Deep Neural Network Approaches to Speaker and Language Recognition

Fred Richardson, *Member, IEEE,* Douglas Reynolds, *Fellow, IEEE,* Najim Dehak, *Member, IEEE*

***Abstract*— The impressive gains in performance obtained using deep neural networks (DNNs) for automatic speech recognition (ASR) have motivated the application of DNNs to other speech technologies such as speaker recognition (SR) and language recognition (LR). Prior work has shown performance gains for separate SR and LR tasks using DNN posteriors of sub-phonetic units and DNN feature representations. In this work we present the application of single DNN for both SR and LR using the 2013 Domain Adaptation Challenge speaker recognition (DAC13)and the NIST 2011 language recognition evaluation (LRE11) benchmarks. Using a single DNN trained on Switchboard data we demonstrate large gains on performance in both benchmarks: a 55% reduction in EER for the DAC13 out-of-domain condition and a 48% reduction in C_{avg} on the LRE11 30s test condition. It is also shown that further gains are possible using score or feature fusion leading to the possibility of a single i-vector extractor producing state-of-the-art SR and LR performance**

Index Terms: i-vector, DNN, bottleneck features, senone posteriors, tandem features, speaker recognition, language recognition

I. Introduction

The impressive gains in performance obtained using deep neural networks (DNNs) for automatic speech recognition (ASR) [1] have motivated the application of DNNs to other speech technologies such as speaker recognition (SR) and language recognition (LR) [2–10]. Two general methods of applying DNN's to the SR and LR tasks have been shown to be effective. The first or "direct" method uses a DNN trained as a classifier for the intended recognition task directly to discriminate between speakers for SR [5] or languages for LR [4]. The second or "indirect" method uses a DNN trained for a different purpose to extract data that is then used to train a secondary classifier for the intended recognition task. Applications of the indirect method have used a DNN trained for ASR to extract frame-level features [2, 3, 11], accumulate a multinomial vector [7] or accumulate multi-modal statistics [6, 8] that were then used to train an i-vector system [12, 13].

Manuscript received Month Day, Year; accepted Month Day, Year. Date of publication Month Day, Year; date of current version Month Day, Year. The associate editor coordinating the review of this manuscript and approving it for publication was Honorific Firstname Lastname.

Fred Richardson and Douglas Reynolds are with MIT Lincoln Laboratory, Lexington, MA 02421 USA (email: frichard@ll.mit.edu, dar@ll.mit.edu).

N. Dehak is with the MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139 USA (e-mail: najim@csail.mit.edu).

This work was sponsored by the Department of Defense under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

An aim of this paper is to examine the use of a single DNN for both SR and LR tasks, with the possibility of using a single i-vector classifier. Prior published work has examined the application of DNNs for SR or LR separately. In Section V we describe development experiments which motivate the focus on two indirect methods. The first indirect method (bottleneck features or BNFs) uses frame-level features extracted from a DNN with a special bottleneck layer [14] and the second indirect method (DNN posteriors) uses posteriors extracted from a DNN to accumulate multi-modal statistics [6]. The features and statistics from these indirect methods are used to train i-vector classifiers for each task and combinations of methods. Contrastive and fusion experiments for well defined SR and LR benchmarks are given in Section V.

II. DNN's for SR and LR

A DNN classifier is essentially a multi-layer perceptron with more than two hidden layers that typically uses random initialization and stochastic gradient descent to initialize and optimize the weights [1, 15]. For speech applications, the input to a DNN is a stacked set of spectral features (e.g., MFCCs, PLPs) extracted from short (20ms) segments (frames) of speech. Typically a context of +/- 5 to 10 frames around the current input frame are used. The output of the DNN is a prediction of the posterior probability of the target classes for the current input frame (see Figure 1)

In the direct method for LR and SR, a DNN is used to predict the language or speaker class for a given frame of speech. Since the entire speech waveform is considered to belong to a single class, the frame-level DNN posteriors must be combined to make a single decision score. This can be accomplished either by simply averaging the DNN predictions or by training a secondary classifier, such as a multinomial, that uses statistics across the whole input derived from the DNN as a single feature vector.

In contrast to the direct method, the indirect method uses a DNN that was trained on a different data set and possibly for a different purpose. In this work, we have used a DNN trained for an ASR task for both LR and SR. The ASR DNN is to predict sub-phonetic units or "senones" for each input frame [1]. In the following two subsections we describe how we use the ASR DNN output posteriors and BNFs in the context of an i-vector classifier.

A. DNN posteriors

A typical i-vector system uses zeroth, first and second order statistics generated using a Gaussian mixture model

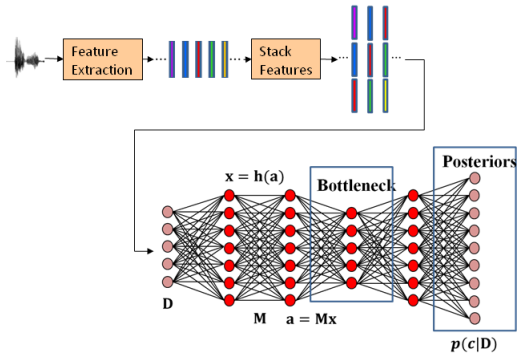

Fig. 1. Example DNN architecture

(GMM) [12]. Statistics are accumulated by first estimating the posterior of each GMM component density for a frame and using these posteriors as weights for accumulating the statistics for each component of the mixture distribution. The zeroth order statistics are the total occupancies across an utterance for each GMM component and the first order statistics are the occupancy weighted accumulations of feature vectors for each component. The i-vector is then computed using a dimension reducing transformation applied to the stacked first order statistics that is non-linear with respect to the zeroth order statistics.

An alternate approach to extracting statistics has been proposed in [6]. Statistics are accumulated in the same way as for the GMM but class posteriors from the DNN are used in place of GMM component posteriors. Once the statistics have been accumulated, the i-vector extraction is performed in the same way as it is from the GMM based posteriors. This approach has been shown to give significant gains for both SR and LR [6, 7, 16].

B. DNN bottleneck features

A DNN can also be used as a means of extracting features for use by a secondary classifier - including another DNN [17]. This is accomplished by sampling the activation of one of the DNN's hidden layers and using this as a feature vector. For some classifiers the dimensionality of the hidden layer is too high and some sort of feature reduction is necessary like LDA or PCA. In [14], a dimension reducing linear transformation is optimized as part of the DNN training by using a special bottleneck hidden layer that has fewer nodes (see Figure 1). The bottleneck layer uses a linear activation and behaves very much like a LDA or PCA transformation on the activation of the previous layer [14, 18]. Matrix factorization was originally proposed in [18] to reduce the number of parameters of the output layer, but in our work we have chosen to use the second to last layer with the hope that the output posterior prediction would not be too adversely affected by the loss of information at the bottleneck layer. BNFs have been shown to work well for both LR and SR [2, 3, 10, 11].

III. I-Vector System

In the experiments, an i-vector classifier, configured as described below, was used for baseline and integrated DNN systems. Speech activity segmentation generated using a GMM based speech activity detector (GMM SAD) was used for all systems. The front-end feature extraction for the baseline LR system uses 7 static cepstra appended to 49 shifted delta cepstra (SDC) for a total of 56 features. Unlike the front-end described in [19], vocal track length normalization (VTLN) and feature domain nuisance attribute projection (fNAP) are not used. The front-end for the baseline SR system uses 20 MFCCs including C0 and their first derivatives for a total of 40 features. Features extracted directly from the DNN bottle neck layer are used for BNF experiments.

All i-vector systems use a 2048 component GMM and 600 dimensional i-vectors. For DNN posterior experiments, the DNN output posteriors are used instead of those from the GMM, but DNN posterior weighted statistics from the data used to train the GMM are needed for normalization. After extraction, i-vectors are length normalized [20], the mean of enrollment files are used as target models, and PLDA is used for scoring.

Hyperparameters for SR and LR tasks were trained using their respective training data with the above i-vector configuration. All LR systems use the discriminative backend described in [19].

IV. Development Experiments

For our initial work towards developing a DNN for LR, we used a small six language sub-set of the NIST 2009 Language Recognition Evaluation (LRE09) corpora that includes Farsi, Hindi, Korean, Mandarin, Russian and Vietnamese [21, 22]. The training partition consists of 20 hours of data per language (10 hours each of VoA and CTS data) for a total of 120 hours of speech. The test data is the subset of the LRE09 evaluation that matches the same six languages. This corpus was selected to match experiments reported in [4]. In all the development experiments we use a cheating backend on the test data and report the EER at 30, 10 and 3 seconds.

We first focused on the direct approach discussed in Section II. The DNN used here was trained using the training partition of the data set and consisted of 819 input nodes (stack of 21 frames of 13 Gaussianized [23] PLP coefficient and their first and second order derivatives), 2-5 hidden layers with 2560 nodes per layer, and 6 output nodes. All hidden layers used a sigmoid activation. The DNN training is preformed on an nVidia Tesla K40 GPU using custom software developed at MIT/CSAIL. Scores for each language were generated by averaging the 6 DNN frame-level output log posteriors for each test file.

Results using the direct approach along with the baseline i-vector system are given in Table I. While there is a substantial degradation in performance relative to the baseline system at the 30s and 10s durations using this technique, there is a slight gain at the short 3s duration for the 2 layer DNN which is consistent with results reported in [4]. It is not clear why there is further performance loss for the 5 layer DNN. Based on these results, we abandoned further direct approach experiments.

We next examined the DNN BNF and the DNN posterior technique discussed in Section II. The DNN for these experiments was trained on 100 hours of Switchboard 1 data [24]

TABLE I
DEVELOPMENT DIRECT PERFORMANCE (EER %)

Layers	Nodes	30 sec	10 sec	3 sec
Baseline		0.684	2.84	11.4
2	2560	2.58	4.27	10.3
5	2560	3.24	5.65	12.8

TABLE II
DEVELOPMENT INDIRECT PERFORMANCE (EER %)

Features	Posteriors	30 sec	10 sec	3 sec
SDC	GMM	0.684	2.84	11.4
BNF	GMM	0.208	1.30	7.37
SDC	DNN	0.268	1.58	10.1
BNF	DNN	0.298	1.35	8.73

using 4,199 state cluster (senone) target labels generated using the Kaldi Switchboard-1 tri4a example system [25]. The same 819 input features were used as the above direct approach DNN. The DNN has 7 hidden layers of 1024 nodes each with the exception of the 6$^{\text{th}}$ bottleneck layer which has 64 nodes. All hidden layers use a sigmoid activation function with the exception of 6$^{\text{th}}$ layer which is linear [14].

The results in Table II show that the BNFs together with GMM posteriors give the best performance and that in general all systems using the DNN significantly out perform the baseline system. Based on these results we focused next on applying this ASR DNN to more challenging LR and SR tasks.

Too many experiments were run during development to present in full, but here are some of the interesting observations. Increasing the amount of training data and the number of parameters in the DNN both by up to a factor of three did not significantly improve performance. The inclusion of the bottle neck layer did not adversely affect the DNN posterior based results. Appending delta features to the PLP features gave a small but significant performance gain. Most interesting, without Gaussianization on the input PLP features performance degraded by well over a factor of two.

V. SR AND LR BENCHMARK EXPERIMENTS

In this section we present experiments with the indirect DNN approaches on some well defined SR and LR benchmarks. The SR systems were trained and evaluated using the 2013 Domain Adaptation Challenge (DAC13) [26]. The DAC13 is a specified set of hyper-parameter, enroll, and test lists developed to exhibit a data domain shift for a SR task and has been reported on in several publications [16, 27, 28]. The LR systems were evaluated on the NIST 2011 Language Recognition Evaluation (LRE11) data [29] which covers 24 languages coming from telephone and broadcast audio and has test durations of 3, 10, and 30 seconds. Details on the LR training and development data can be found in [19].

A. Speaker recognition experiments

Two sets of experiments were run on the DAC13 corpora: "in-domain" and "out-of-domain". For both sets of experiments, the UBM and $\mathbf{T}$ hyper-parameters are trained on Switchboard (SWB) data. The other hyper-parameters (whitening, within, and across covariances) are trained on 2004-2008 speaker recognition evaluation (SRE) data for the in-domain experiments and SWB data for the out-of-domain experiments (see [26] for more details). Tables III and IV summarize the results for the in-domain and out-of-domain experiments with the first row of each table corresponding to the baseline system. While the DNN-posterior technique with MFCCs gives a significant gain over the baseline system for both sets of experiments, as also reported in [6] and [16], an even greater gain is realized using BNF with a GMM. However, using both BNFs and DNN-posteriors degrades performance.

TABLE III
IN-DOMAIN DAC13 RESULTS

Features	Posteriors	EER(%)	DCF*1000
MFCC	GMM	2.71	0.404
MFCC	DNN	2.27	0.336
BNF	GMM	**2.00**	**0.269**
BNF	DNN	2.79	0.388

TABLE IV
OUT-OF-DOMAIN DAC13 RESULTS

Features	Posteriors	EER(%)	DCF*1000
MFCC	GMM	6.18	0.642
MFCC	DNN	3.27	0.427
BNF	GMM	**2.79**	**0.342**
BNF	DNN	3.97	0.454

B. Language recognition experiments

The experiments run on the LRE11 task are summarized in Table V with the first row corresponding to the baseline system and the last row corresponding to a fusion of 5 "post-evaluation" systems (see [19] for details). BNFs with GMM posteriors out performs the other systems configurations including the 5 system fusion. Interestingly, BNFs with DNN-posteriors show more of an improvement over the baseline system than in the speaker recognition experiments.

C. Score and feature fusion

Scores from the four speaker recognition systems in Tables III and IV were fused by combining them with uniform weights. Out of all possible pair-wise combinations, the BNF/GMM+MFCC/DNN systems yielded the best performance. The results are summarized in Table VI. For the out-of-domain case the 4 system fusion is actually worse than fusing just the BNF/GMM+MFCC/DNN systems perhaps due to the poorer performance of the MFCC/GMM system in this condition. For the in-domain case the BNF/GMM+MFCC/DNN system fusion comes very close to fusing all four systems. While it is possible that better performance could be attained

TABLE V
LRE11 RESULTS C_{avg}

Features	Posteriors	30s	10s	3s
SDC	GMM	5.26	10.7	20.9
SDC	DNN	4.00	8.21	19.5
BNF	GMM	**2.76**	**6.55**	**15.9**
BNF	DNN	3.79	7.71	18.2
5-way fusion [19]		3.27	6.67	17.1

TABLE VI
FUSION OF ALL SYSTEM AND THE TOP 2 SYSTEM ON DAC13. THE SYSTEM NOTATION USED IS [FEATURE]/[POSTERIOR].

Fusion	Condition	EER(%)	DCF*1000
All 4 systems	In-domain	1.61	0.236
BNF/GMM + MFCC/DNN	In-domain	1.65	0.237
Tandem/GMM	In-domain	**1.55**	**0.229**
All 4 systems	Out-of-domain	2.88	0.355
BNF/GMM + MFCC/DNN	Out-of-domain	2.54	0.326
Tandem/GMM	Out-of-domain	**2.44**	**0.323**

TABLE VII
LRE11 FUSION C_{avg}

Fusion	30s	10s	3s
All 4 systems	**2.22**	**5.41**	**14.5**
BNF/GMM + SDC/DNN	2.31	5.69	14.7
Tandem/GMM	2.67	6.71	15.9

by estimating the optimal weights for combining scores on held-out data or via cross-validation, we believe that the naive fusion using uniform weights is a good indication of how well fusion works between these different systems. The best in-domain score fusion gives a performance gain of almost 20% relative to the BNF/GMM system alone while the best out-of-domain score fusion gives a relative gain of only about 9%.

Also included in Table VI is the result of stacking 20 MFCC features with the 64 BNFs and retraining the GMM I-vector system with the resulting 84 tandem features [30]. The performance for the tandem feature system is slightly better than score fusion for the DAC13 task. The tandem approach may be of interest in limited resource scenarios where it is not possible to run more than one i-vector system.

Score fusion experiments using the four language recognition systems in Table V were carried out by training a discriminative backend on the development data over all two system combinations and comparing the top performing pair to the fusion of all four systems. As in the DAC13 fusion experiments, the BNF/GMM+MFCC/DNN gave the best performance of all two system combinations. The results are summarized in Table VII. While the fusion gains are relatively modest (roughly a 10% relative improvement across the durations), the fusion of just the BNF/GMM+MFCC/DNN is only slight worse than the fusion of all four systems. DET plots for the out-of-domain performance of the baseline, DNN, fused and tandem systems respectively are shown in Figure 2.

The tandem system performs worse than score fusion on the LRE11 task but is on par with the BNF/GMM system. This may be because the features used for score fusion and for the tandem features are not the same. The MFCC features used in the tandem system perform well on the SR task but are not as suited to the LR task as the SDC features used in the SDC/DNN system. However, the tandem/GMM system's result suggests that one could use the same tandem feature representation for both LR and SR and still realize a gain on the SR task. This may be of interest in situations where i-vectors are extracted with one set of hyper parameters and then used for both the LR and SR tasks.

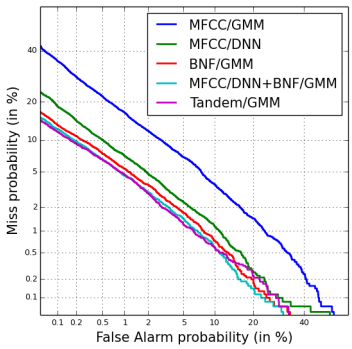

Fig. 2. DAC13 out-of-domain DET plot

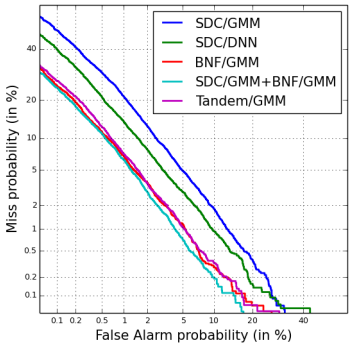

Fig. 3. LRE11 30sec duration

VI. CONCLUSIONS

This paper has described the development of a DNN BNF i-vector system and demonstrated substantial performance gains when applying the system to both the DAC13 SR and LRE11 LR benchmarks. For the DAC13 task the BNF/GMM system was shown to reduce the error rates of the baseline MFCC/GMM system by 26% for EER and 33% for DCF for the in-domain task and 55% for EER and 47% for DCF for the out-of-domain task. On LRE11, the same BNFs decreased EERs at 30s, 10s, and 3s durations by 48%, 39%, and 24%, respectively, and even out performed a 5 system fusion of acoustic and phonetic based recognizers.

Further reductions in error were demonstrated on the DAC13 SR task using score fusion or tandem features. Fusing the BNF/GMM and MFCC/DNN system scores reduces the error rates relative to the BNF/GMM system by 18% for EER and 12% for DCF for the in-domain task and by 9% for EER and 5% for DCF for the out-of-domain task. Using tandem features lead to a larger reduction in error rate of 23% for EER and 15% for DCF for the in-domain task and 13% for EER and 6% for DCF by for the out-of-domain task. Score fusion on the LRE11 task lead to 16%, 13% and 8% reduction in C_{avg} on the 30s, 10s and 3s durations conditions. While the tandem features did not lead to significant changes in performance on the LRE11 task, their good performance on DAC13 suggests the possibility of a single tandem front-end and a single I-vector extractor for both SR and LR applications.

Acknowledgments: The authors would like to thank Patrick Cardinal, Yu Zhang and Ekapol Chuangsuwanich at MIT CSAIL for sharing their DNN expertise and GPU optimized DNN training software.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, pp. 82–97, November 2012.

[2] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "I-vector representation based on bottleneck features for language identification," *IEEE Electronics Letters*, pp. 1569–1580, 2013.

[3] P. Matejka, L. Zhang, T. Ng, H. S. Mallidi, O. Glembek, J. Ma, and B. Zhang, "Neural network bottleneck features for language identification," in *Proceedings of IEEE Odyssey*, 2014, pp. 299–304.

[4] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Proceedings of ICASSP*, 2014, pp. 5374–5378.

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

[6] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proceedings of ICASSP*, 2014, pp. 1714–1718.

[7] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer, "Application of convolutional neural networks to language identification in noisy conditions," in *Proceedings of IEEE Odyssey*, 2014, pp. 287–292.

[8] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proceedings of IEEE Odyssey*, 2014, pp. 293–298.

[9] O. Ghahabi and J. Hernando, "I-vector modeling with deep belief networks for multi-session speaker recognition," in *Proceedings of IEEE Odyssey*, 2014, pp. 305–310.

[10] S. Yaman, J. Pelecanos, and R. Sarikaya, "Bottleneck features for speaker recognition," in *Proceedings of IEEE Odyssey*, 2012.

[11] A. K. Sarkar, C.-T. Do, V.-B. Le, and C. Barras, "Combination of cepstral and phonetically discriminative features for speaker verification," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1040–1044, Sept. 2014.

[12] N. Dehak, P. Kenny, R. Dehak, P. Ouellet, and P. Dumouchel, "Front end factor analysis for speaker verification," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 19, no. 4, pp. 788–798, may 2011.

[13] N. Dehak, P. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via ivectors and dimensionality reduction," in *Proceedings of Interspeech*, 2011, pp. 857–860.

[14] Y. Zhang, E. Chuangsuwanich, and J. Glass, "Extracting deep neural network bottleneck features using low-rank matrix factorization," in *Proceedings of ICASSP*, 2014, pp. 185–189.

[15] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proceedings of ICASSP*, 2013.

[16] D. Garcia-Romero, X. Zhang, A McCree, and D. Povey, "Improving speaker recognition performance in the domain adaptation challenge using deep neural networks," in *submitted to SLT*, 2014.

[17] K. Vesely, M. Karafiat, and F. Grezl, "Convolutive bottleneck network features for lvcsr," in *Proceedings of IEEE ASRU*, 2011, pp. 42–47.

[18] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proceedings of ICASSP*, 2013.

[19] E. Singer, P. Torres-Carrasquillo, D. Reynolds, A. McCree, F. Richardson, N. Dehak, and D. Sturim, "The mitll nist lre 2011 language recognition system," in *Proceedings of IEEE Odyssey*, 2011, pp. 209–215.

[20] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proceedings of Interspeech*, 2011, pp. 249–252.

[21] "The nist 2009 language recognition evaluation," http://www.itl.nist.gov/iad/mig/tests/lre/2009/.

[22] P. Torres-Carrasquillo, E. Singer, T. Gleason, A. McCree, D. A. Reynolds, F. Richardson, and D. Sturim, "The mitll nist lre 2009 language recognition system," in *Proceedings of ICASSP*, 2010, pp. 4994–4997.

[23] B. Xiang, U. Chaudhari, J. Navratil, G. Ramaswamy, and R. Gopinath, "Short-time gaussianization for robust speaker verification," in *Proceedings of ICASSP*, 2002.

[24] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proceedings of ICASSP*, 1992, pp. 517–520.

[25] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J Silovsky, G. Stemmer, and K. Vesel, "The kaldi speech recognition toolkit," in *Proceedings of IEEE ASRU*, 2011.

[26] S. H. Shum, D. A. Reynolds, D. Garcia-Romero, and A. McCree, "Unsupervised clustering approaches for domain adaptation in speaker recognition systems," in *Proceedings of IEEE Odyssey*, 2014, pp. 265–272.

[27] H. Aronowitz, "Inter dataset variability compensation for speaker recognition," in *Proceedings of ICASSP*, 2014.

[28] O. Glembek, J. Ma, P. Matejka, B. Zhang, O. Plchot, L. Burget, and S. Matsoukas, "Domain adaptation via within-class covariance correction in i-vector based speaker recognition systems," in *Proceedings of ICASSP*, 2014.

[29] "The nist 2009 language recognition evaluation," http://www.itl.nist.gov/iad/mig/tests/lang/2009/.

[30] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *Proceedings of ICASSP*, 2000.