BUT-PT System Description for NIST LRE 2017

Pavel Matějka[1,4], Oldřich Plchot[1], Ondřej Novotný[1], Sandro Cumani[2], Alicia Lozano-Diez[1,3], Josef Slavicek[4,1], Mireia Diez[1], František Grézl[1], Ondřej Glembek[1], Kamsali Veera Mounika[1], Anna Silnova[1], Lukáš Burget[1], Lucas Ondel[1], Santosh Kesiraju[1], Johan Rohdin[1]

(1) BUT, Speech@FIT, Czech Republic
(2) Politecnico di Torino, Italy
(3) Audias-UAM, Universidad Autonoma de Madrid, Spain
(4) Phonexia, Czech Republic

{matejkap|iplchot}@vut.cz[1], sandro.cumani@polito.it[2]
alicia.lozano@uam.es[3], josef.slavicek@phonexia.com[4]

Abstract

1. Introduction

Our submission is a collaborative effort of BUT, Politecnico di Torino, Universidad Autonoma de Madrid and Phonexia. The main body of work was conducted during end of September and beginning of October 2017 when the whole team met in Brno and all members were closely working together with common datasets.

All of our individual systems rely on the bottleneck features[1, 2] (BNF) as frontends. Most of our systems are still based on i-vectors and subsequent generative classifier. We also complement the classical i-vector based systems with a system based on embeddings obtained from discriminatively trained end-to-end LRE system. Finally, the primary submission is a fusion of four systems where we utilize two different BNF extractors, non-linear processing of i-vectors and embeddings obtained from the discriminative system.

2. Data

We have utilized all data supplied by NIST (training and dev) for primary (fixed) condition. All data were downsampled to 8kHz.

The annotated Switchboard and Fisher I and II databases were used to train the bottleneck NN. These databases were provided for all participants to allow the use of techniques requiring annotated speech corpora for system development.

The LRE17 training data were used to train the i-vector system (UBM, i-vector extractor). LRE17 development data were split into two parts. First part was used added to the classifier training data and the second part was used as a held-out set to monitor the performance during development and for calibration and fusion during development. For the final calibration and fusion before submission, we have re-calibrated and fused all systems on the full LRE17 dev set. We denote LRE17 training data as *Fixed* set later in this document.

For the open condition and only for the classifier training data, we tried to design an alternative dataset with an emphasis on large channel variability and amount of data. We used data from previous evaluations and other publicly available sources. We denote this as *Open* set later in this document.

2.1. Primary Data

We used a mix of LRE17 train and dev for training our classifiers.

As the training data provided by NIST contained many rather long segments, we have decided to prepare two versions of the training data. First, we used the lre17 training data as is without any modification (we denote this version as ***full***, and second we cut all of the files containing more than 40s of speech into cuts ranging from 2.5 to 40s (we denote this version as ***cuts***. We did not include the original long segment in the resulting training list then. We chose this approach to simplify the processing of data in subsequent classifiers.

In the lre17 dev. we split segments that contain more than 40s of speech into multiple short cuts ranging from 2.5 to 40 seconds of speech. We also kept the original long segments that we cut. This way, we obtained 6090 segments, while the original NIST dev set contains 3660 segments. We decided to put $2/3$ of this data into the training set and leave the remaining $1/3$ as a held out test data. We did not attempt to detect overlapping segments (nested cuts) between the two splits of lre17 dev data, but we ensured that the short cuts created by us do not overlap w.r.t. their original long segments.

2.2. Data for Open condition

We designed an independent dataset for the open-data condition, where we concentrated on obtaining a large diversity in channels and a large amount of training data. We have reused data from previous NIST evaluations and all of the LRE17 train-

This work was partly supported by Czech Ministry of Interior project No. VI20152020025 DRAPAK, by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602, by Google Faculty Research Award and by Grant Agency of the Czech Republic project No. GJ17-23870Y. Mireia Diez was supported by European Union's Horizon 2020 Marie Sklodowska-Curie grant agreement No. 748097 and Johan Rohdin was supported by European Union's Horizon 2020 Marie Sklodowska-Curie / South Moravian Region grant agreement No. 665860.

ing data. We did not attempt to filter the LRE17 training data to remove the overlapping segments with the remaining training data. We have also added part of KALAKA-3 database [3] (British English, European Spanish) and human annotated part of Al Jazeera Dialectal Speech Corpus for the Arabic dialects [4]. As in the training data for fixed condition, we added the two thirds of lre17 dev data.

The training data for the open condition reached the size of 67779 segments (2346 hours of speech). Here, we did not produce any additional short cuts from the long segments.

Table 1: *Sources of data used for open training data condition and their statistics. Rows are sorted according to amounts of detected speech.*

Database	#files	hours
OGI 22 languages	1987	3
unknown broadcast news	64	3.9
LRE17 dev MLS14	1854	6.3
OGI multilingual	1640	8.8
KALAKA-3	409	9.9
Radio Free Europe	478	11.2
Foreign Accented English	4239	13.6
HKUST Mandarin	276	17.2
Al Jazeera Dialectal Speech Corpus	4423	21.8
LRE17 dev VAST	2242	22.1
SpeechDat-East	6202	30.6
NIST SREs	3722	88.6
Callfriend	838	144
Levantine Arabic and Iraqi CTS[1]	3606	168
Previous NIST LREs	3015	256
Fisher English, Arabic	5804	337
VOA broadcasts	10775	366.7
LRE17 train	16205	837.3
Total	67779	2346

2.3. Training data for Bottleneck features

2.3.1. Fisher English

Fisher English database Part 1 and 2 was used for training as no other data was allowed. The final training data was composed of 1800 hours of clean Fisher data augmented with another 3 copies of artificially corrupted Fisher data. We used fant tool [5] to mix reverberated speech and reverberated noise with given SNR with original clean audio file.

We generated artificial room impulse responses (IR) using Room Impulse Response Generator tool from E. Habets [2].

IRs were generated for rooms where each dimension was limited to the range of 2–22 meters and other parameters in the tool were set randomly.

Noises were added at SNRs ranging from 0dB to 45dB. The noises are downloaded from Freesound.org library with these types:

- real fan stationary noises - fan, AC, hvac, street, ventilation - 115 samples from Freesound.org
- real background transient noises - dishes, motor, workshop, doors, city, keyboard, library, office. The character is mainly transient, with some minor portion of stationary noises - 60 samples from freesound.org
- babbling noises: each created by merging speech from 100 random speakers from Fisher database using speech activity detector - 25 samples
- artificially generated noises - various spectral modifications of white noise + 50 and 100 Hz hum - 7 samples

2.3.2. Babel Multilingual

We used the IARPA Babel Program data[3]. This data simulates a case of what one could collect in limited time from a completely new language. All training data from the full language packs from the 17 languages (Assamese, Bengali, Cantonese, Cebuano, Haitian, Kazakh, Kurmanji, Lao, Lithuanian, Pashto, Tagalog, Tamil, Telugu, TokPisin, Turkish, Vietnamese, Zulu) were used. There is 1070 hours of speech data in total with approximately uniform distribution of speech duration per each language. More details about the characteristics of the individual languages can be found in [6].

3. Voice Activity Detection

Our VAD consists of two carefully designed parts: a neural network (NN) which produces per-frame scores, and a post-processing stage which builds the segments based on the scores.

The NN was trained on the Fisher English. The input dimension is 288, while there are 2 hidden layers, each of 400 sigmoid neurons, and the final softmax layer has 2 outputs, corresponding to the classes: speech, non-speech. The NN has 277k parameters.

The input features for the NN consist of 15 log-Mel filterbank outputs and 3 Kaldi-pitch features [7]. We apply per-speaker mean normalization estimated on the whole unsegmented recordings. Then we apply frame splicing with 31 frame-long context, where the temporal trajectory of each feature is scaled by a Hamming window and reduced to 16 dimensions by Discrete Cosine Transform. The final 288-dimensional features are globally mean and variance normalized on the NN input.

In the post-processing, we bypass the NN output softmax function (allowing us to interpret the outputs as log-likelihoods), then we convert the two outputs to logit-posteriors, and then we smooth the score by averaging over consecutive 31 frames. In the final step, the speech segments were extracted by thresholding the posterior at the value of -0.5.

4. Feature extraction

4.1. Stacked Bottleneck Features (SBN)

A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through a NN and reading off the vector of values at the bottleneck layer. We have used a cascade of two such NNs for our experiments. The output of the first network is stacked in time, defining context-dependent input features for the second NN, hence the term Stacked Bottleneck Features (SBN). The NN input features are 24 log Mel-scale filter bank outputs augmented with 2 fundamental frequency based features based on [8]. In summary, 24 log filter bank outputs and 2 fundamental frequency features form 26-dimensional feature vectors.

Mean subtraction is applied at the utterance level. Hamming window followed by DCT consisting of 0th to 5th base

[1] LDC2007S01, LDC2006S45, LDC2005S14, LDC2005S07

[2] http://www.audiolabs-erlangen.de/content/05-fau/professor/00-habets/05-software/01-rir-generator/rir_generator.pdf

[3] Collected by Appen, `http://www.appenbutlerhill.com`

are applied on the time trajectory of each parameter resulting in $(24+2)\times 6 = 156$ coefficients on the first stage NN input.

The dimensionality of the bottleneck layer was set to 30 or 80. The dimensionality of the other hidden layers was set to 1500. The bottleneck outputs from the first NN are sampled at times $t-10$, $t-5$, t, $t+5$ and $t+10$, where t is the index of the current frame. The resulting 150 or 400-dimensional features are inputs to the second stage NN with the same topology as first stage. The 30 or 80 bottleneck outputs from the second NN (referred as SBN) are the final features.

The basic idea in multilingual training of BN is to train NN on more than one language [9] so that the final bottleneck features cover richer acoustic space than when trained for one language only. We used block softmax to divide the output layer into parts according to individual languages [1, 2]. During the training, only the part of the output layer corresponding to the language of the actual target is activated.

We used several type of SBN features.

1. **FSH-30** - trained on Fisher English corpus. First bottleneck is 80 dimensional and the second bottleneck is 30 dimensional. There are 3 hidden layers before BN layer in both NN. The output layer has 9824 outputs - triphones. During training the BN layer is connected directly to the output layer.
2. **FSH-80** - trained on Fisher English corpus. First and second bottleneck layers are 80 dimensional. There are 2 hidden layers before BN layer in both NN. The output layer has 9824 outputs - triphones. There is one hidden layer between bottleneck and output layer during training.
3. **BabelML17-80** - trained on 17 languages from BABEL project. First and second bottleneck layers are 80 dimensional. The output layer has 15558 outputs - tied triphones per each language. This NN is trained in multilingual fashion where we used block softmax as the output layer [2] . This features are only for OPEN condition. The NN topology is same as for FSH-80.

4.2. NN embeddings

For one of the systems submitted, we use a DNN embedding based architecture, which consists of a sequence summarizing DNN trained to learn an utterance (or segment) level representation from the frame-by-frame input features.

The structure of the system was inspired by the embedding DNN system presented in [10] for speaker verification. We can be split the DNN into two parts separated by the pooling (summarizing) layer. The first part of the DNN, up to the pooling layer, works in a frame basis, and consists of two BLSTM layers followed by a fully connected layer, with 256 hidden units (or cells) each. These recurrent layers based on BLSTM have proven their ability to deal with temporal information, especially over short utterances for LID, without stacking of frames in the input features but taking into account the information learnt from previous (and following) frames in the same sequence. Then, mean and standard deviation stats over the temporal dimension of the sequence are pooled together in the sequence summarizing layer. Two fully connected layers follow, which correspond to the embeddings used as sequence (or utterance) level representation for the LID system (concatenation of both is used for the embedding system submitted). Finally, a softmax output layer is used to discriminate among the 14 target languages. This way, from the summarizing layer up to the output layer, the DNN acts over sequences instead of frames.

Sigmoid activation was used as non linear function for all hidden units. Categorical cross-entropy loss function was optimized via Adam optimizer, using 200 samples per minibatch. Dropout of 0.3 was used in recurrent units and gates of the BLSTM layers, and sequences of 3 seconds were considered during training. However, in order to extract embeddings for each utterance, we do not constraint the length of the segment to 3 seconds, but forward the whole segment to obtain one embedding per utterance.

The network was fed with the FSH-30 SBN features described in the previous section, and trained using a balanced dataset from the training list which contained up to 15 hours per language, prioritizing the development segments included as training. The model selection was performed according to validation accuracy in the 1/3 held out test data (with no more than 200 epochs allowed for training). The Gaussian Linear Classifier (GLC) described in section 5.1 was used as a backend, trained on the full training list.

5. Classifiers

5.1. Gaussian linear classifier / Multi-Gaussian classifier

Generative modeling of i-vector point–estimates for language recognition has proven to be an effective alternative to discriminative classifiers based on Logistic Regression or Support Vector Machines. In [11], we have proposed a simple linear classifier based on Gaussian distributions which provides accuracies similar to those of linear discriminative approaches. The model assumes that, for each language, the corresponding i-vector point–estimates $\boldsymbol{\mu}_i$ are generated according to:

$$\boldsymbol{\mu}_i \sim \mathcal{N}\left(\mathbf{m}_\ell, \boldsymbol{\Lambda}^{-1}\right) \ , \tag{1}$$

where $\mathbf{m}_\ell$ is a language–dependent mean vector and $\boldsymbol{\Lambda}^{-1}$ is a covariance matrix, shared among all language distributions. The model parameters can be easily obtained by Maximum–Likelihood estimation. The class–conditional log–likelihood for $\boldsymbol{\mu}_i$ given language ℓ can be computed as:

$$\log P(\boldsymbol{\mu}_i|\ell) = \frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}(\boldsymbol{\mu}_i - \mathbf{m}_\ell)^T\boldsymbol{\Lambda}(\boldsymbol{\mu}_i - \mathbf{m}_\ell) + k \ , \tag{2}$$

where k is a data–independent constant. We denote this classifier as GLC.

Since development and evaluation data comprise different, possibly mismatched, data sources, and we also observed a relevant mismatch between development data and previous LRE data, we also propose a modified Gaussian classifier, named Multi-Gaussian Classifier (MGC), able to better model these different sources. The MGC classifier assumes that i-vectors of each language-source combination are generated by a different Gaussian distribution according to

$$\boldsymbol{\mu}_i \sim \mathcal{N}\left(\mathbf{m}_{\ell,s}, \boldsymbol{\Lambda}^{-1}\right) \ , \tag{3}$$

where s denotes the source. For this evaluation, we considered three sources, namely, VAST, MSL14 and previous LRE data. At test time a language score is computed from a GMM whose components are the Gaussian distributions associated to the target language, assuming uniform weights over the data sources.

5.2. NonLinearTransform

Following the success of Non–Linear PLDA (NL–PLDA) [12] for speaker verification, we propose to apply the technique for

language recognition tasks. To this extent, we trained a NL–PLDA model using the formulations given in [12], assuming that the classes are languages rather than speakers. Although PLDA models (including NL–PLDA) can be directly used to compute language recognition scores, due to lack of time we did not directly use NL–PLDA for scoring, but rather to simply compute a transformation of i–vectors. It is worth noting that, in contrast with our previously proposed i-vector Gaussianization approach [13], the NL–PLDA transformation is aware of language labels. Finally, GLC and MGC classifiers were trained over the transformed i-vectors.

5.3. Neural Network classifier

For one of submits, we used a small Neural network (NN) as one of fused systems. The NN takes i-vectors as input and generates posteriors of language classes at the output softmax layer. The NN has a single hidden layer which works as a Variational information bottleneck [14] of size $K = 64$. When trained on i-vector sets with cuts, the NN shows slightly better performance than GLC and these two models are complementary – they fuse well. We found no way to train successfully the NN on data without cuts (because of over-fitting). Source code of the NN is available at https://github.com/JosefSlavicek/LRE-VIB

6. Calibration and fusion

After the first stage, where produced a vector of scores for each segment, we continued with other two stages that do *pre-calibration* and *fusion* in score-space and are both trained on the dev subset (NIST LRE17 dev) for both the *Fixed* or *Open* condition. During development, we were calibrating and fusing on the $2/3$ of LRE17 dev that we also added into the training data, for the final submission, we calibrated and fused on the whole LRE17 dev including our short cuts. Both stages are implemented by multiclass logistic regression [15].

Our logistic regression solutions to calibration and fusion were simple, to avoid over-training. For *pre-calibration*, an individual system has a trainable scale factor and an offset vector. In *fusion*, every system gets a single trainable scale factor, while every language gets a trainable score offset. The parameters are trained via optimizing prior-weighted multiclass cross-entropy. We used an *uniform prior* (flat) over all 14 languages for both *pre-calibration*, and fusion.

Later on in this document we report actual equalized cost obtained by the means of NIST scoring tool on the held out part of our development set.

6.1. Cluster-dependent system fusion

Some of our systems use cluster dependent subsystems [16] fused into one system by means of a simple average of their scores, which simplifies the development and provides sufficient robustness. Such system is then denoted by **CD**.

Basically it is a simple fusion of 5 (as there are 5 language clusters) i-vector based systems, where the individual UBMs are trained only on data belonging only to the particular cluster (Arabic, Chinese, English, Iberian, Slavic). The training data for T-matrix were however common for all individual cluster-dependent subsystems.

7. Submitted systems and Results

Most of our systems were based on the 30 dimensional bottleneck features trained on Fisher English (FSH-30). We varied the topology and size of GMM and ivector extractor, we added delta coefficients at the input, we experimented with postprocessing of the ivectors and used different classifiers. The Table 2 summarizes the components of separate systems together with results and also the fusion and submitted systems. All ivector based system were trained on the full list only on LRE17 training data. Only systems 5 and 6 were trained also on Fisher and Switchboard data.

8. Processing Speed

Our code is a development code and was not optimized for speed. The measurement is done for one channel of the file *fe_03_04835* from Fisher English 1 corpus with length 597 seconds and 216 seconds of detected speech. The most time consuming part of the system is bottleneck extraction. The processing time of bottleneck feature extraction including loading all models is 50 and 70 seconds for 80 and 30 dimensional bottleneck extractors respectively. GMM and ivector extraction for big model (4096G/800ivec) including loading models is 15sec. Evaluating NN/GLC/MGC is at most 1sec. The processing time for extracting NN embeddings from bottleneck features to embeddings is 39 seconds from 30 dimensional bottleneck features FSH-30. The VAD is negligible to the processing time of the whole system so we compute the realtime factors to the detected speech. The results are in the Table 4. The single best system is 3.27 times faster than RT and the primary fusion is 0.79 times faster than RT.

Processing was run on Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz and each process with less then 8G of RAM.

9. Conclusion

We have built over 30 systems for this evaluation with the main focus to build a single best system. We experimented with denoising NN, automatic discovery units, different flavors of phonotactic systems, different backends, different sizes of ivector systems, different BN features, NN embeddings and frame level language classifiers.

The evaluation plan stated "Teams are encouraged to report whether and how having access to the development set helped improve the performance". The development data helped mainly in the final classifier and also helped in the decision process which techniques to use and which to fuse because our test set consisted of this data.

10. References

[1] K. Vesely, M. Karafiat, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, Dec 2012, pp. 336–341.

[2] Radek Fer, Pavel Matejka, Frantisek Grezl, Oldrich Plchot, Karel Vesely, and Jan Honza Cernocky, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech & Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[3] Luis Javier Rodríguez-Fuentes, Mikel Penagarikano, Amparo Varona, Mireia Diez, and Germán Bordel, "Kalaka-3: a database for the assessment of spoken language recognition technology on youtube audios," *Language Resources and Evaluation*, pp. 1–23, 2015.

[4] Samantha Wray and Ahmed Ali, "Crowdsource a little to label a lot: labeling a speech corpus of dialectal Arabic,"

Table 2: *Submitted systems and its description with results on our dev set.*

number	features	model	classifier	list	$\mathrm{actC_{avg}}$
1	FSH-30+Delta	GMM=4096 ivec=800 CD	MGC	full	0.1457
2	FSH-30+Delta	GMM=4096 ivec=800	NL-PLDA + MGC	cuts,full	0.1734
3	FSH-30+Delta	GMM=4096 ivec=800	MGC	full	0.1684
4	FSH-30	NN embeddings	GLC	full	0.1955
5	FSH-80	GMM=4096 ivec=800(+FSH+SWB)	NN	cuts	0.1990
6	FSH-80	GMM=4096 ivec=800(+FSH+SWB)	MGC	full	0.1805
7	Babel17-80	GMM=4096 ivec=800	MGC	open	0.1854

condition	systems	submission	note	$\mathrm{actC_{avg}}$
fixed	1 2 4 6	primary	Fusion	0.1296
	3 4 6	contrastive 1	Fusion	0.1332
	1	contrastive 2	Single best system CD	0.1457
	3	contrastive 3	Single system	0.1684
	2 3	contrastive 4	Single system - two classifiers	0.1582
open	2 4 6 7	primary	Fusion	0.1314
	1 5 7	contrastive 1	Fusion	0.1287
	7	contrastive 2	Single system	0.1854

Table 3: *Submitted systems and its description with results on evaluation data*

condition	systems	submission	note	$\mathrm{minC_{avg}}$		$\mathrm{actC_{avg}}$		EER	
				notEqual	Equal	notEqual	Equal	notEqual	Equal
fixed	1 2 4 6	primary	Fusion	0.1687	0.1635	0.1710	0.1660	4.295	4.151
	3 4 6	contrastive 1	Fusion	0.1775	0.1743	0.1786	0.1754	4.475	4.421
	1	contrastive 2	Single best system CD	0.1991	0.1832	0.2013	0.1868	5.158	4.807
	3	contrastive 3	Single system	0.2203	0.2011	0.2228	0.2039	5.759	5.256
	2 3	contrastive 4	Single system - 2 classf	0.2065	0.1888	0.2081	0.1913	5.321	4.893
open	2 4 6 7	primary	Fusion	0.1600	0.1599	0.1621	0.1612	4.066	4.061
	1 5 7	contrastive 1	Fusion	0.1689	0.1668	0.1704	0.1692	4.267	4.308
	7	contrastive 2	Single system	0.2071	0.2218	0.2096	0.2263	5.536	6.035

Table 4: *Real time processing speed of the systems not optimized for speed on the 597 seconds long audio with 216 seconds of detected speech. Real time factor is computed only on detected speech (higher is faster).*

condition	submission	RT
fixed	primary	0.79xRT
	contrastive 1	1.52xRT
	contrastive 2	1.44xRT
	contrastive 3	2.51xRT
	contrastive 4	2.48xRT
open	primary	0.83xRT
	contrastive 1	0.77xRT
	contrastive 2	3.27xRT

in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*. 2015, pp. 2824–2828, ISCA.

[5] H. Gnter Hirsch and Harald Finster, "The simulation of realistic acoustic input scenarios for speech recognition systems," in *Proceedings of Interspeech 2005*, 2005.

[6] M. Harper, "The BABEL program and low resource speech technology," in *ASRU 2013*, Dec 2013.

[7] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 2494–2498.

[8] David Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech Coding and Synthesis*, W. B. Kleijn and K. Paliwal, Eds., New York, 1995, Elseviever.

[9] Tanja Schultz and Alex Waibel, "Language independent and language adaptive acoustic modeling for speech recognition," *SPEECH COMMUNICATION*, vol. 35, pp. 31–51, 2001.

[10] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proceedings of Interspeech 2017*, 2017.

[11] David González Martínez, Oldřich Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Proceedings of Interspeech 2011*, 2011, pp. 861–864.

[12] S. Cumani and P. Laface, "Joint estimation of plda and nonlinear transformations of speaker vectors," *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, vol. 25, no. 10, pp. 1890–1900, Oct 2017.

[13] S. Cumani and P. Laface, "Nonlinear i-vector transformations for plda-based speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 908–919, April 2017.

[14] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy, "Deep variational information bottleneck," *CoRR*, vol. abs/1612.00410, 2016.

[15] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2007.

[16] Pavel Matějka, Le Zhang, Tim Ng, Harish Sri Mallidi, Ondřej Glembek, Jeff Ma, and Bing Zhang, "Neural network bottleneck features for language identification," in *Proceedings of Odyssey 2014*. 2014, vol. 2014, pp. 299–304, International Speech Communication Association.