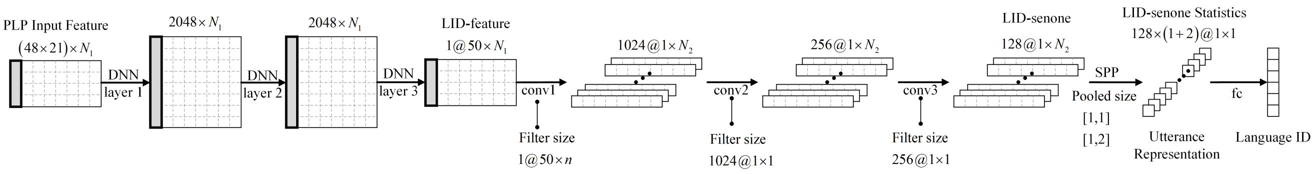

Figure 1: *LID-net Structure.* The whole network spans from frame to utterance level, consisting of *DNN layers*, *convolutional* and *SPP* layers with detailed configuration shown in Table 1. Through *DNN layers*, LID-features can be extracted whose size is $50 \times N_1$. After convolutional layers, LID-senones are obtained and SPP acts as a speech utterance representation extractor, as explained in Section 3. The detailed notation and meaning of N_1 and N_2 is explained in Section 2.

Table 1: *Configuration of LID-net.* Eight layers are divided into DNN, convolution, SPP and fully connected (fc) layers, with N_1 and N_2 parameters used to experimentally explore the effect of context. Separate 30s, 10s and 3s networks are trained independently (despite the 10s and 3s networks having identical structure).

Layer	Stage	Input Size	Configuration
1	DNN layer1	$(48 \times 21) \times N_1$	connections: $(48 \times 21) \times 2048$
2	DNN layer2	$2048 \times N_1$	connections: 2048×2048
3	DNN layer3	$2048 \times N_1$	connections: 2048×50
4	conv1	$1@50 \times N_1$	filter size: $1@50 \times 21$
5	conv2	$1024@1 \times N_2$	filter size: $1024@1 \times 1$
6	conv3	$256@1 \times N_2$	filter size: $256@1 \times 1$
7	SPP 30s	$64@1 \times N_2$	SPP pooled size: $[1,1], [1,2]$
	SPP 10s/3s	$128@1 \times N_2$	SPP pooled size: $[1,1], [1,2]$
8	fc 30s	$64 \times (1+2)@1 \times 1$	filter size: $64 \times (1+2)@1 \times 1$
	fc 10s/3s	$128 \times (1+2)@1 \times 1$	filter size: $128 \times (1+2)@1 \times 1$

of a CNN. Specifically, as shown in Fig. 1, LID-net consists of a DNN-based front-end to derive acoustic features related to LID tasks, followed by a CNN back-end, applying spatial pyramid pooling (SPP) [17] to form an utterance representation. The DNN is configured with a constricted bottleneck (BN) layer to transform acoustic features into a compact representation in a frame-by-frame manner[†]. Convolutional layers then perform nonlinear transformations of BN features into units which are discriminative for different languages. Following the definition of senones in DNNs for ASR, we term these units **LID-senones** and will investigate them further in Section 3. A SPP layer is introduced to form an utterance representation from LID-senones.

Despite having a logical and compelling structure, LID-net contains a large number of parameters needing training. Given the limitations of LID training data, particularly for utterance-level labels, direct training leads to severe over-fitting issues. An incremental training scheme is therefore proposed, inspired by transfer learning techniques. Specifically, we first initialise *DNN layers* from a DNN well-trained by the SwitchBoard corpus as in [18], then train LID-net by incrementally adding and training successive convolutional layers. Results show good performance gains through adding convolutional layers in this way (see Section 5.4).

To summarise, the contribution of this paper are:

- LID-net, a novel end-to-end structure aiming to model LID-senones.
- An incremental training scheme to effectively exploit information from related tasks (*e.g.*, large scale ASR) and address the over-fitting issues in the deep structure.
- Extensive experiments of different LID-net configurations on NIST LRE 2009 highly confusable languages. Comparing these to state-of-the-art i-vector methods based on DBFs, improved performance is achieved.

2. LID-net structure

We propose a comprehensive task-aware neural network spanning frame to utterance level, as shown in Fig. 1. The whole system includes *DNN layers* followed by *Convolutional* and *SPP layers*. The natural ability of task aware NNs to infer discriminative rules allows effective transformation of acoustic features, with pooling to take the place of the i-vector generative approach.

In operation, acoustic features are extracted from frame to utterance level, with certain frames combined together to exploit context in some layers. The first context window size is in *DNN layer1*, using a fixed $10-1-10$ size (which means a sliding context window is used with every 21 frames input to represent the current frame) which is common in DNN based speech systems. 50 dimensional LID-features are then extracted through the *DNN layers* before another context window is applied at layer *conv1*. Its size is controlled by the filter length. If the convolutional filter is $1@50 \times 21$ (meaning a 50×21 rectangle over 1 channel), this means it combines $10-1-10$ frame-level features. The final context is the SPP layer after the convolutional layers. This differs from the first two context windows, as it is intended to extract features at an utterance level while the previous windows are at a frame level.

Layers *DNN layer1* to *DNN layer3* act as a feature transformer. Unlike general acoustic features such as MFCC or PLP, the derived LID-features are task-specific. Since LID systems are effective at a frame or short utterance level [15, 16], these

[†]Note that *DNN layers* could also be implemented as convolutional layers as a consequence of them being initialized by a previously well trained DNN. This will be explained further in Section 4.

layers process acoustic features frame by frame. As mentioned above, the BN construction aims to keep LID-features compact.

The subsequent layers could be viewed as an utterance representation extractor. Previous work has proven that the statistics of senones can be discriminative in languages [19, 10]. We aim for a task-aware version which we name LID-senones, that are constructed from LID-features to make their statistics even more discriminative for LID. The frame-level LID-senones are pooled into an utterance representation by the SPP layer.

The configuration of each layer is shown in the Table 1. An input size configuration of "$1024@1\times N_2$" means a $1\times N_2$ input feature over 1024 channels. SPP pool parameters "$[1,1]$, $[1,2]$" indicate that the pooled feature vector size is $[1,1]$ and $[1,2]$ at each channel then reshaped to different channels respectively.

Note that the 30s, 10s and 3s networks are trained independently. Although 10s and 3s networks have the same structure, they contain different parameters. The 30s network differs from the others before SPP due to lack of training data, so the number of filters in the 30s network is naturally smaller. Unlike the 30s case, the other networks benefit from augmentation techniques to effectively enlarge their training datasets.

3. LID-senones and utterance representation

3.1. LID-senones with convolutional layers

Given two dimension input features, researchers tend to analyse small blocks across both local time and frequency domains, such as 5×5 or 7×7 [14], despite the fact that useful correlations exist in larger time and frequency domain regions [20]. We therefore include the entire frequency domain over several frames in our convolution.

To explain how it works, consider the following hypothetical example. An LID-net with just one convolutional layer (i.e. *LID-feature* output linked directly to *LID-senone* input) has been well-trained, and the configuration of SPP is $[1,1]$; this means pooling size is $[1\times N_2]$, thus the whole speech feature matrix has N_2 frames pooled directly into an utterance vector. In Fig. 1, before the *fc* layer, LID-senone statistics are expected to be obtained through feed-forward calculation so a classifier can directly follow in obtaining language ID. LID-senones do not have specific names like ordinary senones in ASR, which is unimportant as long as they contain sufficient information to discriminate between languages. In our example, the activation of each LID-senone on each frame could be obtained from *conv1*. Therefore we could view the trained filters in *conv1* as being LID-senone detectors.

In fact, we do not know how many frames are needed to effectively learn an LID-senone, so Section 5.3 will explore different sizes of filters. However we can use a simple network to conduct some experiments to briefly provide evidence for the LID-senone idea. Firstly, activation values from the layer prior to the SPP (of size 1024×1) are explored from four frames from a recording. A simple plot of the vectors, in Fig. 2 shows distinct activations from different utterances in (a) and (b), a transition region in (c) and a non-speech region in (d). Note the amplitude scale of the latter. It appears that different LID-senones are active in different regions of an utterance.

To explore further, the activations prior to the *fc* layer are also captured and analysed. Four different utterances (two utterances from two languages) are taken from the training dataset, and 35 LID-senones randomly chosen from the higher dimensional utterance vector. The statistics from these sets of LID-

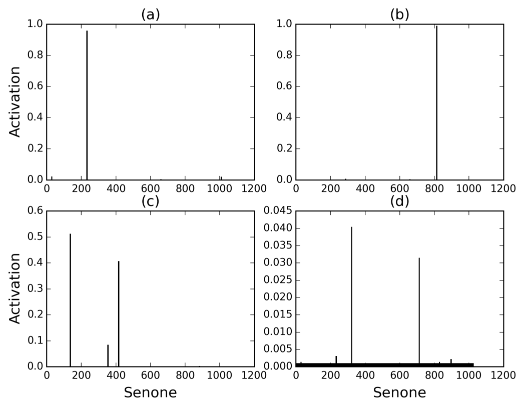

Figure 2: *Activation Values before SPP From Four Different Frames.* The values are operated with softmax for convenience. Filter size in *conv1* is $1@50\times21$. (a) and (b) show the activation values of normal frames, who could represent an activated LID-senone precisely. (c) describes activation of LID-senones at a change point, so more than one LID-senone is activated. (d) shows very small activation values for non-speech frames.

senones are compared for the four utterances in Fig. 3, comprising two utterances of Dari and two of Farsi. Fig. 3 clearly exhibits greater uniformity in the within-language statistics in each case, providing evidence to support the language discriminating ability of LID-senones.

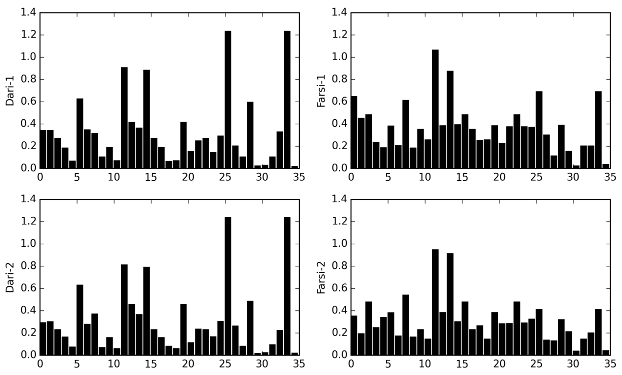

Figure 3: *Statistics on Four Different Utterances.* The left two are different analysis regions of a Dari utterance and the two on the right are from Farsi. Plots show 35 statistical values randomly selected from the LID-senone statistics vector.

3.2. Spatial pyramid pooling for utterance represenation

Spatial Pyramid Pooling can be viewed as a special method of pooling, which was first introduced in image recognition [17]. The motivation of SPP is to solve the problem that input feature size is arbitrary, yet a fixed-dimension representation is required, irrespective of input size.

Unlike a normal pooling layer with a fixed pooling size, SPP has a fixed pooled number. If input feature vector size is $C@M \times M$, and $[N, N]$ SPP is implemented, this means that whatever the input size, the feature vector will be segmented into $N \times N$ parts while the pooling size is $\lceil \frac{M}{N} \rceil$ and stride is $\lfloor \frac{M}{N} \rfloor$, then every sub-spatial space executes max/average pooling. By this means, C channels would give an $N \times N \times C$ feature matrix, which is reshaped to an $(N \times N \times C)@1 \times 1$ four dimensional matrix. Thus to cater to a deep network of arbitrary size, ordinary pooling is replaced with SPP.

SPP could be utilized not only for its tolerance to different input sizes, but can also pool at different scales. In LID-net, a $[1, 1]$ and $[1, 2]$ SPP are applied simultaneously. It is observed that $[1, 1]$ pooling is implemented over the whole N_2 frames while $[1, 2]$ is carried out on the $N_2/2$ frame scale.

4. Incremental training strategy

In DNN for ASR systems, each frame is associated with one label whereas an end-to-end LID system utterance, which might contain hundreds or even thousands of frames, has just one label. If LID-net is trained directly using the target language dataset, performance will be limited by lack of training data, since a large quantity of labeled data is required to train the NNs. We propose an incremental training method to ease this dilemma. Due to the success of adopting pre-trained ASR DNNs for LID tasks [11], we first train a DNN using the SwitchBoard corpus, and partly transfer the parameters to *DNN layer1-DNN layer3* of LID-net. Convolutional layers are then added layer by layer to fine-tune the whole network. The detailed procedure is shown in Fig. 4.

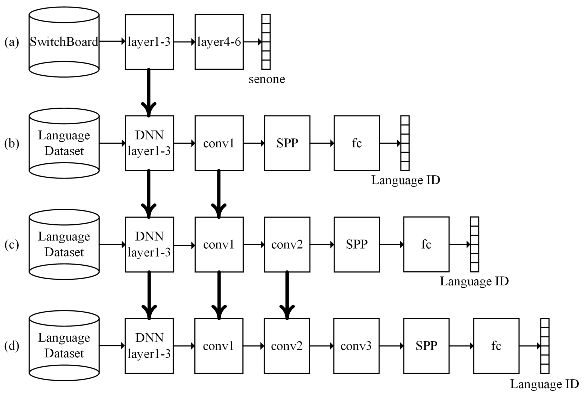

Figure 4: *Incremental Training Strategy Flowchart.* (a) A DNN with one internal BN layer for ASR is trained using the SwitchBoard corpus. (b) *layer1-layer3* parameters from *(a)*, with one convolutional layer appended, is trained until convergence. (c) *DNN layer1-DNN layer3* and *conv1* parameters, with a second convolution layer appended, is trained until convergence. (d) *DNN layer1-DNN layer3* and *conv1-conv2* parameters are transferred to LID-net and another convolutional layer appended, then trained until convergence.

4.1. Transfer learning from large-scale corpus

Transfer learning [21, 22, 23] is very common in NNs for tasks such as recognition and verification, especially in image and speech processing. It is obvious that most of the trainable parameters are at the frontal *DNN layers*, hence severe over-fitting may be caused by training such a deep network when there is a deficiency of training data. In this case, SwitchBoard is utilized to train a six layer DNN with one internal bottleneck layer, using 48 dimension features (15 PLP $+\Delta + \Delta\Delta + 3$ pitch) and 3020 senones, which is identical to LID-net. The structure of the DNN is $48 \times 21 - 2048 - 2048 - 50 - 2048 - 2048 - 3020$. Once the DNN is trained using the ASR data, the parameters in the first three layers are transferred to *DNN layer1-DNN layer3* of LID-net, as previously discussed. The learning rate of parameters in the transferred layers is set to one tenth of the newly added layers, and the number of additional convolutional layers constrained to the training dataset size.

4.2. Incremental training with language corpus

Despite having transferred most of the LID-net parameters from a well trained DNN, the remaining parameters cannot be randomly initialized, as in usual practice, since the network consistently fails to converge. We thus train the network layer-by-layer, and Fig. 4 (b), (c) and (d) illustrates the training sequence. We first train LID-net with just one convolutional layer, for which random Gaussian initialization is used, while other layers are populated with transferred parameters. We train the network until it converges, then add another new convolutional layer which is randomly initialised, and continue training using the same method. By this means, layers can be successively added to an arbitrary depth, called incremental training.

5. Experiments

5.1. Experimental Setup

5.1.1. Language training corpus

All the experiments are evaluated on the six most confused languages from the NIST LRE 2009 dataset: Dari, Farsi, Russian, Ukrainian, Hindi and Urdu. Most of the speech is collected from the Conversational Telephone Speech (CTS) and Voice of America (VOA) radio broadcasts. The training dataset comprises about 150 hours of speech, with Equal Error Rate (EER) used to measure performance. Evaluations are on 30s, 10s and 3s temporal scales, so three independent systems are trained. Data augmentation is used to expand the training dataset for the 10s and 3s data by segmenting the dataset into 3s chunks with 1.5s overlap and 10s chunks with 5s stride respectively. This yields approximately 282 hours of 3s segments and 245 hours of 10s segments, compared to the original 150 hours.

5.1.2. Hybrid temporal evaluation

Since the 30s, 10s and 3s networks are trained independently, three different temporal scale networks are implemented. In which case, during testing, 30s data could be segmented into 10s and 3s speech then use the corresponding well trained network to evaluate them separately. Thus 30s test data yields three temporal scale results for 30s, 10s and 3s, while 10s data uses the same method to yield 10s and 3s temporal scale results. This kind of testing procedure is called a hybrid evaluation method, explored further in Section 5.5.

5.1.3. Baseline systems and proposed networks

For comparison, the following methods are implemented:

LID-net: Proposed LID-net without hybrid temporal evaluation, trained and tested independently using 30s, 10s or 3s data.

LID-HT-net: Proposed LID-net with hybrid temporal evaluation. In the 30s test dataset, time scale could be 30s, 10s or 3s. In the 10s dataset, time scale can be 10s and 3s. The 3s dataset has only a 3s timescale.

BN-GMM/i-vector: This is the first baseline system used for comparison. The i-vector method uses DBF as front-end features and Gaussian posterior probability for back-end modeling [7]. LDA and within class covariance normalization (WCCN) are applied to compensate the variability, and cosine distance is used to obtain the final score.

BN-DNN/i-vector: This is the second baseline system used for comparison. The procedure is the same as **BN-GMM/i-vector** except that the back-end model is derived from a well-trained DNN trained on ASR data.

Test systems are identified using the notation "Nxy", indicating experiment x, variant y.

5.2. Baseline system configuration

Our baselines are the state-of-the-art **BN-GMM/i-vector** and **BN-DNN/i-vector** systems, where both the acoustic feature and the DNN structure configuration with BN layer are exactly the same as the proposed LID-net. The systems are built as follows.

(1) Train a 6 layer DNN ($48 \times 21 - 1024 - 1024 - 50 - 1024 - 1024 - 3020$) with an internal BN layer using the SwitchBoard dataset;

(2) Extract 50-dimenstion DBF from the trained DNN;

(3) Construct zeroth-order and first-order Baum-Welch statistics. The front-end features are all DBF and back-end modeling by Gaussian posterior probability and output of DNN for ASR in **BN-GMM/i-vector** and **BN-DNN/i-vector** respectively.

(4) Train T-matrix using expectation maximization (EM) with 5 iterations;

(5) Extract 400-dimension i-vector of both training and evaluation data;

(6) LDA and WCCN are applied to compensate for the intersession variability. Cosine distance is used to calculate the final score of each speech utterance.

5.3. Evaluation of different convolutional filter sizes

To ascertain how many frames are suitable for constructing LID-senones, we explore different filter sizes of $50 \times n$ in the first convolutional layer, beginning with $n = 1$ and testing with a step size of 5. In this experiment, all LID-net architectures have just one convolutional layer and the number of channels after *conv1* is 1024. Results are presented in Table 2.

Table 2: *Comparison of LID-net with Different Filter Sizes in Convolutional Layer.* Performance is given in EER (%) for all test conditions.

No.	System Name	30s	10s	3s
N11	LID-net 50x1 filter	10.37	12.89	18.51
N12	LID-net 50x6 filter	10.11	12.58	17.39
N13	LID-net 50x11 filter	9.47	12.26	16.76
N14	LID-net 50x16 filter	9.06	11.94	16.80
N15	LID-net 50x21 filter	8.95	**11.65**	**16.24**
N16	LID-net 50x26 filter	**8.91**	11.92	16.54

From Table 2 we can see results from different filter sizes in the convolutional layers. The performance trend in the same temporal scale decreases as the context window become larger. The 10s and 3s network performance is best when $n = 21$. A filter size of 50×26 has slight improvement in 30s and a slight decrease for 10s and 3s. This means that the trained LID-features best indicate LID-senones every 41 frames (i.e. two context windows), which is longer than the 21 frames commonly used for ASR in 7-1-3-7 SDC features [24]. While 21 frames is a good representation of the basic phonemic unit in ASR, LID-senones appear to benefit from more frames. As a consequence, a filter size of 50×21 is selected for all of the following experiments.

5.4. Evaluation of convolutional layer complexity

After one convolutional layer, some coarse LID-senones can be obtained, which are available for calculating statistics information. Still, we wish to extract more succinct LID-senones through additional convolutional layers, which are applied with an incremental training strategy. Table 3 explores how the performance of the system changes when appending additional layers.

Table 3: *Comparison of Different Complexities in Convolutional Layers.* The name "LID-net 1024C+128C" means two convolutional layers with the first having 1024 channels and the second having 128 channels. All the filter sizes of the first convolutional layer are $1@50 \times 21$ and the other layers are $C@1 \times 1$, where C is the number of input channels. The performance is given in EER (%) for all test conditions.

No.	System Name	30s	10s	3s
N15	LID-net 1024C	8.95	11.65	16.24
N21	LID-net 1024C+128C	8.87	9.26	13.72
N22	LID-net 1024C+256C	8.91	8.92	13.52
N23	LID-net 1024C+512C	10.27	9.26	13.83
N24	LID-net 1024C+128C+64C	8.20	9.63	13.26
N25	LID-net 1024C+256C+64C	**7.87**	8.52	13.34
N26	LID-net 1024C+256C+128C	8.76	**7.79**	**13.23**

In Table 3, N15 was the best performing system from Table 2 whose convolutional filter size is 50×21 with 1024 channels. In N21, N22 and N23, the number of channels in the first convolutional layer is fixed, but there is an additional convolutional layer. Three complexities of 128, 256, 512 channels are evaluated respectively. N24, N25 and N26 add a third convolutional layer to the above three systems. The results indicate that performance improves considerably by moving from two to three layers in 30s and 10s networks while results show significant improvement moving from one to two convolutional layers in the 3s network. To explain this, we think the statistics in 30s/10s tests are more abundant than for 3s LID-senones, therefore only a more complex network can fit the distribution of 30s/10s LID-senones. In the 3s network the circumstance is the opposite; the complexity of such a short utterance is simpler, hence a shallow network can effectively model the distribution. Due to the large quantity of 3s training data, the more complicated network still gains some improvement, but this appears to achieve decreasing returns. In the two convolutional layer 30s network, the performance become sharply worse with more parameters, due to lack of training data in 30s compared to 10s and 3s. Still, we see that the performance of N25 in 30s is counterintuitively worse than N26 in 10s, and argue that this is mainly for the same reason.

5.5. Hybrid temporal evaluation

In this section, the best performing LID-nets are selected to be part of LID-HT-net in 30s, 10s and 3s temporal scales respectively. These are namely N25 on 30s, N26 on 10s and 3s. The performance of the hybrid temporal evaluation is shown in Table 4 alongside the current state-of-the-art baseline systems.

Table 4: *Results of Hybrid Temporal Evaluation and State-of-the-Art Systems.* The performance is given in EER (%) for all test conditions.

No.	System Name	30s	10s	3s
B01	BN-GMM/i-vector	6.47	7.49	13.37
B02	BN-DNN/i-vector	5.84	8.02	15.27
N31	LID-HT-30s-net	7.87	-	-
N32	LID-HT-10s-net	**6.36**	**7.79**	-
N33	LID-HT-3s-net	7.07	8.50	**13.23**
N41	LID-HT-net fusion	**6.07**	**7.73**	**13.23**

From the results we can see that the 10s-net achieves best performance on 30s and 10s test datasets. This is because statistical information is insufficient in 3s, limited by speech length. The 10s-net has both sufficient statistics as well as an adequate training dataset. We firmly believe that if adequate training data was available for the 30s-net, the performance would be much better. Finally, we perform a simple fusion on the scoring of the three networks (shown at the bottom of Table 4) indicating further performance gains. This lends support to the suggestion that different temporal networks capitalise on complementary information.

The final LID-net performs well compared with the two baseline systems. Compared to BN-GMM/i-vector, the 30s and 3s results are better but the 10s result is slightly worse. Compared to BN-DNN/i-vector, the 30s result is slightly worse, but the 10s and 3s results are clearly better. Note that the i-vector baselines use both zeroth order and first order Baum-Welch statistics for training an extractor. In LID-net, what the SPP layer does is just equivalent to calculating zeroth order Baum-Welch statistics. The discriminative power of the end-to-end deep network has thus enabled LID-net to perform as well on only zeroth order statistics as the i-vector systems do on both zeroth and first order statistics.

6. Conclusion

In this paper we propose a comprehensive task-aware network spanning frame to utterance level.

The network is deep but straightforward; first LID-features are derived from ordinary acoustic features through several *DNN layers*, followed by several convolutional layers and then one SPP layer. LID-senones are expected to be derived from several frames of context by the convolutional layers, and their statistics are obtained through the SPP layer.

Despite a compelling architecture, when all of the layers are trained together, the network suffers serious over-fitting due to lack of LID training data combined with the large number of parameters. Hence, we propose starting with a DNN trained using ASR data, and then specify an incremental training method that successively transfers training parameters from a smaller trained network to which a convolution layer is added and then fine tuned. At the final layer, language ID labels are used with the back-propagation algorithm for fine-tuning.

Hybrid temporal evaluation is proposed for various time scales in the same test dataset. We consider that different temporal scales contain complementary information to some degree. Results show that LID-net can achieve good performance, and this is achieved without utilizing first order Baum-Welch statistics. The integrated procedure is summarised as follows.

(1) Train a 6 layer DNN ($48 \times 21 - 1024 - 1024 - 50 - 1024 - 1024 - 3020$) with an internal BN layer using the SwitchBoard dataset;

(2) Transfer the first 3 layers to *DNN layer1-DNN layer3* of LID-net;

(3) Using an incremental training strategy, add convolutional layers and retrain;

(4) Use hybrid temporal evaluation to obtain different scales of results on 30s and 10s test datasets.

Despite good performance, LID-net is not perfect. Firstly, the training data for 30s is clearly not adequate for the complexity of the network. We thus intend using the whole NIST LRE 2009 dataset for training, not just the six most confused languages. Secondly, only a simple fusion is performed on different temporal scales. We consider that if a large neural network is trained that combines all 30s, 10s and 3s networks together, it would have six temporal scales including 30s, 15s, 10s, 5s, 3s and 1.5s due to the $[1, 1]$ and $[1, 2]$ SPP layers. This may achieve even better performance due to the existence of complementary information at different temporal scales.

7. References

[1] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans Audio Speech Lang Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction.," *Proc. of InterSpeech*, pp. 857–860, 2011.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Mag*, vol. 29, no. 6, pp. 82–97, 2012.

[4] Yan Song, Xinhai Hong, Bing Jiang, Ruilian Cui, Ian Vince McLoughlin, and Lirong Dai, "Deep bottleneck network based i-vector representation for language identification," *Proc. of InterSpeech*, pp. 398–402, 2015.

[5] Fred Richardson, Douglas Reynolds, and Najim Dehak, "A unified deep neural network for speaker and language recognition," *arXiv preprint arXiv:1504.00923*, 2015.

[6] Bing Jiang, Yan Song, Si Wei, Jun-Hua Liu, Ian Vince McLoughlin, and Li-Rong Dai, "Deep bottleneck features for spoken language identification," *PLoS ONE*, vol. 9, no. 7, 2014.

[7] Yan Song, Bing Jiang, YeBo Bao, Si Wei, and Li-Rong Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[8] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," *Proc. of ICASSP*, pp. 1695–1699, 2014.

[9] Patric Kenny, Vishwa Gupta, Themos stafylakis, Pierre Quellet, and Jahangir Alam, "Deep neural networks for extracting Baum-Welch statistics for speaker recognition," *Proc. of ICASSP*, pp. 1695–1699, 2014.

[10] Luciana Ferrer, Yun Lei, Mitchell McLaren, and Nicolas Scheffer, "Study of senone-based deep neural network approaches for spoken language recognition," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 24, no. 1, pp. 105–116, 2016.

[11] Bing Jiang, Yan Song, Si Wei, Ian MclougHLin, and Li-Rong Dai, "Task-aware deep bottleneck features for spoken language identification," *Proc. of InterSpeech*, pp. 3012–3016, 2014.

[12] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," in *Audio, Speech, and Language Processing,IEEE/ACM Transactions on*, 2014, pp. 1533–1545.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. of NIPS*, pp. 1106–1114, 2012.

[14] Alicia Lozano-Diez, Ruben Zazo-Candil, Javier Gonzalez-Dominguez, Doroteo T. Toledano, and Joaquin Gonzalez-Rodriguez, "An end-to-end approach to language identification in short utterances using convolutional neural networks," in *Proc. Interspeech*, 2015.

[15] Ignacio Lopez-Moreno, Jorge Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pablo Moreno, "Automatic language identification using deep neural networks," *Proc. of ICASSP*, pp. 5337–5341, 2014.

[16] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno, "Automatic language identification using long short-term memory recurrent neural networks," *Proc. InterSpeech*, 2014.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision–ECCV 2014*, pp. 346–361. Springer, 2014.

[18] Yan Song, Ruilian Cui, Xinhai Hong, Ian Mcloughlin, Jiong Shi, and Lirong Dai, ," *Improved Language Identification using Deep Bottleneck Network*, pp. 1695–1699, 2015.

[19] Sabato Marco Siniscalchi, Jeremy Reed, Torbjørn Svendsen, and Chin-Hui Lee, "Universal attribute characterization of spoken language for automatic spoken language recognition," *Computer Speech & Language*, vol. 27, no. 1, pp. 209–227, 2013.

[20] Ian Vince McLoughlin, *Applied Speech and Audio Processing*, Cambridge University Press, 2009.

[21] Lorien Y Pratt, LY Pratt, SJ Hanson, CL Giles, and JD Cowan, "Discriminability-based transfer between neural networks," in *Advances in Neural Information Processing Systems 5*. Citeseer, 1993.

[22] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[23] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[24] Pedro A Torres-Carrasquillo, Elliot Singer, Mary A Kohler, Richard J Greene, Douglas A Reynolds, and John R Deller Jr, "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features.," *Porc. of InterSpeech*, 2002.