Spoken Language Recognition using X-vectors

David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell,
Daniel Povey, Sanjeev Khudanpur

Center for Language and Speech Processing
Human Language Technology Center of Excellence
The Johns Hopkins University, USA

david.ryan.snyder@gmail.com, dgromero@jhu.com, alan.mccree@jhu.com, gsell@jhu.com,
dpovey@gmail.com, khudanpur@jhu.com

Abstract

In this paper, we apply x-vectors to the task of spoken language recognition. This framework consists of a deep neural network that maps sequences of speech features to fixed-dimensional embeddings, called x-vectors. Long-term language characteristics are captured in the network by a temporal pooling layer that aggregates information across time. Once extracted, x-vectors utilize the same classification technology developed for i-vectors. In the 2017 NIST language recognition evaluation, x-vectors achieved excellent results and outperformed our state-of-the-art i-vector systems. In the post-evaluation analysis presented here, we experiment with several variations of the x-vector framework, and find that the best performing system uses multilingual bottleneck features, data augmentation, and a discriminative Gaussian classifier.

1. Introduction

The National Institute for Standards and Technology (NIST) organized its 8th language recognition evaluation (LRE) in 2017, with an ongoing goal of promoting development in language recognition technology and measuring its performance. Like the previous evaluation in 2015, the NIST LRE 2017 focused on distinguishing between dialects and closely related languages. It was separated into an unconstrained *open* training condition and a *fixed* training condition, in which only a specified set of training resources were permitted. In addition to conversational telephone speech and broadcast narrowband speech data, this evaluation also included wideband speech extracted from videos.

Most modern language recognition systems are based on i-vectors [1]. The standard approach consists of a universal background model (UBM), and a large projection matrix $\mathbf{T}$ that are trained to maximize the training data likelihood. The projection maps high-dimensional statistics from the UBM into a low-dimensional representation, known as an i-vector. Once extracted, i-vectors are commonly classified using Gaussian models, logistic regression, or cosine scoring. State-of-the-art i-vector systems incorporate deep neural networks (DNN) trained as acoustic models for automatic speech recognition (ASR). This is generally done in one of two ways: either posteriors from the ASR DNN replace those from a Gaussian mixture model (GMM) or the i-vector system is trained on bottleneck features extracted from the DNN [2, 3, 4, 5, 6].

Using DNNs to directly capture language or speaker characteristics is currently a very active area of research. In language recognition, a popular approach is to train the neural network to classify languages at the frame-level [7, 8, 9, 10]. At test-time, utterance-level scores are computed by averaging log-posteriors across time. In this work, we adapt the x-vector framework we developed recently for speaker recognition [11] to language recognition. This framework consists of a discriminatively trained DNN that maps variable-length speech segments to embeddings called x-vectors. Once extracted, x-vectors are used like i-vectors. This allows for leveraging the classification and backend technology that has been developed for i-vector-based language recognition systems.

2. X-vector System

2.1. Overview

The x-vector system is based on a framework that we developed for speaker recognition [11]. The system is comprised of a feed-forward DNN that maps variable-length speech segments to embeddings that we call *x-vectors*. Once extracted, the x-vectors are classified by the discriminatively trained Gaussian classifier in Section 4.

The network is implemented using the nnet3 neural network library in the Kaldi Speech Recognition Toolkit [12]. The recipe is based on the SRE16 v2 recipe available in the main branch of Kaldi as `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2`. The training classes and features have been modified for language

recognition.

2.2. Architecture

Layer	Layer context	Tot. context	In x out
frame1	$[t-2, t+2]$	5	$5F$x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xL

Table 1: The standard x-vector DNN architecture. X-vectors are extracted at layer *segment6*, before the nonlinearity. The input layer accepts F-dimensional features. The L in the softmax layer corresponds to the number of training languages.

The DNN configuration is outlined in Table 1. The input to the DNN is a sequence of T speech frames. The first 5 layers process the input at the frame-level, with a small temporal context centered at the current frame t. For example, the input layer, *frame1*, splices together the F-dimensional features at frames $t-2$, $t-1$, t, $t+1$ and $t+2$, which gives it a total temporal context of 5 frames. The input to the next layer, *frame2*, is the spliced output of *frame1* at $t-2$, t and $t+2$. This builds on the temporal context established by the previous layer, so that *frame2* sees a total context of 9 frames. This process is continued in the following layers, and results in *frame5* seeing a total context of 15 frames.

The statistics pooling layer aggregates information across the time dimension so that subsequent layers operate on the entire segment. The input to the pooling layer is a sequence of T 1500-dimensional vectors from the previous layer, *frame5*. The output is the mean and standard deviation of the input (each 1500-dimensional vectors). These statistics are concatenated together (to produce a 3000-dimensional vector) and passed through the segment-level layers and finally the softmax output layer. The nonlinearities are rectified linear units (ReLUs). The networks studied in Section 6 have between 4.2 and 4.6 million parameters.

2.3. Training

The network is trained to classify languages using a multiclass cross entropy objective function (Equation 1). The primary difference between training here and in [8, 9, 10] is that our system is trained to classify languages from variable-length segments, rather than frames. Suppose there are L languages in N training segments. Then $P(\text{lang}_l \mid \mathbf{x}_{1:T}^{(n)})$ is the probability of language l given T input frames $\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ...\mathbf{x}_T^{(n)}$. The quantity d_{nl} is 1 if the language label for segment n is l, otherwise it's 0.

$$E = -\sum_{n=1}^{N}\sum_{l=1}^{L} d_{nl} \log P(\text{lang}_l \mid \mathbf{x}_{1:T}^{(n)}) \qquad (1)$$

Training examples are constructed by picking speech chunks from the training data that are 2–4 seconds long. The corresponding target is the language label. The network is trained for several epochs using stochastic natural gradient descent [13].

2.3.1. Data Augmentation

We use augmentation to increase the amount and diversity of the x-vector DNN training data. Our strategy uses speed perturbation, additive noises and reverberation. Reverberation involves convolving room impulse responses (RIR) with audio and is performed with the multicondition training tools in the Kaldi *ASpIRE* recipe [12]. Speed perturbation alters the speed of the speech signal using a specified speed factor [14]. The augmentation datasets we used are described in Section 5.2.3.

We use a 6-fold augmentation strategy that combines the original "clean" training list with 5 augmented copies. To augment a recording, we randomly choose between one of the following:

- **speed perturbation**: Apply a speed factor of 0.9 or 1.1 to slow down or speed up the original recording.
- **music**: A single music file (without vocals) is randomly selected from MUSAN (see Section 5.2.3), trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- **noise**: MUSAN noises are added at one second intervals throughout the recording (0-15dB SNR).
- **reverb**: The training recording is artificially reverberated via convolution with simulated RIRs.

2.4. Embeddings

At test time, 512-dimensional x-vectors are extracted at layer *segment6* of the network (see Table 1), before the nonlinearity. Since the pooling layer aggregates across the input frames, we are able to extract a single x-vector from each speech segment.

Alternatively, the x-vector DNN could be used to classify languages directly. This possibility is explored in Section 6.4. However, we find that extracting embeddings is more flexible, as it facilitates integration with the

backend technology developed for i-vectors. Also, it allows expanding to new languages without needing to retrain the x-vector extractor. The latter option is explored in Section 6.5.

2.5. Features

We experiment with acoustic features and two types of bottleneck features. Before being fed into the x-vector DNN, features are mean-normalized over a 3-second sliding window, and nonspeech frames are removed using an energy-based speech activity detection system.

2.5.1. Acoustic Features

The acoustic features are 23 MFCCs with a frame-length of 25ms.

2.5.2. Fisher English BNFs

For the fixed training condition of LRE17, 60-dimensional linear bottleneck features (BNF) are extracted from an ASR DNN trained on the Fisher English corpus. The DNN is a time-delay acoustic model built using the nnet2 library in Kaldi. Besides the bottleneck layer, it uses the same architecture and training recipe as the system described in Section 2.2 of [15].

Its input features are 40 MFCCs with a frame-length of 25ms. Cepstral mean subtraction is performed over a sliding window of 6 seconds. The DNN has six layers, and a total left-context of 13 and a right-context of 9. The hidden layers use the p-norm (where $p = 2$) nonlinearity and have an input dimension of 3500 and an output dimension 350. The penultimate layer is a 60 dimensional linear bottleneck layer. The softmax output layer computes posteriors for 5297 triphone states. No fMLLR or i-vectors are used for speaker adaptation. Excluding the output layer, which is not needed to compute BNFs, the DNN has 9.2 million parameters.

2.5.3. Multilingual BNFs

For the open training condition, we use 60-dimensional BNFs extracted from an ASR DNN trained on multiple languages. Up to and including the bottleneck layer, the DNN has the exact same architecture as Section 2.5.2. It also uses the same features.

The DNN is trained on 23 languages from the IARPA Babel dataset (Amharic, Cebuano, Guarani, Javanese, Lao, Tagalog, Tokpisin, Zulu, Assamese, Dholuo, Haitian, Kazakh, Lithuanian, Pashto, Tamil, Turkish, Cantonese, Igbo, Kurmanji, Mongolian, Swahili, Telugu, and Vietnamese). Each language is given a separate output layer that computes posteriors over 4300 to 4900 triphone states, depending on the language. The hidden layer parameters are shared across languages.

3. I-vector Baseline

Our baselines consist of two state-of-the-art joint i-vector systems introduced by McCree *et al*., in [6]. These systems were part of our LRE17 submission, and are described in greater detail in [16]. The joint system differs from the classical i-vector system in that both the means and the *weights* of a GMM for a recording are permitted to differ from the UBM. This allows it to model both acoustic and phonotactic variability.

The first joint i-vector system is a senone system, which uses posteriors from an ASR acoustic model trained on Fisher English. The second system uses BNFs extracted from a separate acoustic model, also trained on Fisher English.

4. Classifier

We use the same classifier for x-vectors and i-vectors. Prior to classification, embeddings are whitened, length normalized [17], and dimensionality is reduced using linear discriminant analysis (LDA). Classification is performed by a discriminatively-trained Gaussian classifier [18]. Fusing scores from multiple systems was performed by learning a weight for each system, and averaging weighted scores.

The whitening matrix, LDA, and classifier were trained on an augmented version of the LRE17 training data (Section 5.2). A held-out portion was used for fusion. Training recordings were segmented to durations of 3-60 seconds of speech. In addition to applying augmentation similar to what is described in Section 2.3.1, we also simulate GSM-AMR phone encoding, add babble noise from Fisher, and add synthetic, low-frequency modulated noise.

5. Corpora

5.1. NIST LRE 2017

Cluster	Languages
Arabic	Egyptian Arabic, Iraqi Arabic, Levantine Arabic, Maghrebi Arabic
Chinese	Mandarin, Min Nan
English	British English, General American English
Slavic	Polish, Russian
Iberian	Caribbean Spanish, European Spanish, Latin American Continental Spanish, Brazilian Portuguese

Table 2: NIST LRE 2017 language clusters and languages.

We evaluate performance on the 2017 NIST language recognition evaluation (LRE) [19]. It consists of 14 lan-

guages in 5 language clusters, as illustrated in Table 2. The data came from conversational telephone speech and broadcast narrowband speech data from the LDC MLS14 corpus, as well as wideband speech extracted from videos (VAST corpus). The focus of the evaluation is differentiating between closely related languages/dialects within a cluster.

Training conditions were broken down into *fixed* and *open* training conditions. The fixed condition permits use of past LREs, Fisher English, Switchboard corpora and LRE17 development data, as well as publicly available noise corpora. The open condition lifts these restrictions.

5.2. Training Data

This section describes the training data used by both the i-vector baselines and x-vector systems. The ASR DNNs were trained on additional corpora, as described in Section 2.5.

5.2.1. NIST LRE 2017 Training and Development

This dataset consists of about 18,000 training segments and 4,000 development segments provided by NIST for the evaluation participants. It is comprised of narrowband and broadband data collected by LDC for the MLS14 and VAST corpora and contains speech from the 14 target languages/dialects in Table 2. Due to the small amount of broadband training data, it was downsampled to 8kHz before training.

5.2.2. 2015 NIST IVC

This dataset consists of audio from the ivector challenge (IVC) [20]. This dataset is composed primarily of previous NIST LREs. It consists of data collected over the telephone channel. In total, there are 177,000 speech segments with an average duration of 35 seconds from 57 languages. This dataset is used only in Section 6.5.

5.2.3. Augmentation Datasets

We use the simulated room impulse responses (RIRs) described by Ko *et al*. in [21] for reverberation. Additive noises come from the MUSAN dataset, which consists of over 900 noises, 42 hours of music from various genres and 60 hours of speech from twelve languages [22]. To comply with the requirements of the fixed training condition of the evaluation, we only used the noises and music without vocals. Both MUSAN and the RIR datasets are freely available from `http://www.openslr.org`.

6. Results

Results are reported in terms of the C_{primary} metric described in [19]. This is the average of the actual detection costs using $P_{\text{target}} = 0.1$ and $P_{\text{target}} = 0.5$. In the following tables, the *Overall* results have been equalized by the NIST scoring tool. As a result, scores from each data source (VAST or MLS14) or language have been balanced and contribute equally to the metric.

6.1. Baseline

In this section, we compare performance of two state-of-the-art joint i-vector systems described in Section 3 with the x-vector system. The system ivec_senone uses posteriors from an ASR DNN trained on Fisher English and ivec_bnf is trained on Fisher English BNFs. The system xvec_bnf is an x-vector system trained on the Fisher English BNFs from Section 2.5.2.

System	MLS14	VAST	Overall
1 ivec_senone	0.193	0.217	0.205
2 ivec_bnf	0.170	0.206	0.189
3 xvec_bnf	0.148	0.178	0.163
1,2 fusion	0.151	0.183	0.167
1,3 fusion	0.130	0.168	0.150
2,3 fusion	0.125	0.164	0.145
1,2,3 fusion	0.124	0.163	0.144

Table 3: Comparison with i-vectors on LRE17 evaluation set. All systems conform to the fixed training condition.

In Table 3 we see that xvec_bnf achieves lower C_{primary} than either i-vector system on both the MLS14 and VAST parts of the evaluation. Overall, xvec_bnf outperforms ivec_senone by about 20% and ivec_bnf by about 14% relative.

The last 4 rows of Table 3 report score fusion results. By itself, xvec_bnf achieves a C_{primary} roughly equal to the fusion of the two i-vector systems. Moreover, the x-vector and i-vector systems appear to be very complementary when fused; the fusion of ivec_bnf and xvec_bnf is 23% better than our best baseline, and 13% better than the fusion of the two i-vector systems.

6.1.1. Duration analysis

NIST LREs typically include test segments with durations clustered around 3, 10 and 30 seconds. This facilitates the analysis of performance as a function of duration. For LRE17 only the MLS14 portion of the data was provided with these duration patterns. The VAST data had a continuous distribution over 10 seconds or longer.

Table 4 shows results by duration. It is clear that the shorter segments are harder to classify by all systems. Also, even though the VAST data had longer duration segments, the domain shift with respect to the majority of our training data results in performance that is worse than the 3 second MLS14 subset. Since the x-vector system was trained on examples that are on average 3 seconds long, it was expected that it might achieve its largest gains (compared to the baselines) on the shortest test segments.

System	MLS14			VAST
	3s	10s	30s	
1 ivec_senone	0.216	0.178	0.084	0.217
2 ivec_bnf	0.194	0.156	0.064	0.206
3 xvec_bnf	0.169	0.135	0.061	0.178
1,2 fusion	0.171	0.140	0.062	0.183
1,3 fusion	0.148	0.125	0.055	0.168
2,3 fusion	0.143	0.119	0.050	0.164
1,2,3 fusion	0.142	0.117	0.052	0.163

Table 4: Results split by test segment durations. The VAST subset is 10s or longer.

However, Table 4 demonstrates no clear relationship between the x-vector's relative performance and duration: compared to the baselines, xvec_bnf is 13–22% better on the 3 second segments, 13–24% better on the 10 second segments, and 5–27% better on the 30 second segments.

6.2. X-vector Features

It has been well documented that i-vector-based language recognition systems improve greatly by incorporating ASR DNNs. In this section, we see how this observation generalizes to x-vectors, by comparing systems trained on MFCCs (xvec_mfcc), Fisher English BNFs (xvec_bnf), and multilingual BNFs (xvec_mlbnf). See Section 2.5 for a description of these these features.

System	MLS14	VAST	Overall
xvec_mfcc	0.209	0.203	0.206
xvec_bnf	0.148	0.178	0.163
xvec_mlbnf	0.130	0.149	0.140

Table 5: X-vector performance using MFCCs, Fisher English BNFs and multilingual BNFs.

In Table 5 we see that x-vector performance echoes a similar trend observed in i-vectors: monolingual BNFs perform much better than acoustic features alone, but multilingual BNFs are the best choice [5]. Using just MFCCs, xvec_mfcc achieves performance comparable to ivec_senone from the previous section. However, replacing MFCCs with Fisher English BNFs improves performance by 21%. Finally, substituting MFCCs with multilingual BNFs improves performance even further, by 32%.

6.3. Data Augmentation

In this section, we test the importance of augmenting the x-vector DNN training data. In either system, the features are multilingual BNFs and the Gaussian classifier still uses the same augmentation strategy as described in Section 4.

System	MLS14	VAST	Overall
xvec_mlbnf_no_aug	0.152	0.179	0.166
xvec_mlbnf	0.130	0.149	0.140

Table 6: Comparison of performance with or without augmenting the x-vector DNN training list. In either case, the classifier (Section 4) uses the same augmented list.

In Table 6 we observe that removing augmentation significantly degrades performance. This is likely due to augmentation increasing the limited amount of training data, as well as making the system more robust against degraded audio. This result parallels our observations training x-vectors for speaker recognition in [11].

6.4. Direct Classification vs. Embeddings

The x-vector framework is based on our work in speaker recognition [11] where the goal is to produce embeddings that generalize to unseen speakers. However, in a closed-set language recognition task, the x-vector DNN can be used directly for classification, provided it is trained on the same language classes as required for deployment. In this section, direct classification is compared with using embeddings extracted from the same system.

System	MLS14	VAST	Overall
xvec_mlbnf_direct	0.155	0.256	0.206
xvec_mlbnf	0.130	0.149	0.140

Table 7: Comparison of performance using the x-vector DNN for direct classification, or as features for the Gaussian classifier from Section 4.

In Table 7 we see that using embeddings to train the Gaussian classifier achieves much better performance than using the system directly for classification. In particular, the direct system appears to suffer from the limited amount of VAST training data. While xvec_mlbnf is only 16% better than xvec_mlbnf_direct on MLS14 it is 42% better on VAST. Although it's likely the direct results could be improved with hyper-parameter tuning and calibration in the backend, this underscores the flexibility of our standard x-vector approach. Once extracted, x-vectors can be fed into the same pipeline used for i-vectors, taking advantage of existing classifier and backend technology that assists in domain adaptation and calibration.

6.5. Adding New Languages without Retraining

Another advantage of extracting embeddings from a DNN, rather than using it for direct classification, is

that it opens up the possibility of deploying the system with a different set of languages without having to retrain the x-vector extractor. Given a multilingual training list with sufficient diversity, it may be possible to train the DNN to produce embeddings that generalize to languages or dialects not present in the training data. In this section, we simulate this by training an x-vector DNN called xvec_mlbnf_57lang using the 57 languages of the IVC dataset (see Section 5.2.2). Although different training data is used, the same augmentation strategy (see Section 2.3.1) we used for xvec_mlbnf is applied to xvec_mlbnf_57lang. It is important to note that the IVC dataset may contain a significant acoustic domain mismatch with the LRE17 evaluation data as it is from a different collection.

System	MLS14	VAST	Overall
xvec_mlbnf_57lang	0.153	0.183	0.168
xvec_mlbnf	0.130	0.149	0.140

Table 8: Performance with an x-vector DNN trained on the IVC dataset (Section 5.2.2) or on the LRE17 development data (Section 5.2).

In Table 8 we see that xvec_mlbnf_57lang lags behind the system trained on completely in-domain data and matching languages. Nonetheless, it achieves similar performance as xvec_mlbnf_no_aug. In the future, we plan to expand the number of languages as well as the domains represented in the x-vector training data to further explore this approach.

7. Conclusions

In this paper, we adapted the x-vector framework, which was originally developed for speaker recognition, to the task of language recognition. We found x-vectors achieved excellent performance on the NIST LRE 2017, outperforming several state-of-the-art i-vector systems. We explored several variations to the basic x-vector framework. We found that, like in i-vector systems, bottleneck features greatly improved performance over acoustic features. Echoing similar results in speaker recognition, our experiments showed that augmenting the x-vector DNN training data was a good choice. Finally, although the framework permits direct classification, we found that extracting x-vectors from the DNN and using them as features for a Gaussian classifier produced much better results.

8. References

[1] N. Dehak, P. 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.

[2] Y. Song, B. Jiang, Y. Bao, S. Wei, and L. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

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

[4] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *Signal Processing Letters, IEEE*, vol. 22, no. 10, pp. 1671–1675, 2015.

[5] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černockỳ, "Multilingual bottleneck features for language recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[6] A. McCree, G. Sell, and D. Garcia-Romero, "Augmented data training of joint acoustic/phonotactic dnn i-vectors for nist lre15," in *Proc. Odyssey: Speaker Lang. Recognit. Workshop*, 2016, pp. 204–209.

[7] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5337–5341.

[8] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

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

[10] D. Garcia-Romero and A. McCree, "Stacked long-term tdnn for spoken language recognition," in *INTERSPEECH*, 2016, pp. 3226–3230.

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

[12] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, et al., "The Kaldi speech recognition toolkit," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*, 2011.

[13] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *CoRR*, vol. abs/1410.7455, 2015.

[14] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition.," in *Interspeech*, 2015, pp. 3586–3589.

[15] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 92–97.

[16] A. McCree, D. Snyder, G. Sell, and D. Garcia-Romero, "Language recognition for telephone and video speech: The JHU HLTCOE submission for NIST LRE17," *Submitted to Odyssey*, 2018.

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

[18] A. McCree, "Multiclass discriminative training of i-vector language recognition," in *Proc. Odyssey*, 2014, pp. 166–172.

[19] "NIST 2017 language recognition evaluation plan," `https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf`, 2017.

[20] "The 2015 language recognition i-vector machine learning challenge," `https://www.nist.gov/sites/default/files/documents/itl/iad/mig/lre_ivectorchallenge_rel_v1-1.pdf`, 2015.

[21] T. Ko, V. Peddinti, D. Povey, M. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5220–5224.

[22] D. Snyder, G Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.