A NOVEL LEARNABLE DICTIONARY ENCODING LAYER FOR END-TO-END LANGUAGE IDENTIFICATION

Weicheng Cai[1,3]*, Zexin Cai*[1]*, Xiang Zhang*[3]*, Xiaoqi Wang*[4] *and Ming Li*[1,2*]

[1]School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
[2]Data Science Research Center, Duke Kunshan University, Kunshan, China
[3]Tencent Inc., Beijing, China
[4]Jiangsu Jinling Science and Technology Group Limited

ml442@duke.edu

ABSTRACT

A novel learnable dictionary encoding layer is proposed in this paper for end-to-end language identification. It is inline with the conventional GMM i-vector approach both theoretically and practically. We imitate the mechanism of traditional GMM training and Supervector encoding procedure on the top of CNN. The proposed layer can accumulate high-order statistics from variable-length input sequence and generate an utterance level fixed-dimensional vector representation. Unlike the conventional methods, our new approach provides an end-to-end learning framework, where the inherent dictionary are learned directly from the loss function. The dictionaries and the encoding representation for the classifier are learned jointly. The representation is orderless and therefore appropriate for language identification. We conducted a preliminary experiment on NIST LRE07 closed-set task, and the results reveal that our proposed dictionary encoding layer achieves significant error reduction comparing with the simple average pooling.

Index Terms— language identification (LID), end-to-end, dictionary encoding layer, GMM Supervector, variable length

1. INTRODUCTION

Language identification (LID) can be defined as a utterance level paralinguistic speech attribute classification task, in compared with automatic speech recognition, which is a "sequence-to-sequence" tagging task. There is no constraint on the lexicon words thus the training utterances and testing segments may have completely different content [1]. The goal, therefore, might be to find a robust and duration-invariant utterance level vector representation describing the distributions of local features.

In recent decades, in order to get the utterance level vector representation, dictionary learning procedure is widely used. A dictionary, which contains several temporal orderless center components (or units, words), can encode the variable-length input sequence into a single utterance level vector representation. Vector quantization (VQ) model, is one of the simplest text-independent dictionary models [1]. It was introduced to speaker recognition in the 1980s [2]. The average quantization distortion is aggregated from the frame-level residual towards to the K-means clustered codebook. The Gaussian Mixture Model (GMM) can be considered as an extension of the VQ model, in which the posterior assignments are soft [3, 4]. Once we have a trained GMM, we can simply average the frame-level likelihood to generate the encoded utterance level likelihood score. Besides, we can move forward to accumulate the 0^{th} and 1^{st} order Baum-Welch statistics, and encode them into a high dimensional GMM Supervector [5]. VQ codebook and GMM are unsupervised and there is no exact physical meaning on its components. Another way to learn the dictionary is through phonetically-aware supervised training [6, 7]. In this method, a deep neural network (DNN) based acoustic model is trained. Each component in the dictionary represents a phoneme (or senone) physically, and the statistics is accumulated through senone posteriors, as is done in recently popular DNN i-vector approach [8, 9, 10]. A phonotactic tokenizer can be considered as a dictionary doing hard assignments with top-1 score [1]. Once we have a trained tokenizer, usually a bag-of-words (BoW) or N-gram model is used to form the encoded representation [11, 12].

These existing approaches have the advantage of accepting variable-length input and the encoded representation is in utterance level. However, when we move forward to modern the end-to-end learning pipeline, e.g. the neural network, especially for the fully-connected (FC) network, it usually requires a fixed-length input. In order to feed into the network, as is done in [13, 14, 15, 16], the original input feature sequence has to be resized or cropped into multiple small fixed-size segments in frame level. This might be theoretically and practically not ideal for recognizing language, speaker or other paralinguistic information due to the need of a time-invariant representation from the entire arbitrary and potentially long duration length.

To deal with this issue, recently, in both [17, 18], similar temporal average pooling(TAP) layer is adopted in their neural network architectures. With the merit of TAP layer, the neural network have the ability to train input segments with random duration. In testing stage, the whole speech segments with arbitrary duration can be fed into the neural network.

Compared with the simple TAP, the conventional dictionary learning have the ability to learn a finer global histogram to demonstrate the feature distribution better, and it can accumulate high order statistics. In computer vision community, especially in image scene classification, texture recognition, action recognition tasks, modern convolutional neural network (CNN) usually bound to the conventional dictionary learning methods together to get a better encoding representation. For example, NetVLAD [19], NetFV [20], Bilinear

*This research was funded in part by the National Natural Science Foundation of China (61401524,61773413), Natural Science Foundation of Guangzhou City (201707010363), Science and Technology Development Foundation of Guangdong Province (2017B090901045), National Key Research and Development Program (2016YFC0103905).

Pooling [21], and Deep TEN [22] are proposed and achieved great success.

This motivates us to implement the conventional GMM and Supervector mechanism into our end-to-end LID neural network. As the major contribution of this paper, we introduce a novel learnable dictionary encoding (LDE) layer, which combines the entire dictionary learning and vector encoding pipeline into a single layer for end-to-end deep CNN. The LDE layer imitates the mechanism of conventional GMM and GMM Supervector, but learned directly from the loss function. This representation is orderless which might be suitable for LID and many other test-independent paralinguistic speech attribute recognition tasks. The LDE layer acts as a smart pooling layer integrated on top of convolutional layers, accepting variable length inputs and providing output as an utterance level vector representation. By allowing variable-length inputs, the LDE layer makes the deep learning framework more flexible to train utterances with arbitrary duration. In these sense, it is inline with the classical GMM i-vector [23] method both theoretically and practically.

2. METHODS

2.1. GMM Supervector

In conventional GMM Supervector approach , all frames of features in training dataset are grouped together to estimate a universal background model (UBM). Given a C component GMM UBM model λ with $\lambda_c = \{p_c, \boldsymbol{\mu_c}, \boldsymbol{\Sigma_c}\}, c = 1, \cdots, C$ and an utterance with a L frame feature sequence $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_L}\}$, the 0^{th} and centered 1^{st} order Baum-Welch statistics on the UBM are calculated as follows:

$$N_c = \sum_{t=1}^{L} P(c|\boldsymbol{x_t}, \lambda) \tag{1}$$

$$\boldsymbol{F_c} = \sum_{t=1}^{L} P(c|\boldsymbol{x_t}, \lambda) \cdot \boldsymbol{r_{tc}} \tag{2}$$

where $c = 1, \cdots, C$ is the GMM component index and $P(c|\boldsymbol{x_t}, \lambda)$ is the occupancy probability for $\boldsymbol{x_t}$ on λ_c. $\boldsymbol{r_{tc}} = \boldsymbol{x_t} - \boldsymbol{\mu_c}$ denotes as a residual between t^{th} frame feature and the mean of the GMM's c^{th} component.

The corresponding centered mean supervector $\tilde{\boldsymbol{F}}$ is generated by concatenating all the $\tilde{\boldsymbol{F}}_{\boldsymbol{c}}$ together:

$$\tilde{\boldsymbol{F}}_{\boldsymbol{c}} = \frac{\sum_{t=1}^{L} P(c|\boldsymbol{x_t}, \lambda) \cdot \boldsymbol{r_{tc}}}{\sum_{t=1}^{L} P(c|\boldsymbol{x_t}, \lambda)}. \tag{3}$$

2.2. LDE layer

Motivated by GMM Supervector encoding procedure, the proposed LDE layer has the similar input-output structure. As demonstrated in Fig. 1, given an input temporal ordered feature sequence with the shape $D \times L$ (where D denotes the feature coefficients dimension, and L denotes the temporal duration length), LDE layer aggregates them over time. More specifically, it transforms them into an utterance level temporal orderless $D \times C$ vector representation, which is independent of length L.

Different from conventional approaches, we combine the dictionary learning and vector encoding into a single LDE layer on top of the front-end CNN, as shown in Fig. 2. The LDE layer simultaneously learns the encoding parameters along with an inherent

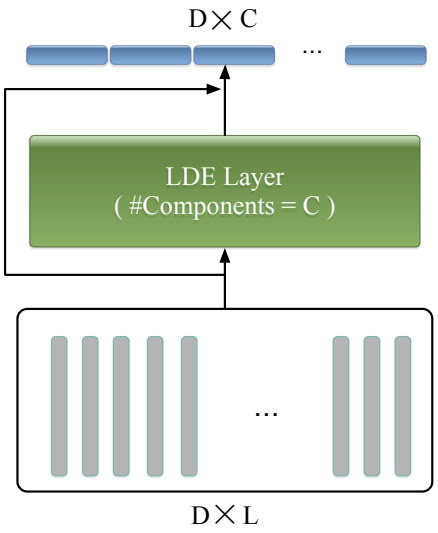

Fig. 1. The input-out structure of LDE layer. It receives input feature sequence with variable length, produces an encoded utterance level vector with fixed dimension

dictionary in a fully supervised manner. The inherent dictionary is learned from the distribution of the descriptors by passing the gradient through assignment weights. During the training process, the updating of extracted convolutional features can also benefit from the encoding representations.

The LDE layer is a directed acyclic graph and all the components are differentiable $w.r.t$ the input $\boldsymbol{X}$ and the learnable parameters. Therefore, the LDE layer can be trained end-to-end by standard stochastic gradient descent with backpropagation. Fig. 3 illustrates the forward diagram of LDE layer. Here, we introduce two groups of learnable parameters. One is the dictionary component center, noted as $\boldsymbol{\mu} = \{\boldsymbol{\mu_1}, \cdots \boldsymbol{\mu_c}\}$. The other one is assigned weights, which is designed to imitate the $P(c|\boldsymbol{x_t}, \lambda)$, noted as $\boldsymbol{w}$.

Consider assigning weights from the features to the dictionary components. Hard-assignment provides a binary weight for each feature $\boldsymbol{x_t}$, which corresponds to the nearest dictionary components. The c^{th} element of the assigning vector is given by $w_{tc} = \delta(\|\boldsymbol{r_{tc}}\|^2 = \min\{\|\boldsymbol{r_{t1}}\|, \cdots \|\boldsymbol{r_{tC}}\|\})$, where δ is the indicator function (outputs 0 or 1). Hard-assignment does not consider the dictionary component ambiguity and also makes the model non-differentiable. Soft-weight assignment addresses this issue by assigning the feature to each dictionary component. The non-negative assigning weight is given by a softmax function,

$$w_{tc} = \frac{\exp(-\beta\|\boldsymbol{r}_{tc}\|^2)}{\sum_{m=1}^{C} \exp(-\beta\|\boldsymbol{r}_{tm}\|^2)} \tag{4}$$

where β is the smoothing factor for the assignment. Soft-assignment assumes that different clusters have equal scales. Inspired by GMM, we further allow the smoothing factor s_c for each dictionary center $\boldsymbol{u_c}$ to be learnable:

$$w_{tc} = \frac{\exp(-s_c\|\boldsymbol{r_{tc}}\|^2)}{\sum_{m=1}^{C} \exp(-s_i\|\boldsymbol{r_{tm}}\|^2)} \tag{5}$$

which provides a finer modeling of the feature distributions.

Given a set of L frames feature sequence $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_L}\}$ and a learned dictionary center $\boldsymbol{\mu} = \{\boldsymbol{\mu_1}, \cdots \boldsymbol{\mu_c}\}$, each frame of

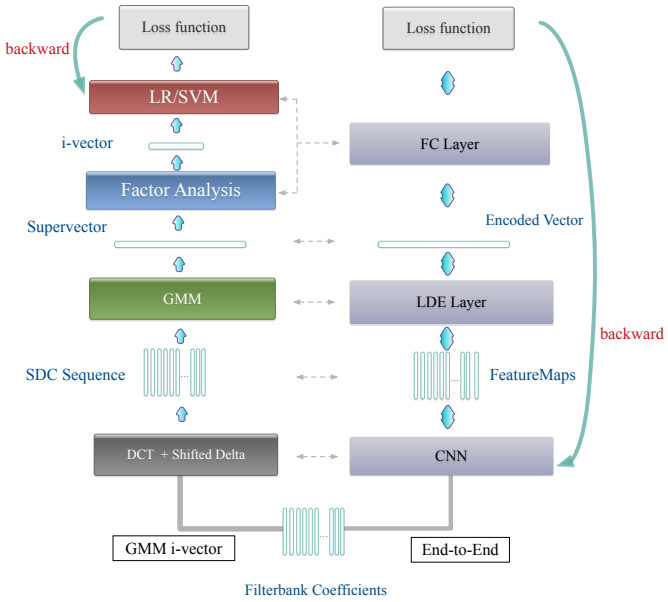

Fig. 2. Comparison of GMM i-vector approach and end-to-end neural network with LDE layer

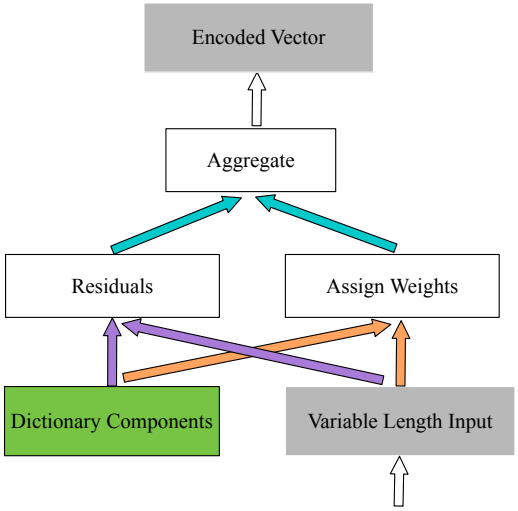

Fig. 3. The forward diagram within the LDE layer

feature $\boldsymbol{x_t}$ can be assigned with a weight w_{tc} to each component $\boldsymbol{\mu_c}$ and the corresponding residual vector is denoted by $\boldsymbol{r_{tc}} = \boldsymbol{x_t} - \boldsymbol{u_c}$, where $t = 1, \cdots L$ and $c = 1, \cdots C$. Given the assignments and the residual vector, similar to conventional GMM Supervector, the residual encoding model applies an aggregation operation for every dictionary component center $\boldsymbol{\mu_c}$:

$$\boldsymbol{e_c} = \sum_{t=1}^{L} e_{tc} = \frac{\sum_{t=1}^{L}(w_{tc} \cdot \boldsymbol{r_{tc}})}{\sum_{t=1}^{L} w_{tc}} \tag{6}$$

It's complicated to compute the the explicit expression for the gradients of the loss ℓ with respect to the layer input $\boldsymbol{x_t}$. In order to facilitate the derivation we simplified it as

$$\boldsymbol{e_c} = \frac{\sum_{t=1}^{L}(w_{tc} \cdot \boldsymbol{r_{tc}})}{L} \tag{7}$$

The LDE layer concatenates the aggregated residual vectors with assigned weights. The resulted encoder outputs a fixed dimensional representation $\boldsymbol{E} = \{\boldsymbol{e_1}, \cdots \boldsymbol{e_C}\}$ (independent of the sequence length L). As is typical in conventional GMM Supervector/i-vector, the resulting vectors are normalized using the length normalization [24].

We implement the LDE layer similar as described in [22], and more detail about the explicit expression for the gradients of the loss ℓ with respect to the layer input and the parameters can refer to [22].

2.3. Relation to traditional dictionary learning and TAP layer

Dictionary learning is usually learned from the distribution of the descriptors in an unsupervised manner. K-means learns the dictionary using hard-assignment grouping. GMM is a probabilistic version of K-means, which allows a finer modeling of the feature distributions. Each cluster is modeled by a Gaussian component with its own mean, variance and mixture weight. The LDE layer makes the inherent dictionary differentiable $w.r.t$ the loss function and learns the dictionary in a supervised manner. To see the relationship of the LDE to K- means, consider Fig. 3 with omission of the residual vectors and let smoothing factor $\beta \to \infty$. With these modifications, the LDE layer acts like K-means. The LDE layer can also be regarded as a simplified version of GMM, that allows different scaling (smoothing) of the clusters.

Letting $C = 1$ and fixing $\boldsymbol{\mu} = 0$, the LDE layer simplifies to TAP layer ($\boldsymbol{e} = \frac{\sum_{t=1}^{L} \boldsymbol{x_t}}{L}$) .

3. EXPERIMENTS

3.1. Data description

We conducted experiments on 2007 NIST Language Recognition Evaluation(LRE). Our training corpus including Callfriend datasets, LRE 2003, LRE 2005, SRE 2008 datasets, and development data for LRE07. The total training data is about 37000 utterances.

The task of interest is the closed-set language detection. There are totally 14 target languages in testing corpus, which included 7530 utterances split among three nomial durations: 30, 10 and 3 seconds.

3.2. GMM i-vector system

For better result comparison, we built a referenced GMM i-vector system based on Kaldi toolkit [25]. Raw audio is converted to 7-1-3-7 based 56 dimensional shifted delta coefficients (SDC) feature, and a frame-level energy-based voice activity detection (VAD) selects features corresponding to speech frames. All the utterances are split into short segments no more than 120 seconds long. A 2048 components full covariance GMM UBM is trained, along with a 600 dimensional i-vector extractor, followed by length normalization and multi-class logistic regression.

3.3. End-to-end system

Audio is converted to 64-dimensional log mel-filterbank coefficients with a frame-length of 25 ms, mean-normalized over a sliding window of up to 3 seconds. The same VAD processing as in GMM

Table 1. Performance on the 2007 NIST LRE closed-set task

System ID	System Description	Feature	Encoding Method	$C_{avg}(\%)$			$EER(\%)$		
				3s	10s	30s	3s	10s	30s
1	GMM i-vector	SDC	GMM Supervector	20.46	8.29	3.02	17.71	7.00	2.27
2	CNN-TAP	CNN FeatureMaps	TAP	9.98	3.24	1.73	11.28	5.76	3.96
3	CNN-LDE(C=16)	CNN FeatureMaps	LDE	9.61	3.71	1.74	8.89	2.73	1.13
4	CNN-LDE(C=32)	CNN FeatureMaps	LDE	8.70	2.94	1.41	8.12	2.45	0.98
5	**CNN-LDE(C=64)**	CNN FeatureMaps	LDE	**8.25**	**2.61**	**1.13**	**7.75**	**2.31**	**0.96**
6	CNN-LDE(C=128)	CNN FeatureMaps	LDE	8.56	2.99	1.63	8.20	2.49	1.12
7	CNN-LDE(C=256)	CNN FeatureMaps	LDE	8.77	3.01	1.97	8.59	2.87	1.38
8	Fusion ID2 + ID5	-	-	**6.98**	**2.33**	**0.91**	**6.09**	**2.26**	**0.87**

Table 2. Our front-end CNN configuration

layer	output size	downsample	channels	blocks
conv1	$64 \times L_{in}$	False	16	-
res1	$64 \times L_{in}$	False	16	3
res2	$32 \times \frac{L_{in}}{2}$	True	32	4
res3	$16 \times \frac{L_{in}}{4}$	True	64	6
res4	$8 \times \frac{L_{in}}{8}$	True	128	3
avgpool	$1 \times \frac{L_{in}}{8}$	-	128	-
reshape	$128 \times L_{out}, L_{out} = \frac{L_{in}}{8}$	-	-	-

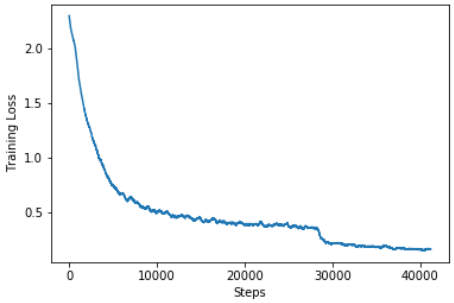

Fig. 4. Loss duraing CNN-LDE training stage, smoothed with each 400 steps

i-vector baseline system is used here. For improving the data loading efficiency, all the utterances are split into short segments no more than 60s long , according to the VAD flags.

The receptive field size of a unit can be increased by stacking more layers to make the network deeper or by sub-sampling. Modern deep CNN architectures like Residual Networks [26] use a combination of these techniques. Therefore, in order to get higher abstract representation better for utterances with long duration, we design a deep CNN based on the well-known ResNet-34 layer architecture, as is described in Table 2.

For CNN-TAP system, a simple average pooling layer followed with FC layer is built on top of the font-end CNN. For CNN-LDE system, the average pooling layer is replaced with a LDE layer.

The network is trained using a cross entropy loss. The model is trained with a mini-batch, whose size varies from 96 to 512 considering different model parameters. The network is trained for 90 epochs using stochastic gradient descent with momentum 0.9 and weight decay 1e-4. We start with a learning rate of 0.1 and divide it by 10 and 100 at 60th and 80th epoch. Because we have no separated validation set, even though there might exist some model checkpoints can achieve better performance, we only use the model after the last step optimization. For each training step, an integer L within [200,1000] interval is randomly generated, and each data in the mini-batch is cropped or extended to L frames. The training loss tendency of our end-to-end CNN-LDE neural network is demonstrated in Fig. 4. It shows that our neural network with LDE layer is traninable and the loss can converge to a small value.

In testing stage, all the 3s, 10s, and 30s duration data is tested on the same model. Because the duration length is arbitrary, we feed the testing speech utterance to the trained neural network one by one.

In order to get the system fusion results of ID8 in Table 1, we randomly crop several additional training data corresponding to the separated 30s, 10s, 3s duration tasks. The score level system fusion weights are all trained on them.

3.4. Evaluation

Table 1 shows the performance on the 2007 NIST LRE closed-set task. The performance is reported in average detection cost C_{avg} and equal error rate (EER). Both CNN-TAP and CNN-LDE system achieve significant performance improvement comparing with conventional GMM i-vector system.

For our purpose in exploring encoding method for end-to-end neural network, we focus the comparison on system ID2 and ID3-ID7. The CNN-LDE system outperforms the CNN-TAP system with all different number of dictionary components. When the numbers of dictionary component increased from 16 to 64, the performance improved insistently. However, once dictionary component numbers are larger than 64, the performance decreased perhaps because of overfitting.

Comparing with CNN-TAP, the best CNN-LDE-64 system achieves significant performance improvement especially with regard to EER. Besides, their score level fusion result further improves the system performance significantly.

4. CONCLUSIONS

In this paper, we imitate the GMM Supervector encoding procedure and introduce a LDE layer for end-to-end LID neural network. The LDE layer acts as a smart pooling layer integrated on top of convolutional layers, accepting arbitrary input lengths and providing output as a fixed-length representation. Unlike the simple TAP, it rely on a learnable dictionary and can accumulate more discriminative statistics. The experiment results show the superior and complementary of LDE comparing with TAP.

5. REFERENCES

[1] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12–40, 2010.

[2] F. Soong, A. E. Rosenberg, J. BlingHwang, and L. R. Rabiner, "Report: A vector quantization approach to speaker recognition," *At & T Technical Journal*, vol. 66, no. 2, pp. 387–390, 1985.

[3] D.A. Reynolds and R.C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Transactions on Speech & Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[4] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker verification using adapted gaussian mixture models," in *Digital Signal Processing*, 2000, p. 1941.

[5] W.M. Campbell, D.E. Sturim, and DA Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

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

[7] M. Li and W. Liu, "Speaker verification and spoken language identification using a generalized i-vector framework with phonetic tokenizations and tandem features," in *INTERSPEECH 2014*.

[8] M. Mclaren, Y. Lei, and L. Ferrer, "Advances in deep neural network approaches to speaker recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4814–4818.

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

[10] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *ASRU 2016*, pp. 92–97.

[11] G. Gelly, J. L. Gauvain, V. B. Le, and A. Messaoudi, "A divide-and-conquer approach for language identification based on recurrent neural networks," in *INTERSPEECH 2016*, pp. 3231–3235.

[12] M. Li, L. Liu, W. Cai, and W. Liu, "Generalized i-vector representation with phonetic tokenizations and tandem features for both text independent and text dependent speaker verification," *Journal of Signal Processing Systems*, vol. 82, no. 2, pp. 207–215, 2016.

[13] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *ICASSP 2014*.

[14] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Proc. INTERSPEECH 2014*, 2014.

[15] R. Li, S. Mallidi, L. Burget, O. Plchot, and N. Dehak, "Exploiting hidden-layer responses of deep neural networks for language recognition," in *INTERSPEECH*, 2016.

[16] M. Tkachenko, A. Yamshinin, N. Lyubimov, M. Kotov, and M. Nastasenko, "Language identification using time delay neural network d-vector on short utterances," 2016.

[17] L. Chao, M. Xiaokong, J. Bing, L. Xiangang, Z. Xuewei, L. Xiao, C. Ying, K. Ajay, and Z. Zhenyao, "Deep speaker: an end-to-end neural speaker embedding system," 2017.

[18] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *SLT 2017*, pp. 165–170.

[19] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR 2016*, 2016, pp. 5297–5307.

[20] S. Simonyan, A. Vedaldi, and A. Zisserman, "Deep fisher networks for large-scale image classification," in *NIPS 2013*, pp. 163–171.

[21] T. Lin, A. Roychowdhury, and S. Maji, "Bilinear cnns for fine-grained visual recognition," in *IEEE International Conference on Computer Vision*, 2016, pp. 1449–1457.

[22] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *CVPR 2017*.

[23] 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.

[24] Daniel Garcia-Romero and Carol Y Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems.," in *INTERSPEECH 2011*, pp. 249–252.

[25] D. Povey and A. et al. Ghoshal, "The kaldi speech recognition toolkit," in *ASRU 2011*.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR 2016*, 2016, pp. 770–778.