DNN BASED EMBEDDINGS FOR LANGUAGE RECOGNITION

Alicia Lozano-Diez[1,2], *Oldřich Plchot*[2], *Pavel Matějka*[2], *Joaquin Gonzalez-Rodriguez*[1]

[1]Audias-UAM, Universidad Autonoma de Madrid, Madrid, Spain
[2]BUT, Speech@FIT, Czech Republic

ABSTRACT

In this work, we present a language identification (LID) system based on embeddings. In our case, an embedding is a fixed-length vector (similar to i-vector) that represents the whole utterance, but unlike i-vector it is designed to contain mostly information relevant to the target task (LID). In order to obtain these embeddings, we train a deep neural network (DNN) with sequence summarization layer to classify languages. In particular, we trained a DNN based on bidirectional long short-term memory (BLSTM) recurrent neural network (RNN) layers, whose frame-by-frame outputs are summarized into mean and standard deviation statistics. After this pooling layer, we add two fully connected layers whose outputs correspond to embeddings. Finally, we add a softmax output layer and train the whole network with multi-class cross-entropy objective to discriminate between languages. We report our results on NIST LRE 2015 and we compare the performance of embeddings and corresponding i-vectors both modeled by Gaussian Linear Classifier (GLC). Using only embeddings resulted in comparable performance to i-vectors and by performing score-level fusion we achieved 7.3% relative improvement over the baseline.

***Index Terms*—** Embeddings, language recognition, LID, DNN

1. INTRODUCTION

Similarly to other fields in speech research [1], neural networks (NN) are becoming an essential building-block of different components also in language recognition systems. One of the most prominent uses of DNNs in today's language recognition systems (as well as in speaker recognition and in speech recognition) are the bottleneck features (BNF). Bottleneck features were originally developed for speech recognition [2, 3, 4] and were later very successfully applied also to language recognition [5, 6, 7, 8, 9] and speaker recognition [10, 11, 12], where they are the basis of state-of-the-art systems and where they gradually replaced traditional acoustic features like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction coefficients (PLP).

Thanks to the Speech@FIT group at Brno University of Technology for hosting Alicia Lozano-Diez during her research stay funded by Ayuda a la movilidad predoctoral para la realización de estancias breves en centros de I+D, 2016, Ministerio de Economía y Competitividad, Spain (EEBB-I-17-12491). This work was supported by projects CMC-V2: Caracterización, Modelado y Compensación de Variabilidad en la Señal de Voz (TEC2012-37585-C02-01), and DSSL: Redes Profundas y Modelos de Subespacios para Detección y Seguimiento de Locutor Idioma y Enfermedades Degenerativas a partir de la Voz (TEC2015-68172-C2-1-P), both funded by Ministerio de Economía y Competitividad, Spain; and by Czech Ministry of Interior project No. VI20152020025 "DRAPAK", by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602", and by Google Faculty Research Award.

The fact that these feature vectors are extracted for every frame of an utterance (as well as senone posteriors are used in [13]), producing a variable-length sequence, poses a challenge in modeling. This has been addressed by the concept of i-vectors, where we extract a compact fixed-length representation of a whole utterance, derived as a point estimate of the latent variable in a factor analysis model [14, 15]. In language recognition, systems based on i-vectors [16, 17] are very common and provide the state-of-the art performance with models as simple as the Gaussian Linear Classifier (GLC) [18].

More recently, the use of DNNs have been explored in text-dependent speaker recognition to replace GMM and i-vectors in order to obtain an utterance level representation usually referred to as *embedding*. The embedding is usually obtained by using a *pooling mechanism*, for example the mean over the framewise outputs of one or more layers in the DNN [19] or by the use of a recurrent NN [20].

Naturally, similar ideas have been applied also for language recognition. For instance, outputs of hidden layers are averaged over time and stacked together in [21], forming a representation for the utterance in a high-dimensional space; or in [22], where one of the layers in the DNN averages frame-by-frame activations, and posteriors produced by the network are used for LID as end-to-end approach. Also, [23] presents a DNN-based system which learns language dependent vectors with angular proximity loss function.

Recently, a fairly simple architecture that can be directly applied or modified also for LID has been developed for text-independent speaker verification in [24]. Here, embeddings are obtained as outputs of two hidden layers (after pooling mean and standard deviation over time), while training the whole network with multi-class cross-entropy for speaker identification. Subsequent modeling of embeddings with Probabilistic Linear Discriminant Analysis [25] (PLDA) achieves a performance comparable to i-vectors.

Motivated by all this, we developed a DNN-based language recognition system with embeddings, following the approach used for speaker verification in [24]. In particular, we use bidirectional long short-term memory (BLSTM) recurrent layers in order to exploit the temporal information of the signal (whose frames are represented by bottleneck features), before the sequence summarizing layer. Then, we pool together the mean and standard deviation statistics from the output of this frame-by-frame part of the DNN (over all frames of an input sequence). The pooled mean and standard deviation are then forwarded through two additional fully connected hidden layers, whose outputs will be used to extract the utterance level embeddings. Finally, a softmax layer is used as output layer, and the network is trained with multi-class cross-entropy objective to discriminate between languages.

Unlike [22], we used the extracted embeddings (instead of posteriors) to train a generative model for classification (GLC). Our architecture is also simpler than the one presented in [21], with smaller embedding layers, and yields better results.

We compare the performance of the system based on embeddings with a corresponding i-vector system that is trained using the same features. We report the performance on the NIST LRE 2015 and achieve comparable results which suggests that systems based on embeddings are a viable approach to be explored for LID. Finally, we further improve the results of the strong baseline by means of a score-level fusion which suggests that the DNN modeling has been able to extract complementary information out of the same bottleneck features.

2. THE NIST LRE 2015 DATASET

We report our results on the most recent and challenging NIST LRE 2015 benchmark [26]. The data that we used for training the DNN and subsequent Gaussian Linear Classifiers are composed of the dataset shipped by NIST for the *fixed* condition of the evaluation. Our bottleneck feature extractor was on the other hand trained on the Fisher English corpus instead of the allowed Switchboard.

The NIST LRE 2015 dataset consists of recordings from 20 different languages, clustered into six groups according to language similarities [26]. A training dataset provided by NIST was split into two disjoint parts: training and development (dev) datasets [27]. These datasets were created by randomly selecting 60% for the training part and 40% for the dev set. The segments belonging to the development set were further split into short cuts of different durations that contain from 3 to 30 seconds of speech. After splitting the data and dividing the dev segments into cuts, we ended up with 3042 segments (248 hours of speech) in training set and 42295 segments (146 hours of speech) in dev set.

Moreover, a balanced training subset (up to 15 h per per language) was randomly selected and used for the DNN training (the one used as embedding extractor) and UBM training of the reference i-vector system in order to partially compensate big differences in the amount of available training data per language. However, no data augmentation was performed for the classes with less than 15 h of speech. The resulting dataset contains 2316 segments out of original 3042. Furthermore, we randomly selected 979 segments (up to 50 per language) from the dev set for the purposes of cross-validation and model selection during DNN training.

Finally, we used the full train set (3042 segments) to train the GLC and the dev set was used to obtain calibration parameters. The systems were evaluated on the full LRE 2015 evaluation dataset, which consists of 164334 test segments of different durations.

3. DNN EMBEDDINGS FOR LANGUAGE IDENTIFICATION

In this section, we describe the proposed architecture of the DNN that is trained to classify languages and at the same time provides fixed-length embeddings that summarize the whole utterance and extract useful information about the language.

3.1. Input features

A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through a DNN and reading off the vector of values at the bottleneck layer. In this work, we feed the embedding DNN with stacked bottleneck feature (SBN) vectors.

In our case, they are extracted from a cascade of two DNNs trained for the task of automatic speech recognition (ASR). Thus, the bottleneck feature vector output by the first network is stacked in time, defining context-dependent input features for the second one (hence the term "stacked"). The input features for this first DNN are 24 log Mel-scale filter bank outputs augmented with 2 fundamental frequency features based on [28], resulting in a 26-dimensional feature vector. Then, mean subtraction is applied at the utterance level, and Hamming window followed by DCT consisting of 0th to 5th base are applied on the time trajectory of each parameter resulting in $(24 + 2) \times 6 = 156$ coefficients to feed the first network. The bottleneck outputs from the first network (80 dimensional) are sampled at times $t - 10$, $t - 5$, t, $t + 5$ and $t + 10$, where t is the index of the current frame. The resulting 400-dimensional features are then used as inputs for the second stage DNN.

Both DNNs were trained using Fisher English corpus and have the same architecture: three hidden layers with 1500 hidden units each, a bottleneck layer (dimensionality set to 80 and 30 for the first and second network, respectively), and an output layer with 9824 outputs (triphones).

The 30 dimensional bottleneck outputs from the second DNN (referred to as SBN) are the final features used in this work.

3.2. Architecture

The DNN used to extract the embeddings is depicted in Fig. 1. The structure was inspired by the architecture used in [24], but we introduced some modifications. Generally speaking, we can split the DNN into two parts separated by the pooling layer.

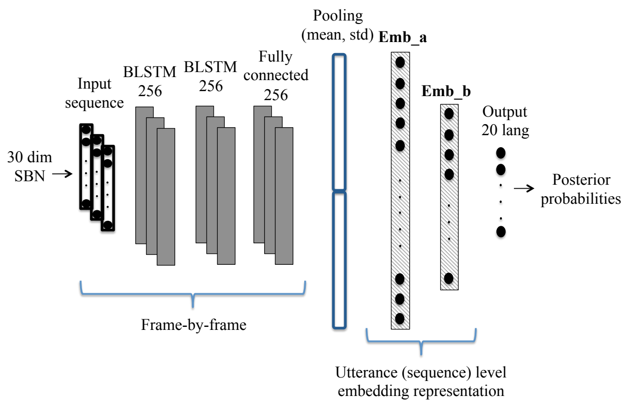

Fig. 1. Architecture of the proposed DNN for language recognition with embeddings.

The first part of the DNN (up to the pooling layer) works on a frame-by-frame basis. It consists of two bidirectional LSTM (BLSTM) layers followed by a fully connected layer, with 256 hidden units each. We used recurrent layers based on LSTM units due to their success in related works ([29, 30]) and its ability to deal with temporal information, especially over short utterances (Time Delay NN architecture (TDNN) was used in [24] instead).

Then, the pooling layer computes mean and standard deviation of the previous layer activation values (over all the frames in a given input sequence), and is further followed by two fully connected layers whose outputs are later used to extract embeddings. Finally, the output is a 20-dimensional softmax layer that provides a vector of language posterior probabilities for each utterance.

Sigmoid activation function was used as a non-linearity for all hidden units.

3.3. Training

The embedding DNN is trained to optimize multi-class cross-entropy loss function, via Adam optimizer. To reduce over-fitting, dropout rate of 30% was used in the BLSTM layers for both cells and gates. Gradients are estimated over batches of 200 samples. The maximum number of iterations (epochs) for all experiments was set to 200, and the final model was selected according to the best accuracy on the validation set.

During training, both train and validation segments were split into 3 second sequences (300 frames) with no overlap and no stacking of frames. This resulted in 173109 samples (sequences) for training (approximately 144 h of speech), and 3631 samples for validation (about 3 h of speech).

3.4. Embedding extraction

Once the DNN is trained, input features for each segment are forwarded through the network up to the embedding layers in order to obtain the embedding-based representation.

It should be pointed out that even though we feed the DNN with 3 second segments during training, this is not done for the embedding extraction. In order to extract embeddings for each utterance, we do not constrain the length of the segment to 3 seconds, but instead we forward the whole segment to obtain one embedding per utterance from each embedding layer.

The extracted embeddings are used either independently or concatenated as a fixed-length utterance representation for the LID backend (GLC).

4. LID SYSTEM BACKEND

In order to compare the performance of i-vectors and the proposed embeddings, both were modeled and calibrated in the same manner. We also briefly mention the fusion of the two systems since it brought an improvement in performance over the i-vector baseline.

4.1. Reference i-vector model

The i-vector based system used as reference in this work follows the classical i-vector pipeline [16] for LID. A diagonal-covariance UBM with 2048 components was trained using the same 30 dimensional stacked bottleneck features described in section 3.1.

For the purposes of GMM and i-vector extractor training, we used the same balanced dataset as for the embedding-DNN (up to 15 h per language). The total i-vector extractor was trained in 10 iterations and the dimensionality of i-vectors was set to 600.

4.2. Gaussian Linear Classifier

A simple Gaussian Linear Classifier [16, 18] is used on top of the i-vectors or embeddings in order to obtain the vector of 20 class-conditional log-likelihoods for each segment. Model of each language is represented by a Gaussian distribution with mean computed over i-vectors of given language and covariance matrix that is computed over all training data and which is shared across all models. This generative model is trained on the full training dataset.

4.3. Calibration and fusion

We trained a multi-class logistic regression on top of the development set scores (log-likelihoods) to obtain a scaling factor and offset vector. All of the scores (dev end eval) are then transformed (calibrated) using these parameters.

In the results section, we report also starred versions of the C_{avg} [26] (dev* and eval*). This means that instead of being trained on a separate held-out calibration set (development set), the calibration parameters were trained on that set itself. The difference between starred and non-starred C_{avg} suggests the dataset shift between development and evaluation sets or calibration problems.

We also provide results of the score level fusion in section 5.3 again obtained by means of logistic regression. Each pre-calibrated system gets a single trainable scale factor, while every language gets a trainable score offset.

The parameters are trained via optimizing multi-class cross-entropy, using a flat prior over all 20 languages for both pre-calibration and fusion.

5. EXPERIMENTS AND RESULTS

In this section, we present the results of our proposed embedding system and the corresponding baseline i-vector system in terms of average C_{avg} as defined for NIST LRE 2015 [26]. First, we analyze the performance achieved with i-vectors and embeddings of different size in Table 1. Then, encouraged by the results of smaller embeddings, we analyze the effect of further reducing the dimensionality with PCA in Tables 2 and 3. Given the fact that our DNN is trained directly for the given task, we also analyze the performance when taking DNN posteriors as scores in Table 4. Finally, we perform a score level fusion and we show in Table 5 that the DNN was able to extract complementary information and our systems fuse well with the i-vector baseline.

5.1. Embeddings

Inspired by the DNN architecture used in [24] for text-independent speaker verification, we started our experiments with two embedding layers of sizes 512 and 300 respectively. We experiment with using the two embeddings independently and stacking both of them. Results can be seen in Table 1.

Table 1. *Comparison of i-vectors with embeddings of different size.*

	$\mathrm{C_{avg}} \times 100$		
System	dev*	eval	eval*
Reference i-vector	4.54	**16.93**	**14.91**
DNN_1 emb_a (512)	5.92	20.26	18.68
DNN_1 emb_b (300)	**4.47**	19.03	16.25
DNN_1 emb_a_conc_b (812)	5.77	20.04	17.87
DNN_2 emb_a (256)	4.71	18.82	16.52
DNN_2 emb_b (150)	4.75	19.04	16.71
DNN_2 emb_a_conc_b (406)	4.88	19.19	16.84
DNN_3 emb_a (128)	5.51	19.02	17.01
DNN_3 emb_b (75)	4.76	19.05	16.62
DNN_3 emb_a_conc_b (203)	5.37	19.30	16.93

When looking at the results on dev set in Table 1, we see that embeddings achieved similar performance as the very competitive i-vector baseline. The differences in performance on the evaluation set suggest that the embeddings extracted from the discriminative model are more sensitive for the domain shift which is happening between dev and eval datasets. Similar differences between C_{avg} and C^{*}_{avg} for both embeddings and i-vectors suggest that both systems received comparable calibration.

By comparing the results of embeddings of different size, we see a better performance of DNN_2 system which produces embeddings of the half size w.r.t. DNN_1 (DNN_3 further halves the embeddings of DNN_2). This behavior was expected as compared to the system in [24] we deal with a closed-set problem and much lower number of classes. These results also suggest that the embeddings of larger size contain more detrimental information about channel as all networks reached the same performance on the training data. Motivated by this observation, we explore further dimensionality reduction via Principal Component Analysis (PCA) in Tables 2 and 3.

As shown in Table 2, reducing the dimensionality of concatenated embeddings from DNN_1 by PCA improves the results. The best performance is achieved by keeping the first 100 dimensions which results in approximately 7% relative improvement on eval.

Table 2. *Applying PCA to concatenated DNN_1 embeddings.*

	$C_{avg} \times 100$		
PCA dim	dev*	eval	eval*
No PCA (812)	5.77	20.04	17.87
500	5.02	19.26	16.55
300	4.38	18.79	15.99
100	**3.89**	**18.67**	**16.05**
25	5.67	19.98	17.91

Finally, we compare the PCA post-processing using the concatenated embeddings from all DNNs in Table 3. We can observe that with smaller embeddings, we are able to reduce the dimensionality further to 25 and still gain some performance improvement. We achieved the best results with embeddings from DNN_2 whose concatenated dimensionality 406 is close to the typical i-vector (400 or 600). In terms of C_{avg}, we achieved the performance of 17.44% which is already close to our i-vector baseline with 16.93%.

Table 3. *Applying PCA to concatenated embeddings.*

	$C_{avg} \times 100$ (eval)		
System	None	100	25
DNN_1 emb_a_conc_b (orig 812)	20.04	18.67	19.98
DNN_2 emb_a_conc_b (orig 406)	19.19	18.11	**17.44**
DNN_3 emb_a_conc_b (orig 203)	19.30	18.70	18.13

5.2. Posteriors

As our DNNs for embedding extraction are in fact trained to discriminate between languages, and softmax layer outputs corresponding posterior probabilities for each of the 20 target languages, we also analyze the use of these posteriors directly as scores. This approach corresponds to the end-to-end training and the results are summarized in Table 4. Although results on the dev set are similar to the i-vector system, this approach seems to generalize slightly worse than raw embeddings.

Comparing results in Tables 3 and 4, we can observe that PCA-reduced embeddings safely outperformed the system based on posteriors. Our results suggest that using embeddings together with a simple model (GLC) is a viable research direction, especially if we consider that the DNN does not necessarily have to be trained exactly on the target languages.

Table 4. *Performance of individual DNN systems when taking posterior probabilities as scores.*

	$C_{avg} \times 100$		
System	dev*	eval	eval*
Ref. i-vector	4.54	16.93	14.91
DNN_1 posteriors	4.90	20.37	17.26
DNN_2 posteriors	3.94	19.68	16.64
DNN_3 posteriors	5.00	19.76	16.95

5.3. Fusion

In this section, we present improvements obtained with a simple score level fusion (as described in section 4.3) between the reference i-vector system and the best embedding-based system presented in previous sections (DNN_2 emb_a_conc_b + 25 dimensional PCA). We also show the fusion of the reference i-vector system and the posterior probability outputs from the network. Looking at Table 5, we can see that both fusions improve results, suggesting that these different approaches can extract complementary information from the same input features.

Table 5. *Comparison of single systems and their fusions.*

	$C_{avg} \times 100$		
System	dev*	eval	eval*
(1) Ref. i-vector	4.54	16.93	14.91
(2) DNN_2 emb_a_conc_b + PCA (25)	3.67	17.44	15.86
(3) DNN_2 posteriors	3.94	19.68	16.64
Score fusion (1)+(2)	**2.99**	**15.69**	13.52
Score fusion (1)+(3)	3.04	16.41	**13.19**

6. CONCLUSIONS

In this work, we presented a DNN architecture with embeddings for language identification. The DNN is trained to discriminate between languages and at the same time learns an utterance level representation of input segments. These fixed-length representations i.e., embeddings are further used to train a generative classifier (GLC). This is a novel approach for the problem of LID and is in line with the research that has proven to be promising for speaker verification [24]. Our results are comparable with a state-of-the-art i-vector system on the NIST LRE 2015 setup. Also, we would like to highlight that using embeddings instead of posteriors from the DNN provided better results, and points towards the direction of training more general DNNs i.e., trained on much larger and variable set of languages, that can provide more general embeddings usable across various LID tasks.

7. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath,

and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[2] V. Fontaine, C. Ris, J-M. Boite, and Multitel Site Initialis, “Nonlinear discriminant analysis for improved speech recognition,” in *Proc. Eurospeech-97, Rhodes*, 1997, pp. 4–2071.

[3] František Grézl, Martin Karafiát, and Lukáš Burget, “Investigation into bottle-neck features for meeting speech recognition,” in *Proc. Interspeech 2009*, 2009, number 9, pp. 2947–2950.

[4] Markus Müller, Sebastian Stüker, and Alex Waibel, “Language adaptive dnns for improved low resource speech recognition,” in *Proceedings of Interspeech 2016*, 2016, pp. 3878–3882.

[5] Alicia Lozano-Diez, Ruben Zazo, Doroteo T. Toledano, and Joaquin Gonzalez-Rodriguez, “An analysis of the influence of deep neural network (dnn) topology in bottleneck feature based language recognition,” *PLoS ONE*, vol. 12, no. 8, pp. e0182580, August 2017.

[6] Radek Fér, Pavel Matějka, František Grézl, Oldřich Plchot, and Jan Černocký, “Multilingual bottleneck features for language recognition,” in *Proceedings of Interspeech 2015*, 2015, vol. 2015, pp. 389–393.

[7] Pavel Matějka, Le Zhang, Tim Ng, Harish Sri Mallidi, Ondřej Glembek, Jeff Ma, and Bing Zhang, “Neural network bottleneck features for language identification,” in *Proceedings of Odyssey 2014*. 2014, International Speech Communication Association.

[8] F. Richardson, D. Reynolds, and N. Dehak, “Deep neural network approaches to speaker and language recognition,” *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, Oct 2015.

[9] 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, pp. e100795, 2014.

[10] Daniel Garcia-Romero and Alan McCree, “Insights into deep neural networks for speaker recognition,” in *Proceedings of Interspeech 2015*, 2015, pp. 1141–1145.

[11] Sibel Yaman, Jason Pelecanos, and Ruhi Sarikaya, “Bottleneck features for speaker recognition,” in *Proceedings of Odyssey 2012*. 2012, International Speech Communication Association.

[12] Alicia Lozano-Diez, Anna Silnova, Pavel Matějka, Ondřej Glembek, Oldřich Plchot, Jan Pešán, Lukáš Burget, and Joaquin Gonzalez-Rodriguez, “Analysis and optimization of bottleneck features for speaker recognition,” in *Proceedings of Odyssey 2016*. 2016, International Speech Communication Association.

[13] Luciana Ferrer, Yun Lei, Mitchell McLaren, and Nicolas Scheffer, “Spoken language recognition based on senone posteriors,” in *Proceedings of Interspeech 2014*, 2014, pp. 2150–2154.

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

[15] P. Kenny, P. Oullet, V. Dehak, N. Gupta, and P. Dumouchel, “A Study of Interspeaker Variability in Speaker Verification,” *IEEE Trans. on Audio, Speech and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[16] David González Martínez, Oldřich Plchot, L. Burget, O. Glembek, and P. Matějka, “Language recognition in ivectors space,” in *Proceedings of Interspeech 2011*, 2011, pp. 861–864.

[17] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas A. Reynolds, and Réda Dehak, “Language recognition via i-vectors and dimensionality reduction,” in *Proceedings of Interspeech 2011*, 2011, pp. 857–860.

[18] Sandro Cumani, Oldřich Plchot, and Radek Fér, “Exploiting i-vector posterior covariances for short-duration language recognition,” in *Proceedings of Interspeech 2015*, 2015.

[19] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, “Deep neural networks for small footprint text-dependent speaker verification,” in *Proceedings of ICASSP*, May 2014.

[20] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, “End-to-end text-dependent speaker verification,” in *Proceedings of ICASSP*, March 2016.

[21] Ruizhi Li, Harish Sri Mallidi, Oldřich Plchot, Lukáš Burget, and Najim Dehak, “Exploiting hidden-layer responses of deep neural networks for language recognition,” in *Proceedings of Interspeech 2016*, 2016.

[22] Jan Pešán, Lukáš Burget, and Jan Černocký, “Sequence summarizing neural networks for spoken language recognition,” in *Proceedings of Interspeech 2016*, 2016, pp. 3285–3289.

[23] G. Gelly and J.L. Gauvain, “Spoken language identification using lstm-based angular proximity,” in *Proceedings of Interspeech 2017*, 2017.

[24] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in *Proceedings of Interspeech 2017*, 2017.

[25] S. J. D. Prince, “Probabilistic linear discriminant analysis for inferences about identity,” in *Proc. International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007.

[26] “The 2015 NIST Language Recognition Evaluation Plan (LRE15),” http://www.nist.gov/itl/iad/mig/upload/LRE15_EvalPlan_v23.pdf.

[27] Oldřich Plchot et al., “Bat system description for nist lre 2015,” in *Proceedings of Odyssey 2016*. 2016, International Speech Communication Association.

[28] David Talkin, “A robust algorithm for pitch tracking (RAPT),” in *Speech Coding and Synthesis*, W. B. Kleijn and K. Paliwal, Eds., New York, 1995, Elseviever.

[29] 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,” in *Proceedings of Interspeech 2014*, 2014.

[30] Ruben Zazo, Alicia Lozano-Diez, Javier Gonzalez-Dominguez, Doroteo T. Toledano, and Joaquin Gonzalez-Rodriguez, “Language identification in short utterances using long short-term memory (lstm) recurrent neural networks,” *PLOS ONE*, vol. 11, no. 1, pp. 1–17, 01 2016.