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

Optimizing a Speaker Embedding Extractor Through Backend-Driven Regularization

Luciana Ferrer[1], *Mitchell McLaren*[2]

[1]Instituto de Investigación en Ciencias de la Computación (ICC), CONICET-UBA, Argentina
[2]Speech Technology and Research Lab (StarLab), SRI International, USA

lferrer@dc.uba.ar, mitchell.mclaren@sri.com

Abstract

State-of-the-art speaker verification systems use deep neural networks (DNN) to extract highly discriminant representations of the samples, commonly called speaker embeddings. The networks are trained to maximize the cross-entropy between the estimated posteriors and the speaker labels. The pre-activations from one of the last layers in that network are used as embeddings. These sample-level vectors are then used as input to a backend that generates the final scores. The most successful backend for speaker verification to date is the probabilistic linear discriminant analysis (PLDA) backend. The full process consists of a linear discriminant analysis (LDA) projection of the embeddings, followed by mean and length normalization, ending with PLDA for score computation. While this procedure works very well compared to other approaches, it seems to be inherently suboptimal since the embeddings extractor is not directly trained to optimize the performance of the embeddings when using the PLDA backend for scoring. In this work, we propose one way to encourage the DNN to generate embeddings that are optimized for use in the PLDA backend, by adding a secondary objective designed to measure the performance of a such backend within the network. We show modest but consistent gains across several speaker recognition datasets.

Index Terms: speaker recognition, embeddings, probabilistic linear discriminant analysis, deep neural networks, gaussian backend

1. Introduction

Most state-of-the-art speaker verification systems consist of three stages [1, 2]: (1) a frame-level feature extractor, (2) a stage that transforms these features into a sample-level vector representation, and (3) a stage that uses the sample-level vectors to compute a score for the trial. The established approach for the second stage is to use a deep neural network (DNN) trained to classify a large set of speakers available in the training set using a cross-entropy objective. The output of a sample-level layer in that DNN is then used as the vector representation, or *embedding*, for the sample. The final stage then consists of a linear discriminant analysis (LDA) transformation followed by length normalization and, finally, probabilistic LDA (PLDA) to generate likelihood ratios for the trials.

One instance of DNN architecture for the second stage, implemented as the standard recipe for speaker verification in the Kaldi toolkit [3], is the x-vector architecture. The system that uses x-vector embeddings modeled by the backend described above was proven hard to beat by any significant margin during the 2018 NIST speaker recognition evaluation. Many alternative approaches were proposed, some of which integrated the parameterized part of the scoring stage into the DNN, resulting in a third stage that only computes a cosine similarity or dot product between vectors to be used as score. These approaches are based on changing the training objective to ones designed to generate embeddings that are a good metric for speaker verification, without the need for an additional transformation. The objectives that have been explored for speaker verification are angular margin loss, central loss and triplet loss [4, 5, 6, 7, 8].

Notably, to our knowledge, no publication has presented results on the use of these alternative losses when using the x-vector architecture. Furthermore, the alternative approaches all use the layer prior to the output layer to extract the embeddings, since that is the layer that is trained to be a good metric for speaker verification. Yet, it is well-known that using this layer as embeddings in the standard x-vector architecture is suboptimal. The best results are obtained when using the previous layer [9, 10]. Our attempts to train x-vector architectures using angular or central loss have failed to give gains. The use of triplet loss seems to be more promising but much more involved to tune and significantly slower to run.

For these reasons, in this work, we pursue a somewhat opposed approach. Rather than aiming to create speaker embeddings that make the final backend stage unnecessary (or very slim and parameter-less), we design a DNN architecture to create embeddings that are optimized for the PLDA backend[1]. To this end, we propose to add a secondary objective function that resembles as close as possible the backend that will ultimately will be used for scoring. The embeddings are transformed with LDA and mean and length-normalized. Finally, a Gaussian backend is used to compute posteriors for all speakers in training. These posteriors are then used to compute a cross-entropy with the true speaker labels, generating a secondary loss that can be combined with the primary loss to regularize the training of the network. The parameters of LDA and Gaussian backend are reestimated once in a while during training, using a subset of the training data.

We show results using a standard x-vector architecture for the embeddings extractor DNN. We test on several different standard datasets, showing a modest but consistent advantage from the use of the proposed approach.

2. Baseline Architecture

As mentioned in the introduction, despite many interesting and elegant attempts to improve upon it, the x-vector architecture [2] was still the state of the art for speaker verification on the NIST 2018 speaker recognition evaluation (SRE). This architecture is shown in the left branch part of Figure 1. It

[1]Note that, throughout the paper, we use the term "PLDA backend" to refer to the complete pipeline including the LDA projection, mean and length normalization and, finally, PLDA.

http://dx.doi.org/10.21437/Interspeech.2019-1820

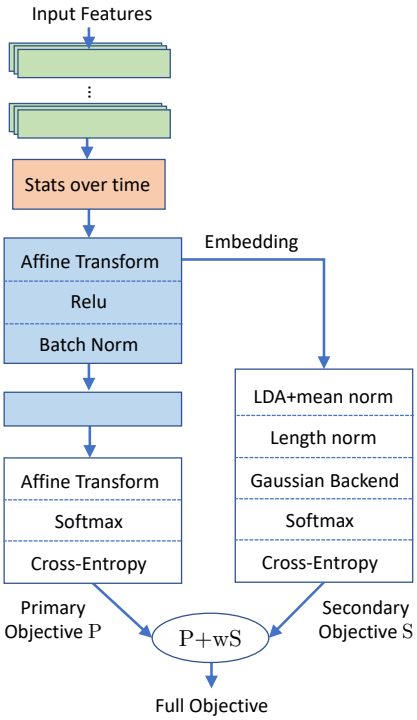

Figure 1: *Architecture of the proposed system. The branch on the left is identical to the x-vector architecture. The branch on the right, computes the proposed secondary objective taking the embeddings as input. The green blocks are frame-level layers, while the light-blue blocks are segment-level layers. The layer from which the embeddings are extracted has been zoomed in to show its components. Note that the other light-blue layer is identical to the previous layer, except that it has not been expanded. The two white output layers have also been zoomed in to show their respective components.*

consist of three time-delay frame-level layers, two more frame-level layers without time delay, a pooling layer that converts the frame-level activations into a single fixed-length vector per sample and, finally, two sample-level layers followed by the output layer. The pooling layer computes the mean and standard deviation of the input activations. All activations in the network are rectified linear units (relu), except in the output layer. Batch normalization is used on the activations from all layers, except the output. The output layer computes an affine transform of its input, and then transforms the outputs using softmax. The resulting activations are the estimated posteriors for each of the training speakers. The objective function to be optimized by the network is the average cross-correlation between these posteriors and the ground truth.

The speaker embeddings are extracted from the layer just after pooling, before the activation function is applied. These embeddings are then processed as follows: (1) linear discriminant analysis is used to project the embeddings into a lower dimensional space with most of the speaker discriminative information, (2) the projected vectors are mean normalized, (3) the resulting vector is length-normalized [11], and (4) this vector is used for scoring with a probabilistic LDA backend (PLDA) [12]. The LDA step assumes that the within-speaker covariances of the embeddings are identical across speakers. The PLDA step assumes the same thing about the LDA-projected mean and length normalized embeddings.

PLDA allows for a proper way to compute likelihood ratios during scoring. Specifically, the simplified PLDA (SPLDA) model commonly used for speaker verification assumes that the input vectors can be modeled as

$$m_i = \mu + V y_{s_i} + z_i, \tag{1}$$

where m_i is the input vector representing a certain sample from speaker s_i; μ is a fixed bias; y_{s_i} is a vector of size R_y, the dimension of the speaker subspace; and z_i is a noise term that is meant to model the variability in m_i not due to the speaker. The model assumes that

$$\begin{aligned} y_{s_i} &\sim N(0, I), \\ z_i &\sim N(0, D^{-1}), \end{aligned}$$

All these latent variables are assumed independent from each other. This model is equivalent to assuming the following distributions [13]

$$m_i|\gamma_{s_i} \sim N(\gamma_{s_i}, D^{-1}), \tag{2}$$

$$\gamma_{s_i} \sim N(\mu, VV^T). \tag{3}$$

where we can see that VV^T is the between-speaker and D^{-1} is the within-speaker covariance matrix of the m_i vectors. The training of PLDA parameters is done using an EM algorithm. Yet, for the simplified version of PLDA described above, a simple smart initialization where the V and D matrices are estimated based on the within- and between-speaker covariances of the training data (see Section 2.1 in [14] for more details) leads to a very good model that does not require EM iterations.

The backend procedure described above works extremely well for speaker embeddings, giving large gains with respect to simple cosine similarity scoring (see, for example, Table 1 in [6]). In fact, every step (LDA, mean and length norm and PLDA) seems essential to get optimal performance. We will not attempt to contend this procedure. Rather, we will acknowledge that it has been the state of the art for speaker verification for a few years and adopt it as-is.

Notably, the embeddings that are used so successfully as input to the PLDA backend were optimized for quite a different process. They were trained to maximize cross-entropy between the speaker label and the posteriors, computed from the embeddings after applying relu activations, batch-normalization, a full feed-forward layer with its own activations and batch-normalization and, finally, a softmax transformation. Motivated by this gap between how embeddings are trained and how they are used, several researchers have proposed different ways to train the networks to make the backend unnecessary. These approaches replace the cross-entropy loss with one that attempts to optimize the embeddings for direct scoring without a backend. As mentioned in the introduction, in this work, we attempt to achieve a somewhat opposed goal: optimize the embeddings for their use with the PLDA backend that we know works so well already.

3. Backend-driven Regularization

In order to optimize the embeddings for their use as input to the PLDA backend, we add a secondary objective during training of the network. This secondary objective is used as a sort of regularization term in the sense that it introduces additional information or assumptions into the model (the assumptions made by the PLDA backend). The secondary objective is, as the primary one, an average cross-entropy, but in this case

the posteriors are estimated by transforming the embeddings using a process that is meant to replicate as closely as possible the one used in the PLDA backend. The right part of Figure 1 depicts the secondary branch needed to compute this new objective. The secondary objective is linearly combined with the primary one using a weight w to scale the secondary objective. This weight can be fixed or changed throughout the training iterations.

In the secondary branch, the embeddings are first transformed using an LDA model, and then mean and length normalized, exactly as in the backend process. The next step in the backend is SPLDA. The SPLDA model could be used to compute the likelihoods and then the posteriors needed to calculate the cross-entropy objective. Yet, since SPLDA is so similar to a Gaussian backend in practice, in our implementation we replace the SPLDA step with a Gaussian backend. The Gaussian backend [15] assumes, as PLDA does, that all speakers share the same covariance matrix. It computes likelihoods for each speaker assuming a Gaussian distribution with a speaker-specific mean and shared covariance. Specifically, the log-likelihood of sample i with respect to the model for speaker j can be estimated with this simple formula:

$$l_{i,j} = m_i^T D\mu_{s_j} - \frac{1}{2}\mu_{s_j}^T D\mu_{s_j} \quad (4)$$

where, as before, m_i is the vector representing sample i (in this case, the embedding projected and mean and length normalized), D is the inverse of the empirical within-speaker covariance, and μ_{s_j} is the empirical mean of the samples for speaker s_j. This is a simple affine transform that can be easily coded within the network for backpropagation.

After the log-likelihoods are computed, a softmax transform is applied to convert them into posteriors. Finally, a cross-entropy objective is computed with respect to the speaker labels, exactly as in the primary branch.

The proposed secondary branch consists of four parameters: the LDA projection, the global mean for mean-normalization, the matrix containing vectors $D\mu_{s_j}$ for all j, and the vector containing $-1/2\mu_{s_j} D\mu_{s_j}$ for all j. These four parameters are estimated using a subset of the training data and updated every N mini-batches.

4. Experimental Setup

In this section we describe the system configuration and datasets used for our experiments.

4.1. Speaker Recognition System

The input features for the network are power-normalized cepstral coefficients (PNCC) [16] which, in our experiments, gave better results than the more standard mel frequency cepstral coefficients (MFCCs). We extract 30 PNCCs with a bandwith going from 100 to 7600 Hz and root compression of 1/15. The features are mean and variance normalized over a rolling window of 3 seconds. Silence frames are discarded using a speech activity detection system based on a DNN with two hidden layers containing 500 and 100 nodes. The DNN was trained using 19-dimensional mel-frequency cepstral coefficients (MFCC) features (excluding C_0), stacked with 31 frames, and mean and variance normalized over a 3 second rolling window. Frames with a log-posterior for the speech class above -1.5 are used for the extraction of embeddings.

System training data included 234K signals from 14,630 speakers. This data was compiled from NIST SRE 2004–2008, NIST SRE 2012, Mixer6, Voxceleb1, and Voxceleb2 (train set) data. Voxceleb1 data had 60 speakers removed that overlapped with Speakers in the Wild (SITW). All waveforms were up- or down-sampled to 16 KHz before further processing. In addition, we down-sampled any data originally of 16 kHz or higher sampling rate (74K files) to 8 kHz before up-sampling back to 16 kHz, keeping two "raw" versions of each of these waveforms. We determined this multi-bandwidth exposure of the same signals allowed the embeddings system to operate well in both 8kHz and 16kHz bandwidths.

Augmentation of data was applied using four categories of degradations as in [1], including music and noise, both at 10 to 25 dB signal-to-noise ratio, compression, and low levels of reverb. We used 412 noises compiled from both freesound.org and the MUSAN corpus. Music degradations were sourced from 645 files from MUSAN and 99 instrumental pieces purchased from Amazon music. For reverberation, examples were collected from 47 real impulse responses available on echothief.com and 400 low-level reverb signals sourced from MUSAN. Compression was applied using 32 different codec-bitrate combinations with open source tools. We augmented the raw training data to produce 2 copies per file per degradation type (randomly selecting the specific degradation and SNR level, when appropriate) such that the data available for training was 9-fold the amount of raw samples. In total, this resulted in 2,778K files for training the speaker embedding DNNs.

The architecture of our embeddings extractor DNN follows the Kaldi recipe [3]. The DNN is implemented in Tensorflow, trained using an Adam optimizer using chunks of speech between 200 and 350 milliseconds. Overall, we extract about 20K chunks of speech from each of the speakers. DNNs were trained over 3 epochs over the data using a mini batch size of 96 examples. Each epoch is divided into 197 iterations. After each iteration, we update the secondary branch parameters using 50 chunks per speaker and evaluate the models on held-out validation data. We used dropout with a probability linearly increasing from 0.0 up to 0.1 at 1.5 epochs then linearly decreasing back to 0.0 at the final iteration. The learning rate started at 0.0005, increasing linearly after 60 iterations reaching 0.03 at the final iteration.

The training data for the PLDA backend was a random of subset of 343K samples of the data used for training the embeddings DNN, excluding the 16 kHz data re-sampled to 8 kHz data. In this case, we use full segments to train the backend rather than chunks - a significant difference from what we do to update the secondary branch parameters during training. Both within the secondary branch and in the backend, the LDA projection transforms the embeddings from 512 dimensions to 200 dimensions. After LDA and mean and length normalization, we score with SPLDA using a full-rank speaker subspace.

The described system configuration for the baseline system was very similar to the one used for the SRI submission to the 2018 NIST speaker recognition evaluation. More details on the system configuration can be found in [17].

4.2. Developments and Evaluation Datasets

We use several different datasets for development and evaluation of the proposed approach. Following is a brief description of each of these sets.

FVC Aus: Interviews and conversational excerpts from over 500 Australian English speakers from the forensic voice comparison dataset [18]. Audio was recorded using close

Table 1: *The development and evaluation datasets with number of speakers and target/impostor (tgt/imp) trial counts.*

Role	Dataset	Speakers	#tgt	#imp
Dev	SITW dev	119	2.6k	335k
	SRE16 dev	20	4.8k	19.3k
	SRE18 dev	35	7.8k	100.3k
Eval	FVC Aus	544	182k	35.8m
	LASRS	333	41k	4.8m
	RATS source	336	66k	21.9m
	SITW eval	180	3.7k	717k
	SRE16 eval	201	37k	1.9m
	SRE18 eval	289	60.7k	2.0m

talking microphones, and cut down to 10-60 seconds of speech content for the evaluation protocol.

RATS src: Telephone calls in five non-English languages from over 300 speakers. The data was the source data (not transmitted) part of the DARPA RATS program [19] for the SID task. Trials are formed exhaustively with the exclusion of cross-language trials and test segments are cut to include 10-60 seconds of speech content. Speakers are enrolled with 1, 2 or 3 segments each consisting of 60 seconds of speech content.

SITW: The Speakers in the Wild (SITW) dataset contains speech samples in English from open-source media [20]. SITW contains naturally occurring noises; reverberation; codec; and channel variability. The enroll and test utterances for SITW vary in duration from 6 to 240 seconds.

SRE16: The NIST Speaker Recognition Evaluation (SRE) 2016 dataset [21] includes variability due to domain/channel and language mismatches. It also has variability in test segment duration which are uniformly distributed between 10s and 60s. The non-English conversational telephone speech (CTS) is recorded over a wide variety of handset types.

SRE18: We focus on the CMN2 subset of the SRE'18 dataset [22]. This subset has similar characteristics to the SRE'16 dataset, with the exception of focusing on different languages, and including speech recorded over VOIP instead of just PSTN calls.

LASRS: The corpus is composed of 100 bilingual speakers from each of three languages, Arabic, Korean and Spanish [23]. Each speaker is asked to perform a series of tasks in English and also in their native language. Each task is recorded using several recording devices and repeated in two separate sessions, recorded in different days.

SITW, SRE16 and SRE18 have a well-defined development sets. We use those 3 sets to tune the parameters of our models. The rest of the sets are used for evaluation of the final systems.

5. Results

We show results in terms of minimum detection cost function (DCF), using the SRE 2010 [24] costs and priors (a probability of target trial of 0.01, cost of misses of 10 and cost of false alarms of 1) and equal error rate (EER). The schedule for the secondary objective weight w is lightly tuned using the average relative gain with respect to the baseline on the three development sets. The chosen schedule is to start with a weight of 0, followed by a linear increase to 0.2 between iterations 40 and 80, keeping it fixed at 0.2 until the final iteration.

Figure 2 shows the values of the two objectives for the two systems as a function of the iteration number. The primary objective is not noticeably affected by the regularization term. On the other hand, the secondary objective is significantly improved after the weight for this branch is increased from

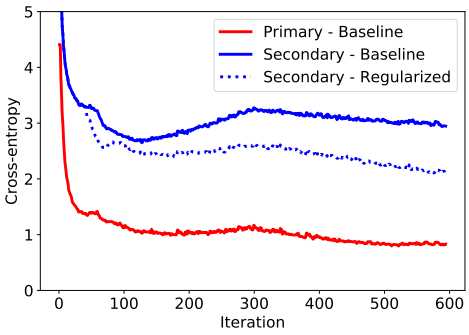

Figure 2: *Average cross-entropy on the validation set after each iteration. The plots corresponds to 3 epochs through the data. The primary curve for the regularized system is not shown since it overlaps almost perfectly with that of the baseline system.*

Table 2: *Development and evaluation results for the baseline (Base), the regularized (Reg) systems, and the relative gain. The bolded gains indicate the cases where the Reg system's performance is outside a 90% confidence interval of the Base system computed using a bootstrap procedure [25].*

	EER			DCF		
Dataset	Base	Reg	Rel	Base	Reg	Rel
FVC Aus	0.5	0.6	-7.4	0.027	0.031	-14.8
LASRS	6.1	5.8	4.7	0.325	0.303	**6.8**
RATS src	4.1	4.0	2.9	0.173	0.161	6.9
SITW dev	2.3	2.4	-1.7	0.116	0.103	11.2
SITW eval	2.0	2.1	-2.5	0.107	0.103	3.7
SRE16 dev	13.1	11.7	10.8	0.581	0.490	15.7
SRE16 eval	11.7	10.1	**13.4**	0.601	0.502	**16.5**
SRE18 dev	9.2	9.0	1.3	0.354	0.333	5.9
SRE18 eval	9.7	8.9	8.3	0.393	0.356	**9.4**

0 to 0.2. Note that the slight increase in monitors around iteration 300 is due to the increased dropout probability peaking at that iteration. Finally, table 2 shows the final results over all development and evaluation datasets. We can see modest but consistent gains across most datasets, with the only exception being the FVC Aus dataset, which is exceptionally clean compared to most of our training data.

The results shown here correspond to one particular system configuration. We have also tested a few other configurations (using MFCCs instead of PNCCs) and using different amounts of training data (less speakers or fewer chunks per speaker). In all cases, results show a similar trend of modest but consistent gains across datasets.

6. Conclusions

We proposed a new approach to regularize the training of speaker embeddings extractor DNNs. The approach is aimed at making the resulting embeddings better fit for modeling with the standard PLDA backend. The regularization is achieved by adding a secondary cross-entropy objective computed with respect to a new set of posteriors. These new posteriors are obtained by transforming the embeddings with operations resembling the PLDA backend. We show results on a wide range of conditions and show that the proposed approach gives gains on most datasets. This paper is meant as a proof-of-concept. Many dimensions of the approach still need to be explore and we believe there is potential for further improvements.

7. References

[1] M. McLaren, D. Castan, M. Nandwana, L. Ferrer, and E. Yilmaz, "How to train your speaker embeddings extractor," in *Proc. of Speaker Odyssey*, Les Sables d'Olonne, France, June 2018.

[2] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, Calgary, Canada, April 2018.

[3] "NIST SRE 2016 xvector recipe."

[4] Z. Huang, S. Wang, and K. Yu, "Angular softmax for short-duration text-independent speaker verification," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018.

[5] 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-18*, Les Sables d'Olonne, France, June 2018.

[6] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "SpeakerGAN: Recognizing speakers in new languages with generative adversarial networks," *NIPS workshop IRASL*, 2018.

[7] S. Novoselov, V. Shchemelinin, A. Shulipa, A. Kozlov, and I. Kremnev, "Triplet loss based cosine similarity metric learning for text-independent speaker recognition," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018.

[8] J. Rohdin, A. Silnova, M. Diez, O. Plchot, P. Matejka, and L. Burget, "End-to-end DNN based speaker recognition inspired by i-vector and PLDA," in *Proc. ICASSP*, Calgary, Canada, April 2018.

[9] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, Stockholm, August 2017.

[10] M. Nandwana, M. McLaren, D. Castan, J. van Hout, and A. Lawson, "Analysis of complementary information sources in the speaker embeddings framework," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018.

[11] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, Florence, Italy, Aug. 2011.

[12] S. Prince, "Probabilistic linear discriminant analysis for inferences about identity," in *Proceedings of the International Conference on Computer Vision*, 2007.

[13] A. Sizov, K. A. Lee, and T. Kinnunen, "Unifying probabilistic linear discriminant analysis variants in biometric authentication," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2014, pp. 464–475.

[14] L. Ferrer and M. McLaren, "Joint PLDA for simultaneous modeling of two factors," *Journal of Machine Learning Research*, vol. 20, January 2019.

[15] M. F. BenZeghiba, J.-L. Gauvain, and L. Lamel, "Language score calibration using adapted gaussian back-end," in *Proc. Interspeech*, Brighton, Sep. 2009.

[16] C. Kim and R. Stern, "Power-normalized cepstral coefficients (PNCC) for robust speech recognition," in *Proc. ICASSP*, Kyoto, Mar. 2012.

[17] M. McLaren, L. Ferrer, D. Castan, M. K. Nandwana, R. Travadi, P. Georgiou, and S. Narayanan, "The SRI-CON-USC NIST 2018 SRE system description," in *Proceedings of SRE18 Workshop*, Athens, Greece, December 2018.

[18] G. Morrison, C. Zhang, E. Enzinger, F. Ochoa, D. Bleach, M. Johnson, B. Folkes, S. De Souza, N. Cummins, and D. Chow, "Forensic database of voice recordings of 500+ australian english speakers," 2015. [Online]. Available: http://databases. forensic-voice-comparison. net

[19] K. Walker and S. Strassel, "The RATS radio traffic collection system," in *Proc. Odyssey-12*, Singapore, Jun. 2012.

[20] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *Proc. Interspeech*, San Francisco, September 2016.

[21] "NIST 2016 speaker recognition evaluation plan." [Online]. Available: https://www.nist.gov/sites/default/files/documents/itl/iad/mig/SRE16_Eval_Plan_V1-0.pdf

[22] "NIST 2018 speaker recognition evaluation plan." [Online]. Available: https://www.nist.gov/document/ sre18evalplan2018-05-31v6.pdf

[23] S. D. Beck, R. Schwartz, and H. Nakasone, "A bilingual multi-modal voice corpus for language and speaker recognition (LASR) services," in *Proc. Odyssey-04*, Toledo, Spain, May 2004.

[24] "NIST SRE10 evaluation plan," http://www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf.

[25] N. Poh and S. Bengio, "Estimating the confidence interval of expected performance curve in biometric authentication using joint bootstrap," in *Proc. ICASSP*, Honolulu, Apr. 2007.