Liu *et al. EURASIP Journal on Audio, Speech, and Music Processing* (2019) 2019:19
https://doi.org/10.1186/s13636-019-0166-8

EURASIP Journal on Audio, Speech, and Music Processing

RESEARCH Open Access

Introducing phonetic information to speaker embedding for speaker verification

Yi Liu[1], Liang He[1*], Jia Liu[1] and Michael T. Johnson[2]

Abstract

Phonetic information is one of the most essential components of a speech signal, playing an important role for many speech processing tasks. However, it is difficult to integrate phonetic information into speaker verification systems since it occurs primarily at the frame level while speaker characteristics typically reside at the segment level. In deep neural network-based speaker verification, existing methods only apply phonetic information to the frame-wise trained speaker embeddings. To improve this weakness, this paper proposes phonetic adaptation and hybrid multi-task learning and further combines these into c-vector and simplified c-vector architectures. Experiments on National Institute of Standards and Technology (NIST) speaker recognition evaluation (SRE) 2010 show that the four proposed speaker embeddings achieve better performance than the baseline. The c-vector system performs the best, providing over 30% and 15% relative improvements in equal error rate (EER) for the core-extended and 10 s–10 s conditions, respectively. On the NIST SRE 2016, 2018, and VoxCeleb datasets, the proposed c-vector approach improves the performance even when there is a language mismatch within the training sets or between the training and evaluation sets. Extensive experimental results demonstrate the effectiveness and robustness of the proposed methods.

Keywords: Speaker verification, Deep neural networks, Speaker embedding, Phonetic adaptation, Multi-task learning, C-vector

1 Introduction

Automatic speaker verification (ASV) has made great strides in the last two decades, moving from traditional Gaussian mixture model (GMM) approaches [1] to the i-vector framework [2] and neural network-based speaker embedding [3]. Based on Bayesian factor analysis, the i-vector framework converts a variable-length speech utterance into a fixed-length vector representing speaker characteristics. A variety of backend classifiers can be applied to suppress session variability and increase speaker discrimination. Even though the i-vector approach performed well in previous National Institute of Standards and Technology (NIST) speaker recognition evaluations (SREs), it is known to suffer from many problems in practical applications. The i-vector is a point estimate of the total variability factor, ignoring the covariance [2, 4]. The performance of i-vector systems deteriorates dramatically when the utterance is short, because this point estimate does not model the uncertainty [5]. I-vectors are also vulnerable to language and channel mismatch as shown in recent NIST SREs [6, 7]. Moreover, the performance of the i-vector model tends to asymptote quickly as the amount of data increases, which means that it is unable to fully exploit the availability of large-scale training data [3].

Deep neural networks (DNNs) have been used for speech processing tasks for a number of years [8–10]. Recently, neural network-based *speaker embedding* has drawn much attention in the speaker verification community. Motivated by the i-vector concept, speaker embedding encodes the speaker characteristics of an utterance into a fixed-length vector using neural networks. The first such method was the *d-vector* approach, initially proposed for text-dependent speaker verification [11]. The network was trained frame-by-frame and the d-vector was extracted by averaging all the activations of a selected hidden layer from an utterance. This network architecture was extended to text-independent verification in [12].

*Correspondence: heliang@tsinghua.edu.cn
[1]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, 100084 Beijing, China
Full list of author information is available at the end of the article

Critics of this approach argue that the frame-wise training is not a good option since speaker information tends to reside within long-term segments [13, 14].

To address this problem, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) were then introduced to directly capture segment information [13, 15, 16]. Network architectures and training strategies used in image and face recognitions have been adapted to speaker verification [17–19]. By using approaches such as statistics pooling [3], self attention [14] and learnable dictionary encoding (LDE) [20], neural networks are able to extract meaningful low-dimensional vectors from utterances. More effective loss functions have also been proposed to further encourage discriminative learning of speaker embeddings [21–24]. Speaker embedding has shown state-of-the-art performance comparative to i-vectors in many conditions. Speaker embedding also benefits from its ability to utilize big data [25], which is valuable in commercial applications. Based on these advantages, speaker embedding is quickly replacing the i-vector approach as the next generation of speaker verification technology.

Of the many components of a speech signal, speaker traits, and phonetic content, representing *who spoke what*, are two predominant factors for human communication. The mixing of speaker traits and phonetic contents creates challenges for speaker verification. Although speaker embedding has achieved superior performance, most current systems still do not take phonetic content into account. However, in many cases, networks cannot separate speaker information from the intermingled signal and different techniques should be applied. For example, in automatic speech recognition (ASR), speaker adaptation is used to reduce the impact of the speaker factor to improve accuracy [26]. In a similar way, it should be possible to reduce the impact of phonetic information on the speaker embedding.

This is a difficult task, however, because phonetic information is dominant at the frame level while speaker information is typically extracted at the segment level. To overcome this *level mismatch problem*, we propose several methods in this paper to explicitly introduce frame-level phonetic information into the segment-level speaker embedding extraction. The first of these is phonetic adaptation. Similarly to speaker adaptation in ASR, phonetic adaptation uses phonetically rich vectors to remove the influence of phonetic content. This enables the network to focus on speaker traits which are insensitive to content variation. The second approach uses hybrid multi-task learning to extract the information shared between the speaker and phonetic components. This makes the network more robust against noise and improves the model generalization. Since these two approaches are designed from different perspectives, the phonetic adaptation and the hybrid multi-task learning can be combined into a novel *c-vector* (phonetic information combined vector). A simplified c-vector approach is further presented to reduce the model size.

This paper is an extension of our previous work presented in [27]. The new contributions of this paper are as follows:

- A new c-vector approach has been proposed, combining phonetic adaptation and hybrid multi-task learning. A simplified c-vector architecture has also been presented.
- Extensive experiments on 8-kHz NIST SREs and 16-kHz VoxCeleb [19, 28] have been conducted. Severe language mismatch has been introduced into the experiments to assess the generalization of the proposed approaches.
- Data augmentation has been added and a better baseline than that reported in our previous work [27] has been built. In addition, larger datasets have been used to train the phonetic-related models in this paper. These modifications evaluate the effectiveness of the proposed approaches when more training data is available.

The experiments in this paper demonstrate that our proposed systems significantly outperform conventional speaker embedding. The best results are obtained with the c-vector approach. On the NIST SRE 2010 dataset, the resulting relative improvement in equal error rate (EER) is over 30% for the core-extended condition and 15% for the 10 s-10 s condition. Results on NIST SRE 2016, 2018, and VoxCeleb further validate the effectiveness of our methods in the language mismatched condition.

The outline of the paper is as follows. The existing literature about the use of phonetic information in speaker verification is briefly reviewed in Section 2. Section 3 describes the baseline system, and Section 4 presents our proposed approaches to introduce phonetic information in the speaker embedding neural network. Our experimental setup and results are given in Section 5. The last section concludes the paper.

2 Phonetic information in speaker verification

From an acoustic perspective, speaker traits, phonetic content and other components are intermingled throughout the speech signal. How to separate the speaker traits from speech content is the key problem in speaker verification.

Gaussian mixture models have been successfully used in speaker verification for several decades. In GMM-based speaker verification, features are required to be softly aligned to the corresponding Gaussian mixtures to compute the sufficient statistics. This frame alignment plays an important role in the GMM framework. To improve

alignment accuracy, fine-grained GMMs were first proposed to model individual phoneme groups [29, 30]. DNN acoustic models were later introduced to improve the frame alignment in [31]. In the DNN approach, the phonetic content is modeled by senones, which are sub-phonetic classes in speech recognition. The posteriors on these senones are estimated by DNNs and are then used to compute the statistics for the i-vector modeling. This model was extended in [32] where broader phonetic units were investigated. These works showed that comparison of speakers within the same phonetic category reduces the impact of the phonetic variability. Bottleneck (BN) features extracted from ASR acoustic models, which have rich phonetic information, have also been used in many approaches, with and without DNN-based alignment [33]. Overall, the i-vector approach based on DNN alignment and BN features greatly outperforms conventional systems. The existing work in i-vector-based systems suggests the importance of considering phonetic information in speaker verification.

For neural network-based speaker embedding, the d-vector network in [34] used the concatenation of raw features and the outputs of an ASR network to represent phonetic information. In [35], conventional Mel-frequency cepstral coefficients (MFCCs) were replaced by ASR BN features to train the speaker embedding extractor. A collaborative joint training was presented in [36], in which the speaker and speech recognition networks were interconnected. Using an RNN architecture, the outputs of one task were fed into another at the next time step. This feedback enabled the speaker network to receive the information from the speech recognition task.

Multi-task learning has also been investigated for speaker verification. Multi-task learning has been shown to be useful across many different tasks [37]. Speaker traits and phonetic information, two key components of speech, have been combined through the use of multi-task networks. In [38], phonetically-related classification was considered as a parallel task to the speaker classification network. The extracted features were effective for text-dependent speaker verification. The same idea has been used in some other works as well [35]. The rationale is that by exploring the common information shared between the speaker and phonetic components, multi-task learning can prevent overfitting and improve model generalization.

However, to the best of our knowledge, none of these networks involving phonetic information consider the level mismatch problem and can only be trained at the frame level (i.e. in the d-vector style). This is not applicable to state-of-the-art segment-level speaker embeddings. Another problem in current multi-task learning for speaker verification is that most multi-task networks share all hidden layers between the speaker and phonetic-discriminant tasks, which is not ideal since this ignores the fact that speaker traits and phonetic content are quite different and likely need several individual layers to extract their own features. Therefore, it is necessary to propose novel architectures to combine the phonetic information with the speaker embedding.

3 The x-vector baseline

The baseline speaker embedding used in this paper is *x-vector* [3]. X-vector is popular in the speaker verification community and has been provided as the official system on recent NIST SREs. The architecture is illustrated in Fig. 1.

The x-vector network consists of frame-level and segment-level sub-networks, connected by a statistics pooling layer. The frame-level network can be seen as a speaker feature extractor. Given the input sequence $\vec{X}^{(k)} = [\vec{x}_1^{(k)}, \ldots, \vec{x}_{T_k}^{(k)}]$ from utterance k with T_k frames, the frame-level network tries to transform the acoustic features $\vec{x}_t^{(k)}$ into speaker-discriminant features $\vec{f}_t^{(k)}$. A CNN variant, time-delay neural network (TDNN) whose input of each layer is the sliced outputs of the previous layer, is used. We omit the index k for brevity, denoting the frame-level network as

$$\vec{f}_t = \mathcal{F}\left(\vec{x}_t | \Theta_f\right) \tag{1}$$

where $\mathcal{F}(\cdot)$ denotes the feed-forward function and Θ_f is the parameters for frame-level network.

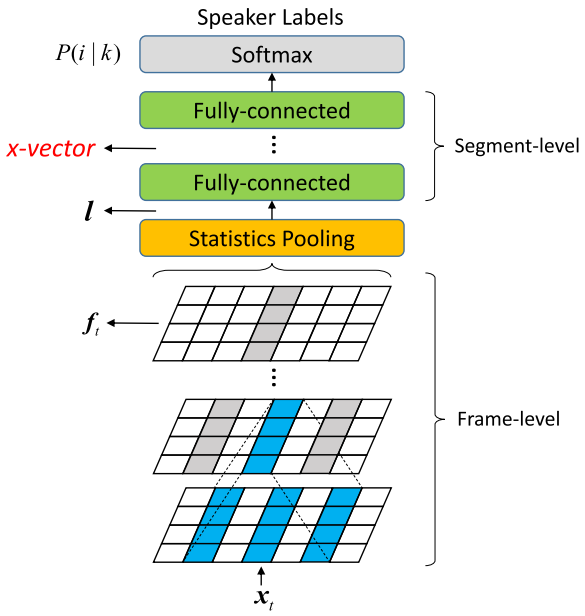

Fig. 1 The x-vector architecture. This architecture can be partitioned into frame- and segment-level sub-networks

Next, a statistics pooling layer aggregates the speaker features $\vec{f}_t$ and concatenates the mean and standard deviation as a segment-level representation $\vec{l}$.

$$\vec{l} = \left[\vec{m}^T, \vec{\sigma}^T\right]^T \tag{2}$$

$$\vec{m} = \frac{1}{T}\sum_{t=1}^{T}\vec{f}_t \tag{3}$$

$$\vec{\sigma} = \left(\frac{1}{T}\sum_{t=1}^{T}(\vec{f}_t - \vec{m})^2\right)^{1/2} \tag{4}$$

Fully-connected layers with parameters Θ_l are then implemented as the segment-level network. The output of the segment-level network is fed into a softmax layer and the posterior $P(i|k)$ of speaker i is calculated as

$$P(i|k) = \text{softmax}\left(\mathcal{F}\left(\vec{l}|\Theta_l\right)\right) \tag{5}$$

The x-vector network parameter $\Theta = \{\Theta_f, \Theta_l\}$ is trained by minimizing the cross entropy loss. After training, the pre-activation of a hidden layer at the segment-level network is extracted as the speaker embedding. The x-vector backend processing is similar to that of i-vector. Mean normalization is first applied. Then, linear discriminant analysis (LDA) can be used to reduce the dimension of the embedding and length normalization is also performed [39]. Finally, probabilistic linear discriminant analysis (PLDA) scoring is introduced to generate the verification scores.

4 Proposed methods

In this section, we will describe our methods to tackle the level mismatch problem and introduce frame-level phonetic information into the training and extraction of the segment-level speaker embedding. Both phonetic adaptation and hybrid multi-task learning are proposed, which are then further combined into an integrated c-vector network.

4.1 Phonetic adaptation

A pooling strategy is used in the speaker embedding neural networks to aggregate frame-level speaker features into segment-level representations. Statistics pooling, which concatenates the first- and second-order statistics (i.e., the mean and standard deviation) as outputs, is applied in the x-vector architecture. In speech signals, speaker features are influenced by phonetic contents. To make the pooling more effective, phonetic information should be considered in the frame-level network.

Motivated by speaker adaptation as used in speech recognition, we propose a *phonetic adaptation* method. In speech recognition, a speaker code (e.g. i-vector) is used as an auxiliary input to help the network reduce the impact of speaker changes [26]. Similarly, phonetic adaptation can be done by feeding phonetically rich vectors into the x-vector network. With the phonetic vectors, the frame-level network is able to learn the phonetic-dependent transforms which is useful for the pooling layer.

In this paper, BN features extracted from an ASR acoustic model are selected as the phonetic vectors. As shown in Fig. 2, a phonetic-discriminant ASR acoustic model with parameters Θ_a is appended to the original network. The phonetic vector $\vec{p}_t$ is the activation extracted from a hidden layer of the appended model.

$$\vec{p}_t = \mathcal{F}\left(\vec{x}_t|\Theta'_a\right) \tag{6}$$

where Θ'_a denotes the parameters of the sub-network used to extract the phonetic vector. Since this sub-network is a part of the ASR acoustic model, Θ'_a can be derived from Θ_a. The frame-level network then becomes

$$\vec{f}_t = \mathcal{F}\left(\vec{x}_t, \vec{p}_t|\Theta_f\right) \tag{7}$$

For initialization, an ASR acoustic model with a BN layer is first pre-trained. The activations of the BN layer are connected to the x-vector frame-level network as phonetic vectors. The phonetic vectors can be extracted before the training of the x-vector network. In this case, the additional acoustic model is only used as a phonetic feature extractor. In our experiments, however, we find that fine-tuning the acoustic model with a small learning rate during the x-vector training improves the performance. The unused layers in the acoustic model are removed and the remaining part is updated with the x-vector network. The fine-tuning makes the phonetic vectors more adapted to the speaker verification task. The procedure is described in Algorithm 1.

Algorithm 1 Training speaker embedding using phonetic adaptation.

Require: α, the learning rate. c, the learning rate scaling factor. m, the batch size. n_s, the number of the training steps.

Pre-train an ASR acoustic model Θ_a. Derive the sub-network Θ'_a from the pre-trained model.

for $n = 1, \ldots, n_s$ **do**

Sample a mini-batch $\left\{\vec{\boldsymbol{X}}^{(k)}, y^{(k)}\right\}_{k=1}^{m}$ where $\vec{\boldsymbol{X}}^{(k)}$ is the feature sequence of utterance k and $y^{(k)}$ is the speaker label.

$L = \frac{1}{m}\sum_{k=1}^{m}\text{CrossEntropy}\left(\vec{\boldsymbol{X}}^{(k)}, y^{(k)}|\Theta_f, \Theta_l, \Theta'_a\right)$

$\Theta_f \leftarrow \Theta_f - \alpha \cdot \nabla_{\Theta_f} L$

$\Theta_l \leftarrow \Theta_l - \alpha \cdot \nabla_{\Theta_l} L$

$\Theta'_a \leftarrow \Theta'_a - c \cdot \alpha \cdot \nabla_{\Theta'_a} L$

end for

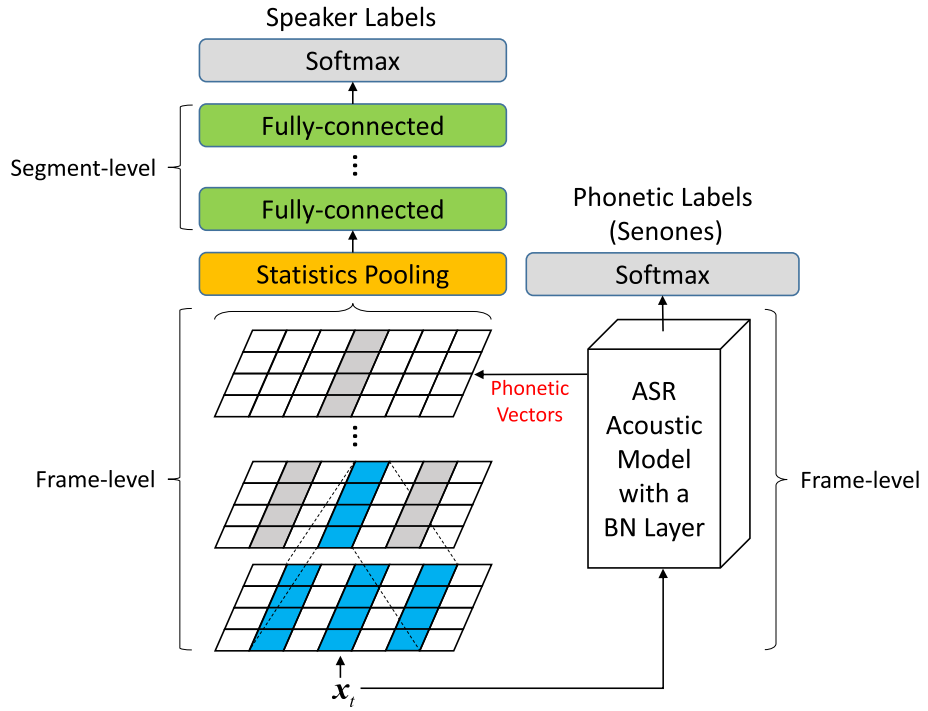

Fig. 2 Phonetic adaptation using phonetic vectors. An additional frame-level ASR acoustic model with a BN layer is appended to the conventional x-vector architecture. The phonetic vectors are extracted from the BN layer

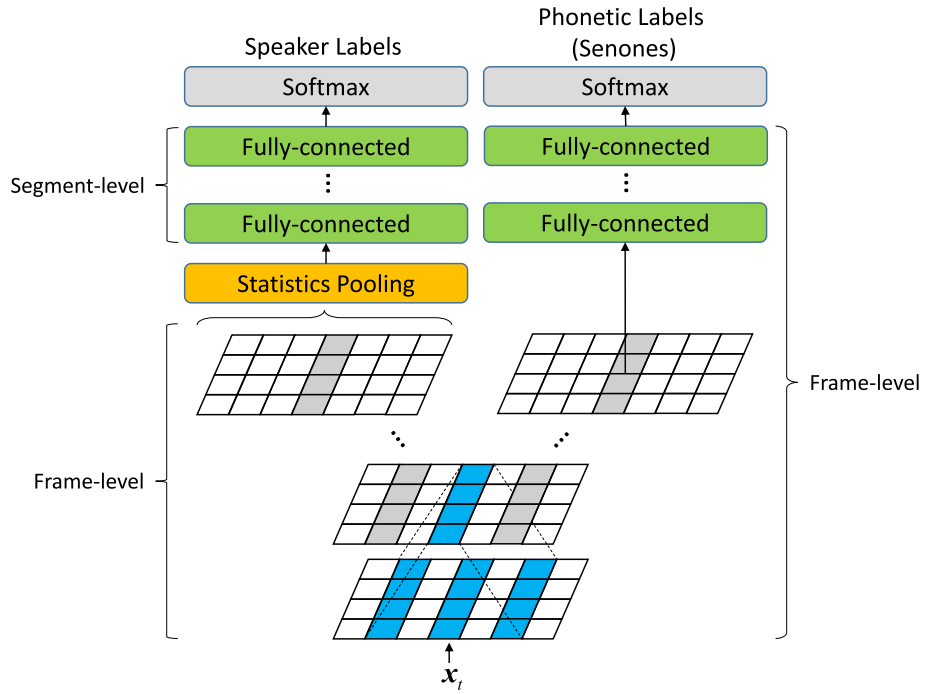

Fig. 3 Hybrid multi-task learning. The phonetic-discriminant network shares layers with the frame-level part of the x-vector architecture

4.2 Hybrid multi-task learning

Phonetic adaptation filters out the phonetic variability by introducing auxiliary vectors. However, although the speaker and phonetic components are different, they still share some common information. Some factors, such as formant, pitch trajectory, and spectral energy distribution, are essential for both speaker traits and phonetic contents. By using phonetic unit classification as a parallel task, multi-task learning can discover more informative features which are less sensitive to nuisance factors.

The conventional multi-task learning approach in speaker verification suffers from a level mismatch problem, because it requires all the tasks to operate at the same level. This only works for the frame-wise d-vector and is not suitable for the x-vector training. To address this, a hybrid multi-task learning framework is proposed in this section. In this hybrid framework, only the frame-level hidden layers in the x-vector network are shared with the phonetic-discriminant task. This architecture is able to process the frame-level phonetic information and the segment-level speaker embedding at the same time.

Multi-task networks also often share all the layers between different tasks. In our framework, the number of shared layers is set to be a hyper-parameter. This hyper-parameter controls the trade-off between the common and individual information in the speaker and phonetic tasks. The hybrid multi-task learning network is shown in Fig. 3.

In Fig. 3, the frame-level network of the x-vector architecture is partitioned into a shared and a non-shared parts whose parameters are Θ_s and Θ_{ns}, respectively. The parameters of the remaining layers in the phonetic-discriminant network are denoted as Θ_p. The speaker feature of the frame-level network is now

$$\vec{f}_t = \mathcal{F}\left(\vec{x}_t|\Theta_s, \Theta_{ns}\right) \tag{8}$$

The training strategy of our multi-task framework is similar to the multi-lingual acoustic model training [40]. The training data consists of speaker and phonetic examples. The speaker examples contain the reference speaker labels for each utterance while the phonetic examples contain the corresponding phonetic units for frames. The transcriptions of the phonetic units are obtained by forced alignment using a hidden Markov model (HMM). The two tasks are trained alternately. At each step, we randomly choose a mini-batch composed of the speaker examples from the pooled training data with probability $p_s = N_s/(N_s + N_p)$ and select the phonetic mini-batch otherwise. Here, N_s and N_p are the number of remaining speaker and phonetic examples. When the speaker exam-

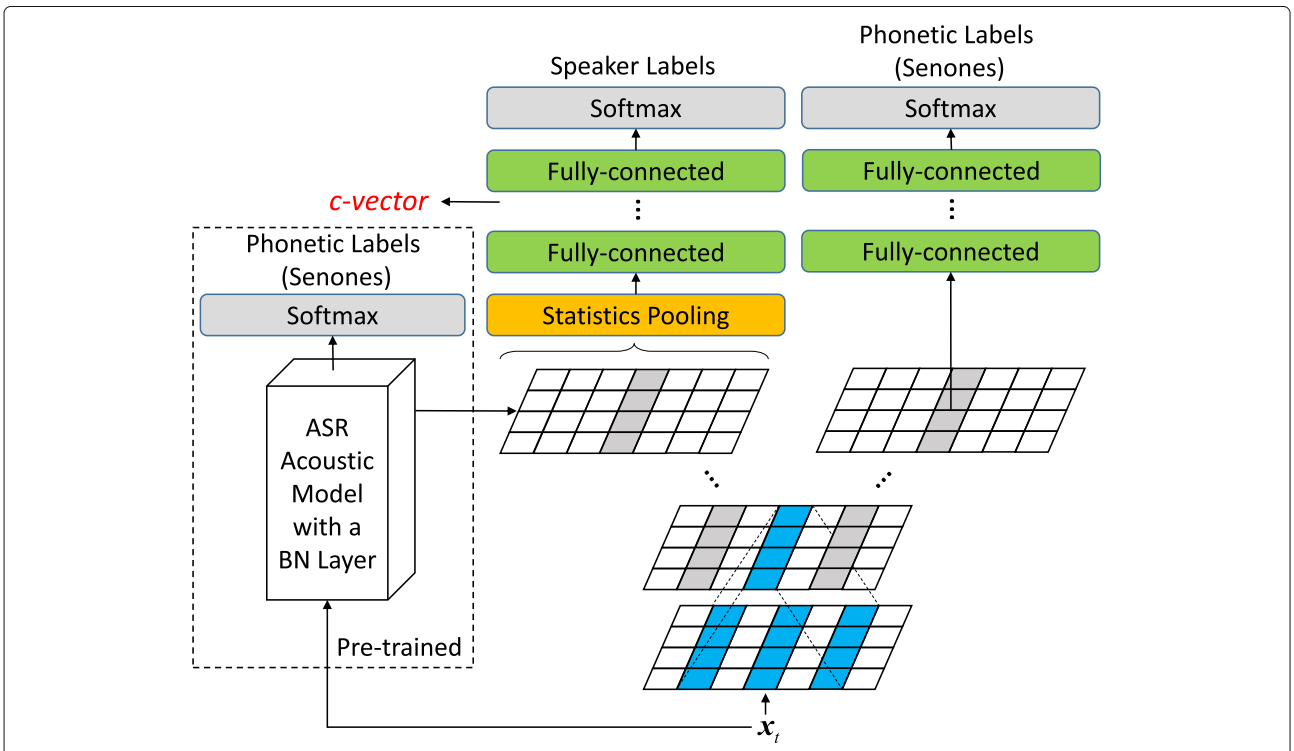

Fig. 4 The c-vector architecture. The x-vector networks using phonetic adaptation and multi-task learning are combined into a unified architecture

ples are selected, the parameters $\{\Theta_s, \Theta_{ns}, \Theta_l\}$ are updated and $\{\Theta_s, \Theta_p\}$ are trained when the phonetic examples are used. It is possible to set different learning rates to balance the importance of these tasks. The complete training procedure is described in Algorithm 2.

Algorithm 2 Training speaker embedding using hybrid multi-task learning.

Require: α_1, m_1, the learning rate and the batch size for the speaker task. α_2, m_2, the learning rate and the batch size for the phonetic task. n_s, the number of training steps. N_s, N_p, the total number of available training examples for speaker and phonetic tasks.

for $n = 1, \ldots, n_s$ **do**

Sample a speaker mini-batch $\left\{\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, y^{(k)}\right\}_{k=1}^{m_1}$ with probability $p_s = \frac{N_s}{N_s+N_p}$. Sample a phonetic mini-batch $\left\{\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, z^{(k)}\right\}_{k=1}^{m_2}$ otherwise. $y^{(k)}$ and $z^{(k)}$ are the speaker and phonetic unit labels, respectively.

if a speaker mini-batch is sampled **then**

$L_1 = \frac{1}{m_1}\sum_{k=1}^{m_1} \text{CrossEntropy}(\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, y^{(k)}|\Theta_s, \Theta_{ns}, \Theta_l)$

$\Theta_s \leftarrow \Theta_s - \alpha_1 \cdot \nabla_{\Theta_s} L_1$

$\Theta_{ns} \leftarrow \Theta_{ns} - \alpha_1 \cdot \nabla_{\Theta_{ns}} L_1$

$\Theta_l \leftarrow \Theta_l - \alpha_1 \cdot \nabla_{\Theta_l} L_1$

Reduce N_s to $N_s - m_1$

else

$L_2 = \frac{1}{m_2}\sum_{k=1}^{m_2} \text{CrossEntropy}(\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, z^{(k)}|\Theta_s, \Theta_p)$

$\Theta_s \leftarrow \Theta_s - \alpha_2 \cdot \nabla_{\Theta_s} L_2$

$\Theta_p \leftarrow \Theta_p - \alpha_2 \cdot \nabla_{\Theta_p} L_2$

Reduce N_p to $N_p - m_2$

end if

end for

4.3 The c-vector

In our phonetic adaptation approach, the phonetic content of an utterance is considered to have negative impact on the speaker verification task. In contrast, hybrid multi-task learning exploits the useful phonetic information to improve the model generalization. The different perspectives of these methods create an opportunity to further combine them into a unified architecture. In this section, we propose a c-vector using both techniques to accomplish this goal.

Figure 4 shows the c-vector architecture, which is a straightforward combination of Figs. 2 and 3. The phonetic vector extracted from a pre-trained acoustic model is introduced to the multi-task network. The phonetic vector is only used in the speaker task while the phonetic-discriminant network is kept unchanged. The integrated network can be jointly optimized following a similar strategy to that of Algorithm 2 except that we need to fine-tune the acoustic model as in Algorithm 1. Since the phonetic vector is only appended to the speaker task, the parameters of the acoustic model providing phonetic vectors will not be updated when the phonetic examples are selected. This will make the ASR acoustic model only focus on the speaker task. The training procedure of the c-vector is summarized in Algorithm 3.

Algorithm 3 The c-vector.

Require: α_1, m_1, the learning rate and the batch size for the speaker task. α_2, m_2, the learning rate and the batch size for the phonetic task. c, the learning rate scaling factor. n_s, the number of training steps. N_s, N_p, the total number of available training examples for speaker and phonetic tasks.

Pre-train an ASR acoustic model Θ_a. Derive the sub-network Θ'_a from the pre-trained model.

for $n = 1, \ldots, n_s$ **do**

Sample a speaker mini-batch $\left\{\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, y^{(k)}\right\}_{k=1}^{m_1}$ with probability $p_s = \frac{N_s}{N_s+N_p}$. Sample a phonetic mini-batch $\left\{\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, z^{(k)}\right\}_{k=1}^{m_2}$ otherwise.

if a speaker mini-batch is sampled **then**

$L_1 = \frac{1}{m_1}\sum_{k=1}^{m_1} \text{CrossEntropy}(\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, y^{(k)}|\Theta_s, \Theta_{ns}, \Theta_l, \Theta'_a)$

$\Theta_s \leftarrow \Theta_s - \alpha_1 \cdot \nabla_{\Theta_s} L_1$

$\Theta_{ns} \leftarrow \Theta_{ns} - \alpha_1 \cdot \nabla_{\Theta_{ns}} L_1$

$\Theta_l \leftarrow \Theta_l - \alpha_1 \cdot \nabla_{\Theta_l} L_1$

$\Theta'_a \leftarrow \Theta'_a - c \cdot \alpha_1 \cdot \nabla_{\Theta'_a} L$

Reduce N_s to $N_s - m_1$

else

$L_2 = \frac{1}{m_2}\sum_{k=1}^{m_2} \text{CrossEntropy}(\vec{\boldsymbol{X}}^{(\boldsymbol{k})}, z^{(k)}|\Theta_s, \Theta_p)$

$\Theta_s \leftarrow \Theta_s - \alpha_2 \cdot \nabla_{\Theta_s} L_2$

$\Theta_p \leftarrow \Theta_p - \alpha_2 \cdot \nabla_{\Theta_p} L_2$

Reduce N_p to $N_p - m_2$

end if

end for

In the c-vector architecture, two independent phonetic branches are used. This is necessary since these two sub-networks are optimized by different objective functions. However, there is also a need to limit the model size. We notice that, in the multi-task learning, the phonetic-discriminant network also provides frame-wise phonetic information. Based on the c-vector architecture, a simplified model is proposed in Fig. 5. In the new model, the pre-trained acoustic model is first removed. A BN layer is then incorporated in the phonetic-discriminant network and the phonetic vectors are extracted from this layer.

Although the speaker-discriminant network in the simplified c-vector uses the activations of the BN layer in the phonetic-discriminant network in the feed-forward step,

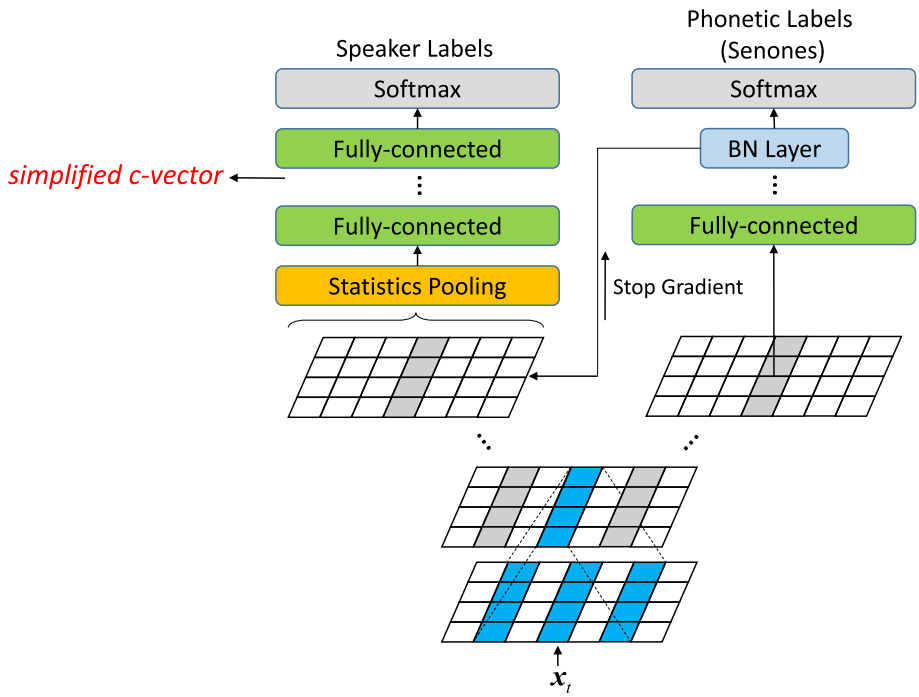

Fig. 5 The simplified c-vector architecture. The additional acoustic model is removed and the phonetic vectors come from the BN layer of the phonetic-discriminant network. The gradient-based training is stopped at the interconnected link between the two sub-networks

the gradient from this sub-network should not be back-propagated through the phonetic-discriminant network. The reason is that the phonetic-discriminant network is optimized for phonetic unit classification in the multi-task learning framework. The speaker information introduced into the phonetic-discriminant network may affect the training procedure and weaken the effectiveness of the multi-task learning. To prevent this impact, when optimizing the speaker-discriminant network, the gradient-based training is stopped at the connection introducing phonetic vectors by setting the gradient to zero. It should be pointed out that, in the simplified version of the c-vector, the phonetic-discriminant network cannot be adapted freely; thus, the phonetic vectors are not optimized for the speaker verification task.

In our proposed methods, the ASR acoustic model and the speaker and phonetic-discriminant networks are trained alternately. This procedure does not require the training data to be both speaker and phonetically transcribed which is different than many conventional multi-task networks. This flexibility is quite desirable in practice since we may collect the speaker data from one source and the phonetic data from other sources. Even although many speaker verification datasets do not have phoneme or text transcriptions, we can use other ASR datasets to introduce the phonetic information to our speaker embeddings.

5 Experiments

5.1 Datasets

The performance of the proposed approaches is presented on NIST SREs and VoxCeleb datasets.

Experiments are first carried out on the NIST SRE 2010 core-extended and 10 s–10 s condition 5 [41]. Both conditions involve English conversational telephone speech. The core-extended condition consists of 2-min enrollment and test utterances while the duration of the utterances in the 10 s-10 s condition range from 8 to 12 s. To validate our proposed methods when utterances of different languages are presented, the NIST SRE 2016 [42] and 2018 [43] datasets are then used. The NIST SRE 2016 evaluation set contains trials spoken in Tagalog and Cantonese. Although two sources, namely call my net 2 (CMN2) and video annotation for speech technology (VAST), are included in NIST SRE 2018, only the CMN2 subset is used in this paper. The CMN2 subset is composed of speech spoken in Tunisian Arabic. The performance on the VAST subset is not reported since it exhibits quite different attribute from telephone speech. Different training data and adaptation technologies need to be investigated to achieve good results on the VAST subset [7].

For NIST SREs, the training data consists of 5 Switchboard datasets (Switchboard-2 Phase 1/2/3, Switchboard Cellular Part 1/2) and NIST SRE 2004–2008 telephone

Table 1 The number of speakers, target trials, and impostor trials for the test sets

Test set			# Speakers	# Target trials	# Impostor trials
SRE 2010	core-extended	Male	1906	3465	175873
		Female	2361	3704	233077
	10s-10s	Male	264	262	10570
		Female	266	284	12662
SRE 2016	Tagalog		404	17764	1003568
	Cantonese		398	19298	946098
	Pooled		802	37062	1949666
SRE 2018	CMN2 Dev		125	7830	100265
	CMN2 Eval		940	60991	2033832
VoxCeleb1 test set			40	18860	18860

excerpts. Unlike [25], Mixer 6 is excluded in our experiments since it was used in NIST SRE 2010 test sets. This comprises 64,742 utterances from 6394 speakers, resulting in 5524 h in total. The NIST SRE 2016 and 2018 unlabeled data, representing 52 and 72 h of data, is used for domain adaptation [25]. Two English corpora, the 318-h Switchboard-1 and 1904-h Fisher English, are used to extract phonetic information. Since data augmentation is shown to improve the performance of a speaker verification system [44], it has become a standard pre-processing step. Two noise and reverberation datasets, MUSAN [45] and RIRs [46], are introduced to augment the training data. The augmentation follows the same recipe described in [25].

We also examine our approaches in an independent dataset out of the NIST SREs. The VoxCeleb dataset is extracted from videos in YouTube [19, 28]. In this experiment, the results are evaluated on the VoxCeleb1 *test* set. The training set includes the *dev* portion of VoxCeleb1 and the entire VoxCeleb2, comprising 2780 h of data and 7323 speakers. Although the VoxCeleb1 dataset only contains English data, the VoxCeleb2 dataset consists of speech from speakers of different nationalities, accents, and languages, making the training set multilingual. Different from NIST SREs, the VoxCeleb dataset is sampled at 16 kHz. The 960-h Librispeech [47] is used to introduce phonetic information. This corpus only contains read English speech. The data augmentation is also applied to the training data.

The statistics of all the test sets are shown in Table 1 and Table 2 summarizes the training data usage in the experiments.

5.2 Experimental setup

5.2.1 Baseline x-vector system

The x-vector system used in our experiments follows the standard setup in Kaldi SRE16 V2 recipe [48]. The input features are 23-dim and 30-dim MFCCs for 8 kHz and 16 kHz audio, respectively. The frame-level network is a 5-layer TDNN with the slicing parameter $\{-2,-1,0,1,2\}$, $\{-2,0,2\}$, $\{-3,0,3\}$, $\{0\}$, $\{0\}$ which means the input of each layer is the contextual sliced outputs of the previous layer. For instance, at time t, MFCCs from time $(t-2)$, $(t-1)$, (t), $(t+1)$ and $(t+2)$ are concatenated as the input of the first hidden layer. A statistics pooling layer is then applied followed by 2 fully connected layers. Each hidden layer consists of a linear transform, following by a rectified linear unit (ReLU) activation and a batch-normalization. All the hidden layers have 512 nodes except for the one before the statistics pooling, which has 1500 nodes instead. The output is predicted by a softmax layer and the size is equal to the number of training speakers.

Table 2 Datasets used in the experiments. The adaptation set is only used for NIST SRE 2016 and 2018

Test set	Sampling rate	Phonetic training set	Speaker training set	Adaptation set
SRE 2010 core-extended & 10s-10s condition 5	8kHz	Switchboard-1 Fisher English	Switchboard-2 Phase 1/2/3 Switchboard Cellular Part 1/2 NIST SRE 2004-2008	-
SRE 2016				SRE16 unlabeled data
SRE 2018				SRE18 unlabeled data
VoxCeleb1 test set	16kHz	Librispeech	VoxCeleb1 dev set VoxCeleb2	-

Table 3 Phonetic adaptation results on the male part of the NIST SRE 2010 core-extended condition

	EER(%)	minDCF08	minDCF10
x-vector	1.96	0.0109	0.3900
x-vector-pa ($c = 0$)	1.68	0.0095	0.3585
x-vector-pa ($c = 0.1$)	1.44	0.0084	0.2979
x-vector-pa ($c = 0.2$)	1.62	0.0085	0.3166
x-vector-pa ($c = 0.3$)	1.61	0.0089	0.3290
x-vector-pa ($c = 0.4$)	1.55	0.0089	0.3324
No pre-training	1.82	0.0095	0.3759

Natural gradient for stochastic gradient descent (NG-SGD) [49] is used to train the network. The batch size is 64 and the number of training epochs is 3. The learning rate starts from 0.001 and linearly decreases to 0.0001 at the end of the training. No dropout is applied. This setup follows the same recipe described in [25]. Unless otherwise specified, all the neural networks in this paper are optimized using this setup.

After training, the pre-activation of the first hidden layer at the segment-level network is extracted as the x-vector. Mean normalization is first performed and the dimension of the x-vector is then reduced through LDA. After LDA, the dimension of the x-vector is 150 for NIST SREs and 200 for VoxCeleb. The embedding is unit-length normalized and PLDA scoring is finally applied. For NIST SREs, the SRE 2004–2008 corpora with augmented excerpts are used to train the LDA/PLDA, while for VoxCeleb, the entire training set is used.

To deal with the domain mismatch in NIST SRE 2016 and 2018, an unsupervised PLDA adaptation proposed in Kaldi is applied. Due to domain mismatch, the total covariance estimated in the new domain is different from the covariance indicated in the out-of-domain PLDA. In the experiments, 75% of the excess in-domain covariance is attributed to the within-class covariance of the PLDA model while the remaining 25% is attributed to the between-class covariance. The PLDA parameters are then re-estimated based on the new within- and between-class covariances. The adaptation is performed on the unlabeled data of NIST SRE 2016 and 2018. Refer to the Kaldi source code [1] for more details.

5.2.2 Speaker embedding with phonetic information

In our proposed methods, the x-vector architecture is treated as a speaker-discriminant network and keeps the same setting as the baseline. Phonetic information is explicitly introduced by phonetic-discriminant networks. The senone transcriptions forced aligned by GMM-HMM are used to represent the phonetic contents on Fisher and Switchboard. The number of senones is 3800.

[1] https://github.com/kaldi-asr/kaldi/blob/master/src/ivector/plda.cc

Table 4 Phonetic adaptation results on VoxCeleb

	EER(%)	minDCF08	minDCF10
x-vector	2.68	0.0144	0.4645
x-vector-pa ($c = 0$)	2.52	0.0137	0.4111
x-vector-pa ($c = 0.1$)	2.26	0.0126	0.3651
x-vector-pa ($c = 0.2$)	2.24	0.0132	0.3289
x-vector-pa ($c = 0.3$)	2.32	0.0124	0.3533
x-vector-pa ($c = 0.4$)	2.32	0.0126	0.3531

For phonetic adaptation, a 5-layer TDNN network is trained as the ASR acoustic model. The slicing parameter is $\{-2,-1,0,1,2\}$, $\{-1,0,1\}$, $\{-1,0,1\}$, $\{-3,0,3\}$, $\{-6,-3,0\}$. The 5-th layer is the BN layer containing 128 nodes and other layers have 650 nodes. The activations of the BN layer in the acoustic model are connected to the 5-th layer of the x-vector network. The effect of the fine-tuning learning rate scaling factor c is discussed in this section. When optimizing the phonetic-discriminant networks, the batch size is set to 256, which is a value often used in ASR acoustic model training.

In hybrid multi-task learning, the phonetic-discriminant network uses the same architecture as the x-vector network except for two differences. The statistics pooling layer is excluded in the phonetic-discriminant network and the number of the nodes in the 5-th layer is reduced to 512. Although the learning rates α_1 and α_2 for the two tasks can be different, keeping them equal performs well in our experiments. The results sharing different numbers of layers will be investigated below.

The c-vector combines the phonetic adaptation and multi-task learning networks and uses the same parameters. In the simplified c-vector architecture, the multi-task learning network is used, with some modifications. The number of nodes in the last hidden layer of the phonetic-discriminant network is reduced to 128 and the activations of this layer are fed into the 5-th layer of the x-vector network.

We use *x-vector-pa*, *x-vector-mt*, *c-vector* and *sc-vector* to denote the proposed systems respectively.

The conventional i-vector and DNN-based i-vector are also used in some evaluations for complete comparison. All the models trained in this paper are gender-independent. We analyze the influence of the parameters in our models on the male part of the NIST SRE 2010 core-extended condition and VoxCeleb. Results on SRE 2016 and 2018 are also reported to validate our approaches.

EER, the minimum detection cost function of NIST SRE 2008 (minDCF08) and SRE 2010 (minDCF10) [41] are used as the main performance metrics. The primary cost measures for SRE 2016 and 2018, denoted as minDCF16 and minDCF18, are reported for these two evaluations respectively [42, 43].

Table 5 Results obtained by sharing different numbers of layers in hybrid multi-task learning

	EER(%)	minDCF08	minDCF10
x-vector	1.96	0.0109	0.3900
x-vector-mt (1-layer sharing)	1.67	0.0091	0.3465
x-vector-mt (2-layer sharing)	1.61	0.0082	0.3009
x-vector-mt (3-layer sharing)	1.52	0.0073	0.2752
x-vector-mt (4-layer sharing)	1.50	0.0086	0.3383

The results are given on the male part of the NIST SRE 2010 core-extended condition

The Kaldi toolkit [48] is used to build all the systems in this paper. The code has been released [2].

5.3 Results and discussion

5.3.1 Phonetic adaptation

This section presents the results of phonetic adaptation in speaker embedding training. In order to show the effectiveness of the acoustic model fine-tuning, several systems are trained with different learning rate scaling factors. Table 3 shows that all the systems using phonetic adaptation outperform the x-vector baseline on the male part of the NIST SRE 2010 core-extended condition. Since the phonetic information is introduced to this network, phonetic adaptation without fine-tuning (i.e., $c = 0$) reduces the EER by 14%. The performance can be further improved if we update the appended network during the x-vector training. The best result is obtained when the learning rate scaling factor is 0.1. The small learning rate means that the acoustic model only needs a slight adjustment to achieve good performance. Compared to the baseline, the x-vector using fine-tuned phonetic vectors improves the performance by 26%, 22%, and 23% in EER, minDCF08, and minDCF10, respectively.

Due to the introduction of an additional network, the number of model parameters increases. It is not initially clear whether the improvement comes from the phonetic information or the bigger model. Hence, we train a network combining the x-vector architecture and the acoustic model from scratch where there is no phonetic information considered. This model has the same topology with *x-vector-pa*. As shown in the last row of Table 3, the larger network does improve the performance. But the performance gain is smaller and it still performs worse than our proposed systems. The result validates the importance of the phonetic information extracted from the pre-trained acoustic model.

Table 4 presents the results on VoxCeleb. Phonetic adaptation is also effective in this dataset. Since the acoustic model is pre-trained only using English speech, the extracted phonetic vectors are not a good match with the multi-lingual VoxCeleb training set. Fine-tuning alleviates this mismatch to some extent. Compared with NIST SRE 2010, a higher learning rate scaling factor needs to be used. In this case, 0.2 or 0.3 seems to be a good option. The speaker embedding using phonetic adaptation improves the EER of the baseline by relative 6% and 16%, without and with fine-tuning ($c = 0.2$), respectively.

[2] https://github.com/mycrazycracy/speaker-embedding-with-phonetic-information

Table 6 Results of systems using hybrid multi-task learning with different configurations on VoxCeleb

	EER(%)	minDCF08	minDCF10
x-vector	2.68	0.0144	0.4645
x-vector-mt (1-layer sharing)	2.58	0.0132	0.4027
x-vector-mt (2-layer sharing)	2.73	0.0145	0.3977
x-vector-mt (3-layer sharing)	2.83	0.0151	0.4700
x-vector-mt (4-layer sharing)	2.92	0.0151	0.5001

5.3.2 Multi-task learning

Table 5 gives the performance of several systems sharing different numbers of frame-level layers between the speaker and phonetic-discriminant networks on NIST SRE 2010. From Table 5, we find that although the multi-task learning adds benefit in this condition, sharing more layers does not always improve the performance. The best overall result on our development set is obtained when 3 layers are shared. We decrease the EER from 1.96, when no multi-task learning is used, to 1.52%, when 3 layers are shared, resulting in 22% relative reduction. The minDCF08 and minDCF10 in this configuration also improves by 33% and 29% compared to the baseline.

In contrast to the previous results, the VoxCeleb experiments reported in Table 6 show that, the multi-task learning only improves the results slightly in this dataset. By sharing 1 layer, the system outperforms the baseline by 4%, 8%, and 13% on EER, minDCF08, and minDCF10, respectively. However, the performance degrades rapidly when more layers are shared. We hypothesize that this is due to the language mismatch between Librispeech and

Table 7 Results obtained from different c-vector configurations on the male part of the NIST SRE 2010 core-extended condition

	EER(%)	minDCF08	minDCF10
x-vector	1.96	0.0109	0.3900
$c = 0.1$ + 2-layer sharing	1.12	0.0066	0.2748
$c = 0.1$ + 3-layer sharing	1.21	0.0065	0.2449
$c = 0.2$ + 2-layer sharing	1.18	0.0067	0.3017
$c = 0.2$ + 3-layer sharing	1.24	0.0071	0.2623
$c = 0.3$ + 2-layer sharing	1.29	0.0074	0.2531
$c = 0.3$ + 3-layer sharing	1.46	0.0074	0.2536

Table 8 Results obtained from different c-vector configurations on VoxCeleb

	EER(%)	minDCF08	minDCF10
x-vector	2.68	0.0144	0.4645
$c = 0.2$ + 1-layer sharing	2.18	0.0129	0.2994
$c = 0.2$ + 2-layer sharing	2.32	0.0120	0.3798
$c = 0.3$ + 1-layer sharing	2.33	0.0121	0.3030
$c = 0.3$ + 2-layer sharing	2.24	0.0123	0.3170

the VoxCeleb training set. In this case, sharing more layers cannot provide more useful information for the speaker-discriminant network. On the contrary, with more layers shared, the number of the individual layers in the speaker-discriminant network decreases, making the extraction of speaker characteristics more difficult.

5.3.3 The c-vector

The advantages of phonetic adaptation and multi-task learning are combined in the proposed c-vector. The tunable hyper-parameters now include both the learning rate scaling factor and the shared layers.

We start our analysis on NIST SRE 2010. Based on the above experiments, the learning rate scaling factor ranges from 0.1 to 0.3. There are 2 or 3 layers shared between the speaker and phonetic-discriminant networks. Table 7 shows that the EER and minDCFs on the male part of the core-extended condition can be greatly reduced if proper parameters are selected. For each configuration, the c-vector performs better than the systems using only phonetic adaptation or multi-task learning. Consistent with the former experiments, better performance is obtained using a smaller learning rate scaling factor. Sharing 2 layers results in better EER, while 3-layer sharing is better for minDCF10. The minimum EER (1.12%) is achieved in the second row while the best minDCF08 (0.0065) and minDCF10 (0.2449) are obtained in the third row. To make a trade-off between these operation points, we set $c = 0.1$ and 3-layer sharing in our c-vector.

Table 8 presents the performance of the c-vector approach on VoxCeleb. The best performance is obtained when the learning rate scaling factor is 0.2 and 1 layer is shared, resulting in 19%, 10%, and 36% relative reduction on EER, minDCF08, and minDCF10, respectively.

Table 9 Comparison of different simplified c-vector systems on the male part of the NIST SRE 2010 core-extended condition

	EER(%)	minDCF08	minDCF10
x-vector	1.96	0.0109	0.3900
sc-vector (1-layer sharing)	1.47	0.0085	0.2848
sc-vector (2-layer sharing)	1.34	0.0085	0.3019
sc-vector (3-layer sharing)	1.24	0.0071	0.3035

Table 10 Comparison of different simplified c-vector systems on VoxCeleb

	EER(%)	minDCF08	minDCF10
x-vector	2.68	0.0144	0.4645
sc-vector (1-layer sharing)	2.52	0.0142	0.4333
sc-vector (2-layer sharing)	2.60	0.0143	0.4469

As explained previously, compared with the c-vector used in NIST SRE 2010, the learning rate scaling factor is increased while the number of the shared layers is decreased.

5.3.4 The simplified c-vector

We further evaluate the simplified c-vector (sc-vector) on these two datasets. As shown in Table 9, the sc-vector approach sharing 3 layers achieves the best overall performance on the male part of NIST SRE 2010 core-extended condition. This is consistent with the results observed for the c-vector.

Table 10 shows the results obtained from different sc-vector configurations on VoxCeleb. Unlike the results in Table 9, the sc-vector does not significantly improve the performance on VoxCeleb. As shown in the above experiments, phonetic adaption with fine-tuning is more helpful than the multi-task learning in this dataset, which means the fine-tuning is vital in the language mismatch condition. However, for the sc-vector, the phonetic-discriminant sub-network is only optimized by the out-of-domain phonetic unit classification, which limits the power of the phonetic vectors.

5.3.5 Comparison of systems on NIST SRE 2010 and VoxCeleb

Tables 11 and 12 summarize the results of the x-vector baseline and all our proposed methods with the best system configurations on NIST SRE 2010. Two i-vector systems are also included. The setups of the i-vector systems are the same as that of the Kaldi SRE10 recipe [3].

From the results, we find that when using fine-tuning the speaker embedding with phonetic adaptation achieves better results than the baseline x-vector in almost all conditions. The only exception is the minDCF08 in the female 10 s–10 s condition which is also very close to the baseline. With multi-task learning, the proposed speaker embedding generally improves the performance, except for the minDCF10 in the male 10 s–10 s condition. The relative improvements in the core-extended condition are about 20% and about 10% in the 10 s–10 s condition. From Table 11 and Table 12, it is difficult to conclude which one is better because they each have advantages in different conditions. Overall, the sc-vector is able to

[3] https://github.com/kaldi-asr/kaldi/tree/master/egs/sre10

Table 11 Summary of results obtained with different systems on the male part of the NIST SRE 2010 core-extended and 10s-10s conditions

	Core-extended			10 s–10 s		
	EER(%)	minDCF08	minDCF10	EER(%)	minDCF08	minDCF10
i-vector	2.33	0.0127	0.4132	10.29	0.0521	0.9695
DNN/i-vector	0.89	0.0047	0.1969	7.25	0.0334	0.9160
x-vector	1.96	0.0109	0.3900	7.62	0.0428	0.8321
x-vector-pa ($c = 0$)	1.68	0.0095	0.3585	6.86	0.0406	0.8053
x-vector-pa ($c = 0.1$)	1.44	0.0084	0.2979	6.45	0.0385	0.7366
x-vector-mt (3-layer sharing)	1.52	0.0073	0.2752	6.10	0.0369	0.8808
sc-vector (3-layer sharing)	1.24	0.0071	0.3035	6.80	0.0351	0.7854
c-vector ($c = 0.1$ + 3-layer sharing)	1.21	0.0065	0.2449	6.40	0.0334	0.5878

deliver better performance than previous systems and the c-vector generally performs the best on NIST SRE 2010. Even though the x-vectors with phonetic adaptation and hybrid multi-task learning have reduced the EER and minDCFs compared to the baseline for both genders, the c-vector approach further improves the performance. The only case where the c-vector approach performs worse than the multi-task learning system is with regard to EER on the male part of the 10 s–10 s condition. In contrast, the minDCF10 is significantly better, leading to 33% relative reduction. On the male part of the core-extended condition, the c-vector significantly outperforms the original x-vector by 38%, 40%, and 37% in EER, minDCF08, and minDCF10, respectively. In the 10 s–10 s condition, the improvement on EER is 16% and over 20% based on minDCFs. The performance is similar on the female part. For the sc-vector, the model size is reduced by removing the appended acoustic model. The cost of this is that the phonetic vector fine-tuning is unavailable, and as a result the sc-vector performs worse than the c-vector.

As shown in Tables 11 and 12, the i-vector framework can also benefit from the use of phonetic information on NIST SRE 2010. The i-vector system using DNN-based alignments (*DNN/i-vector*) outperforms the vanilla i-vector, especially when the utterance duration is long. The improvement of incorporating phonetic information in the i-vector system is almost 50% in some conditions. Even so, compared with our proposed c-vector, the DNN-based i-vector only achieves better results on the male part of the core-extended condition and performs much worse in the 10 s–10 s conditions.

Next, we investigate the performance of the different systems on VoxCeleb. The results are shown in Table 13. The i-vector systems are not included due to inferior results. Compared with NIST SRE 2010, language mismatch exists between the speaker and phonetic training sets on VoxCeleb. From Table 13, it is clear that the phonetic adaptation performs better than the multi-task learning and a more aggressive learning rate scaling factor should be applied. We see that the sc-vector fails to outperform the x-vectors with phonetic adaptation and hybrid multi-task learning in this dataset. The likely reason is that the sc-vector cannot utilize the fine-tuning to adapt the acoustic model so that the extracted phonetic vectors are not suitable for the new domain. Actually,

Table 12 Summary of results obtained with different systems on the female part of the NIST SRE 2010 core-extended and 10s-10s conditions

	Core-extended			10 s–10 s		
	EER(%)	minDCF08	minDCF10	EER(%)	minDCF08	minDCF10
i-vector	2.02	0.0110	0.4104	10.56	0.0530	0.9683
DNN/i-vector	1.13	0.0055	0.2369	9.50	0.0415	0.8697
x-vector	1.45	0.0079	0.3234	9.83	0.0431	0.9451
x-vector-pa ($c = 0$)	1.34	0.0072	0.2932	8.45	0.0459	0.8732
x-vector-pa ($c = 0.1$)	1.29	0.0061	0.2768	7.75	0.0434	0.8627
x-vector-mt (3-layer sharing)	1.18	0.0064	0.2423	8.45	0.0399	0.8486
sc-vector (3-layer sharing)	1.21	0.0062	0.2345	6.34	0.0404	0.7746
c-vector ($c = 0.1$ + 3-layer sharing)	0.99	0.0049	0.2189	7.04	0.0401	0.7782

Table 13 Summary of results obtained with different systems on VoxCeleb

	EER(%)	minDCF08	minDCF10
x-vector	2.68	0.0144	0.4645
x-vector-pa ($c = 0$)	2.52	0.0137	0.4111
x-vector-pa ($c = 0.2$)	2.24	0.0132	0.3289
x-vector-mt (1-layer sharing)	2.58	0.0132	0.4027
sc-vector (1-layer sharing)	2.52	0.0142	0.4333
c-vector ($c = 0.2$ + 1-layer sharing)	2.18	0.0129	0.2994

the performance of the sc-vector is similar to the second row in Table 13. It seems that the introduction of the BN layer in the phonetic-discriminant network training has a negative impact, so that the multi-task learning in the sc-vector does not further improve the results of the speaker embedding using phonetic adaptation without any fine-tuning. Again, the c-vector approach performs the best on VoxCeleb.

5.3.6 Results on NIST SRE 2016 and 2018

Only English speech is used in NIST SRE 2010. In this case, the i-vector and speaker embedding systems both achieve better results when including phonetic information. In recent NIST SREs, language and channel mismatch was introduced between the training and test data [42, 43]. Although we have examined the new speaker embeddings in a multi-lingual VoxCeleb dataset, it is still interesting to investigate the performance of our proposed methods in the more challenging NIST SRE 2016 and 2018. According to the experimental results on NIST SRE 2010 and VoxCeleb, in the language mismatched condition, a larger learning rate scaling factor and fewer shared layers should be used for our proposed approaches. All the results reported in this section are obtained by setting the learning rate scaling factor as 0.2 in the phonetic adaptation, and there is 1 layer shared in the multi-task learning.

The results of the two subsets of NIST SRE 2016, Tagalog and Cantonese, are reported in Table 14. From Table 14, we find that due to the severe language mismatch, the DNN-based i-vector system performs worse than the conventional i-vector. The reason for this is that the DNN trained on the English corpus cannot accurately compute the senone posteriors in Tagalog and Cantonese. Table 14 also demonstrates that the speaker-embedding systems outperform both i-vector systems on NIST SRE 2016. The x-vector baseline reduces the EER from 15.73% (i-vecto) to 9.46%.

Unlike the i-vector systems, our proposed methods still benefit from the added phonetic information even in this language mismatched condition. The first observation is that the x-vector using phonetic adaptation outperforms the baseline and the fine-tuning further improves the performance. The hybrid multi-task learning is also beneficial for the speaker embedding. Compared with the phonetic adaptation, hybrid multi-task learning performs worse in the Tagalog subset, while the results are better in the Cantonese subset. The sc-vector achieves similar results to that of the multi-task learning in the Tagalog subset and results in a better EER in Cantonese. The last row in Table 14 shows that the c-vector performs the best. Compared to the conventional x-vector, the c-vector improves the EER and minDCF16 on the pooled set by relative 15% and 10%, respectively.

Table 15 summarizes the results of different systems on the NIST SRE 2018 CMN2 development and evaluation sets. The x-vector using fine-tuned phonetic vectors performs much better than the multi-task learning. Without fine-tuning, the sc-vector does not significantly improve the performance. The c-vector approach performs similarly to the x-vector with fine-tuned phonetic adaptation (the 3rd row in Table 15) on the dataset. This confirms

Table 14 Summary of results obtained with i-vector systems and different speaker embeddings in the Tagalog and Cantonese subsets of NIST SRE 2016

	Tagalog		Cantonese		Pooled	
	EER(%)	minDCF16	EER(%)	minDCF16	EER(%)	minDCF16
i-vector	21.37	0.8901	10.07	0.6564	15.73	0.7861
DNN/i-vector	22.25	0.9059	11.47	0.6950	16.90	0.8127
x-vector	13.60	0.7877	5.33	0.4429	9.46	0.6365
x-vector-pc ($c = 0$)	13.06	0.7794	4.86	0.4207	8.96	0.6214
x-vector-pv ($c = 0.2$)	12.60	0.7593	4.55	0.4127	8.58	0.6045
x-vector-mt (1-layer sharing)	12.82	0.7629	4.36	0.3878	8.59	0.5941
sc-vector (1-layer sharing)	12.92	0.7675	4.00	0.3835	8.44	0.5951
c-vector ($c = 0.2$ + 1-layer sharing)	12.00	0.7451	4.04	0.3629	8.04	0.5692

The pooled results are also demonstrated

Table 15 Summary of results on the CMN2 subsets of NIST SRE 2018. The actDCF18 is also reported on the evaluation set

	Dev		Eval		
	EER(%)	minDCF18	EER(%)	minDCF18	actDCF18
x-vector	8.86	0.587	10.94	0.630	0.631
x-vector-pv ($c = 0$)	8.10	0.556	9.92	0.597	0.604
x-vector-pv ($c = 0.2$)	7.82	0.521	9.62	0.557	0.562
x-vector-mt (1-layer sharing)	8.40	0.554	10.07	0.589	0.593
sc-vector (1-layer sharing)	8.35	0.543	10.06	0.585	0.587
c-vector ($c = 0.2$ + 1-layer sharing)	7.95	0.511	9.57	0.579	0.584

the important role of the fine-tuning in the language mismatched condition. When the c-vector approach is used, the EER and minDCF18 of the baseline is reduced by 13% and 8% on the evaluation set.

The actDCF18 is also reported on the evaluation set of the NIST SRE 2018 CMN2 subset. Logistic regression-based score calibration is used. The calibration parameters are first trained on the development set using the Bosaris toolkit [50] and then applied on the evaluation set. In Table 15, the actDCFs show a similar trend with the minDCFs and the proposed speaker embeddings still perform better than the baseline in actDCF18.

Although the improvements due to applying the phonetic information in these conditions are smaller than those of NIST SRE 2010, these results show the effectiveness and robustness of our proposed approaches when a language mismatch exists.

6 Conclusions

Although phonetic information has been reported to be effective in both the i-vector and frame-level d-vector frameworks, it is rarely used in state-of-the-art speaker embeddings. In this paper, we propose several approaches to overcome the level mismatch problem and introduce frame-level phonetic information into segment-level speaker embedding. The first approach is based on applying phonetic adaptation using phonetic vectors. The phonetic vectors, which are extracted from a fine-tuned ASR acoustic model, are used as auxiliary inputs into the x-vector network. The second approach uses hybrid multi-task learning to exploit the shared information between speaker traits and phonetic content, which improves model generalization. We finally propose a c-vector architecture combining these two approaches, as well as a simplified c-vector which uses phonetic vectors extracted from the phonetic-discriminant network in the multi-task learning approach. On NIST SRE 2010 core-extended and 10 s-10 s condition 5, the proposed speaker embeddings using phonetic adaptation and hybrid multi-task learning significantly outperform the conventional x-vector, with the best performance achieved by our combined c-vector approach. Moreover, the results on the language mismatched NIST SRE 2016, 2018 and VoxCeleb show that the proposed approaches perform well even if different languages are presented. The relationship between the performance and different system configurations have been carefully analyzed across different conditions. These results provide strong support for the benefit of including phonetic information into the speaker embedding-based speaker verification systems.

Abbreviations

ASR: Automatic speech recognition; ASV: Automatic speaker verification; BN: Bottleneck; CMN2: Call my net 2; CNN: Convolutional neural network; DNN: Deep neural network; EER: Equal error rate; GMM: Gaussian mixture model; HMM: Hidden Markov model; LDA: Linear discriminant analysis; LDE: Learnable dictionary encoding; MFCC: Mel-frequency cepstral coefficient; NG-SGD: Natural gradient for stochastic gradient descent; NIST: National Institute of Standards and Technology; PLDA: Probabilistic linear discriminant analysis; ReLU: Rectified linear unit; RNN: Recurrent neural network; SRE: Speaker recognition evaluation; TDNN: Time-delay neural network; VAST: Video annotation for speech technology

Acknowledgements

We would like to thank the editor and anonymous reviewers for their careful work and thoughtful suggestions that help to improve this paper substantially.

Authors' contributions

YL proposed the three methods to introduce phonetic information into speaker embedding and implemented these systems using Kaldi. He also wrote the draft of this paper. LH and YL made many discussions about the details of the system implementation and the paper structure. JL gave advices on the paper. MTJ revised the manuscript and polished the English usage. All authors read and approved the final manuscript.

Funding

This work was supported by the National Natural Science Foundation of China under grant no. 61403224 and no. U1836219.

Availability of data and materials

The Switchboard, Fisher English and all the NIST SRE datasets are available from Linguistic Data Consortium (LDC, https://www.ldc.upenn.edu/).
The VoxCeleb 1&2 are available from http://www.robots.ox.ac.uk/~vgg/data/voxceleb/.
The Librispeech corpus and the noise and reverberation datasets, MUSAN and RIR, can be downloaded from http://www.openslr.org/resources.php.
The source code has been published in https://github.com/mycrazycracy/speaker-embedding-with-phonetic-information.

Competing interests

The authors declare that they have no competing interests.

Author details
[1]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, 100084 Beijing, China. [2]Department of Electrical and Computer Engineering, University of Kentucky, 453 F. Paul Anderson Tower, KY 40506-0046 Lexington, USA.

Received: 4 March 2019 Accepted: 11 November 2019
Published online: 05 December 2019

References

1. D. A. Reynolds, T. F. Quatieri, R. B. Dunn, Speaker verification using adapted gaussian mixture models. Digit. Sig. Process. **10**(1-3), 19–41 (2000)
2. N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification. IEEE Trans. Audio Speech Lang. Process. **19**(4), 788–798 (2011)
3. D. Snyder, D. Garcia-Romero, D. Povey, S. Khudanpur, in *Proc. INTERSPEECH*. Deep neural network embeddings for text-independent speaker verification, (2017), pp. 999–1003. https://doi.org/10.21437/interspeech.2017-620
4. P. Kenny, Joint factor analysis of speaker and session variability: Theory and alogorithms. Technical Report, CRIM-06/08-13 (2008)
5. A. K. Sarkar, D. Matrouf, P. M. Bousquet, J.-F. Bonastre, in *Proc. INTERSPEECH*. Study of the effect of i-vector modeling on short and mismatch utterance duration for speaker verification (International Speech Communications Association, 2012), pp. 2662–2665. https://www.iscaspeech.org/archive/interspeech_2012/i12_2662.html
6. O. Plchot, P. Matejka, A. Silnova, et al., in *Proc. INTERSPEECH*. Analysis and description of ABC submission to NIST SRE 2016, (2017), pp. 1348–1352. https://doi.org/10.21437/interspeech.2017-1498
7. J. Villalba, N. Chen, D. Snyder, et al., in *Proc. INTERSPEECH*. State-of-theart speaker recognition for telephone and video speech: The JHU-MIT submission for NIST SRE18, (2019), pp. 1488–1492. http://dx.doi.org/10.21437/Interspeech.2019-2713
8. G. Hinton, L. Deng, D. Yu, *et al*, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Sig. Process. Mag. **29**(6), 82–97 (2012)
9. H. Ze, A. Senior, M. Schuster, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Statistical parametric speech synthesis using deep neural networks, (2013), pp. 7962–7966. https://doi.org/10.1109/icassp.2013.6639215
10. Y. Xu, J. Du, L.-R. Dai, C.-H. Lee, An experimental study on speech enhancement based on deep neural networks. IEEE Sig. Process. Lett. **21**(1), 65–68 (2014)
11. E. Variani, X. Lei, E. McDermott, I. L. Moreno, J. Gonzalez-Dominguez, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Deep neural networks for small footprint text-dependent speaker verification, (2014), pp. 4052–4056. https://doi.org/10.1109/icassp.2014.6854363
12. L. Li, Y. Chen, Y. Shi, Z. Tang, D. Wang, in *Proc. INTERSPEECH*. Deep speaker feature learning for text-independent speaker verification, (2017), pp. 1542–1546. https://doi.org/10.21437/interspeech.2017-452
13. S.-X. Zhang, Z. Chen, Y. Zhao, J. Li, Y. Gong, in *Proc. IEEE Spoken Language Technology Workshop (SLT)*. End-to-end attention based text-dependent speaker verification, (2016), pp. 171–178. https://doi.org/10.1109/slt.2016.7846261
14. Y. Zhu, T. Ko, D. Snyder, B. Mak, D. Povey, in *Proc. INTERSPEECH*. Self-attentive speaker embeddings for text-independent speaker verification, (2018), pp. 3573–3577. https://doi.org/10.21437/interspeech.2018-1158
15. G. Heigold, I. Moreno, S. Bengio, N. Shazeer, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. End-to-end text-dependent speaker verification, (2016), pp. 5115–5119. https://doi.org/10.1109/icassp.2016.7472652
16. C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, Z. Zhu, Deep speaker: an end-to-end neural speaker embedding system (2017). arXiv preprint arXiv:1705.02304
17. S. Novoselov, O. Kudashev, V. Shchemelinin, I. Kremnev, G. Lavrentyeva, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Deep CNN based feature extractor for text-prompted speaker recognition, (2018), pp. 5334–5338. https://doi.org/10.1109/icassp.2018.8462358
18. N. Li, D. Tuo, D. Su, Z. Li, D. Yu, in *Proc. INTERSPEECH*. Deep discriminative embeddings for duration robust speaker verification, (2018), pp. 2262–2266. https://doi.org/10.21437/interspeech.2018-1769
19. J. S. Chung, A. Nagrani, A. Zisserman, in *Proc. INTERSPEECH*. Voxceleb2: deep speaker recognition, (2018). https://doi.org/10.21437/interspeech.2018-1929
20. W. Cai, Z. Cai, X. Zhang, X. Wang, M. Li, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. A novel learnable dictionary encoding layer for end-to-end language identification, (2018), pp. 5189–5193. https://doi.org/10.1109/icassp.2018.8462025
21. C. Zhang, K. Koishida, in *Proc. INTERSPEECH*. End-to-end text-independent speaker verification with triplet loss on short utterances, (2017), pp. 1487–1491. https://doi.org/10.21437/interspeech.2017-1608
22. S. Yadav, A. Rai, in *Proc. INTERSPEECH*. Learning discriminative features for speaker identification and verification, (2018), pp. 2237–2241. https://doi.org/10.21437/interspeech.2018-1015
23. R. Li, N. Li, D. Tuo, M. Yu, D. Su, D. Yu, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Boundary discriminative large margin cosine loss for text-independent speaker verification, (2019), pp. 6321–6325. https://doi.org/10.1109/icassp.2019.8682749
24. Y. Liu, L. He, J. Liu, Large margin softmax loss for speaker verification (2019). arXiv preprint arXiv:1904.03479
25. D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, S. Khudanpur, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. X-vectors: Robust DNN embeddings for speaker recognition, (2018), pp. 5329–5333. https://doi.org/10.1109/icassp.2018.8461375
26. G. Saon, H. Soltau, D. Nahamoo, M. Picheny, in *IEEE Workshop on Automatic Speech Recognition and Understanding*. Speaker adaptation of neural network acoustic models using i-vectors, (2013), pp. 55–59. https://doi.org/10.1109/asru.2013.6707705
27. Y. Liu, L. He, J. Liu, M. T. Johnson, in *Proc. INTERSPEECH*. Speaker embedding extraction with phonetic information, (2018), pp. 2247–2251. https://doi.org/10.21437/interspeech.2018-1226
28. A. Nagrani, J. S. Chung, A. Zisserman, in *Proc. INTERSPEECH*. Voxceleb: a large-scale speaker identification dataset, (2017), pp. 2616–2620. https://doi.org/10.21437/interspeech.2017-950
29. A. Park, T. J. Hazen, in *Proceedings of the 7th International Conference on Spoken Language Processing*. ASR dependent techniques for speaker identification (International Speech Communication Association, 2002), pp. 1337–1340. https://www.iscaspeech.org/archive/icslp_2002/i02_1337.html
30. B. J. Baker, R. J. Vogt, S. Sridharan, in *Proc. INTERSPEECH*. Gaussian mixture modelling of broad phonetic and syllabic events for text-independent speaker verification (International Speech and Communication Association, 2005), pp. 2429–2432. https://www.iscaspeech.org/archive/interspeech_2005/i05_2429.html
31. Y. Lei, N. Scheffer, L. Ferrer, M. McLaren, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. A novel scheme for speaker recognition using a phonetically-aware deep neural network, (2014), pp. 1695–1699. https://doi.org/10.21236/ada613971
32. Y. Tian, L. He, M. Cai, W.-Q. Zhang, J. Liu, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Deep neural networks based speaker modeling at different levels of phonetic granularity, (2017), pp. 5440–5444. https://doi.org/10.1109/icassp.2017.7953196
33. M. McLaren, Y. Lei, L. Ferrer, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Advances in deep neural network approaches to speaker recognition, (2015), pp. 4814–4818. https://doi.org/10.1109/icassp.2015.7178885
34. L. Li, D. Wang, Y. Chen, Y. Shi, Z. Tang, T. F. Zheng, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Deep factorization for speech signal, (2018), pp. 5094–5098. https://doi.org/10.1109/icassp.2018.8462169
35. M. H. Rahman, I. Himawan, M. Mclaren, C. Fookes, S. Sridharan, in *Proc. INTERSPEECH*. Employing phonetic information in DNN speaker embeddings to improve speaker recognition performance, (2018), pp. 3593–3597. https://doi.org/10.21437/interspeech.2018-1804
36. Z. Tang, L. Li, D. Wang, R. Vipperla, Collaborative joint training with multitask recurrent model for speech and speaker recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **25**(3), 493–504 (2017)
37. G. Pironkov, S. Dupont, T. Dutoit, in *Signal Processing Conference (EUSIPCO)*. Speaker-aware long short-term memory multi-task learning for speech recognition, (2016), pp. 1911–1915. https://doi.org/10.1109/eusipco.2016.7760581

38. Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, K. Yu, Deep feature for text-dependent speaker verification. Speech Communication. **73**, 1–13 (2015). https://doi.org/10.1016/j.specom.2015.07.003
39. D. Garcia-Romero, C. Y. Espy-Wilson, in *Proc. INTERSPEECH*. Analysis of i-vector length normalization in speaker recognition systems (International Speech Communication Association, 2011), pp. 256–259. https://www.isca-speech.org/archive/interspeech_2011/i11_0249.html
40. T. Sercu, C. Puhrsch, B. Kingsbury, Y. LeCun, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Very deep multilingual convolutional neural networks for lvcsr, (2016), pp. 4955–4959. https://doi.org/10.1109/icassp.2016.7472620
41. A. F. Martin, C. S. Greenberg, in *Proc. INTERSPEECH*. The NIST 2010 speaker recognition evaluation (International Speech Communication Association, 2010), pp. 2726–2729. https://www.iscaspeech.org/archive/interspeech_2010/i10_2726.html
42. NIST 2016 Speaker Recognition Evaluation Plan. https://www.nist.gov/system/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf. Accessed 27 Nov 2019
43. NIST 2018 Speaker Recognition Evaluation Plan. https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf. Accessed 27 Nov 2019
44. M. McLaren, D. Castan, M. K. Nandwana, L. Ferrer, E. Yilmaz, in *Speaker Odyssey*. How to train your speaker embeddings extractor (International Speech Communication Association, 2018). https://www.iscaspeech.org/archive/Odyssey_2018/abstracts/34.html
45. D. Snyder, G. Chen, D. Povey, Musan: A music, speech, and noise corpus (2015). arXiv preprint arXiv:1510.08484
46. T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, S. Khudanpur, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. A study on data augmentation of reverberant speech for robust speech recognition, (2017), pp. 5220–5224. https://doi.org/10.1109/icassp.2017.7953152
47. V. Panayotov, G. Chen, D. Povey, S. Khudanpur, in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Librispeech: an ASR corpus based on public domain audio books, (2015), pp. 5206–5210. https://doi.org/10.1109/icassp.2015.7178964
48. D. Povey, A. Ghoshal, G. Boulianne, *et al*, in *IEEE Workshop on Automatic Speech Recognition and Understanding*. The Kaldi speech recognition toolkit (IEEE Signal Processing Society, 2011). http://publications.idiap.ch/index.php/publications/show/2265
49. D. Povey, X. Zhang, S. Khudanpur, Parallel training of deep neural networks with natural gradient and parameter averaging (2014). arXiv preprint arXiv:1410.7455
50. N. Brümmer, E. De Villiers, The bosaris toolkit: theory, algorithms and code for surviving the new dcf (2013). arXiv preprint arXiv:1304.2865

Publisher's Note