INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Exploiting Hidden-Layer Responses of Deep Neural Networks for Language Recognition

Ruizhi Li[1], *Sri Harish Mallidi*[1], *Lukáš Burget*[2], *Oldřich Plchot*[2], *Najim Dehak*[1]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, USA
[2]Speech@FIT group, Brno University of Technology, Brno, Czech Republic
{ruizhili,mallidi,najim}@jhu.edu, {burget,iplchot}@fit.vutbr.cz

Abstract

The most popular way to apply Deep Neural Network (DNN) for Language IDentification (LID) involves the extraction of bottleneck features from a network that was trained on automatic speech recognition task. These features are modeled using a classical I-vector system. Recently, a more direct DNN approach was proposed, it consists of estimating the language posteriors directly from a stacked frames input. The final decision score is based on averaging the scores for all the frames for a given speech segment. In this paper, we extended the direct DNN approach by modeling all hidden-layer activations rather than just averaging the output scores. One super-vector per utterance is formed by concatenating all hidden-layer responses. The dimensionality of this vector is then reduced using a Principal Component Analysis (PCA). The obtained reduce vector summarizes the most discriminative features for language recognition based on the trained DNNs. We evaluated this approach in NIST 2015 language recognition evaluation. The performances achieved by the proposed approach are very competitive to the classical I-vector baseline.

Index Terms: LID, I-vector, DNN, hidden layers

1. Introduction

Recently, deep neural networks (DNNs) have shown significant improvements in many speech recognition tasks over state-of-the-art Gaussian mixture model (GMM) systems [12]. The reason for the improvements is attributed to DNNs ability to model complex, non-linear manifolds that may be separating features from different classes of speech sounds [1, 2, 4, 12]. Motivated by this success, [2] used DNNs for language recognition task. The approach involves training a DNN to classify language targets directly, at each input frame. For a given test segment, language scores are computed by time-averaging log-posterior probabilities of the DNN output layer [4]. The approach showed promising results by performing better than I-vector system [6, 11] in short-segment conditions (less than 3 seconds) [1, 2]. Whereas, I-vector system is shown to be performing better in longer segment (10, 30 and 120 second) conditions.

In this work we propose a technique to improve frame-by-frame DNN LID. The logistic regression layer used at the output of DNN can cause significant loss in the information required to discriminate languages. We hypothesize that hidden layer activations contain better language discriminant information. Furthermore, the hidden layer information can be complementary to output activations, used in direct DNN-LID. Results from the LID experiments support our hypothesis.

The LID experiments are performed on NIST Language Recognition Evaluation (LRE) 2015 corpus [5]. Compared to direct DNN approach, we achieved 38 % relative improvement in development set. Also, we outperform state-of-the-art I-vector system [3, 10, 11] in evaluation (eval) set of NIST LRE 2015. Combination of proposed technique and state-of-the-art I-vector system resulted in a 4 % relative improvement.

The rest of the paper is organized as follows: section 2 describes NIST LRE 15 database we use in this work. The direct DNN approach as well as the proposed DNN approach are discussed in section 3. Section 4 shows experimental analysis of the proposed system with respect to various hyper-parameters. Comparison with state-of-the-art I-vector system [9] is presented in section 5. Section 6 presents conclusions and future work along the proposed approach.

2. NIST LRE 2015

Table 1: *Language clusters present in NIST LRE 2015 database[5].*

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

All the experimental results shown in this paper are performed on *primary condition* of NIST LRE 2015. The database contains 20 target languages, which are grouped into 6 clusters

This work was supported in parts by the National Science Foundation via award number IIA-0530118, .Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Google, NSF, IARPA, DoD/ARL, or the U.S. Government. This work was supported by the DARPA RATS Program. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Also, the authors would like to thank Prof. Hynek Hermansky for suggesting fusion of DNN I-vector and I-vector systems.

http://dx.doi.org/10.21437/Interspeech.2016-1584

[5]. Table 1 describes the language clusters in the corpus. The task is designed to recognize dialect within its language cluster. The training data provided for the primary condition has been divided into training (train) and development (dev) sets. The train set is composed of 60 % of the original training data and dev set consist of the remaining 40 %. The segments belonging to the dev set are further divided into short cuts, ranging from 3 to 30 seconds of speech [9]. This split resulted in 3042 segments (248 hours of speech) for train set, and 42295 segments (146 hours of speech) for dev set [9]. The results are reported on dev and evaluation (eval) sets.

3. DNN for Language ID

3.1. Frame-by-frame DNN classification

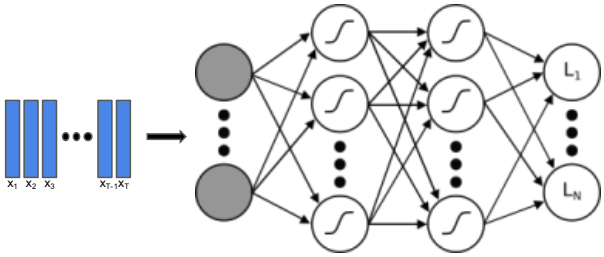

Figure 1: *Frame-by-frame DNN Language Identification*

Figure 1 shows the architecture of the DNN used for LID [1, 2]. It is a fully-connected feed-forward DNN consisting of 3 hidden layers. Each hidden layer has 2560 neurons with Rectified Linear activation function [2]. The output softmax layer's dimensionality is equal to the number of target languages N_ℓ. We used 40 dimensional log-Mel filterbank features to train the neural network. A 3-second sliding window-based mean and variance normalization is applied to the log-Mel features. A context window of 21 frames (10 frames on each side of the current frame) is applied at the input, resulting in 840 dimensional feature vector at every frame. Just before presenting the features at input layer of the DNN, a global mean and variance normalization is applied. This results in a neural network with 16 million parameters (including weights and biases), for a 20 target language case.

The network is trained using stochastic gradient descent (SGD) to minimize cross-entropy between the frame level language labels and its outputs. The SGD uses minibatches of size 200 frames. We start with an initial learning rate of 0.001, and the learning rate has been halved after every epoch. We did not use any cross-validation set for scheduling learning rate.

Once the network is trained, scores of a test utterance are computed by accumulating the log-likelihood as:

$$s_\ell = \frac{1}{N}\sum_{t=1}^{N} \log p(L_\ell|x_t,\theta) \qquad (1)$$

where $p(L_\ell|x_t,\theta)$ is the posterior probability of language ℓ, obtained at the output of neural network [1, 2]. x_t is input feature vector at frame t and N is the total number of frames in the test segment. θ defines the neural network.

3.2. Modeling hidden information

In this section, we describe proposed technique to model information present in hidden layers of the DNN. This approach is illustrated in figure 2. Given a speech utterance, we time-average activations of hidden and output layers. The time-averaged activations of all the layers are stacked to form a single, duration independent vector, referred to as DNN super-vector. Dimensionality of the DNN super-vector is reduced using principal component analysis (PCA). This results in a low dimensional fixed length representation of input speech utterance, similar to classical I-vector used in speaker and language recognition tasks [3, 11]. This vector is referred to as DNN I-vector from here on.

Gaussian linear classifier (GLC) has been shown to be an effective classifier on I-vectors for language recognition [6, 9]. We also propose to use (GLC) in the DNN I-vectors system. In this probabilistic model,the class–conditional log–likelihood for $\boldsymbol{\mu}_h$ given language ℓ can be computed as:

$$\log P(\boldsymbol{\mu}_h|\ell) = \frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}(\boldsymbol{\mu}_h - \mathbf{m}_\ell)^T\boldsymbol{\Lambda}(\boldsymbol{\mu}_h - \mathbf{m}_\ell) + k\,, \qquad (2)$$

where k is a data–independent constant. The model parameters can be easily obtained by Maximum–Likelihood estimation [9, 14].

3.3. Performance Comparison

In order to compare direct DNN system with proposed DNN I-vector system, we trained a single neural network to classify all 20 languages. The architecture of this network is described in section 3.1. The first row of table 2 shows the $\mathrm{C_{avg}}$ obtained from direct DNN approach on dev and eval sets. The performance of DNN I-vector system is reported in the second row of the same table. Both systems are based on the same neural network. For each utterance, the hidden layer activations are time-averaged to compute the super-vector. A DNN I-vector of dimension 1800 is computed by reducing the dimensionality of the corresponding super-vector. A Gaussian liner classifier is trained on the DNN I-vectors to classify the 20 target languages. From the table 2, we can conclude that performance of the DNN I-vector system is significantly better than the direct DNN system [1]. We obtained a relative improvement of 40 % on dev set and 20 % on eval set.

Table 2: *Comparison of* $\mathrm{C_{avg}}$ *between baseline and proposed DNN-LID systems. DEV* REFERS TO CALIBRATED.*

		$\mathrm{C_{avg}} \times 100$	
	System	dev*	eval
1	direct DNN	13.61	45.39
2	DNN I-vector	9.90	38.12

3.4. Per–cluster DNN

Results in the previous section are obtained by training a DNN to classify all the 20 target languages. Since the task in NIST-LRE15 challenge is to recognize dialects given a language family, we trained a separate DNN for each cluster. First row of table 3 show the results of direct DNN system with a single network. Performance of direct DNN system, with one network per–cluster, is shown in second row of table 3. It is evident from the table 3 that per–cluster DNN system seems to outperform the single DNN which was trained to classify all target languages of NIST 2015 LRE.

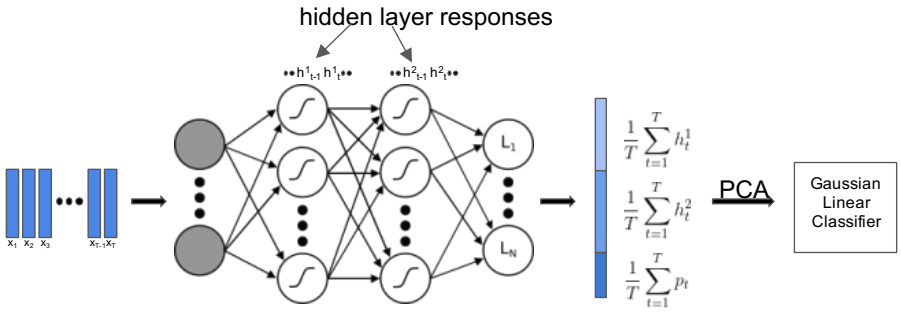

Figure 2: *Proposed DNN I-vector system for LID.*

Results of the DNN I-vector system with single network and per–cluster networks are reported in rows 3 and 4, respectively. Similar to direct DNN approach, per–cluster DNN I-vector system is also performing better than single DNN I-vector system. Note that in DNN I-vector case, per–cluster and single network systems generate super-vectors of different dimensions. In order to have same number of parameters for the final GLC back-end, we reduce dimensionality of super-vectors to 1800. Based on these performances, we only use per–cluster DNN architecture for further analysis in the rest of the paper.

Table 3: *Comparison of* C_{avg} *between single-DNN and per-cluster systems*

		$C_{avg} \times 100$	
	System	dev*	eval
1	direct DNN (1 net)	13.61	45.39
2	direct DNN (6 nets)	11.69	44.71
3	DNN I-vector (1 net)	9.90	38.12
4	DNN I-vector (6 nets)	7.47	36.72

4. Experimental analysis

This section presents an empirical analysis of DNN I-vector system with respect to various hyper-parameters in the system architecture.

4.1. Input feature representation

Results of the DNN I-vector systems presented in previous sections use networks which are trained on log-Mel filterbank features. However, bottleneck features which are extracted from neural networks trained for speech recognition task, have shown significant improvements over acoustic features in many LID works [7, 8, 10, 13]. In this section, we compare between two DNN I-vector systems trained on log-Mel filterbanks and bottleneck features. In particular, we use Stacked BottleNeck (SBN) features [7, 8].

4.1.1. Stacked Bottleneck feature extraction:

SBN extraction pipeline consist of cascade of two neural networks. Each neural network has a 80 dimensional bottleneck layer before last hidden layer. The first neural network is trained on log-Mel filterbank features. The output of the first network is stacked in time ($t-10, t-5, t, t+5$ and $t+10$), and given as input to second neural network. Both neural networks are trained on Switchboard database (audio and transcriptions) [9]. The 80 dimensional bottleneck features, obtained from the second neural network are referred to as Stacked Bottleneck features. More details about the stacked bottleneck features can be found in [8, 7, 9]. Table 4 compares log-Mel filterbanks and SBN features. From the table, it is evident that the performance of SBN features is significantly better than log-Mel filterbanks. We obtained 44 % and 34 % relative improvement in dev and eval sets, respectively. The reason for this improvement can be attributed to, robustness to acoustic mis-matches and higher discriminative information present in the SBN features [9]. For the rest of the experiments, we use DNNs which are trained on SBN features. We also keep the same dimensionality of the DNN I-vectors (1800).

Table 4: *Performance of DNN I-vector system with two input features.*

		$C_{avg} \times 100$	
	System	dev*	eval
1	log-Mel filter banks	7.47	36.72
2	stacked bottlenecks	4.21	23.92

4.2. Per–cluster PCA

In the previous experiments, super-vectors from multiple DNNs, one for each cluster, are stacked to form a single super-vector. DNN I-vector is generated from this super-vector by reducing its dimensionality using PCA transform. For example, in row 4 of table 3, PCA transforms 46,100 dimensional super-vector to 1800 dimensional DNN I-vector. Estimation of PCA transform on this high-dimensional space can be data hungry and computationally expensive. This problem becomes more severe if the number of hidden layers in the neural networks are increased. One approach to mitigate the problem, is to train per–cluster PCA transforms. That is, instead of having a single PCA transform which operates on super-vector obtained from multiple clusters, we employ 6 PCA transforms, one for each cluster, and generate per–cluster DNN I-vector. Per-cluster DNN I-vectors are then stacked and given as input to GLC. Using per–cluster PCA approach, we can generate DNN I-vectors of much lower dimensionality. Table 5 presents a comparison between DNN I-vector systems, using single PCA transform and per–cluster PCA transforms. From this table, it is evident that

C_{avg} of per-cluster PCA system is slightly worse than single PCA strategy, in both dev and eval sets. But, we get a significant reduction in the number of parameters (41 million to 6 million) compared single PCA system. Also, per–cluster PCA system is faster and has smaller memory footprint than single PCA system.

Table 5: *Performance of DNN I-vector system with different PCA configurations.*

		$C_{avg} \times 100$	
	System	dev*	eval
1	Single PCA	4.21	23.92
2	Per-cluster PCA	4.35	24.09

Table 6: *Performance of DNN I-vector system with different hidden responses.*

		$C_{avg} \times 100$	
	System	dev*	eval
1	Pre+Per–cluster DNNs+Per–cluster PCA	4.47	22.92
2	Post+Per–cluster DNNs+Per–cluster PCA	4.35	24.09
3	Pre+Per–cluster DNNs+single PCA	4.15	22.58
4	Post+Per–cluster DNNs+single PCA	4.21	23.92

4.3. Pre-nonlinearity *vs* Post-nonlinearity

In all previous experiments, the super-vectors were obtained by time averaging hidden layer outputs (post-nonlinearity). Table 6 shows the comparison with systems, where super-vectors are obtained by time averaging hidden layer pre-activation responses (pre-nonlinearity). We can conclude from the table that, super-vectors computed from pre-nonlinearity values are a better choice than super-vectors computed after the activation fuction was applied (post-nonlinearity). The reason for this behavior could be that, PCA transform, applied on super-vectors, assumes the data is being generated from a Gaussian conditional distribution [14]. Pre-nonlinearity values have a better fit to Gaussian distribution than post-nonlinearity values. The difference might disappear by using a subspace method which is more suitable for post-nonlinearity values

5. Comparison with I-vector system

In this section, we compare DNN I-vector system with state-of-the-art Gaussian mixture model (GMM) based I-vector system [9, 11]. We briefly describe the baseline I-vector system [9, 11] used in our experiments. The baseline I-vector system consists of an universal background model (UBM) comprised by 2048 components, with diagonal-covariance. The UBM is trained on the balanced training set in 5 iterations (up to 15 hours of randomly selected training utterances per language). The dimensionality of the total variabilty space is 600. More details about this system can be found in [9, 11].

Table 7 shows the C_{avg} results obtained from various systems. Rows 1 and 2 present the results of direct DNN approach proposed by [4]. Results of various DNN I-vector systems are shown in rows 3, 4, and 5. It can be observed from the table that the proposed approach results in close to 30 % relative improvement over direct DNN approach. It is important to remind the reader that the DNNs used in DNN I-vector systems are the same as the ones used in the direct DNN systems. The improvement observed is due to modeling hidden layer responses using proposed approach.

Performance of the proposed DNN I-vector systems are competitive to the standard GMM I-vector system (row 6). Fusion of the systems resulted in 14 % relative improvement in dev set and 4 % relative improvement in eval set. This illustrates the complementary nature of the proposed technique. The significant improvement observed in dev set can be attributed be to fusion weights, which are tuned to dev set.

Table 7: *Comparison and fusion of C_{avg} between DNN I-vector and I-vector systems*

		$C_{avg} \times 100$		
	System	dev*	eval	eval*
1	Direct DNN (1 net)	9.15	37.35	29.66
2	Direct DNN (6 net)	5.94	33.86	25.46
3	DNN I-vector (1 net)	5.65	23.95	23.93
4	DNN I-vector (6 nets + 1 PCA)	4.15	**22.58**	**20.73**
5	DNN I-vector (6 nets + 6 PCA)	4.45	22.92	20.89
6	GMM I-vector	3.69	23.31	20.93
7	score_fusion(4, 6)	3.18	–	20.10
8	score_fusion(5, 6)	3.19	–	20.15
9	score_fusion(4, 5, 6)	3.17	–	20.11

6. Conclusions

In this work, we proposed a technique to improve DNN based Language recognition. The approach is based on modeling hidden layer information of DNNs, and extracting an I-vector like representation for a given speech segment. The technique is referred to as DNN I-vector system. This approach is shown to be significantly better than direct DNN baseline. We also showed that proposed approach is fairly robust to the choice of hyper-parameters in the model. Experimental results indicated that DNN I-vector system is competitive and complementary to GMM I-vector system. In future, we would like to explore different ways of extracting DNN super-vector and different subspace methods to compute DNN I-vector.

7. References

[1] I. Lopez-Moreno, J. Gonzalez-Dominguez, D. Martinez, O. Plchot, J. Gonzalez-Rodriguez, and P. J. Moreno, "On the use of Deep Feedforward Neural Networks for Automatic Language Identification", *Journal of Computer Speech & Language*, 2015.

[2] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. J. Moreno, "Automatic Language Identification Using Deep Neural Networks", in *Proc. ICASSP*, 2014.

[3] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, R. Dehak, "Language Recognition via Ivectors and Dimensionality Reduction", in *Proc. Interspeech*, 2011, pp. 857-860.

[4] J Gonzalez-Dominguez, I Lopez-Moreno, P. J. Moreno, and J Gonzalez-Rodriguez, "Frame-by-frame language identification in short utterances using deep neural networks", *Neural Networks*, 64:49-58, 2015.

[5] NIST, "The 2015 NIST Language Recognition Evaluation Plan (LRE15)", http://www.nist.gov/itl/iad/mig/upload/LRE15_EvalPlan_v22-3.pdf.

[6] D. Martínez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language Recognition in iVector Space", *Proc. Interspeech*, 2011.

[7] K. Veselý, M. Karafiát, F. Grézl, M. Janda, and E. Egorova, "The Language-independent Bottleneck Features", in *Proc. SLT*, 2012.

[8] R. Fér, P. Matějka, F. Grézl, O. Plochot, and J. H. Černocký, "Multilingual Bottleneck Features for Language Recognition", in *Proc. Interspeech*, 2015.

[9] O. Plchot, P. Matějka, R. Fér, O. Glembek, O. Novotný, J. Pešán, K. Veselý, L. Ondel, M. Karafiát, F. Grézl, S. Kesiraju, L. Burget, N. Brümmer, A. Swart, S. Cumani, SH. Mallidi, R. Li, "BAT System Description for NIST LRE 2015", in *Proc. Odyssey*, 2016.

[10] Y. Song, B. Jiang, Y. Bao, S. Wei, L. Dai, "I-vector Representation based on Bottleneck Feature for Language Identification", in *IEEE Electronics Letters*, 2013

[11] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-End Factor Analysis for Speaker Verification", IEEE Trans. on Audio, Speech and Language Processing, vol. 19, pp. 788-798, May 2011.

[12] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition", *IEEE Signal Processing Magazine*, vol. 29, no. 6, Nov. 2012.

[13] P. Matějka, L. Zhang, Tim Ng, H. S. Mallidi, O. Glembek, J. Ma, and B. Zhang, "Neural Network Bottleneck Features for Language Identification", in *Proc. Odyssey*, 2014.

[14] Christopher M. Biship, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2007.