The Speaker and Language Recognition Workshop (Odyssey 2018)
26-29 June 2018, Les Sables d'Olonne, France

Analysis of DNN-based Embeddings for Language Recognition on the NIST LRE 2017

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

[1]Audias-UAM, Universidad Autonoma de Madrid, Madrid, Spain
[2]Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Czechia
alicia.lozano@uam.es, {iplchot, matejkap, inovoton}@fit.vutbr.cz

Abstract

In this work, we analyze different designs of a language identification (LID) system based on embeddings. In our case, an embedding represents a whole utterance (or a speech segment of variable duration) as a fixed-length vector (similar to the i-vector). Moreover, this embedding aims to capture information relevant to the target task (LID), and it is obtained by training a deep neural network (DNN) 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 for each utterance. After this pooling layer, we add two fully connected layers whose outputs are used as embeddings, which are afterwards modeled by a Gaussian linear classifier (GLC). For training, we add a softmax output layer and train the whole network with multi-class cross-entropy objective to discriminate between languages. We analyze the effect of using data augmentation in the DNN training, as well as different input features and architecture hyper-parameters, obtaining configurations that gradually improved the performance of the embedding system. We report our results on the NIST LRE 2017 evaluation dataset and compare the performance of embeddings with a reference i-vector system. We show that the best configuration of our embedding system outperforms the strong reference i-vector system by 3% relative, and this is further pushed up to 10% relative improvement via a simple score level fusion.

1. Introduction

Deep neural networks (DNN) are nowadays a fundamental part of speech processing systems [1] and, in particular, their use in language recognition systems has outperformed many classical frameworks such as i-vectors based on acoustic features like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction coefficients (PLP).

These DNNs have been included in different stages on language recognition pipeline. One of their most common applications is to function as feature extractors, where a DNN is trained to discriminate between phonetic units and then, the output of a relatively small hidden layer (bottleneck) is used as a new frame-by-frame representation of the input signal, replacing MFCC or other traditional features. These bottleneck features are widely used in language recognition [2, 3, 4, 5, 6] and speaker recognition [7, 8, 9, 10], after their success in speech recognition [11, 12].

The fact that these feature vectors are extracted for every frame of an utterance (as is the case also for MFCCs or PLPs), producing a variable-length sequence, poses a challenge in further 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 maximum a posteriori (MAP) point estimate of the latent variable in a factor analysis model [13, 14]. In language recognition, systems based on i-vectors [15, 16] are very common and provide the state-of-the art performance with classifiers as simple as the Gaussian Linear Classifier (GLC) [17].

Recently, approaches based on DNNs with a *pooling mechanism*, for obtaining an utterance level representation (usually referred to as *embedding*), have been explored in speaker recognition [18, 19, 20, 21].

In [22] a similar architecture to the one used in [20] has been applied for the task of language identification (LID), showing comparable performance to the one achieved by a competitive i-vector system on the challenging NIST LRE 2015 dataset [23]. In that work, embeddings are obtained as outputs of two hidden layers that follow after the pooling layer, which is computing the mean and standard deviation over all frame-by-frame outputs from a previous layer. The whole network is trained with multi-class cross-entropy objective to classify languages and extracted embeddings are then modeled same as i-vectors with a simple Gaussian Linear Classifier (GLC).

In this work, we continue the research started in [22] while exploring and analyzing DNN embeddings for language recognition on the most recent NIST LRE 2017. In particular, we describe the embedding subsystem submitted to this last NIST LRE [24], and analyze further improvements made after the evaluation period. We study the influence of increasing the size of our original system architecture, as well as the performance when using different input features and a wide range of augmented training data. Performance for different test segment duration is also analyzed.

We compared the system based on embeddings with a cor-

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 program.

10.21437/Odyssey.2018-6

responding i-vector system, both trained on the same features, and we were able to gradually improve the performance of the embedding system via increasing the model size and the amount of training data until we clearly outperformed the system based on i-vectors. We achieved further improvement by a score-level fusion of both systems, i-vector and embedding, suggesting that embeddings are able to extract complementary information to i-vectors.

2. The NIST LRE 2017 database

In this section, we describe the original database provided by NIST for LRE 2017, and explain how the data has been augmented by adding noise and reverberation.

2.1. Original data provided by NIST

We experiment with the most recent NIST LRE 2017 [25]. This benchmark consists of five clusters of similar languages, with a total of fourteen different languages, as shown in Table 1.

Table 1: *Language and clusters of the NIST LRE 2017 dataset.*

Cluster	Languages
Arabic	Egyptian Arabic, Iraqi Arabic, Levantine Arabic, Maghrebi Arabic
Chinese	Mandarin, Min Nan
English	British English, General American English
Slavic	Polish, Russian
Iberian	Caribbean Spanish, European Spanish, Latin American, Continental Spanish, Brazilian Portuguese

In particular, we focused on the primary (fixed) condition (except for the experiments with multilingual bottleneck features) and we used all data supplied by NIST (training and development) for this condition. All data was down-sampled to 8kHz.

For training the neural network used as bottleneck feature extractor, we used the annotated Fisher I and II databases provided for the evaluation, with three copies of noisy and reverberated variants of the original audio files. For the experiments with multilingual bottleneck features, this neural network was trained using the 17 languages from BABEL program dataset [1].

In order to train the embedding neural network, we used the set of training data released as such by the organizers, which consists of 16205 utterances. We added two thirds of the development data to this set, using the remaining third as a validation set. We cut long segments from the development set expanding it to 6090 segments. Therefore, 20301 segments were used for training the embedding extractor and 1994 were used for validation (model selection).

For training the UBM and i-vector extractor of the reference system, just the LRE 2017 training data was used.

Regarding the final classification stage, the training data and the two thirds of the development set were also utilized to train the Gaussian Linear Classifier (GLC) used as backend for both embeddings and i-vectors. Calibration and fusion parameters were obtained using the whole development set (6090 segments).

[1] Collected by Appen, `http://www.appenbutlerhill.com`

Finally, the systems were evaluated on the full LRE 2017 evaluation dataset, which consists of 25451 test segments of approximately 3, 10 or 30 seconds of speech.

2.2. Augmented dataset

In order to analyze how the data augmentation affects the performance of embedding systems, we used different techniques to obtain corrupted copies of the DNN training data (including the two thirds of development data used as training). The augmented data were then used together with the original audio files to train the DNN used for embedding extraction. In particular, we added noise and reverberation and we also changed the tempo (speed) of the recordings.

For the noisy dataset, we prepared a set of noises that consists of three sources of different types of noise:

- 272 samples taken from the Freesound library [2] (real fan, HVAC, street, city, shop, crowd, library, office and workshop).
- 7 samples of artificially generated noises: various spectral modifications of white noise + 50 and 100 Hz hum.
- 25 samples of babbling noises by merging speech from 100 random speakers from Fisher database using speech activity detector.

These noises were then added to original training data at SNR levels sampled from two ranges: 0-8 dB and 8-20 dB.

Regarding reverberation, we used a set that consists of real Room Impulse Responses (RIR) from several databases: AIR [26], C4DM [27, 28], MARDY [29], OPENAIR [30], RVB 2014 [31], RWCP [32]. Together, they form a set of RIR that simulates different types of rooms: small rooms, big rooms, lecture room, restrooms, halls, stairs, etc. All room models have more than one impulse response per room, i.e. different RIR was used for source of the signal and source of the noise to simulate different locations of their sources.

We also changed the audio playback speed (tempo) to 0.9 and 1.1 of original speed. We used SoX tool which changes speed of audio files but keeps their original pitch.

Finally, these perturbations are combined so that reverberation is applied to audio files already corrupted with noise, or noise is added on top of audio files where the tempo has been modified (referred to as *noised tempos* in the experimental part).

3. Stacked bottleneck 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 [33], 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 $(24+2)\times 6 = 156$

[2] `http://www.freesound.org`

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.

DNNs used as bottleneck feature extractors have the same architecture: three hidden layers with 1500 hidden units each, a bottleneck layer and an output layer. The dimensionality of the bottleneck layer is set to 80 for the first network and either 30 or 80 (depending on the experiment) for the second one. For the experiments with monolingual bottleneck features (SBN30), we trained the networks with Fisher English (I and II), and with 9824 outputs (triphones). For experiments with multilingual bottleneck features (SBN80, ML), we used BABEL dataset with 17 languages and 15558 outputs (tied triphones per each language). This network is trained using block softmax as output layer [34].

The outputs from the bottleneck layer of the second DNN (referred to as SBN) are the final features used in this work as input vectors for both embedding and i-vector systems.

4. DNN-based embedding system

In this section, we describe the embedding DNN used in this work. This DNN is trained to classify the 14 target languages, and it is used as embedding extractor. It is meant to provide fixed-length embeddings that summarize the whole segment or utterance and extract useful information about the language. Thus, it is not used as end-to-end system for LID from the output posterior probabilities but as embedding extractor. Classification is further performed through a Gaussian linear backend (described in section 6).

4.1. Architecture

The architecture of the embedding DNN used in this work follows the same structure as the one used in [22], with some changes for different experiments.

The DNN consists of two parts separated by the pooling (summarizing) layer. The first part, which works on a frame-by-frame basis, comprises two bidirectional LSTM (BLSTM) layers with 256 cells each, followed by a fully connected layer. These recurrent layers based on BLSTMs have proven their ability to deal with temporal information without stacking of input features (especially for very short segments in LID) [35] as they take into account the information learned from previous (and following) frames in the input sequence of features. After these BLSTM layers, we add a single fully connected layer, which size is set to 256 (for *small* DNN) or 1500 (for *large* DNN) depending on the experiment.

A subsequent pooling layer computes mean and standard deviation statistics over the frame-by-frame outputs of the previous layer, summarizing the information of a given input sequence. The output of this pooling layer is forwarded through two additional fully connected layers, whose output values will be later used as embedding representations of the input utterance. The size of these embedding layers are set to 512 and 300 for the *small* architecture, or 512 each for the *large* network.

Finally, the output is a 14-dimensional softmax layer that provides a vector of language posterior probabilities for each utterance. We used a sigmoid activation function as a non-linearity for all hidden units. An example of this architecture is depicted in Figure 1.

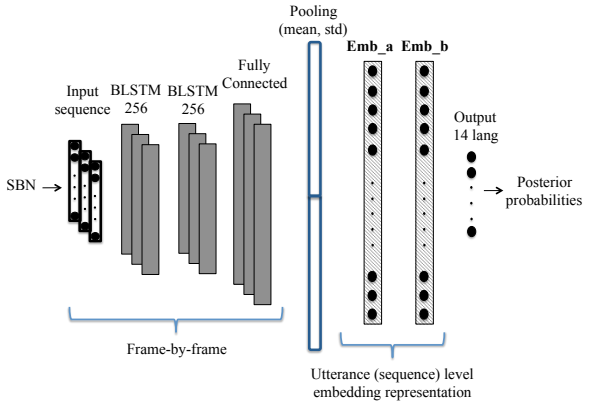

Figure 1: *Architecture of the proposed embedding DNN for language recognition. The size of the layers that are not specified in this figure varies according to the experiment.*

4.2. Training

The DNN used to extract the embedding representations of utterances is first trained to classify sequences among the set of 14 target languages. The loss function that is optimized is multi-class cross-entropy, via Adam optimizer. To reduce overfitting, dropout rate of 30% was used in all layers, including also dropout for gates on the recurrent BLSTM layers. The maximum number of iterations (epochs) for all experiments was set to 400, and the final model was selected according to the best accuracy on the validation set.

During training, gradients are estimated over batches of 210 samples. These batches are created randomly selecting 9 seconds of speech (3 fragments of 3 seconds of speech) from 70 different input audio files. The length of input sequences is set to 3 seconds of speech (300 frames). One epoch is completed when all the files from the input list are seen by the network. The training list is different depending on the experiment, and therefore, different number of hours are used for training depending on the available data for that experiment (amount of augmented data), ranging from approximately 50.7 to 558 h per epoch (from the experiments with just original audio files to the experiment with 11 copies).

Validation segments are the same in all the experiments, and were split into 3 second sequences, resulting in 23782 samples for validation (about 19 h of speech).

4.3. Embedding extraction

For the embedding extraction, input features for each segment are forwarded through the already trained DNN up to the embedding layers in order to obtain the embedding-based representation. Thus, the output layer is not used in this stage.

Even though we feed the DNN with a fixed length of sequences of 3 seconds during training for our experiments (except one experiment presented in section 7.4), the length of the segments for the embedding representation extraction is not constraint. Instead, for this forward pass, the whole segment is used as input to the DNN to obtain one embedding per utterance from each embedding layer.

Finally, the extracted embeddings are concatenated as a fixed-length utterance representation for the LID backend (GLC).

5. Reference i-vector system

In order to compare performance of embeddings for the LID task, we use as reference two state-of-the-art i-vector based systems. These systems follow the classical i-vector pipeline [15] for LID.

In particular, two diagonal-covariance UBMs with 2048 and 4096 Gaussian components respectively were trained using the same 30 dimensional stacked bottleneck features that we used to train the embedding DNN, described in section 3. We also trained the systems after adding their delta coefficients, i.e. using 60-dimensional input feature vectors.

For the purposes of GMM and i-vector extractor training, we used only the original LRE 2017 training dataset (with no augmentation and without the two thirds of development data). The total i-vector extractor was trained in 10 iterations and the dimensionality of i-vectors was set to 600 and 800 for each of the systems.

6. LID backend

6.1. Gaussian Linear Classifier (GLC)

The Gaussian Linear Classifier (GLC) is a simple and commonly used backend for LID [15, 17].

This backend is used in our experiments on top of both i-vectors and embeddings, in order to obtain the vector of 14 class-conditional log-likelihoods for each segment.

The model of each language is represented by a Gaussian distribution with mean computed over i-vectors or embeddings of each given language and a shared covariance matrix that is computed over all training data as a weighted average of within-class covariance matrices. This generative model is trained on the full training list, with no data augmentation.

6.2. Calibration and fusion

After scoring, our score vectors obtained as outputs of the GLC are pre-calibrated. Also, in section 7.5, we present a score level fusion of two systems (i-vector and embeddings) which are previously pre-calibrated.

For the purpose of pre-calibration and fusion, a simple solution was chosen to avoid over-training. Thus, a multi-class logistic regression model is trained using the scores (log-likelihoods) from the development set (including our short cuts).

For pre-calibration, each individual system has a trainable scale factor and an offset vector. For fusion, each system gets a single trainable scale factor, while every language gets a trainable score offset. The parameters are trained via optimizing prior-weighted multi-class cross-entropy, using an uniform (flat) prior over all 14 languages for both pre-calibration and fusion.

6.3. Evaluation metrics

In order to compare the results of the experiments in this work, we use the primary metric for LRE 2017 $C_{primary}$ as defined in the evaluation plan [25], computed with the available NIST tool.

Moreover, we show some results using the $\mathrm{C_{avg}}$ metric used in previous LRE 2015 evaluation [23].

7. Experiments and results

7.1. Results for reference i-vector systems

To establish baselines for our embedding systems, we list the results of the i-vector systems based on the same 30 dimensional SBN features in Table 2. We present two variants with smaller and larger UBM and i-vector size. The performance of a larger system with SBN30D features is clearly the best and would be also among the best single systems we developed for NIST LRE 2017 [36]. For the smaller system we also present the variant without including the delta coefficients to have an exact match of the input features with many of our embedding systems.

Table 2: *Results of i-vector reference systems comparing different input features (SBN30 with and without deltas), and size of UBM and i-vector dimensionality. GLC backend is used on top of the i-vectors.*

	$C_{primary} \times 100$	
Input features	2048/600	4096/800
SBN30	23.58	-
SBN30D (+deltas)	19.80	18.53

7.2. Experiments on input features and different architectures for embeddings

In this section we present results of embedding systems with larger and smaller model and compare three different input features and two training sets. Results are summarized in Table 3.

Table 3: *Results comparing different input features (SBN), training data lists and architectures of the DNN. Both columns show results on stacked embeddings (a+b) with GLC backend.*

	$C_{primary} \times 100$	
	Small (812)	Large (1024)
Max. 15h per language, SBN30	24.07	22.20
Full list, SBN30	22.18	19.86
Full list, SBN30D (+deltas)	21.69	18.95
Full list, SBN80 (ML)	23.41	20.18
Full list, SBN30 + 2 noises	20.60	19.03
Full list, SBN80 (ML) + 2 noises	20.62	18.32

The two architectures used for these experiments, referred to as *small* and *large*, follow the structure shown in Figure 1. They consist of an input layer (with 30, 60 or 80 input units depending on the features), followed by two BLSTM layers (256 cells each) and a fully connected layer, which size is set to 256 for the *small* network and 1500 units for the *large* one. Then, after the pooling layer, the two hidden layers whose outputs are used as embeddings, have 512 and 300 units for the *small* network, and 512 units each for the *large* architecture. Output layers are both 14 dimensional layers, corresponding to the target languages of this work.

As we can see, the *large* configuration outperforms the *small* one in all the cases. The stacked embeddings are 1024-dimensional for the big network, and 812-dimensional for the other, which might influence these gains. Moreover, increasing the size of the layer before pooling allows the network to capture more information from the frame based part, and these

larger statistics that are further propagated through the DNN should contain a better summary for the utterance level side of the network.

In Table 3, we compare results on two different training lists. The first row shows results using a training list that constrains the maximum number of hours per language to 15 h, which was our submission for the NIST LRE 2017. This was done in order to partially compensate the unbalanced dataset available. However, using the full training list has proven to work better in our setup.

Furthermore, we explore three different input features. All of them are stacked bottleneck (SBN) features extracted as described in section 3. We experimented with 30-dimensional SBN (SBN30), trained with just English data, and their 60-dimensional variant that contains delta coefficients (SBN30D). According to these results, including this temporal information at the input seems beneficial for the task, even though the architecture based on BLSTM is also taking into account the context of each frame given in the input sequence. These delta coefficients also helped the i-vector reference system. We also used 80-dimensional SBN from a multilingual network, which outperforms systems using SBN30 when noisy data is included in the training list. This suggests that increasing the input feature size (and therefore the whole model size) makes the network more prone to overfitting or simply too large to train on a smaller dataset and that can be partially compensated by augmenting the training data with their noisy versions. The last two rows of the table show that adding noisy data outperforms results with just original data for both architectures and different input features. Next section further explores and extends this training data augmentation.

7.3. Experiments on data augmentation for embeddings

Deep neural networks are known to be prone to overfitting, which in our setup was supported by the existing gap in performance between the held-out subset separated from the development set (whose remaining two thirds were included in the training set for the embedding DNN) and the actual evaluation dataset.

Increasing the amount of training data partially solves that problem. In our case, we extended the training dataset by performing data augmentation through addition of noise, reverberation and tempo variations of original audio files as described in section 2.2.

In Table 4, we compare the performance of embedding systems trained with up to 11 copies of the original training data with different augmentations. These results are also presented graphically in Figure 2 showing the $C_{primary} \times 100$ metric and grouped by number of copies. Although we will discuss results using the primary metric of the NIST LRE 2017, Table 4 also includes results on $\mathrm{C_{avg}}$ as defined in LRE 2015 for the sake of comparison with results split per test segment durations in section 7.4.

General trends show that increasing the number of copies of the data yields improvements in performance. In particular, adding any noisy version of the data (combined or not with other corruptions) for training the embedding extractor makes the system more robust against data mismatch and brings gradual performance gains. The only two cases in which data augmentation does not improve the system trained only on original data are the ones in which just reverberation or tempo variations without any noise addition are performed over the original audio files. In these cases, the performance is even slightly degraded.

Table 4: *Results with different data augmentations for DNN training with GLC backend. Results are shown as Equalized* $C_{primary} \times 100$ *from NIST LRE 2017 and* $\mathrm{C_{avg}} \times 100$ *from LRE 2015.*

Training data	$C_{primary}$	$\mathrm{C_{avg}}$
Original	19.86	3.97
+ 1 reverb (clean)	20.33	4.06
+ 1 reverb (noise 8-20dB)	19.01	3.90
+ 1 noise (8-20dB)	19.40	3.77
+ 2 noises	19.03	3.85
+ 2 tempos	20.06	3.97
+ 2 reverbs	19.40	3.99
+ 2 noised tempos	18.84	3.73
+ 2 tempos + 1 noise (8-20dB)	19.57	4.00
+ 2 noises + 2 tempos	19.37	3.81
+ 4 noised tempos	18.95	3.93
+ all (4) noises (8-20dB)	18.64	3.79
+ all (7) noises	18.69	3.82
+ all (10) augmentations	17.96	3.67

We have also performed a similar analysis for the i-vector system, where we did not see any benefits from data augmentation [24].

7.4. Results per duration

In order to evaluate how the embedding systems are influenced by the use of 3 second sequences for training, we show some examples of performance split by test segment durations (3, 10, 30 seconds and all pooled together) in Figure 3.

In this figure, we show the results in terms of $\mathrm{C_{avg}} \times 100$ and we experiment with the system from the fourth row in Table 4 (with one noisy copy of the data). Blue bars correspond to training with fixed-length 3 second sequences and yellow bars correspond to the system where each minibatch was selected to contain sequences of random duration between 2 and 3 seconds. Despite the restriction in input sequences length, the blue system still performs better even for the shortest duration condition (3s) and keeps a small performance gain also for longer durations. So far, we were not able to exploit variable training segment durations to achieve better performance or robustness of our DNN embedding architecture.

In Figure 4, we also present the comparison of performance between the best embedding system (the last row of Table 4) and the best i-vector system (SBN30D, 4096/800). The embedding system outperforms the i-vector system for all durations, increasing the gap in performance for shorter durations.

7.5. Fusion i-vector and embeddings

Both i-vector and embedding extractors are trained on top of the same bottleneck features. However, these utterance representations are extracted with very different models, which are likely to leverage different information contained in the initial frame representation of the signal.

This hypothesis is supported by the 10% relative improvement w.r.t. the best reference i-vector system (see Table 5), obtained with a simple score level fusion of the best i-vector and embedding system. This result suggests that i-vectors and embeddings are complementary representations of the same utter-

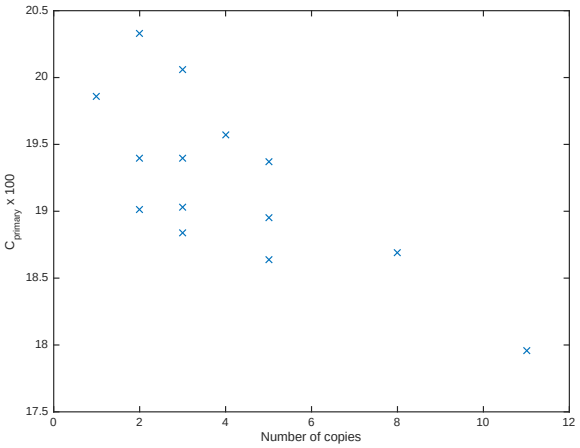

Figure 2: *Influence of the number of copies of the original data used to train the DNN in the performance ($C_{primary} \times 100$) of the resulting embeddings for LID.*

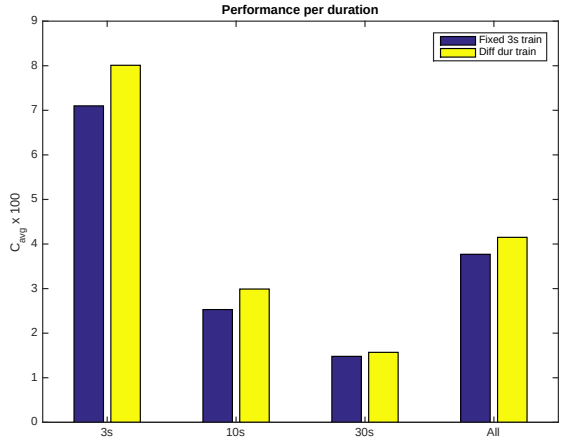

Figure 3: *Comparison of performance (in terms of* $\mathrm{C_{avg}} \times 100$*) split by test segment duration between two embedding systems with fixed or variable sequence length in the DNN training.*

ances, which can be exploited to create a better and more robust LID system.

8. Conclusions

Recently, DNNs trained to extract utterance level embeddings have become a fair competitor to traditional i-vectors for both speaker [20] and language recognition [22]. In this work, we explored an architecture for the embedding DNN based on BLSTMs and we analyzed the effect of data augmentation and different configurations of bottleneck features on DNN training and performance of the LID system. We have also compared how are the embedding systems performing with segments of different durations and compared the results with i-vectors.

We have performed our analyses on the most recent NIST LRE 2017 and our results suggest that the proposed DNN systems are data-hungry, and that adding noise to the original training data in combination with other perturbations such as reverberation or speed changes significantly improves the performance.

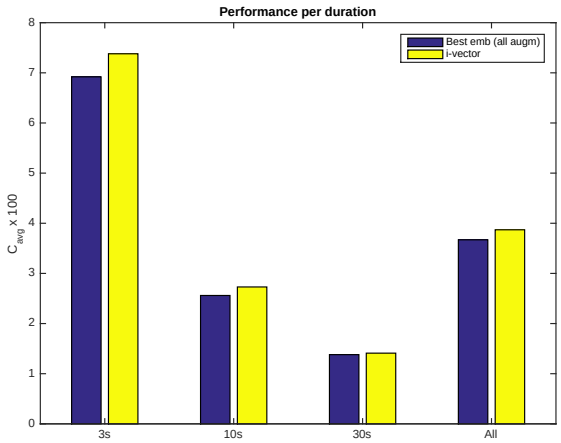

Figure 4: *Comparison of performance ($\mathrm{C_{avg}} \times 100$) per duration between the best embedding system (all augmentations) and the best reference i-vector results.*

Table 5: *Results of i-vector system, best embedding system and score level fusion.*

System	$C_{primary} \times 100$
i-vector (4096/800)	18.53
Embeddings (best, all augm)	17.96
Fusion (score level)	16.66

The best configuration of the embedding system outperforms already the strong i-vector system, and a simple score level fusion of both approaches is able to further improve the overall LID performance. We consider DNN embeddings to be a promising line for research in language recognition and related fields, since they provide compact utterance representations and achieve state-of-the-art results.

9. 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] 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.

[3] 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, International Speech Communication Association.

[4] 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.

[5] 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.

[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, pp. e100795, 2014.

[7] Daniel Garcia-Romero and Alan McCree, "Insights into deep neural networks for speaker recognition," in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, 2015, pp. 1141–1145.

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

[9] Mitchell McLaren, Yun Lei, and Luciana Ferrer, "Advances in deep neural network approaches to speaker recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, 2015, pp. 4814–4818.

[10] 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.

[11] 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.

[12] 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, International Speech Communication Association.

[13] 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.

[14] 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.

[15] 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.

[16] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas A. Reynolds, and Réda Dehak, "Language recognition via i-vectors and dimensionality reduction," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, 2011, pp. 857–860.

[17] Sandro Cumani, Oldřich Plchot, and Radek Fér, "Exploiting i-vector posterior covariances for short-duration language recognition," in *Proceedings of Interspeech 2015*. 2015, International Speech Communication Association.

[18] 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.

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

[20] 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.

[21] K. Chen and A. Salman, "Learning speaker-specific characteristics with a deep neural architecture," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1744–1756, Nov 2011.

[22] Alicia Lozano-Diez, Oldřich Plchot, Pavel Matějka, and Joaquin Gonzalez-Rodriguez, "Dnn based embeddings for language recognition," in *Proceedings of ICASSP*, April 2018.

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

[24] Oldřich Plchot, Pavel Matějka, Ondřej Novotný, Sandro Cumani, Alicia Lozano-Diez, Josef Slavicek, Mireia Diez, František Grézl, Ondřej Glembek, Kamsali Veera Mounika, Anna Silnova, Lukáš Burget, Lucas Ondel, Santosh Kesiraju, and Johan Rohdin, "Analysis of but-pt submission for nist lre 2017," in *Proceedings of Odyssey 2018*, 2018.

[25] "NIST 2017 Language Recognition Evaluation Plan," https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf.

[26] "Aachen impulse response database," http://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/.

[27] "C4dm (center for digital music) RIR database," http://isophonics.net/content/room-impulse-response-data-set.

[28] R. Stewart and M. Sandler, "Database of omnidirectional and b-format room impulse responses," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2010, pp. 165–168.

[29] "Multichannel acoustic reverberation database at york," http://www.commsp.ee.ic.ac.uk/ sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/.

[30] "Openair impulse response database," http://www.openairlib.net/auralizationdb.

[31] "Reverb challenge," reverb2014.dereverberation.com.

[32] "Rwcp sound scene database," http://www.openslr.org/13/.

[33] 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.

[34] Radek Fer, Pavel Matejka, Frantisek Grezl, Oldrich Plchot, Karel Vesely, and Jan Honza Cernocky, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech & Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[35] Hasim Sak Joaquin Gonzalez-Rodriguez Pedro J. Moreno Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Proceedings of Interspeech 2014*, 2014.

[36] Oldřich Plchot, Pavel Matějka, Radek Fér, Ondřej Glembek, Ondřej Novotný, Jan Pešán, Karel Veselý, Lucas Ondel, Martin Karafiát, František Grézl, Santosh Kesiraju, Lukáš Burget, Niko Brummer, Preez du Albert Swart, Sandro Cumani, Harish Sri Mallidi, and Ruizhi Li, "Bat system description for nist lre 2015," in *Proceedings of Odyssey 2016*. 2016, International Speech Communication Association.