INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Unsupervised Training of Siamese Networks for Speaker Verification

Umair Khan and Javier Hernando

TALP Research Center, Department of Signal Theory and Communications,
Universitat Politecnica de Catalunya Barcelona, Spain
{umair.khan, javier.hernando}@upc.edu

Abstract

Speaker labeled background data is an essential requirement for most state-of-the-art approaches in speaker recognition, e.g., x-vectors and i-vector/PLDA. However, in reality it is difficult to access large amount of labeled data. In this work, we propose siamese networks for speaker verification without using speaker labels. We propose two different siamese networks having two and three branches, respectively, where each branch is a CNN encoder. Since the goal is to avoid speaker labels, we propose to generate the training pairs in an unsupervised manner. The client samples are selected within one database according to highest cosine scores with the anchor in i-vector space. The impostor samples are selected in the same way but from another database. Our double-branch siamese performs binary classification using cross entropy loss during training. In testing phase, we obtain speaker verification scores directly from its output layer. Whereas, our triple-branch siamese is trained to learn speaker embeddings using triplet loss. During testing, we extract speaker embeddings from its output layer, which are scored in the experiments using cosine scoring. The evaluation is performed on VoxCeleb-1 database, which show that using the proposed unsupervised systems, solely or in fusion, the results get closer to supervised baseline.

Index Terms: i-vector, impostor selection, CNN, triplet loss

1. Introduction

Deep learning approaches have shown their success in image and speech technologies which has inspired the community to apply these approaches in speaker recognition as well [1, 2, 3]. At the frontend of speaker recognition, deep learning approaches are capable of learning deep features [4, 5] and, the so-called bottle neck features (BNF) [6, 7]. These features are further used within a conventional GMM-UBM framework or in i-vector [8] extraction process. At the backend, it has been successfully applied in combination with i-vectors such as in [9, 10, 11, 12]. Deep learning approaches are also applicable to learn a vector based representation of speech, which is commonly referred to as speaker embeddings, such as in [13, 14, 15, 16, 17, 18]. Speaker embeddings are typically extracted from an intermediate layer of a Deep Neural Network (DNN) which is trained as a classifier. The inputs to the network are feature vectors, like the Mel-Frequency Cepstral Coefficients (MFCC) or in some cases spectrograms. Whereas the output of the network is fed with the class (speaker) labels for the background data. Therefore, these deep learning approaches are typically constrained to labeled background data.

Probabilistic Linear Discriminant Analysis (PLDA) [19] is the most efficient backend for i-vectors [8]. PLDA leads to a superior performance as compared to cosine scoring but at the cost of labeled background data. However, in practice, it is difficult to access large amount of labeled data. In i-vector based speaker recognition, the lack of labeled background data results in a significant performance gap between cosine and PLDA scoring techniques [12]. Although, in [20, 21], automatic labeling techniques were proposed but they could not appropriately estimate the true labels. These approaches perform reasonably well but the results are still far from that of PLDA with actual labels [22]. On the other hand, in [23, 24] autoencoder based unsupervised post-processing of i-vectors was proposed, by the authors, which has improved the performance. These approaches were aimed to increase the discriminative power of i-vectors. Hence, they are applicable as a backend in the i-vector space only.

In this work, we put an effort to reduce the demand of labeled background data for speaker verification. The goal is: (1) to obtain end-to-end speaker verification scores, and (2) to extract speaker embeddings, from spectrograms without using speaker labels. We propose two different siamese networks [25], i.e., double-branch and triple-branch, which consist of two and three branches, respectively. Each branch is composed of a Convolutional Neural Network (CNN) encoder, inspired by the VGG architecture [26], which was recently adapted for speaker verification task in [27, 28, 29]. Typically, siamese networks are trained by feeding the training samples in pairs, e.g., [anchor, client] and [anchor, impostor]. Since the goal is to avoid speaker labels, we propose to generate the pairs of training samples in an unsupervised manner. The client samples are selected within one database according to the highest cosine scores with the anchor sample, in the i-vector space. The impostor samples are selected in the same way but from another database, provided that the two databases does not contain utterances from same speakers.

We propose the use of double-branch siamese network as a binary classifier by minimizing binary cross entropy loss. The selected client and impostor samples, paired one by one with the anchor samples, are fed into the network, and their respective binary labels of 1/0 are provided at the output in order to compute the loss. After training, we obtain decision scores for the speaker verification trials from the output of the network. Thus, our double-branch siamese deploys an end-to-end speaker verification system. On the other hand, the proposed triple-branch network is trained by minimizing triplet loss [30, 31]. The selected client and impostor samples, paired both at the same time with the anchor samples, are fed into the network in pairs of three. Each branch encodes the input sample into a vector based representation which is used to compute the triplet loss. In the testing phase, we extract speaker embeddings for the test data using a branch of the network. These embeddings are scored for the experimental trials using cosine scoring. The evaluation was performed on VoxCeleb-1 database [27]. The results show that using our proposed systems, despite of being unsupervised, the result get closer to a similar but fully supervised baseline. Moreover, fusion of the two proposed systems can further improve the performance.

http://dx.doi.org/10.21437/Interspeech.2020-1882

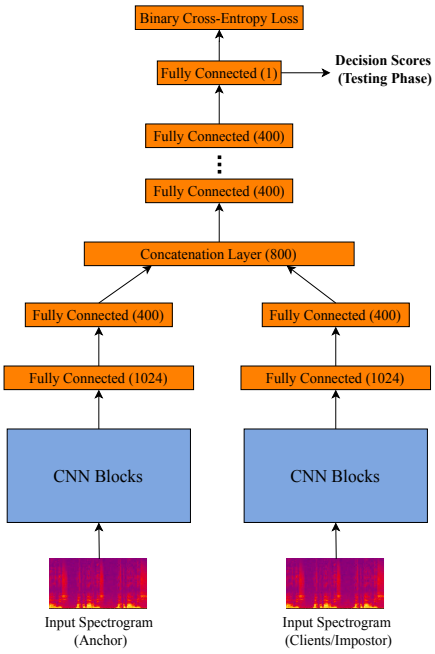

Figure 1: *Block diagram of our double-branch siamese network. In testing phase decision scores for speaker verification are obtained from the last FC layer.*

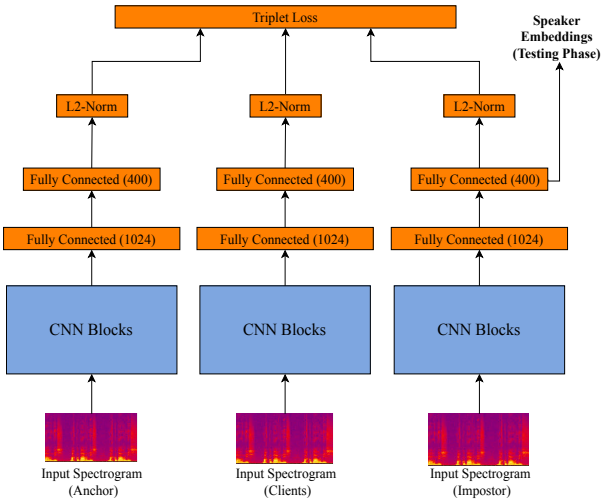

Figure 2: *Block diagram of our triple-branch siamese network. Speaker embeddings are extracted from the last FC layer of any CNN encoder branch.*

The rest of the paper is organized as follows. Section 2 explains the proposed method for training the siamese networks and the selection process of clients and impostor samples. 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

Training a Deep Neural Network (DNN), either in end-to-end fashion or to extract speaker embeddings, usually requires speaker labels for the background data, which is difficult to access in reality. In order to do so, when no labels are available for the background data, we propose the use of two siamese networks which are fully unsupervised unlike the conventional DNN classifiers. Typically, a siamese network is trained using pairwise training samples, i.e., anchor, client and impostor. Since we do not use speaker labels, we propose to generate the training pairs in an unsupervised manner. These training pairs are fed at the input of the network. We propose two different networks, i.e., a double-branch siamese network using binary cross-entropy loss, and a triple-branch siamese network using triplet loss.

After training, we obtain decision scores for the experimental trials at the output of the double-branch network. Whereas the triple-branch network is used to extract speaker embeddings for the test data. These speaker embeddings are scored using cosine scoring in order to perform the experimental trials.

2.1. Double-branch siamese network

Figure 1 shows the block diagram of our proposed double-branch siamese network. There are two identical branches, i.e., the CNN encoder. Mel-spectrogrm features of a training pair of an anchor along with a client or an impostor sample is fed as input to the network. The two branches share weights and biases with each other. After the CNN encoder, the outputs of the two branches are concatenated which is followed by five Fully Connected (FC) layers. The last layer is connected to the binary class labels, i.e., 1/0, indicating if the anchor sample is paired by a client/impostor sample, respectively. During training, binary cross-entropy loss is minimized to update the network weights. Once the network is trained with the selected client and impostor samples in unsupervised manner, we perform the evaluation in an end-to-end fashion. A pair of reference and test utterances, which is involved in an experimental trial, is fed into the network. The decision scores are obtained directly from the output of the last fully connected layer of the network. In this way, our double-branch siamese network deploys an unsupervised end-to-end speaker verification system which does not require any scoring backend.

2.2. Triple-branch siamese network

A block diagram of our proposed triple-branch siamese network is shown in Figure 2. As indicated by the name, there are three branches, each of which is the CNN encoder followed by l2-normalization layer. Each branch is fed by the Mel-spectrogrm features of a training pair of an anchor along with a client and an impostor sample. The CNN encoder encodes the Mel-spectrogram inputs into a vector based representation i.e., speaker embeddings. After the l2-normalization layer, triplet loss [30] is computed between the embeddings of anchor, client and impostor samples. It is worth noting that all the three branches share weights with each other, like in the double-branch network. The triplet loss is computed as follows:

$$L_{triplet} = max(d(a,c) - d(a,i) + m, 0) \qquad (1)$$

where $d(.)$ is the distance between two samples, and a, c and i denote the anchor, client and impostor samples, respectively. m is the margin value which defines how far away the dissimilarities should be. Once the network is trained, we extract speaker embeddings using any CNN encoder branch of the network. These speaker embeddings are scored using cosine scoring technique for the experimental trials.

Algorithm 1: Proposed algorithm for selection of clients and impostor i-vectors

Input : Training i-vectors $a_i \in A$ & $b_i \in B$, $1 \leq i \leq n$

Output: Client and impostor i-vectors C_{ij} and I_{ij}, $1 \leq i \leq n$ & $1 \leq j \leq k$

for *each training i-vector* a_i **do**
  for *each training i-vector* a_t, $1 \leq t \leq n$ **do**
    Compute $ClientScores_{i,t} = cosine(a_i, a_t)$
  end
  From $ClientScores_{i,t}$, select k highest ones.
  if $ClientScores_{i,t} \geq threshold$ **then**
    $C_{i,j} = a_t$
  end
  for *each training i-vector* b_t, $1 \leq t \leq n$ **do**
    Compute $ImpostorScores_{i,t} = cosine(a_i, b_t)$
  end
  From $ImpostorScores_{i,t}$, select k highest ones.
  if $ImpostorScores_{i,t} \geq threshold$ **then**
    $I_{i,j} = b_t$
  end
end

2.3. Selection of clients and impostor samples

The selection process of client and impostor samples is carried out in the i-vector space using two databases, i.e., A and B. Suppose Spk_A and Spk_B denote the speakers appearing in database A and B, respectively. We assume that the speakers in database A do not appear in database B, i.e., $Spk_A \cap Spk_B = \phi$. Algorithm 1 summarizes how the selection of the clients and impostor i-vectors is carried out in an unsupervised manner. First of all we extract i-vectors for all the utterances in both the databases A and B. Then, all the i-vectors in A are scored among each other using cosine scoring technique. For every i-vector in A we select a fix k number of similar i-vectors as potential client i-vectors. After this we apply a $threshold$ to the cosine scores of these k selected potential clients. The potential clients with scores higher than the $threshold$ are selected as the final client i-vectors.

In order to select the impostor i-vectors, we score all the i-vectors in A with those in B, using cosine scoring. For every i-vectors in A, we select k number of i-vectors from B that are closest according to the cosine scores. Since, the speakers in A does not appear in B, these k selected i-vectors are the potential impostors. After this, we apply a threshold to their corresponding cosine scores in order to select the hardest impostors among them. In this way, every i-vector in A has been assigned k client and k impostor i-vectors. Suppose we have n number of i-vectors in each of the databases A and B. Then, we have a total of $(2n \times k)$ samples for training our networks. The value of k is determined experimentally and will be discussed in section 4. For the double-branch siamese network we make training pairs of two, i.e., [anchor, client] and [anchor, impostor], for which the binary labels are 1 and 0 respectively. Whereas for the triple-branch siamese network we make pairs of three samples, i.e., [anchor, client, impostor], in order to compute the triplet loss according to Equation 1. It is worth noting the client and impostor selection is carried out in i-vector space while the actual inputs to the networks are Mel-spectrogram features of the utterances.

Table 1: *Architecture of the VGG based CNN Encoder. In and Out Dim. refers to the input and output feature maps of the layer. Feat Size is the dimension of every output feature map.*

Layer	Size	In dim	Out dim	Stride	Feat size
conv1-1	3x3	1	128	1x1	80xN
conv1-2	3x3	128	128	1x1	80xN
mpool-1	2x2	-	-	2x2	40xN/2
conv2-1	3x3	128	256	1x1	40xN/2
conv2-2	3x3	256	256	1x1	40xN/2
mpool-2	2x2	-	-	2x2	20xN/4
conv3-1	3x3	256	512	1x1	20xN/4
conv3-2	3x3	512	512	1x1	20xN/4
mpool-3	2x2	-	-	2x2	10xN/8
SAP	-	N/8	1	-	512x10
fc-1	-	1	1	-	1024
fc-2	-	1	1	-	400

2.4. CNN encoder

The CNN encoder block is inspired by the VGG architecture [26], which was recently adapted for speaker verification task in [27, 28, 29]. It consists of three main blocks, where each block contains two convolutional and one maxpooling layer. The three blocks are followed by a self attention pooling (SAP) layer [32], and two FC layers of 1024 and 400 neurons, respectively. The CNN encoder encodes the input Mel-spectrograms of shape $(80 \times N)$ into 400 dimensional vectors, i.e., speaker embeddings, where N is the number of frames. More details of the CNN encoder architecture are given in Table 1.

3. Experimental setup and database

The training was performed on the development partition of VoxCeleb-2 database [28] which contains 5994 speakers having 1,092,009 utterances in total. The evaluation was performed on the test partition of VoxCeleb-1 [27], which contains 40 speakers having 4,874 utterances in total. The development partition of VoxCeleb-2 was used to train the two siamese networks. For the i-vector extraction process, the same partition was used to train the Universal Background Model (UBM) and the Total Variability (TV) matrix. MFCC features of 20 dimensions, appended by delta coefficients, were extracted for all the utterances, and a 1024 component UBM was trained to extract i-vectors of length 400. From the test partition of VoxCeleb-1, 37,720 speaker verification 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).

For the clients and impostor selection we split the development partition of VoxCeleb-2 into two equal parts, in order to generate databases A and B as discussed in Section 2.3. Mel-spectrograms of 80 dimensions were computed, of which a randomly selected window of length N was input to the networks. The CNN encoder, depicted in Table 1, was identical for both the networks. The value of N, in Table 1, was set to 350 frames. The margin value m while computing the triplet loss defined in Equation 1 was set to 0.8. The value of $threshold$ in the selection of client and impostor samples was set to 0.2 and 0.0 (inspired from our work in [23, 33]), respectively. All the CNN and fully connected layers were activated using ReLU function, whereas the last layer of the double-branch network used sigmoid activation. The training was carried out using Adam optimizer with the initial learning rate and batch size of 0.0001 and

Table 2: *EER in % for the proposed double- and triple-branch siamese networks, compared with the supervised baselines.*

Approach	k	EER(%)
(1) iv-PLDA [23, 33]	-	9.54
(2) Baseline (Softmax)	-	6.81
(3) Baseline (AMSoftmax)	-	5.71
(4) Double-branch	2	7.81
(5) Double-branch	5	7.73
(6) Double-branch	10	6.90
(7) Triple-branch	10	6.95
Fusion of (6) & (7)	10	6.07

35, respectively. The training continued for a maximum of 500 epochs using early stopping criteria with patience equal to five.

The baseline system was trained using the whole development partition of VoxCeleb-2. The architecture of the baseline is composed of a CNN encoder followed by a classification layer at the end. In order to have a fair comparison, we used exactly the same architecture for the CNN encoder as that of the siamese networks shown in Table 1. The only difference is the last layer which has 5994 neurons (number of speakers) activated by softmax and AMsoftmax [34] functions. We minimized the categorical cross-entropy loss using Adam optimizer with the same parameters as of the siamese networks. In the testing phase, we extract speaker embeddings from the CNN encoder of this network.

4. Results

We have compared our proposed unsupervised systems with the supervised baseline in terms of Equal Error Rate (EER). Table 2 shows the EER in % for different values of k. The fist three rows of Table 2 depict results for the selection of the value of k using our double-branch siamese network. For all the experimental trials we obtain speaker verification scores directly from the output of the double-branch network. From the Table we can see that as we increase the value of k, the performance of the system improves. The best EER of 6.90% was achieved using k equal to 10. We have tried further higher values of k but no substantial improvement was seen, despite increasing the computational cost.

Setting the value of k equal to 10, we have trained our triple-branch siamese network using triplet loss, as discussed in Section 2.2. Using this network, first we extract speaker embeddings for all the test data which requires a scoring backend. Then, we score the obtained speaker embeddings using cosine scoring technique. From the Table we can see that the triple-branch siamese network has achieved an EER of 6.95% which is almost similar to that of the double-branch network. The EER of 6.90% and 6.95%, obtained by the two systems respectively, are very close to that of the supervised Softmax baseline.

Moreover, if we perform a score level fusion of our two proposed systems, i.e, double- and triple-branch networks, we obtain an EER of 6.07%. This has outperformed the softmax baseline by a relative improvement of 10.86%. However, the fusion strategy could not overcome the second baseline with AMSoftmax activation.

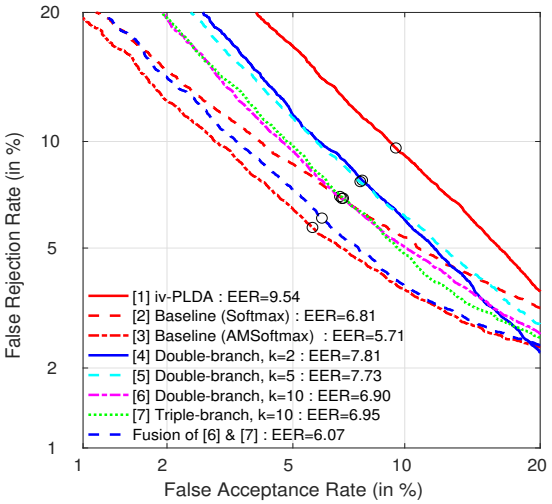

Figure 3: *DET curves for the proposed double- and triple-branch siamese network, compared with the supervised baselines. Different plots are shown for different values of k.*

Figure 3 shows a comparison of the Detection Error Tradeoff (DET) curves for the supervised baselines and our proposed systems. Different plots are shown for scores obtained with different values of the k using the double-branch siamese network. As discussed earlier, we observed that k equal to 10 is the best choice for our experiments. After this we plotted the DET curve for triple-branch for k equal to 10 only. It can be observed that both the proposed systems show comparable performance with the softmax baseline in the EER region. Furthermore, in the low FRR regions the DET plot of the proposed systems become closer to the AMSoftmax baseline. However, this relation seems to be reversed in the low FAR regions. Similarly, the DET curve for the fusion of the two proposed systems is even closer to the AMSoftmax baseline in low FRR regions.

5. Conclusions

In this work, we proposed two different siamese networks for speaker verification without using speaker labels. The first network has two branches and was trained as a binary classifier, whereas the second network has three branches and was trained to learn speaker embeddings, where each branch was a CNN encoder. Since the goal was to avoid speaker labels, the pairs of training samples were generated in an unsupervised manner. The client samples were selected within one database according to the highest cosine scores with the anchor, in i-vector space. Whereas, the impostor samples were selected in the same way but from another database. After training, we obtain decision scores using the double-branch network, whereas from the triple-branch network we extract speaker embeddings for the test data. The evaluation was performed on VoxCeleb-1 database. The experiments have shown that using our proposed systems, despite of being unsupervised, the result was closer to the fully supervised baselines.

6. Acknowledgements

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

7. References

[1] 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.

[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] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. Rouvier, P.-M. Bousquet, and B. Favre, "Speaker diarization through speaker embeddings," in *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE, 2015, pp. 2082–2086.

[15] 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.

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

[17] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *Proc. Interspeech 2018*, pp. 2252–2256, 2018.

[18] S. Novoselov, A. Shulipa, I. Kremnev, A. Kozlov, and V. Shchemelinin, "On deep speaker embeddings for text-independent speaker recognition," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 378–385.

[19] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[20] 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.

[21] 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.

[22] 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.

[23] U. Khan, M. India, and J. Hernando, "Auto-encoding nearest neighbor i-vectors for speaker verification," *Proc. Interspeech 2019*, pp. 4060–4064, 2019.

[24] U. Khan and J. Hernando, "Dnn speaker embeddings using autoencoder pre-training," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[25] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

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

[28] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Interspeech*, 2018.

[29] M. India, P. Safari, and J. Hernando, "Self Multi-Head Attention for Speaker Recognition," in *Proc. Interspeech 2019*, 2019, pp. 4305–4309.

[30] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[32] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[33] U. Khan, M. India, and J. Hernando, "i-vector transformation using k-nearest neighbors for speaker verification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7574–7578.

[34] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.