S-VECTOR: A DISCRIMINATIVE REPRESENTATION DERIVED FROM I-VECTOR FOR SPEAKER VERIFICATION

Yusuf Ziya Işık[1,2], *Hakan Erdogan*[2], *Ruhi Sarıkaya*[3]

[1]TUBITAK BILGEM , Gebze, Turkey
[2]Faculty of Engineering and Natural Sciences, Sabancı University, Turkey
[3]Microsoft Corporation, Redmond, WA, USA

yusuf.ziya@tubitak.gov.tr, haerdogan@sabanciuniv.edu,ruhi.sarikaya@microsoft.com

ABSTRACT

Representing data in ways to disentangle and factor out hidden dependencies is a critical step in speaker recognition systems. In this work, we employ deep neural networks (DNN) as a feature extractor to disentangle and emphasize the speaker factors from other sources of variability in the commonly used i-vector features. Denoising autoencoder based unsupervised pre-training, random dropout fine-tuning, and Nesterov accelerated gradient based momentum is used in DNN training. Replacing the i-vectors with the resulting speaker vectors (s-vectors), we obtain superior results on NIST SRE corpora on a wide range of operating points using probabilistic linear discriminant analysis (PLDA) back-end.

Index Terms— speaker verification, denoising autoencoder, random dropout

1. INTRODUCTION

With the recent success of deep neural networks on a number of machine learning tasks including speech recognition, representation learning has also gained significant attention of research community [1]. Learning good representations of data is critical for the performance of the machine learning algorithms. Representations learned using deep hierarchies outperformed hand-crafted features on several tasks in natural language processing [2–4], and image processing domains [5, 6].

One of the key methods that helps learning deep representations is unsupervised pre-training. The main motivation of unsupervised pre-training is that learning the underlying structure of the data may also help solving the particular task at hand. One of the methods for unsupervised learning of representations is denoising autoencoders [7]. Denoising autoencoders learn representations that are robust against corruptions on the input.

In this paper, we use unsupervised pre-training with denoising autoencoders and random dropout fine-tuning [8] to emphasize the hidden factors related with speaker identity in the i-vector representation. The new representation called speaker vectors, or s-vectors in short, eases the speaker detection task, and improves the performance of back-end classifiers. We experiment with the state-of-the-art PLDA back-end on NIST SRE corpora. The new s-vector representation outperformed the i-vector representation on a wide range of operating points especially in the low-false alarm rate region.

This work is not the first attempt in the literature to adopt the recent advancements in deep neural networks to the speaker verification problem. In [9], bottleneck features are proposed for speaker verification as an alternative to Mel Frequency Cepstral Coefficients (MFCC). In [10], Gaussian Bernoulli Restricted Boltzmann Machines (GB-RBM) are used as a replacement to the PLDA model. Two GB-RBMs are used to model *target* and *non-target* classes. Each GB-RBM takes the concatenation of two i-vectors to compare as the input vector. The log-likelihood ratio given by the two models are used as the final score during the test phase. A second attempt to replace PLDA using Gaussian RBMs is presented in [11]. They proposed a topology where the hidden layer has separate sub-blocks for the speaker and channel factors. This method may be used as a generative model to obtain log-likelihood ratio scores or as a method to reduce the dimensionality of i-vectors by projecting to speaker factors space. Unlike the works in [10, 11], we do not attempt to replace PLDA with a neural network. Instead, we present a method to transform the i-vectors to a new representation, which we call s-vector, more suitable for the speaker verification task. In [12], Gaussian mixture model, typically used to generate sufficient statistics necessary for i-vector extraction, is replaced with a deep neural network (DNN). Our approach, instead, uses a DNN to expose discriminative information in i-vectors that can not be utilized by linear models. In this respect, this work and [12], are complementary to each other.

The rest of the paper is organized as follows. In section 2 we present the denoising autoencoder, and in section 3 random dropout method is introduced. In section 4, we describe how we use these techniques to obtain the s-vector representation. Experimental results are given in 5, and finally in section 6, we conclude.

2. DENOISING AUTOENCODERS

Denoising autoencoder [7] is first proposed as a means of initializing a deep neural network with unsupervised pre-training. To obtain a good representation to initialize a deep neural network, deep autoencoders use robustness to corruption of the input as a training criterion. The denoising auto encoder first encodes a noisy version of the input, and then tries to decode the input itself from the learned encoding of the noisy version. Let us denote the input vector as x, and the corrupted version of the input as $\tilde{x}$. We first map the corrupted input $\tilde{x}$ to $y = h(W\tilde{x} + b)$, where h is a non-linearity function, W denotes the weights, and b is the bias term. We then decode y using $z = h(W'y + b')$. W' may optionally be selected as the transpose of W. We want z as close as possible to the original input x. For binary or near binary inputs, one possible corruption process, which is called partial destruction, is randomly zeroing out some portion of the data. For real valued inputs, adding Gaussian noise to the input may be more appropriate.

Denoising autoencoders aim to obtain representations that are robust to partial corruption or even destruction of the input. It is argued that robustness may only be possible if the learned representations capture the underlying hidden factors generating the data. In [13], using denoising autoencoders as building blocks with a final back-propagation fine-tuning step, deep architectures are built obtaining state-of-the art results on several image classification tasks.

3. RANDOM DROPOUT

Random dropout is a technique used in fine-tuning steps of the neural network learning process [8]. In random dropout, we zero out the output of each hidden unit with a probability of p during each presentation of a training sample. After the training phase, we use all of the hidden units, but multiply each of the weights by $(1-p)$ to compensate the extra amount of weights used during the test phase. Although it seems to be a simple ad hoc trick, the random drop out technique has a sound foundation and it is very powerful.

The first motivation to use random dropout is that by randomly dropping some of the hidden nodes for each training sample presentation, we prevent the hidden nodes to depend on each other. This makes it possible to use the whole capacity of the network and avoids overfitting. Random dropout could also be seen as a way of ensemble averaging. At each presentation of each training sample, we are actually working on a different network. All the networks sampled from our original architecture share the weights with each other, which can be seen as a strong regularization. At the test phase, we are using all the nodes and this could be seen as a form of ensemble averaging. So random dropout is a very effective way of training and using a very large ensemble of neural networks.

We can also apply the random dropout technique on the visible units perhaps with a smaller value of p. When applied to visible units, random dropout is similar to denoising autoencoders using a destructive noise process.

4. A MORE ROBUST REPRESENTATION THAN THE I-VECTOR

The i-vector model is an unsupervised method to represent the total variability space. The i-vector representation is believed to contain information related to the physical factors generating the speech utterance such as vocal tract length, microphone used, or room acoustics [14]. In this work, we try to find a new representation that suppresses the factors irrelevant to speaker verification task, and highlights the speaker identity related factors. The new representation will be more robust to variations in the subspaces of total variability space irrelevant to speaker identity. This may make the speaker detection task easier for the following shallow back-end classifiers.

To obtain such a representation from i-vectors, we start with denoising autoencoder based unsupervised pre-training. Since i-vectors are continuous input representations, Gaussian noise is added to the input during model training. Using the feature detectors learned by the denoising autoencoder as the initial starting point, we train a multi-layer perceptron (MLP) with random dropout fine-tuning procedure. During random dropout fine-tuning, we use the speakers in the training set as our output classes and employ a soft-max regression layer for the output. After the training phase of the MLP, we take the output of the last hidden layer as our final feature for speaker verification. We call this feature as the speaker vector representation, or s-vector in short.

5. EXPERIMENTS

5.1. Datasets

For NIST SRE12 [15], NIST provided lists of speech segments belonging to each of the 1918 SRE12 target speakers. These training speech segments are from SRE06, SRE08, and SRE10 corpora. The I4U consortium, one of the participants of NIST SRE12, divided these training segments into two speaker verification tasks *Dev* and *Eval*. Each task is composed of two lists: *Train* and *Test*. The utterances in Dev-Test and Eval-Test are non-overlapping. The Dev-Train and Dev-Test utterances are included in Eval-Train. The training and test utterances in the lists have different Lingustic Data Consortium labels. For each segment two noisy versions are generated, one having 6 dB SNR and the other having 15 dB SNR. We use 10 HVAC noise files and crowd noise files generated by summing several hundreds of utterances from NIST SRE corpora to generate noisy versions of the dataset. More detailed information about I4U development lists can be found in [16]. We used data in *Dev* list for training models in our experiments. The *Eval* list is used for testing.

5.2. Baseline Systems

We used 39 dimensional MFCC features containing 19 static, 19 delta and 1 delta-energy coefficients. We used an energy based bi-Gaussian classifier for voice activity detection. We applied feature warping with a 3s window to MFCC features after voice activity detection. We trained a 2048 mixture gender independent universal background model (UBM) using segments from NIST SRE04, SRE05 corpora as well as segments from the Dev-Train list. Noisy segments were not used in the UBM training. We used the same utterances for training gender-dependent i-vector models with the i-vector dimension set to 600.

For the PLDA baseline, we used linear discriminant analysis to further reduce the dimension of the i-vectors to 200. The i-vectors were then centered, whitened and unit normalized. We used these i-vectors to train a two-covariance PLDA model. In training linear discriminant analysis and two-covariance model, we also used the noisy versions of the original utterances. When the target speaker had multiple training segments, we used the average of the training i-vectors for training the target speaker model. Znorm score normalization was used for PLDA baseline system.

5.3. Neural Network System

We used the same utterances used in PLDA training to build DNN models. We used the utterances in *Dev-Test* list as validation data for optimization of the parameters and for early stopping. For the corruption process, we used additive Gaussian noise with zero mean and variance equal to 0.2. We used mini-batch stochastic gradient descent both for pre-training and fine-tuning the model. We experimented with various batch sizes and selected 200 as the optimal size. We chose a pre-training learning rate of 0.001 and a fine-tuning learning rate of 0.005. We employed early stopping both for back propagation and random dropout fine-tuning. For random dropout fine-tuning, we used 0.5 for the probability of omitting hidden nodes, and 0.2 for omitting the input nodes.

In [17], Sutskever et al. used Nesterov accelerated gradient (NAG) based momentum with stochastic gradient descent to train deep neural networks from random initializations. We applied NAG based momentum as explained in [17], and found it very useful especially for deep architectures. We used a maximum of 600 fine-tuning epochs. We used the Theano library [18] for our neural network implementation.

We experimented with sigmoid, tanh and rectified linear activation units for hidden nodes, obtaining the best results with tanh activation units. Hence, for all the results reported in this paper, we used tanh activation for the hidden nodes.

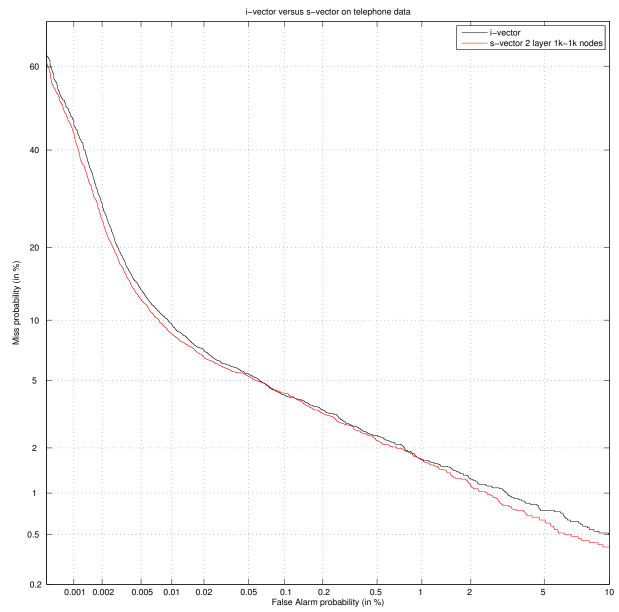

Fig. 1. *DET curve for i-vector and s-vector systems on telephone data. For extracting the s-vectors, a neural network having 2 hidden layers with 1k nodes on each layer is used.*

The output layer is a soft-max regression layer with a separate node for each of the speakers in the training set.

5.4. Results

We used the *Eval* task lists from I4U in our experiments. Only clean test data are used. We used equal error rate (EER), minimum value of decision cost function of NIST SRE08 evaluation (DCF08), and minimum value of decision cost function of NIST SRE10 evaluation (DCF10) as our performance metrics.

We compared the two representations using a PLDA backend. The performance of the two features are analyzed for various values of the feature dimension after LDA. For the s-vector case, we experimented with both single and two layer networks to understand if a deep representation is necessary for our task. For the single hidden layer case, we tried 400, 600, 1000 and 2000 as the number of hidden nodes. For the two hidden layers case, we tried three configurations for the number of hidden nodes; 1000-1000, 1000-600, and 600-600. The EER and minDCF values are shown in Table 1 for the telephone test data and in Table 2 for the interview test data.

For the telephone test data, we have achieved between 5 to 11% relative improvement with s-vector representation in EER and DCF10 metrics for every dimension of the LDA projected feature vectors. The configuration with 2 hidden layers each with 1000 nodes seems to have the best overall performance. Even with a single layer, the s-vector representation obtains better performance than the i-vector representation. The deep hierarchies calculate more abstract representations

Table 1. *Comparison of the i-vector and s-vector systems for clean telephone data.*

Systems	Dimension 200			Dimension 300			Dimension 400		
	DCF08	DCF10	EER	DCF08	DCF10	EER	DCF08	DCF10	EER
i-vector baseline	0.0517	0.1829	1.5	0.0509	0.1812	1.5	0.0519	0.1884	1.62
s-vector L1-h1000	0.0514	0.1753	1.52	0.0509	0.171	1.47	0.0528	0.1763	1.55
s-vector L1-h2000	**0.0492**	0.176	1.36	**0.0493**	0.176	**1.39**	0.0513	0.1831	1.49
s-vector L1-h600	0.0509	0.1739	1.5	0.0496	0.1745	1.47	0.0506	0.1768	1.52
s-vector L1-h400	0.0538	0.1798	1.65	0.0539	0.1768	1.7	0.0537	0.1791	1.66
s-vector L2-h1k-1k	0.0522	**0.1691**	**1.34**	0.0508	**0.1711**	1.43	0.0503	**0.1726**	**1.44**
s-vector L2-h1k-.6k	0.052	0.177	1.46	0.051	0.1747	1.55	0.0502	0.174	1.52
s-vector L2-h.6k-.6k	0.0517	0.1774	1.6	0.0509	0.1742	1.52	**0.0497**	0.1745	1.57

Table 2. *Comparison of the i-vector and s-vector systems for clean interview data.*

Systems	Dimension 200			Dimension 300			Dimension 400		
	DCF08	DCF10	EER	DCF08	DCF10	EER	DCF08	DCF10	EER
i-vector baseline	0.0273	0.1351	**0.6**	**0.023**	0.1313	**0.5**	0.0239	0.1363	**0.48**
s-vector L1-h1000	0.0283	**0.1264**	0.69	0.0236	0.1246	0.6	**0.0236**	0.1283	0.52
s-vector L1-h2000	**0.0256**	0.1277	0.6	0.0243	0.1258	0.55	0.0239	0.1323	0.51
s-vector L1-h600	0.028	0.1308	0.63	0.0256	0.1273	0.62	0.0253	0.1275	0.6
s-vector L1-h400	0.0299	0.1331	0.7	0.0277	0.1265	0.65	0.0263	0.1293	0.57
s-vector L2-h1k-1k	0.0277	0.1273	0.67	0.0245	**0.1222**	0.63	0.0254	0.1239	0.55
s-vector L2-h1k-.6k	0.0291	0.1308	0.69	0.0271	0.1255	0.57	0.0268	**0.1236**	0.59
s-vector L2-h.6k-.6k	0.03	0.1292	0.69	0.0265	0.1292	0.6	0.0251	0.1267	0.61

as we go up on the hierarchy. The input of our neural network system, which is the i-vector representation, is already an abstract one. This may be the reason for obtaining performance improvements even with a single layer network. In Figure 1, we give the DET plots for the i-vector baseline and the s-vector representation obtained using a neural network having 2 hidden layers with 1000 nodes on each layer for the telephone test data. We can see that the s-vector representation consistently outperforms the i-vector representation in a wide range of operating points, especially in the very low and high false alarm rate regions.

For the interview test data, we still obtain relative performance improvements in the range 6 to 9.3% for the very low false alarm region measured by minimum of the DCF10 metric. However, we did not observe similar achievements in the equal error rate region.

Table 3. *Percentage of the within-class energy in total energy for the i-vector and s-vector representations.*

	Clean Data	Clean & Noisy Data
i-vector	58.76	62.786
s-vector	52.06	53.439

To better understand if we achieved our goal of suppressing factors irrelevant to speaker identity, we computed the percentage of energy due to within-class variance in the total energy. We used 600 dimensional s-vectors obtained with a single hidden layer neural network. For the clean data case, major sources of intra-speaker variability are channel, microphone and speaking style differences between sessions of the same speaker. In Table 3, we see a significant drop in within-class energy percentage using the s-vector representation. When we also use noise added data in addition to clean data, we see a smaller increase in within-class energy percentage for the s-vector compared to the i-vector. This indicates that s-vector is more robust to noise compared to the i-vector.

6. CONCLUSIONS

We proposed a method to obtain more discriminative and robust features from the i-vector representation. This is achieved by using a deep neural network to emphasize the speaker factors and suppress the other factors. We obtained better performance by using the s-vectors as input to a state-of-the-art PLDA back-end classifier. The new representation is also very cheap to compute at the test phase. As a future work, we plan to jointly train a neural network for tasks like gender and language recognition besides speaker verification with the hope of obtaining more robust and disentangled features.

7. ACKNOWLEDGEMENT

Part of this work is funded by the TÜBİTAK grant number 110G082.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent, “Representation learning: A review and new perspectives.,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, Aug. 2013.

[2] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, “Natural language processing (almost) from scratch,” *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[3] Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran, “Deep belief nets for natural language call-routing,” in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5680–5683.

[4] Puyang Xu and Ruhi Sarikaya, “Convolutional neural network based triangular crf for joint intent detection and slot filling,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 78–83.

[5] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al., “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, vol. 19, pp. 153–160, 2007.

[7] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[8] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” Tech. Rep. 1207.0580, arXiv, 2012.

[9] Sibel Yaman, Jason Pelecanos, and Ruhi Sarikaya, “Bottleneck features for speaker recognition,” in *Odyssey 2012 - The Speaker and Language Recognition Workshop*, 2012.

[10] Mohammed Senoussaoui, Najim Dehak, Patrick Kenny, Réda Dehak, and Pierre Dumouchel, “First attempt of boltzmann machines for speaker verification,” in *Odyssey 2012 - The Speaker and Language Recognition Workshop*, 2012.

[11] Themos Stafylakis, Patrick Kenny, Mohammed Senoussaoui, and Pierre Dumouchel, “Plda using gaussian restricted boltzmann machines with application to speaker verification.,” in *Proc. Interspeech*, 2012.

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

[13] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *The Journal of Machine Learning Research*, vol. 9999, pp. 3371–3408, 2010.

[14] Patrick Kenny, “A small footprint i-vector extractor,” in *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[15] Craig S Greenberg, Vincent M Stanford, Alvin F Martin, Meghana Yadagiri, George R Doddington, John J Godfrey, and Jaime Hernandez-Cordero, “The 2012 NIST speaker recognition evaluation,” in *Proc. Interspeech*, 2013.

[16] Rahim Saeidi, Kong Aik Lee, Tomi Kinnunen, Tawfik Hasan, Benoit Fauve, Pierre-Michel Bousquet, Elie Khoury, et al., “I4U submission to NIST SRE 2012: A large scale collaborative effort for noise-robust speaker verification,” in *Proc. Interspeech*, 2013.

[17] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1139–1147.

[18] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010, Oral Presentation.