INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Auto-Encoding Nearest Neighbor i-vectors for Speaker Verification

Umair Khan, Miquel India, Javier Hernando

TALP Research Center, Department of Signal Theory and Communications,
Universitat Politecnica de Catalunya Barcelona, Spain

{umair.khan, javier.hernando}@upc.edu, miquel.india@tsc.upc.edu

Abstract

In the last years, i-vectors followed by cosine or PLDA scoring techniques were the state-of-the-art approach in speaker verification. PLDA requires labeled background data, and there exists a significant performance gap between the two scoring techniques. In this work, we propose to reduce this gap by using an autoencoder to transform i-vector into a new speaker vector representation, which will be referred to as ae-vector. The autoencoder will be trained to reconstruct neighbor i-vectors instead of the same training i-vectors, as usual. These neighbor i-vectors will be selected in an unsupervised manner according to the highest cosine scores to the training i-vectors. The evaluation is performed on the speaker verification trials of VoxCeleb-1 database. The experiments show that our proposed ae-vectors gain a relative improvement of 42% in terms of EER compared to the conventional i-vectors using cosine scoring, which fills the performance gap between cosine and PLDA scoring techniques by 92%, but without using speaker labels.

Index Terms: deep learning, autoencoders, i-vectors, speaker verification

1. Introduction

Deep learning approaches have been applied in speaker recognition after showing their success in image and speech technologies [1, 2, 3, 4]. As a front-end, deep learning approaches are capable of learning deep features [5, 6, 7] and, the so-called bottle neck features (BNF) [8, 9]. These features are further used within a conventional GMM-UBM framework or in i-vector extraction process for speaker recognition. Deep learning approaches are applicable to learn a compact representation of speech utterances, which is commonly referred to as speaker embeddings, such as in [10, 11, 12, 13, 14]. As a backend, it has been successfully applied in combination with i-vectors such as in [15, 16, 17].

The compact representation of speech utterances known as i-vector [18] has been the state-of-the-art approach in speaker recognition, over the last years. Cosine and Probabilistic Linear Discriminant Analysis (PLDA) scoring are the two commonly used techniques to decide if two i-vectors belong to the same speaker. PLDA leads to a superior performance as it requires labeled background data. However, in practice, it is difficult to access large amount of labeled data. In i-vector based speaker verification, the lack of labeled data results in a significant performance gap between cosine and PLDA scoring techniques. In [19, 20], some unsupervised automatic labeling techniques are proposed but they cannot appropriately estimate the true labels. These approaches perform reasonably well but the results are still far from that of PLDA with actual labels [21].

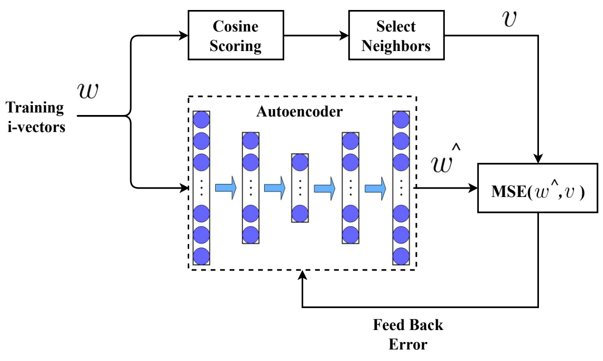
In [22, 23, 24], several attempts have been made to improve the performance of speaker verification, using unsupervised learning such as Restricted Boltzmann Machines (RBMs) and Deep Belief Networks (DBNs). As an example of front-end, in [22, 25], a vector representation of speakers was proposed by means of RBM adaptation. As a backend, in [26], various imposter selection algorithms are proposed, in order to reduce the performance gap between cosine and PLDA scoring techniques without the use of labeled data. They applied DBN adaptation as a backend for i-vector based speaker verification. These algorithms rely on training a separate model for each target speaker which is costly in terms of computations. However, the results are still far from i-vector/PLDA approach with actual speaker labels.

In this work, we put an effort to reduce the demand of labeled background data for i-vector based speaker verification. The goal is to reduce the performance gap between cosine and PLDA scoring techniques without using speaker labels. Unlike the conventional PLDA backend for i-vectors, and DNN based classifiers, autoencoder training is an unsupervised process which does not require labeled data. We propose to train an autoencoder in a new framework, in order to compensate session variability among i-vectors when no labeled background data is available. We train the autoencoder to reconstruct neighbor i-vectors, rather than to reconstruct the same training i-vectors. After the training, we extract speaker vectors for the testing i-vectors, which are referred to as autoencoder vectors or shortly ae-vectors. For the experimental trials, we score the ae-vectors using cosine scoring. The ae-vectors have shown high discriminative power compared to i-vectors. The experimental results show that while training the autoencoder in the proposed manner, a relative improvement of 42% is gained, over the baseline system using cosine scoring technique. This has reduced the performance gap between cosine and PLDA scoring techniques, by 92%.

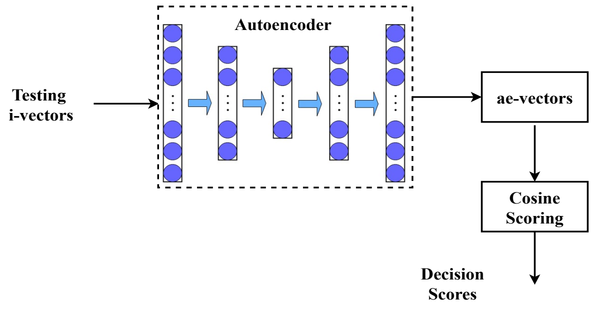
The rest of the paper is organized as follows. Section 2 explains the proposed method for training the autoencoder and the selection process of neighbor i-vectors. Section 3 describes the experimental setup and the database. The results obtained are discussed in Section 4. Finally in section 5, some conclusions are drawn as the findings of this paper.

2. Proposed method

Probabilistic Linear Discriminant Analysis (PLDA) scoring technique for i-vectors requires speaker labels, which is difficult to access in reality. On the other hand, cosine scoring technique avoids speaker labels at the cost of degrading the performance. In order to reduce the performance gap between these two scoring techniques when no labels are available for background data, we propose a new framework of autoencoder training which is fully unsupervised unlike the conventional DNN classifiers and PLDA. Furthermore, we propose to train the autoencoder in order to reconstruct similar i-vectors instead of the same training i-vectors. In this way, the autoencoder trained in



(a) *Proposed training of the autoencoder.*



(b) *Autoencoder vector extraction for the test i-vectors.*

Figure 1: *Block diagram of the proposed training and testing phases of the autoencoder.*

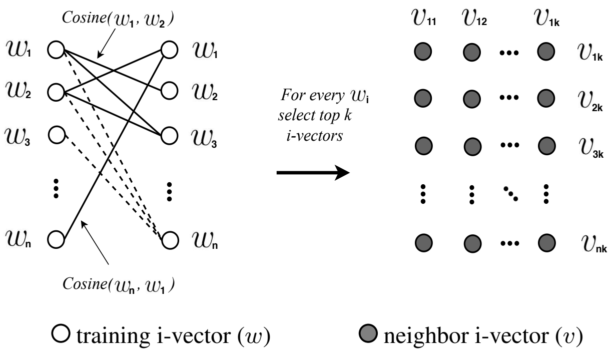
such unsupervised manner is capable of compensating session variability among i-vectors without using speaker labels.

Figure 1a shows the block diagram of the training phase of our proposed system. For an input i-vector w a similar i-vector v is set at the output and the training is carried out in this manner. The similar i-vectors are selected in an unsupervised manner according to the cosine scores to the training i-vector. After training, we extract speaker vectors for the testing i-vectors, which are used in the experiments as shown in Figure 1b. The main steps involved in extracting ae-vectors are explained as follows.

2.1. Autoencoder training

The conventional architecture of an autoencoder consists of an *encoder* and a *decoder* as shown in the enclosed block of Figure 1a. The *encoder* is a function that encodes the input i-vector w into a shorter dimensional space, and the *decoder* is a function that decodes it back in order to reconstruct w. The conventional training is carried out by minimizing the Mean Square Error (MSE) between the input w and the reconstructed $w\hat{}$. Thus the loss function is : $MSE(w\hat{},w)$, where $w\hat{} = decoder(encoder(w))$.

In this paper, we propose to train the autoencoder by minimizing the loss function : $MSE(w\hat{},v)$, as shown in Figure 1a, where v is a similar i-vector to w and $w\hat{} = decoder(encoder(w))$. We propose an automatic selection of similar i-vectors. Multiple similar i-vectors can be considered for every training i-vector w. In section 4, we will compare the results of our proposed training with that of a conventional training i.e., reconstructing the same training i-vectors.



Cosine(w_1, w_2)
w_1
w_2
w_3
w_n
w_1
w_2
w_3
w_n
Cosine(w_n, w_1)
For every w_i select top k i-vectors
v_{11} v_{12} v_{1k}
v_{1k}
v_{2k}
v_{3k}
v_{nk}
○ training i-vector (w)
● neighbor i-vector (v)

Figure 2: *Algorithm for selection of neighbor i-vectors.*

2.2. Selection of neighbor i-vectors

All the training i-vectors are scored among each other using cosine scoring technique. For every i-vector we select a set of similar i-vectors as neighbor i-vectors. A straightforward approach to select the neighbor i-vectors, is to apply a $threshold$ to the cosine scores between the training i-vectors. The i-vectors with scores higher than the $threshold$ are selected as neighbor i-vectors. This may lead to a variable number of neighbor i-vectors for every training i-vector.

Another approach is to consider a constant value k, and select k number of neighbor i-vectors for every training i-vector. This approach selects a uniform number of neighbors for every training i-vector which will lead to a balanced training.

Figure 2 shows a visualization of the selection process of the neighbor i-vectors for a constant k number of neighbors. Suppose w_i is a training i-vector, where $i = (1, \ldots, n)$ and n is the total number of training i-vectors. First, we score all the training i-vectors among each other using cosine scoring technique. Then, we select the top k i-vectors with highest scores as neighbor i-vectors for every w_i. The selected neighbor i-vectors are denoted by v_{ij}, where v_{ij} is the j^{th} neighbor of i^{th} training i-vector. Algorithm 1 summarizes how the selection of the neighbor i-vectors is carried out:

Algorithm 1: Proposed neighbor i-vectors selection algorithm for a constant k

```
Input  : Training i-vectors $w_i$, $1 < i < n$
Output: Neighbor i-vectors $v_{ij}$, $1 < i < n$ and
        $1 < j < k$
1 for each training i-vector $w_i$ do
2     for each training i-vector $w_t$, $1 < t < n$ do
3         if $i \neq t$ then
4             Compute $score_{i,t} = cosine(w_i, w_t)$
5         end
6     end
7     Select the corresponding $k$ i-vectors with the highest
        scores as $v_{i,j}$
8 end
```

Thus, we have a total of $n \times (k-1)$ samples for the autoencoder training. The values of $threshold$ and k are determined experimentally and will be discussed in section 4. Thus, the autoencoder is able to learn information about the session variability of i-vectors, without necessarily using actual speaker labels.

Table 1: *EER and minDCF for the proposed ae-vectors and i-vectors evaluated for different values of threshold using cosine scoring.*

Approach	threshold	EER(%)	minDCF
[1] i-vector	-	17.61	0.8390
[2] ae-vector	-	16.84	0.8569
[3] ae-vector	0.4	14.92	**0.8376**
[4] ae-vector	0.3	12.91	0.8594
[5] ae-vector	0.2	11.65	0.8420
[6] ae-vector	0.1	**11.19**	0.8527
Fusion of [1] & [6]	-	**10.17**	**0.7568**

2.3. Autoencoder vector extraction

Once the autoencoder is trained with the selected neighbor i-vectors, we transform the testing i-vectors into a new speaker vector representation, using the autoencoder as shown in Figure 1b. We extract the desired speaker vectors at the output of the autoencoder. These are referred to as autoencoder vectors or shortly ae-vectors. In the experiments, ae-vectors have shown to increase the discriminative quality of i-vectors without using speaker labels. Using ae-vectors, we perform the trials of the experiments with cosine scoring technique.

3. Experimental setup and database

The experiments were performed on VoxCeleb-1 database [27]. It contains 148,642 development and 4,874 test utterances, which belong to 1211 and 40 speakers, respectively. The development set was used to train the autoencoder using Keras deep learning library [28]. For the baseline i-vector/PLDA system, the development set was used to train the Universal Background Model (UBM), the Total Variability (TV) matrix and the PLDA parameters. MFCC features of 20 dimensions, appended by delta coefficients, were extracted for all the utterances in the development and test sets. A 1024 component UBM was trained to extract i-vectors of length 400. The PLDA was trained with 20 iterations and the number of eigenvoices was empirically set to 200. The UBM training, TV matrix training and i-vector extraction process were carried out using Alize toolkit [29]. From the test set, 37,720 experimental trials were scored. Half of them are client trials while the other half are impostor trials. The performance was evaluated using the Equal Error Rate (EER) and the minimum of the Decision Cost Function (minDCF) calculated using $C_M = C_{FA} = 1$, and $P_T = 0.01$, as in [27].

The autoencoder, used in this paper, is a fully connected feed forward network which consists of 3 hidden layers. The encoder and decoder parts are symmetrical as shown in Figure 1. The hidden layer 1 and 3 have 300 neurons each, while hidden layer 2 consists of 200 neurons. The input and output layers consist of 400 neurons each. The autoencoder training was carried out with 100 epochs using Stochastic Gradient Descent (SGD) optimizer. All the layers of the autoencoder used ReLU activation except the last layer which used linear activation. The learning rate was set to 0.01 with a decay of 0.0002 and the batch size was set to 100.

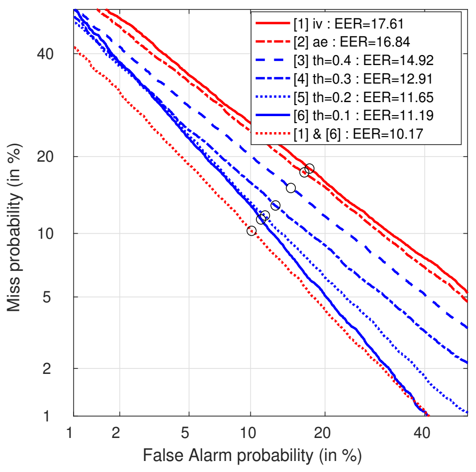


Figure 3: *DET curves for the proposed ae-vectors and i-vectors evaluated for different values of threshold using cosine scoring.*

4. Results

We have compared our proposed ae-vectors with baseline i-vectors and ae-vectors extracted using a conventionally trained autoencoder i.e., same output as input. In Tables 1 & 2, and in Figures 3 & 4, approach [2] corresponds to ae-vectors extracted using a conventionally trained autoencoder. Table 1 compares the performance of our proposed ae-vectors with the baseline i-vectors by setting a threshold to select number of neighbor i-vectors. All the vectors are scored using cosine scoring technique. Different values of $threshold$ were evaluated in order to tune the system for obtaining best results. From the table it is clear that our proposed ae-vectors has outperformed the baseline system. As we decrease the value of $threshold$, the performance of the system improves. The best EER of 11.19% was obtained for a $threshold$ equal to 0.1 which gains a relative improvement of 36% over the baseline i-vectors. This leads to fill the performance gap between cosine and PLDA scoring techniques by 80%. The best minDCF was obtained for a $threshold$ equal to 0.4.

A score level fusion of i-vectors and the ae-vectors with $threshold$ equal to 0.1, has further improved the performance, both in terms of EER and minDCF. An EER of 10.17% was obtained for the fusion. The optimum weights for the fusion were obtained empirically, and were set to 0.55 and 0.45 for i-vectors and ae-vectors, respectively. Figure 3 shows a comparison of the Detection Error Trade-off (DET) curves for the baseline and the proposed system. Different plots are shown for different ae-vectors obtained with different values of the $threshold$ parameter. It can be observed that all the ae-vectors, show better performance in all working regions, compared to the baseline i-vectors.

In Table 2, we have shown a performance comparison of our proposed ae-vectors with the baseline i-vectors using a constant k for the selection of neighbor i-vectors. We have experimented with different values of k in order to tune the system for obtaining best results. From the table it is clear that a fix value of k has shown improvement compared to the $threshold$ approach as well as to the baseline. Starting with k equal to 1, an EER of 15.32% was obtained. Thus, by considering only one nearest neighbor for every i-vector, a relative improvement

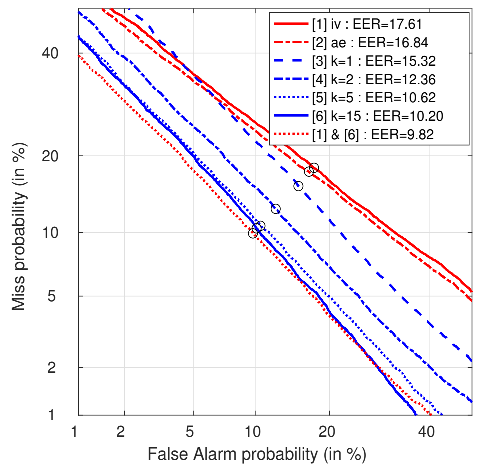


Figure 4: *DET curves for the proposed ae-vectors and i-vectors with different values of k using cosine scoring.*

Table 2: *EER and minDCF for the proposed ae-vectors and i-vectors for different values of k using cosine scoring.*

Approach	k	EER(%)	minDCF
[1] i-vector	-	17.61	0.8390
[2] ae-vector	-	16.84	0.8569
[3] ae-vector	1	15.32	0.9218
[4] ae-vector	2	12.36	0.8382
[5] ae-vector	5	10.62	**0.8053**
[6] ae-vector	15	**10.20**	0.8066
Fusion of [1] & [6]	-	**9.82**	**0.7625**

of 13% is gained, compared to the baseline system. As we increase the value of k, the performance of the system improves. The best EER of 10.20% was obtained for k equal to 15 which gains a relative improvement of 42% over the baseline i-vectors. This has filled the performance gap between cosine and PLDA scoring techniques by 92%. The best result in terms of minDCF, was obtained with k equal to 5. This approach allows a balanced training which improved the performance of the system. A further increase in the value of k may include very far neighbors, which degraded the performance.

A score level fusion of i-vectors and ae-vectors with k equal to 15, has further improved the performance, both in terms of EER and minDCF. An EER of 9.82% was obtained for the fusion, which is very close to the results for i-vector/PLDA using actual speaker labels. The optimum weights for the fusion were tuned experimentally and were set to 0.49 and 0.51 for i-vectors and ae-vectors, respectively. The DET curves for the baseline and the proposed system, using the second method, are shown in Figure 4. Different plots are shown for different ae-vectors obtained with different values of k. It can be observed that all the ae-vectors, have shown better performance than the i-vectors in all working regions.

If we perform a score level fusion between i-vector/PLDA and the ae-vectors with k equal to 15, which gives the best results, an EER of 9.0% is obtained. This gains a relative improvement of almost 6% over i-vector/PLDA. The fusion has improved the system in terms of minDCF as well. The optimum weights for the fusion were set to 0.04 and 0.96 for i-vectors and ae-vectors, respectively. The EER comparison and DET curves for the fusion are shown in Table 3 and Figure 5, respectively.

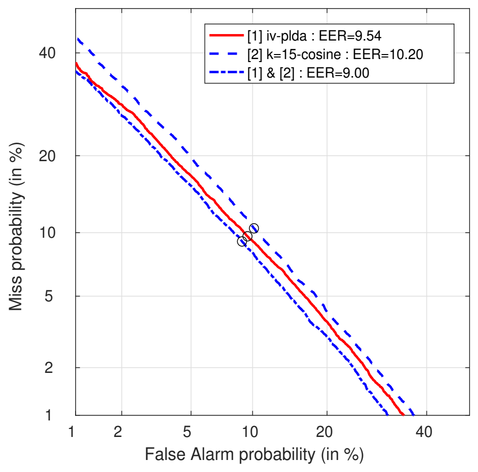


Figure 5: *DET curves for score level fusion between i-vector/PLDA and the proposed ae-vectors with k equal to 15.*

Table 3: *EER and minDCF for score level fusion between i-vector/PLDA and the proposed ae-vectors with k equal to 15.*

Approach	Scoring	EER(%)	minDCF
[1] i-vector	PLDA	9.54	0.7768
[2] ae-vector ($k = 15$)	Cosine	10.20	0.8066
Fusion of [1] & [2]	-	**9.00**	**0.7338**

5. Conclusions

A new framework of autoencoder training has been proposed in this work to increase the discriminative power of i-vectors for speaker verification. The main objective was to fill the performance gap between the cosine and the PLDA scoring techniques when no labeled background data is available. We have trained an autoencoder in order to reconstruct a set of neighbor i-vectors, instead of reconstructing the same training i-vectors. The neighbor i-vectors were selected as the closest to the training i-vector according to the cosine scores. After the training, speaker vectors were extracted for the test i-vectors as the output of the autoencoder. These speaker vectors were referred to as ae-vectors. For the experimental trials, ae-vectors were scored using the cosine scoring. The evaluation was performed on the speaker verification trials of VoxCeleb-1 database. The results have shown that our proposed ae-vectors gain a relative improvement of 42% in terms of EER over the baseline system. This has filled the gap between cosine and PLDA scoring systems by 92%.

6. Acknowledgements

This work has been developed in the framework of DeepVoice Project (TEC2015-69266-P), funded by Spanish Ministry.

7. References

[1] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, “A novel scheme for speaker recognition using a phonetically-aware deep neural network,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1695–1699.

[2] 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, 10 2015.

[3] 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, 11 2011.

[4] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, “Deep neural networks for extracting baum-welch statistics for speaker recognition,” in *Proc. Odyssey*, 2014, pp. 293–298.

[5] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[6] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, “Deep feature for text-dependent speaker verification,” *Speech Communication*, vol. 73, pp. 1–13, 2015.

[7] J. Jorrín, P. García, and L. Buera, “Dnn bottleneck features for speaker clustering,” *Proc. Interspeech 2017*, pp. 1024–1028, 2017.

[8] L. Deng, D. Yu *et al.*, “Deep learning: methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[9] T. Yamada, L. Wang, and A. Kai, “Improvement of distant-talking speaker identification using bottleneck features of dnn.” in *Interspeech*, 2013, pp. 3661–3664.

[10] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, “Deep neural networks for small footprint text-dependent speaker verification,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[11] Y. Z. Isik, H. Erdogan, and R. Sarikaya, “S-vector: A discriminative representation derived from i-vector for speaker verification,” in *23rd European Signal Processing Conference (EUSIPCO)*. IEEE, 2015, pp. 2097–2101.

[12] G. Bhattacharya, J. Alam, and P. Kenny, “Deep speaker embeddings for short-duration speaker verification,” in *Proc. Interspeech 2017*, 2017, pp. 1517–1521.

[13] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[14] K. Okabe, T. Koshinaka, and K. Shinoda, “Attentive statistics pooling for deep speaker embedding,” *arXiv preprint arXiv:1803.10963*, 2018.

[15] M. Senoussaoui, N. Dehak, P. Kenny, R. Dehak, and P. Dumouchel, “First attempt of boltzmann machines for speaker verification,” in *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[16] T. Stafylakis, P. Kenny, M. Senoussaoui, and P. Dumouchel, “Preliminary investigation of boltzmann machine classifiers for speaker recognition,” in *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[17] T. Stafylakis, P. Kenny, M. Senoussaoui, and P. Dumouchel, “Plda using gaussian restricted boltzmann machines with application to speaker verification,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[18] 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, 2011.

[19] E. Khoury, L. El Shafey, M. Ferras, and S. Marcel, “Hierarchical speaker clustering methods for the nist i-vector challenge,” in *Odyssey: The Speaker and Language Recognition Workshop*. Citeseer, 2014, pp. 254–259.

[20] S. Novoselov, T. Pekhovsky, and K. Simonchik, “Stc speaker recognition system for the nist i-vector challenge,” in *Odyssey: The Speaker and Language Recognition Workshop*, 2014, pp. 231–240.

[21] C. S. Greenberg, D. Bansé, G. R. Doddington, D. Garcia-Romero, J. J. Godfrey, T. Kinnunen, A. F. Martin, A. McCree, M. Przybocki, and D. A. Reynolds, “The nist 2014 speaker recognition i-vector machine learning challenge,” in *Odyssey: The Speaker and Language Recognition Workshop*, 2014, pp. 224–230.

[22] P. Safari, O. Ghahabi, and J. Hernando, “From features to speaker vectors by means of restricted boltzmann machine adaptation,” in *ODYSSEY 2016-The Speaker and Language Recognition Workshop*, 2016, pp. 366–371.

[23] O. Ghahabi and J. Hernando, “Restricted boltzmann machines for vector representation of speech in speaker recognition,” *Computer Speech & Language*, vol. 47, pp. 16–29, 1 2018.

[24] O. Ghahabi and J. Hernando, “Deep belief networks for i-vector based speaker recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1700–1704.

[25] U. Khan, P. Safari, and J. Hernando, “Restricted Boltzmann Machine Vectors for Speaker Clustering,” in *Proc. IberSPEECH*, 2018, pp. 10–14.

[26] O. Ghahabi and J. Hernando, “Deep learning backend for single and multisession i-vector speaker recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 807–817, 4 2017.

[27] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: a large-scale speaker identification dataset,” in *Interspeech*, 2017.

[28] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[29] A. Larcher, J. F. Bonastre, B. G. B. Fauve, K. A. Lee, C. Lévy, H. Li, J. S. D. Mason, and J. Y. Parfait, “Alize 3.0-open source toolkit for state-of-the-art speaker recognition.” in *Interspeech*, 2013, pp. 2768–2772.