Odyssey 2016
21-24 Jun 2016, Bilbao, Spain

Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks

Ruben Zazo, Alicia Lozano-Diez,
Joaquin Gonzalez-Rodriguez

ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain
{ruben.zazo, alicia.lozano} @uam.es

Abstract

Long Short-Term Memory recurrent neural networks (LSTM RNNs) provide an outstanding performance in language identification (LID) due to its ability to model speech sequences. So far, previously published LSTM RNNs solutions for LID deal with highly controlled scenarios, balanced datasets and limited channel variability. In this paper we evaluate an end-to-end LSTM LID system, comparing it against a classical i-vector system, on different environments based on data from Language Recognition Evaluations (LRE) organized by NIST. In order to analyze the behavior we train and test our system on a balanced and controlled subset of LRE09, on the development data of LRE15 and, finally, on the evaluation set of LRE15. Our results show that an end-to-end recurrent system clearly outperforms the reference i-vector system in a controlled environment, specially when dealing with short utterances. However, our deep learning approach is more sensitive to unbalanced datasets, channel variability and, specially, to the mismatch between development and test datasets.

1. Introduction

Most of the state-of-the-art systems for LID [1, 2] rely on acoustic modeling [3, 4]. The basic approach of these systems consists of an i-vector extractor (as in speaker verification) followed by a classification stage [5, 6]. Recently, new approaches such as Deep feed forward Neural Networks (DNNs) have shown to outperform i-vector based approaches when enough data for training is available ($\geq$20h per language); specially when dealing with short test utterances ($\leq$3s) [7, 8, 9].

Even though the performance shown by DNNs in LID is remarkable, they rely on stacking several acoustic frames as an input in order to model time context longer than a frame [7]. However, Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) have the ability to store information from previous inputs during long time periods [10, 11, 12]; which makes them much more suitable to model data sequences in tasks such as handwriting recognition [13] or speech recognition [14].

Recently, it has been shown that LSTM RNNs provide an outstanding performance in (LID) [15]. In that paper, the solution implemented runs over a large machine infrastructure and include proprietary LSTM RNNs, both provided by Google Inc. This fact makes its use hardly reproducible or simply inaccessible for many research groups. Motivated by those results, we published an adapted version of this algorithm using an open source implementation running over a single GPU [16]. In this work we study and evaluate our LSTM based system in more challenging conditions including more similar languages (dialects), unbalanced data, duration variability and database mismatch. In order to perform this comparison we will use data from the Language Recognition Evaluations (LRE) organized by the National Institute of Standards and Technologies (NIST) in 2009 and 2015. Our results show that our LSTM RNN system performs significantly better than the reference i-vector system when dealing with short utterances and a controlled environment, but its performance degrades in severe mismatched conditions.

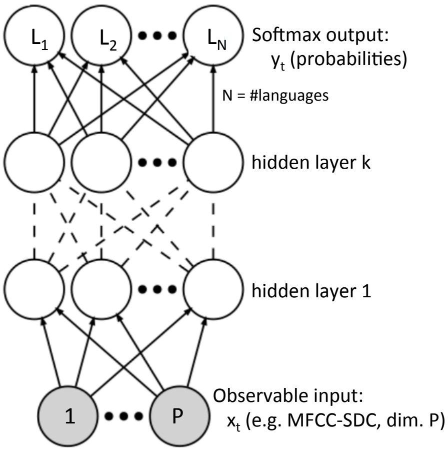


Figure 1: DNN network topology. We use one or two hidden layers replacing each unit with a LSTM memory block with forget gates and peepholes.

2. System Description

2.1. Long Short-Term Memory RNNs

Deep feed forward Neural Networks (DNNs) have proven to outperform classical approaches in tasks such as speech recognition [14] or language identification [7]. The typical architecture of a fully connected Deep feed forward Neural Network is shown in Figure 1. Recurrent Neural Networks (RNNs) are a special type of DNNs where connections between units form a directed cycle, creating an internal state of the network which acts as a memory. It allows RNNs to exhibit dynamic temporal behavior making them a better approach to model temporal sequences but, as shown in [17], its training process has some issues that makes its performance not as good as expected.

The underlying idea of a LSTM neural network is to re-

doi: 10.21437/Odyssey.2016-33

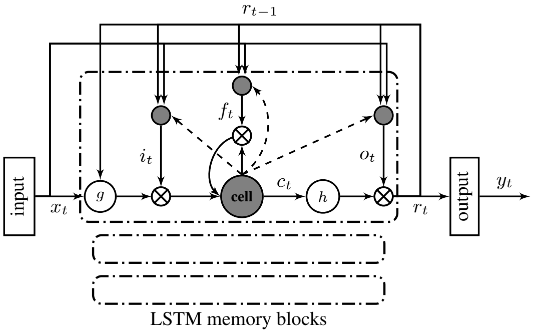


Figure 2: Long Short-Term Memory recurrent neural network architecture. A single memory block is shown for clarity.

place the hidden units in a traditional RNN with *memory blocks*. These memory blocks store the temporal state of the network which changes with the input of the network at each time step. As it can be seen in Figure 2, a memory block contains a memory cell and three adaptive, multiplicative units called gates, which control the flow of information. The input (i_t) and output (o_t) gates control respectively the flow of input activations into the memory cell and the output flow of cell activations into the rest of the network. The forget gate (f_t) allows the flow of information from the memory block to the cell as an additive input, therefore adaptively forgetting or resetting the cell's memory. The input, output and forget gates are respectively similar to the write, read and reset signals in a memory. These features make them easier to train properly than conventional RNNs. The input and output gates help solving the vanishing error problem in the traditional RNN[17]: in the absence of a new input or error signals to the cell, the local error remains constant. The forget gate allows the network to have an adaptive and limited memory buffer avoiding infinite loops.

The LSTM architecture described in [12] also has the ability to learn precise timing of the outputs using *peephole* connections (dashed arrows in Fig. 2). These connections allow communication between gates of the same memory block; outperforming the traditional architecture specially when precise timing of the outputs is important.

With this architecture, LSTM RNNs compute a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$. For LID the input sequence, x_i, are the parameters (e.g. MFCC) of the frame i while the output sequence, y_i, is a vector with as many elements as target languages we have; every element in the output vector stands for the probability of that frame belonging to each language. More information about the training process of a LSTM can be found in [10, 12].

2.2. System description

In all our experiments the training dataset is split into random chunks of 2 seconds from which MFCC-SDC (Shifted Delta Coefficients) with the configuration 7-1-3-7 are computed using Kaldi [18]. The network is fed with these MFCC-SDC with no stacking of acoustic frames: a single MFCC-SDC is given as an input at each time step.

Our system consists of one or two hidden layers followed by an output layer and are implemented using CURRENNT [19] running over a single-GPU. The hidden layers are unidirectional LSTM layers with forget gates and peepholes while the output layer is a softmax layer with the same number of units as languages we have in our experiments. The softmax layer utilizes a cross entropy error function for the back propagation and returns a probability for each input frame and language.

In order to deal with unbalanced data and make the training process faster we train each iteration with a different subset of the training data, which consists of picking random chunks of 2 seconds until we have about 6 hours of audio per language.

The memory blocks in the LSTM hidden layers store the temporal state of the network which changes with the input to the neural network at each time step. When the system gives a probability for a given frame of belonging to one of the languages as an output, it relies not only on the frame input but on every previous frame in that sequence or file. Therefore, the last outputs are computed when the system has information of almost the whole file so they are the most reliable. For scoring, we compute an utterance level score for each target language by averaging the log of the softmax output for that language but taking into account just the last frame scores for every file (details are given in Section 5).

Finally, multiclass linear logistic regression calibration using FoCal Multiclass toolkit [20] was applied to the outputs of every neural network and the reference i-vector system. Moreover, to analyze whether the information learned by the reference and proposed systems is complementary, linear logistic regression fusion of the two individual systems described (LSTM and i-vector) was performed and evaluated.

3. Reference System

3.1. i-vector based LID System

The i-vector system follows the standard procedure described in [21]. It is based on an Universal Background Model consisting of 1024 Gaussian components, trained on the same data described in section 2: MFCC-SDC with the configuration 7-1-3-7. From Baum-Welch statistics computed over this UBM, we derive a Total Variability (TV) subspace of 400 dimensions using PCA followed by 10 EM iterations. Both MFCC-SDC and TV matrix were obtained using Kaldi [18].

Having at maximum 8 different classes in our experiments, a standard classification scheme based on Linear Discriminant Analysis (LDA) would have projected our data into a space with 7 or less dimensions, loosing relevant information for LID. Therefore Cosine Distance scoring without LDA has been used for this task. Thus, the similarity measure (score) for a given test utterance i-vector w, and the mean i-vector w_L of the language L is given by

$$S_{(w,w_L)} = \frac{\langle w, w_L \rangle}{||w|| ||w_L||} \tag{1}$$

The total number of parameters of the i-vector system accounts for the TV matrix. It is given by NxFxD, being N, F and D the number of Gaussians components (1024), the feature dimension (56) and the i-vector dimensions (400). In our model, this makes a total of $\sim$23M of parameters.

4. Datasets and Evaluation Metrics

4.1. Dataset Description

In [16] we already showed the performance of the proposed system in a controlled environment. In this paper we aim to evaluate the performance of our system on different scenarios. In order to train, develop and test the systems we have used a subset of NIST LRE 2009 and NIST LRE 2015.

Cluster	Target Languages
Arabic	Egyptian, Iraqi, Levantine, Maghrebi, Modern Standard
Chinese	Cantonese, Mandarin, Min, Wu
English	British, General American, Indian
French	West African, Haitian Creole
Slavic	Polish, Russian
Iberian	Caribbean Spanish, European Spanish, Latin American Spanish, Brazilian Portuguese

Table 1: Target languages and language clusters in NIST LRE 2015.

4.1.1. A balanced subset of NIST Language Recognition Evaluation 2009

The first dataset chosen for our experiments is the same used in [16]: a controlled environment extracted from NIST LRE 2009 [22]. The dataset of the evaluation consisted of a mixture of two types of data: Conversational Telephone Speech (CTS) and Broadcast news data from "Voice of America" (VOA). In order to avoid unbalanced mix of CTS and VOA, all the data considered in our experiments belongs to VOA. Further, to avoid the disparity on training material for every language (from ∼10 to ∼950 hours) we selected 8 representative languages for which up to 200 hours of audio are available: US English (eng), Spanish (spa), Dari (dar), French (fre), Pashto (pas), Russian (rus), Urdu (urd), Chinese Mandarin (chi).

For the test set, we deal only with short utterances (≤3s), motivated by the degradation on performance seen on i-vector systems for short durations. Thus, we selected the trials from the NIST LRE 2009 3s condition belonging to VOA for the specified languages, yielding a total of 2942 test segments and 23536 trials.

4.1.2. NIST Language Recognition Evaluation 2015

The dataset used for the second set of experiments is the one provided by NIST for the core task (limited training data) of the LRE'15 [23]. This dataset includes CTS and Broadcast Narrow Band Speech (BNBS) within the training data, and 20 different languages grouped according to 6 clusters (see Table 1), with no inter-cluster trials. The total amount of hours available for training and development per language ranges from about half an hour to more than 100 hours. For more information, see [23].

Differently from the other dataset used in this work, the emphasis of these experiments is on discriminating among languages that are similar to each other and frequently mutually intelligible. Moreover, some clusters are composed of a set of languages with completely different amount of data available to train the system, which makes the task more challenging, since it requires dealing with unbalanced clusters.

In order to evaluate the performance of the system in this challenging task, two subsets have been selected as evaluation sets. The first subset mimics the experiments done in the evaluation time, when only the development data was available. Thus, this evaluation set is a 15% of the development data, which was not used to train the system. This subset was split in segments of 3, 10 and 30 seconds to evaluate the performance in different durations. The second subset is the real evaluation test-set, which was not limited to segments of given durations but covered a broad range of speech durations.

4.2. Evaluation Metrics

Two different metrics were used:

- C_{avg}, as described in the LRE 2009 [22][24] evaluation plan, has been used as the main error measure to evaluate the capabilities of the system to identify languages in a one-vs-all way. C_{avg} is a cost function that penalizes taking bad decisions, therefore it considers both discrimination and the ability of setting optimal thresholds (i.e. calibration).
- EER_{avg}, the mean of the Equal Error Rate computed language by language has been used for easier comparison being an extensively used metric in the community.

5. Experimental Results

5.1. Experiments in a controlled environment: a subset of NIST LRE 2009

5.1.1. Discarding initial frame scores

In uni-directional, left-to-right LSTMs, such as the one used in our experiments, an output is based on previous and present inputs in the sequence. Therefore, the last output scores are the most reliable. Figure 3 shows how the average performance of 5 different architectures varies with the percentage of initial frame scores discarded (we will describe the different architectures later in this section, just relative improvement matters now). Selecting the last 10% of the scores leads to improved robustness of the utterance level score; from now on, the results will be shown when 90% of the initial frame scores are discarded.

5.1.2. System performance

Table 2 summarizes the results obtained in terms of EER_{avg} and C_{avg}. We highlight that 4 out of 5 proposed architectures for the LSTM RNN system outperform the reference i-vector based system up to 15% in terms of C_{avg}. This fact is particularly interesting taking into account that the proposed architectures have from 5 to 21 times fewer parameters (see *Size* in Table 2) than the reference system.

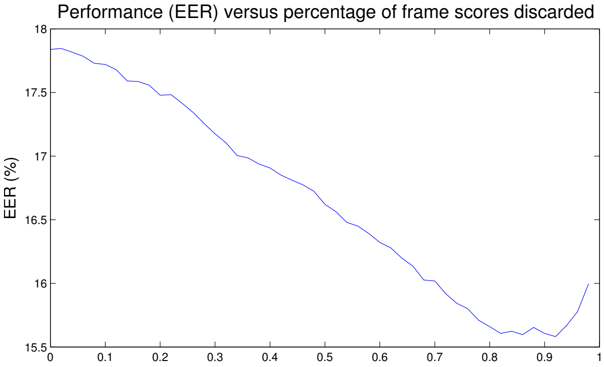
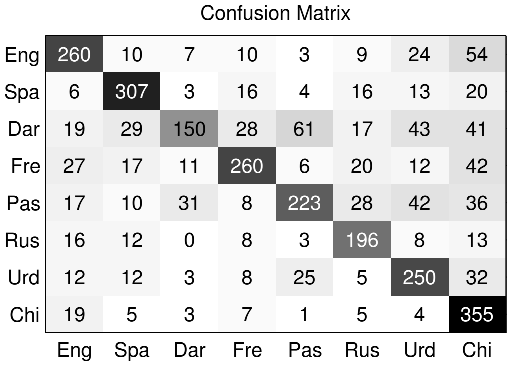


Figure 3: In RNN based systems, the output score is computed based on previous and present inputs, being the last outputs the most reliable scores. Discarding the less reliable scores may lead to a better performance. In this figure we show the average performance (EER_{avg}) of 5 LSTM systems versus percentage of initial frame scores discarded.

		Complexity		Performance (3s)		
ID		Size	Size Gain (%)	$EER_{avg}(\%)$	C_{avg}	Improvement (%)
#1	reference i-vector based system	∼23M	-	16.94	0.1632	-
#2	lstm_1_layer_512_units	∼1.2M	∼ 94.3	18.05	0.1817	-
#3	lstm_1_layer_750_units	∼2.5M	∼ 88.1	15.83	0.1613	∼1.2
#4	lstm_1_layer_1024_units	∼4.4M	∼ 79	15.18	0.1559	∼4.5
#5	lstm_2_layer_256_units	∼850k	∼ **95.9**	15.13	0.1585	∼2.9
#6	lstm_2_layer_512_units	∼3.3M	∼ 84.3	**13.66**	**0.1383**	**∼15.3**
#7	Fusion (#1 + #6)	*∼26M*	-	*11.98*	*0.1153*	*∼29.4*

Table 2: Systems performance and size on LRE'09 subset (3s test segments). The third column stands for the relative gain in terms of size of the proposed systems with respect to the reference system while the last column stands for the relative improvement in terms of C_{avg}.



	Eng	Spa	Dar	Fre	Pas	Rus	Urd	Chi
Eng	260	10	7	10	3	9	24	54
Spa	6	307	3	16	4	16	13	20
Dar	19	29	150	28	61	17	43	41
Fre	27	17	11	260	6	20	12	42
Pas	17	10	31	8	223	28	42	36
Rus	16	12	0	8	3	196	8	13
Urd	12	12	3	8	25	5	250	32
Chi	19	5	3	7	1	5	4	355

Figure 4: Confusion matrix of the best LSTM RNN system, lstm_2_layer_512_units (system #6), on a balanced subset of NIST LRE 2009

In order to analyze both the confusion and the discrimination performance of the systems considering all the languages pairs, Figure 4 shows the confusion matrix of the best system, #6 in Table 2.

In addition, we study the complementarity between our neural network systems and the classical i-vector approach. In the last row of Table 2 we can see the result of the fusion between the best LSTM RNN system (#6) and the i-vector system (#1) which performs roughly 15% better then the best single system in terms of C_{avg}.

5.2. Experiments in the development set of NIST LRE 2015

The system proposed in our experiments using NIST LRE 2015 consists of one neural network per cluster. All the neural networks have the same architecture which corresponds to the best performing system in NIST LRE 2009: two hidden layers of 512 units followed by an output layer. The hidden layers are uni-directional LSTM layers while the output layer is a softmax with as many units as languages in the cluster.

In this section we want to analyze the results of the best performing system in a controlled environment when dealing with a more challenging scenario. In these experiments we have 6 clusters with 2 to 6 similar languages in each and the data available to train is neither balanced nor controlled at all (details are given in Section 4). For the following experiments we have not fine tuned our system because we want to test the performance of the best system in the previously controlled environment when facing a more difficult task; closer languages, two different sources of data, completely unbalanced datasets, etc.

Table 3 summarizes the results on an unseen 15% of the development set. Two major messages can be extracted from these results. First of all, note that the LSTM RNN based system performs better than the i-vector system when the test utterances are short enough (most of the 3s utterances and some of the 10s) while the i-vector approach is solidly better when dealing with long utterances. For example, in the 3 seconds subset, as we show in Fig. 5, the recurrent neural network approach have an improvement higher than 20% in terms of C_{avg} with respect to the reference system. Secondly, we can observe that the fusion of the two systems behaves considerably better and is more robust than any of the single systems, outperforming the deep learning approach even in short durations and the i-vector system when facing long ones.

5.3. Experiments in the test set of NIST LRE 2015

The last scenario we wanted to test our system in corresponds to the fixed-training task of the NIST Language Recognition Evaluation 2015. In this experiment we have the same set-up as we had in the previous one but a big mismatch is observed between

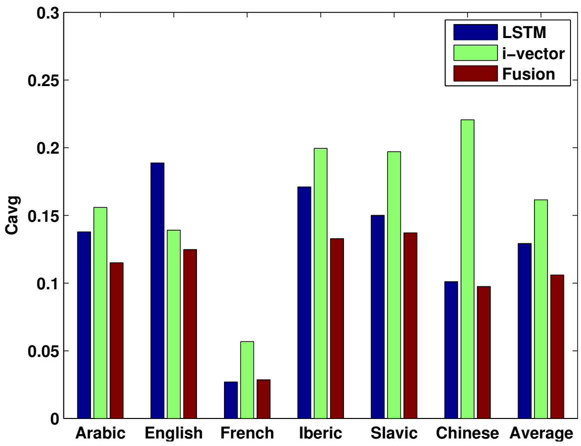


Figure 5: Performance of the proposed system compared to the reference i-vector system on our 3 seconds subset of the development set of NIST LRE 2015

		Mean C_{avg} / EER(%) per cluster						
	System	Arabic	English	French	Iberic	Slavic	Chinese	**Average**
3 sec	LSTM	**0.1379 / 13.28**	0.1888 / **11.40**	**0.0270 / 2.92**	**0.1711 / 14.40**	**0.1501 / 15.24**	**0.1011 / 8.73**	**0.1293 / 11.00**
	i-vector	0.1559 / 15.86	**0.1391** / 11.47	0.0568 / 6.17	0.1996 / 19.90	0.1971 / 19.84	0.2206 / 19.24	0.1615 / 15.41
	Fusion	*0.1150 / 11.27*	*0.1248 / 7.86*	*0.0286 / 2.25*	*0.1328 / 10.67*	*0.1371 / 13.69*	*0.0975 / 7.83*	*0.1060 / 8.93*
10 sec	LSTM	0.0976 / 9.65	0.1932 / 9.63	0.0304 / 2.12	0.1673 / **11.43**	0.1059 / 10.85	**0.0906 / 7.44**	0.1142 / 8.51
	i-vector	**0.0750 / 7.63**	**0.0747 / 4.28**	**0.0198 / 1.06**	**0.1449** / 12.26	**0.1004 / 10.18**	0.1133 / 8.32	**0.0880 / 7.29**
	Fusion	*0.0609 / 5.81*	*0.0461 / 4.44*	*0.0127 / 1.06*	*0.1003 / 7.92*	*0.0717 / 7.12*	*0.0648 / 3.83*	*0.0594 / 5.03*
30 sec	LSTM	0.0859 / 8.93	0.1876 / 6.79	0.0104 / 1.88	0.1473 / 10.33	0.0868 / 8.68	0.0995 / 7.16	0.1029 / 7.30
	i-vector	**0.0308 / 3.23**	**0.0199 / 0.76**	**0 / 0**	**0.1278 / 8.60**	**0.0423 / 4.59**	**0.0493 / 3.77**	**0.0450 / 3.49**
	Fusion	*0.0306 / 2.84*	*0.0387 / 0.28*	*0 / 0*	*0.0984 / 5.38*	*0.0331 / 2.99*	*0.0460 / 1.92*	*0.0411 / 2.24*

Table 3: Results on the NIST 2015 development dataset (testing on an unseen 15% of the development dataset) in terms of EER and C_{avg}.

the development data set and the test set. In these conditions, we wanted to analyze the impact of this mismatch in the proposed system and compare it with the degradation suffered by the classical i-vector approach.

Figure 6 summarizes our results. We can observe the performance of the two different systems analyzed in function of some duration bins ranging from those shorter to 3 seconds to those longer than 30 seconds. The last four bars in Figure 6 are the result of the two analyzed systems in the evaluation test set. First of all, we can see that in this scenario the i-vector system outperforms the LSTM RNN based system, showing that our deep learning approach suffers from a bigger degradation in presence of such a mismatch. Secondly, we can see that even though the performance of the i-vector system has a similar result than the LSTM for short utterances, it becomes more accurate when the test utterances become longer while the LSTM seem not to gain so much accuracy in presence of long utterances due to the big mismatch between training and test data.

Finally, we can also see in Fig. 6 the result of two different fusions of the analyzed systems. The first fusion is the submitted linear logistic regression fusion learned over training data. The result of this fusion is not better than the best system itself in any of the durations, probably because training a fusion with data that does not represent the test data led to a wrong fusion which is not able to extract complementary information. In order to prove this hypothesis we have also shown a post-eval fusion (not valid for the evaluation task) with 2-fold cross-validation (Fusion CV in Fig. 6), splitting the test-set in two subsets and training the fusion on one dataset and applied to the other one and vice-versa, leading to an optimistic fusion. As the results show, this fusion performs better than the single-systems in every duration proving that the information learned by the two individual systems is complementary.

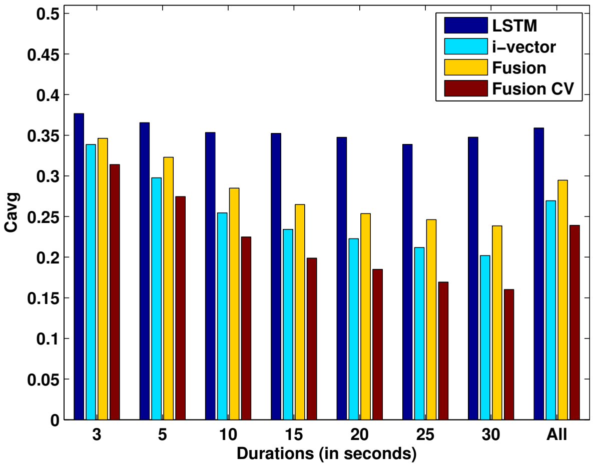


Figure 6: Results of our proposed LSTM RNN system and the reference i-vector system on NIST LRE 15 fixed-training task divided by duration bins, followed by the overall results.

6. Conclusions

In this work, we analyzed an end-to-end Long Short-Term Memory Recurrent Neural Network based system for LID proposed in [16] in several conditions from NIST Language Recognition Evaluations 2009 and 2015 and compared its performance with respect to a classical i-vector system.

Results show that the proposed system using significantly fewer parameters (∼1-5M vs ∼23M) clearly outperforms the reference system on a controlled environment with balanced datasets, limited channel variability and no big mismatch between development and test, specially when dealing with short utterances.

Furthermore, when facing a more challenging scenario with highly-unbalanced datasets and closer languages the performance of the neural network system gets closer to the classical i-vector approach, being still better on short utterances while the i-vector is more accurate on longer ones. Moreover, the information learned by LSTM RNN systems in this scenario is complementary to the information learned by the i-vector system, being the fusion of both systems solidly better in all the durations.

Finally, we have tested our system on the real test of NIST LRE 2015, which has all the variability previously mentioned but it also adds a mismatch between the training data and the test data. Our work shows that the deep learning approaches are more sensitive than the i-vector based systems to this severe mismatch and cannot surpass its performance.

Our findings in this work show that a deep learning end-to-end approach for language recognition with ∼85% less parameters than a classical i-vector system can achieve robust and comparable results in several challenging scenarios. Nevertheless, this kind of systems strongly depend on the similarity between the development and the test dataset and seem to need further research on variability compensation.

7. Acknowledgements

This work has been supported by project *CMC-V2: Caracterizacion, Modelado y Compensacion de Variabilidad en la Senal de Voz* (TEC2012-37585-C02-01), funded by *Ministerio de Economia y Competitividad*, Spain.

8. References

[1] Y.K. Muthusamy, E. Barnard, and R.A. Cole, "Reviewing automatic language identification," *Signal Processing Magazine, IEEE*, vol. 11, no. 4, pp. 33–41, 1994.

[2] E. Ambikairajah, Haizhou Li, Liang Wang, Bo Yin, and V. Sethu, "Language identification: A tutorial," *Circuits and Systems Magazine, IEEE*, vol. 11, no. 2, pp. 82–108, 2011.

[3] Pedro A. Torres-Carrasquillo, Elliot Singer, Mary A. Kohler, and J. R. Deller, "Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features," in *ICSLP*, 2002, vol. 1, pp. 89–92.

[4] J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D.T. Toledano, and J. Gonzalez-Rodriguez, "Multilevel and Session Variability Compensated Language Recognition: ATVS-UAM Systems at NIST LRE 2009," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 6, pp. 1084–1093, 2010.

[5] D. Martinez, O. Plchot, L. Burget, Ondrej Glembek, and Pavel Matejka, "Language Recognition in iVectors Space.," in *INTERSPEECH*, 2011, pp. 861–864.

[6] Elliot Singer, Pedro Torres-Carrasquillo, Douglas A Reynolds, Alan McCree, Fred Richardson, Najim Dehak, and Doug Sturim, "The mitll nist lre 2011 language recognition system," in *Odyssey - The Speaker and Language Recognition Workshop*, 2012.

[7] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno, "Automatic Language Identification using Deep Neural Networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5337–5341.

[8] Alicia Lozano-Diez, Javier Gonzalez-Dominguez, Ruben Zazo, Daniel Ramos, and Joaquin Gonzalez-Rodriguez, "On the use of convolutional neural networks in pairwise language recognition," in *Advances in Speech and Language Technologies for Iberian Languages*, pp. 79–88. Springer, 2014.

[9] 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 *InterSpeech 2015*, 2015.

[10] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385, Springer, 2012.

[11] Felix A. Gers, Jrgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Mar. 2003.

[13] Volkmar Frinken, Francisco Zamora-Martinez, Salvador Espana-Boquera, Marıa José Castro-Bleda, Andreas Fischer, and Horst Bunke, "Long-short term memory neural networks language modeling for handwriting recognition," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 701–704.

[14] Alex Graves, Navdeep Jaitly, and A-R Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[15] 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 *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, 2014, pp. 2155–2159.

[16] R Zazo, A Lozano-Diez, J Gonzalez-Dominguez, D T Toledano, and J Gonzalez-Rodriguez, "Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks.," *PloS one*, vol. 11, no. 1, 2015.

[17] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu, "Advances in optimizing recurrent networks," in *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8624–8628.

[18] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding*. 2011, IEEE Signal Processing Society.

[19] Felix Weninger, Johannes Bergmann, and Bjorn Schuller, "Introducing currennt: the munich open-source cuda recurrent neural network toolkit," *Journal of Machine Learning Research*, vol. 15, 2014.

[20] Niko Brümmer, "Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scorestutorial and user manual," *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, 2007.

[21] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and Reda Dehak, "Language Recognition via i-vectors and Dimensionality Reduction.," in *INTERSPEECH*. 2011, pp. 857–860, ISCA.

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

[23] "2015 Language Recognition Evaluation," Available from: http://www.nist.gov/itl/iad/mig/lre15.cfm, 2015.

[24] N. Brümmer, *Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech*, Ph.D. thesis, Department of Electrical and Electronic Engineering, University of Stellenbosch,, 2010.