A Framework for Adapting DNN Speaker Embedding Across Languages

Weiwei Lin, Man-Wai Mak, Na Li, Dan Su and Dong Yu

***Abstract*—Language mismatch remains a major hindrance to the extensive deployment of speaker verification (SV) systems. Current language adaptation methods in SV mainly rely on linear projection in embedding space; i.e., adaptation is carried out after the speaker embeddings have been created, which underutilizes the powerful representation of deep neural networks. This paper proposes a maximum mean discrepancy (MMD) based framework for adapting deep neural network (DNN) speaker embedding across languages, featuring multi-level domain loss, separate batch normalization, and consistency regularization. We refer to the framework as MSC. We show that (1) minimizing domain discrepancy at both frame- and utterance-levels performs significantly better than at utterance-level alone; (2) separating the source-domain data from the target-domain in batch normalization improves adaptation performance; and (3) data augmentation can be utilized in the unlabelled target-domain through consistency regularization. By combining these findings, we achieve an EER of 8.69% and 7.95% in NIST SRE 2016 and 2018, respectively, which are significantly better than the previously proposed DNN adaptation methods. Our framework also works well with backend adaptation. By combining the proposed framework with backend adaptation, we achieve an 11.8% improvement over the backend adaptation in SRE18. When applying our framework to a 121-layer Densenet, we achieved an EER of 7.81% and 7.02% in NIST SRE 2016 and 2018, respectively.**

Keywords*—*Speaker verification; domain adaptation; data augmentation; maximum mean discrepancy; transfer learning

Code: https://github.com/enmwmak/msc-domain-adaptation

I. Introduction

The success of machine learning relies on the assumption that training data and test data are sampled from the same distribution [1], [2]. In practice, a lot of factors can undermine this assumption. This is especially the case when we want to deploy an existing system to a new environment, where the data have different properties than the training data. For speaker verification, this could happen when the new environment has some specific noise and channel conditions or involves speakers speaking different languages than the training speakers do. Directly using the existing systems in these situations could result in poor performance. Fortunately, it is often possible to collect a small amount of data from the new environment. These data are typically referred to as the target-domain data in the literature [3]. The other data are referred to as the source-domain data. The process of adapting a model to the production environment is referred to as domain adaptation (DA). Depending on whether or not the target-domain data are labeled, DA can be divided into supervised DA and unsupervised DA. As the labeling process is time-consuming and expensive, unsupervised DA is more attractive in real-world applications.

The domain mismatch investigated in this paper is language mismatch. In NIST speaker recognition evaluation 2016 (or SRE16 in short) [4], the language mismatch problem was brought to SV researchers for the first time. The evaluation introduces various new challenges to speaker recognition [4], [5], among which the multilingual setup brought the most attention. Unlike previous SREs, both development (Dev) and evaluation (Eval) data in SRE16 comprise utterances spoken in non-English languages. Table I shows the composition of SRE16 data. Because all of the SRE16 data are non-English, systems trained on previous years' SRE data perform poorly on this evaluation set. Training using only SRE16 data is also not feasible, as there are only 2,472 segments in total and a very small number of them are labelled. Besides, the languages of the labelled development data are different from that of the evaluation data.

Table I. The composition of SRE16 data. "Labelled" means speaker labels are provided. "Unlabelled" means speaker labels are not provided.

Dataset	Category	Language
Dev	Unlabelled	Cantonese and Tagalog
Dev	Unlabelled	Mandarin and Cebuano
Dev	labelled	Mandarin and Cebuano
Eval	Enrollment	Cantonese and Tagalog
Eval	Test	Cantonese and Tagalog

State-of-the-art SV systems are comprised of a deep neural network and a backend model [6]. DA is typically carried out in the backend. In the Kaldi's SRE16 recipe, adaptation is carried out in the PLDA model's mean and covariance matrix. It is very effective and adopted by many researchers [7], [8]. Another very popular DA method for the backend is correlation alignment (CORAL), which essentially whitens the source-domain data and recolors them with a whitening matrix estimated from the target-domain data [9]. In [10], the author proposed a hybrid method combining PLDA model adaptation and CORAL and showed that it is superior to the

Part of this work was done during Weiwei Lin's Internship at Tencent AI Lab. This work was supported by the RGC of Hong Kong SAR, Grant No. PolyU 152137/17E and Tencent AI Lab Rhino-Bird Gift Fund. W.W. Lin and M.W. Mak are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail:weiwei.lin@connect.polyu.hk; enmwmak@polyu.edu.hk). N. Li, D. Su, and D. Yu are with Tencent AI Lab.

The following publication W. Lin, M. -W. Mak, N. Li, D. Su and D. Yu, "A Framework for Adapting DNN Speaker Embedding Across Languages," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2810-2822, 2020 is available at https://doi.org/10.1109/TASLP.2020.3030499.

Table II. SUMMARY OF THE X-VECTOR NETWORK. THE KERNEL IS SPECIFIED AS KERNEL SIZE, STRIDE, AND DILATION.

		X-vector network		Large x-vector network	
l	**Layer**	**Kernel**	**Channel_in × Channel_out**	**Kernel**	**Channel_in × Channel_out**
1	Conv	5,1,1	23 × 512	5,1,1	23 × 1024
2	Conv	3,1,2	512 × 512	3,1,2	1024 × 1024
3	Conv	3,1,3	512 × 512	3,1,3	1024 × 1024
4	Conv	1,1,1	512 × 512	1,1,1	1024 × 1024
5	Conv	1,1,1	512 × 1536	1,1,1	1024 × 2000
–	Statistics pooling	–	1536 × 3072	–	2000 × 4000
6	FC	–	3072 × 512	–	4000 × 512
7	FC	–	512 × 512	–	512 × 512
–	Softmax	–	512 × N	–	512 × N

Table III. SUMMARY OF THE NUMBER OF PARAMETERS AND THE NUMBER OF FLOATING-POINT OPERATIONS IN A FORWARD PASS FOR OUR DENSENET121, THE X-VECTOR NETWORK AND THE LARGE X-VECTOR NETWORK FOR AN INPUT SIZE OF 23 × 400.

Model	# of flops	# of parameters
X-vector network	1000.043M	4.276M
Large x-vector network	3000.705M	11.639M
Densenet80	572.172M	5.256M
Densenet121	958.143M	10.334M

Table IV. DENSENET ARCHITECTURE FOR SPEAKER EMBEDDING. THE GROWTH RATE FOR THE NETWORKS IS 40. NOTE THAT EACH "CONV" LAYER SHOWN IN THE TABLE CORRESPONDS TO THE SEQUENCE BN-RELU-CONV. "CONV 3" DENOTES 1-D CONVOLUTION WITH KERNEL SIZE 3.

Layers	Densenet-80	Densenet-121
Convolution	conv 3	conv 3
Dense Block (1)	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 6$	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 6$
Transition Layer (1)	conv 2 stride 2	conv 2 stride 2
Dense Block (2)	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 10$	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 12$
Transition Layer (2)	conv 2 stride 2	conv 2 stride 2
Dense Block (3)	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 14$	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 24$
Transition Layer (3)	conv 2 stride 2	conv 2 stride 2
Dense Block (4)	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 10$	$\begin{bmatrix}\text{conv 1}\\ \text{conv 3}\end{bmatrix} \times 16$
Stats-pooling Layer	-	-
FC1	492 × 512 Linear	2560 × 512 Linear
FC2	512 × 256 Linear	512 × 256 Linear
Softmax Layer	256 × # of classes	256 × # of classes

individual methods. A more complicated backend adaptation was proposed in [11]–[13]. The authors proposed to use an auto-encoder to minimize the maximum mean discrepancy between the source-domain data and the target-domain data. The method can also address multi-source domain mismatch. In [14], the author proposed to combine a variational auto-encoder (VAE) [15] with a domain adversarial neural network (DANN) for x-vectors domain adaptation. The DANN [16] part aims to retain speaker identity information and learn a feature space that is robust against domain mismatch, while the VAE part is to impose variational regularization on the learned features so that they follow a Gaussian distribution. In [17], the authors extended their variational DANN by incorporating an information maximization criterion [18] into the objective function.

DA also has numerous applications in automatic speech recognition (ASR). For Gaussian mixture model–hidden Markov models (GMM–HMMs), well known effective adaptation techniques include maximum-likelihood linear regression (MLLR) [19]–[21], constrained MLLR (cMLLR)] [22], and vector Taylor series (VTS) expansion [23]. Many of these techniques can be used for tackling environment and speaker mismatches. There are also numerous techniques for adapting DNNs for ASR, which can be divided into three categories: linear transformation, conservative training, and subspace methods. A detailed discussion of DNN adaptation methods for ASR can be found in [24].

More recently, domain adversary learning and variational auto-encoders have been explored for ASR adaptation. For instance, the authors in [25] framed robust speech recognition as a domain adaptation problem. They used DA to suppress the training–test mismatch in real-world speech recognition. Specifically, the DNNs were trained with two discriminative tasks (main and auxiliary tasks). The main task aims to predict phoneme labels and the auxiliary task is to discriminate between the source-domain and target-domain data. The networks were trained to minimize the phoneme classification loss while maximizing the domain confusion. The authors in [26] trained variational auto-encoders on both source- and target-domain data to learn latent representations. The representations of source-domain data are modified such that they match the target domain without changing the attributes related to the recognition task. The modified source-domain data were used to train a speech recognizer.

DNN adaptation is relatively new in SV. Because DNN provides a larger parameter space to explore, it is potentially more powerful than backend adaptation. Most of the DA methods aim to minimize the divergence between the source-domain data and the target-domain data. In the context of speaker verification, this means minimizing the discrepancy between the source-domain speaker embeddings and the target-domain speaker embeddings. In [27], the authors proposed to use adversarial learning to adapt the speaker embeddings. Specifically, Wasserstein GANs [28] were used to minimize the discrepancy between the source-domain and the target-domain speaker embeddings. The authors also explored using other information such as language labels and phone numbers and found that they are beneficial. However, their method requires speaker labels to perform well, which limits the method's applicability. In [29], several GAN variants were proposed. Both adaptation and verification were carried out end-to-end. However, the performance of the system is not as good as the x-vector system with PLDA adaptation in Kaldi.

Most of the previous works in DNN adaptation focus on utterances-level adaptation; i.e., the domain divergence is estimated from the speaker embeddings. Although backpropagation could tune the parameters of the whole network to minimize the domain divergence, we argue that it is still better to explicitly adapt frame-level features. The reason is that training data typically only has a fixed duration range. As a result, adapting utterance-level features may only benefit a specific duration range. Adapting frame-level features, on the other hand, are less prone to fit a specific duration range. Another innovative aspect of our method is making use of unlabelled target-domain data. Data augmentation is an important part of DNN-based SV. However, the augmentation is done only for the labeled data. It was unclear how to apply data augmentation to unlabelled data. To leverage the unlabelled data, we propose to add a consistency regularization that minimizes the divergence between augmented and clean target-domain data. In this way, the consistency regularization makes the target speaker embeddings robust to adverse perturbations.

This paper is an extension of our work in [13]. Specifically, we extended our work in the following regards. First, we added auxiliary batch normalization to each layer of the neural networks. This is, to the best of our knowledge, the first use of separate batch normalization units in domain adaptation. Second, we conducted detailed experiments investigating the effect of target-domain utterance duration on DA. The experiments show that utterance-level adaptation tends to overfit a specific duration range, which validates the assumption of our multi-level adaptation. To the best of our knowledge, the duration mismatch problem in DA has not been investigated in the previous research. Third, we conducted a detailed investigation of the effect of kernels in MMD-based adaptation. In summary, this paper investigates the use of **m**ultilevel domain loss, **s**eparate batch normalization, and **c**onsistency regularization for speaker embeddings. We abbreviate the proposed method as MSC.

II. Deep Neural Networks for Speaker Embedding

A. X-vector Architecture

The x-vector network consists of three parts: Frame-level time-delay neural networks (TDNNs), utterance-level fully-connected (FC) layers, and a statistics pooling layer that bridges the frame-level layers and utterance-level layers [6], [30]. A TDNN is a special form of convolutional neural networks (CNNs). It skips the computation at chosen temporal positions while maintaining the same receptive field size as a CNN. A statistics pooling layer concatenates the mean and standard deviation of the activations from the last convolutional layer. The concatenated means and standard deviations are passed to two FC layers. The network is trained to minimize the standard cross-entropy loss. The network is trained using small chunks of acoustic sequences derived from the clean and augmented utterances. The typical chunk length ranges from 200 ms to 400 ms. After the network is trained, the embedding of each utterance is extracted from the first affine layer after the statistics pooling layer. A backend consisting of LDA and PLDA models is trained using the embeddings as input [31].

B. Densenet Architecture for Speaker Embedding

DenseNets are proposed in [32] for computer vision. A Densenet comprises two types of blocks, namely, dense block and transition block. In a dense block, each layer is connected by all of the outputs from the previous layers. To prevent the number of feature maps from growing excessively, a transition block is introduced to reduce the feature map size. Suppose each convolutional layer produces k feature maps, then the l-th layer inside the block has $k_0 + k \times (l-1)$ feature maps, where k_0 is the number of channels in the input layer. The parameter k is referred to as the growth rate. In this work, we use a dense network composed of 1-dimensional convolution. We used the same statistics pooling layer as that of the x-vector network. Because max-pooling and average pooling do not work well in speaker recognition, we replaced the max-pooling by stride-2 convolution layers. Table IV shows our network architecture. Table III summarizes the number of parameters and the number of floating-point operations in a forward pass for our Densenet80 and Densenet121 for an input size of 23×400. We also compared our DenseNets with a large x-vector network in which the numbers of channels per layer are doubled. The the number of floating-point operations and model size of the x-vector networks are presented in Table III.

III. PLDA and Backend Adaptation

A. Probabilistic Linear Discriminant Analysis

Probabilistic linear discriminant analysis (PLDA) [31] has been a popular backend for x-vector systems. Given a set of D-dimensional length-normalized [33] DNN embedding vectors $\{\mathbf{x}_{ij}; i = 1,\dots,N; i = 1,\dots,H_i;\}$ from N speakers, each with H_i sessions, PLDA assumes that the embedding vectors can be expressed as the following factor analysis model:

$$\mathbf{x}_{ij} = \mathbf{m} + \mathbf{V}\mathbf{z}_i + \boldsymbol{\epsilon}_{ij}, \tag{1}$$

where $\mathbf{m}$ is the global mean of the embedding, $\mathbf{V}$ defines the speaker subspace, $\mathbf{z}_i$ is the speaker factor and $\boldsymbol{\epsilon}_{ij}$ is the residue whose covariance matrix represents non-speaker variability.

B. Correlation Alignment for Backend Adaptation

Correlation alignment (CORAL) is a popular domain adaptation technique and has been very successful in SV [10], [34]. It was introduced to SV in [34]. CORAL aims to minimize the distance between the second-order statistics (covariance) of the source features $\mathbf{C}_S$ and target features $\mathbf{C}_T$. When using with speaker embeddings, CORAL has the advantage of fast adaptation without re-training the whole network for a new domain. CORAL aims to find a transformation matrix $\mathbf{A}$ that minimizes the distance:

$$\left\| \mathbf{A}^\top \mathbf{C}_S \mathbf{A} - \mathbf{C}_T \right\|_F^2. \tag{2}$$

IV. Maximum Mean Discrepancy

Maximum mean discrepancy is a distance measure on the space of probability [35]. Given two sets of samples $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^M$ from distributions P_x and P_y, MMD measures the similarity of P_x and P_y by computing the mean squared difference of the statistics of the samples:

$$\mathcal{D}(\mathcal{X}, \mathcal{Y}) = \left\| \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i) - \frac{1}{M} \sum_{j=1}^{M} \phi(\mathbf{y}_j) \right\|^2 . \tag{3}$$

When $\phi(.)$ is the identity function, the MMD computes the mean squared distance between the sample sets. Eq. 3 can be expanded as:

$$\begin{aligned} \mathcal{D}(\mathcal{X}, \mathcal{Y}) &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \phi(\mathbf{x}_i)^\mathsf{T} \phi(\mathbf{x}_{i'}) \\ &- \frac{2}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi(\mathbf{x}_i)^\mathsf{T} \phi(\mathbf{y}_j) + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} \phi(\mathbf{y}_j)^\mathsf{T} \phi(\mathbf{y}_{j'}). \end{aligned} \tag{4}$$

The dot product terms can be replaced by kernel functions $k(\cdot, \cdot)$:

$$\begin{aligned} \mathcal{D}(\mathcal{X}, \mathcal{Y}) &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} k(\mathbf{x}_i, \mathbf{x}_{i'}) \\ &- \frac{2}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} k(\mathbf{x}_i, \mathbf{y}_j) + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} k(\mathbf{y}_j, \mathbf{y}_{j'}). \end{aligned} \tag{5}$$

In the case of quadratic (Quad) kernels, we have:

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\mathsf{T} \mathbf{y} + c)^2. \tag{6}$$

The MMD becomes:

$$\begin{aligned} \mathcal{D}(\mathcal{X}, \mathcal{Y}) = c \left\| \frac{1}{N} \sum_i \mathbf{x}_i - \frac{1}{M} \sum_j \mathbf{y}_j \right\|_F + \\ \left\| \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i^\mathsf{T} - \frac{1}{M} \sum_j \mathbf{y}_j \mathbf{y}_j^\mathsf{T} \right\|_F , \end{aligned} \tag{7}$$

where $\|.\|_F$ represents the Frobenius norm. We can see that with a second-degree polynomial kernel, MMD can match up to the second-order statistics and c can be adjusted to control the contribution of the first and the second moments. If we set c to 0, Eq. 7 becomes

$$\mathcal{D}_{\text{MMD}} = \left\| \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^\mathsf{T} - \frac{1}{M} \sum_{j=1}^{M} \mathbf{y}_j \mathbf{y}_j^\mathsf{T} \right\|_F^2 . \tag{8}$$

In deep CORAL [36], the domain discrepancy is defined as:

$$\ell_{CORAL} = \frac{1}{4d^2} \left\| \mathbf{C}_S - \mathbf{C}_T \right\|_F^2 , \tag{9}$$

where d is the dimension of the deep-layer's activations, $\mathbf{C}$ is a covariance matrix, and the subscripts S and T stand for the source-domain and target-domain, respectively. Eq. 9 can also be written as:

$$\begin{aligned} \ell_{CORAL} = \frac{1}{4d^2} \Bigg\| \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T} \\ - \frac{1}{M} \sum_{j=1}^{M} (\mathbf{y}_j - \bar{\mathbf{y}})(\mathbf{y}_j - \bar{\mathbf{y}})^\mathsf{T} \Bigg\|_F^2 , \end{aligned} \tag{10}$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are the means of source-domain data $\{\mathbf{x}\}_{i=1}^N$ and target-domain data $\{\mathbf{y}\}_{j=1}^M$, respectively. Besides the scaling constant, the difference between Eq. 8 and Eq. 10 is that Eq. 8 uses the difference between the second-moments while Eq. 10 uses the difference between the centered second moments (covariances).

Another popular kernel is the radial basis function (RBF) kernel:

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2} \left\| \mathbf{x} - \mathbf{y} \right\|^2 \right) , \tag{11}$$

where σ is the width parameter. With the RBF kernel, the feature space is of infinite dimension and contains all moments of data. Minimizing MMD using the RBF kernel is equivalent to matching all moments of two distributions [37]. It is also possible to use a mixture of RBF kernels [38]:

$$k(\mathbf{x}, \mathbf{y}) = \sum_{q=1}^{K} \exp\left(-\frac{1}{2\sigma_q^2} \left\| \mathbf{x} - \mathbf{y} \right\|^2 \right) , \tag{12}$$

where σ_q is the width parameter of the q-th RBF kernel.

Another popular measure for the discrepancy between probability distributions is Kullback–Leibler (KL) divergence. Both MMD and KL divergence measure the distance between two distributions $p(\mathbf{x})$ and $q(\mathbf{x})$. The main difference is that KL divergence requires parametric forms for $p(\mathbf{x})$ and $q(\mathbf{x})$, while MMD only requires the samples from $p(\mathbf{x})$ and $q(\mathbf{x})$. For example, assume that we have two sets of samples from two Gaussian distributions. For KL divergence, we must estimate the parameters of the two distributions (e.g., the means and covariance matrices for Gaussian distributions) and then compute the KL divergence using the estimated parameters. For MMD, we can directly compute the distance between the two distributions using sampled data. In domain adaptation, it is not easy to find parametric distributions that are significantly expressive for the target- and source-domain data. MMD solves this problem nicely because of its non-parametric nature. More precisely, the distance between any distributions can be estimated without finding any parameters first. Therefore, MMD is a better choice than KL divergence for domain adaptation.

V. The Proposed MSC Domain Adaptation Framework

A. Multi-level Adaptation

Assume that we have a labelled dataset $\{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^I$ from the source-domain. Denote $\Theta = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^L$ as the set of

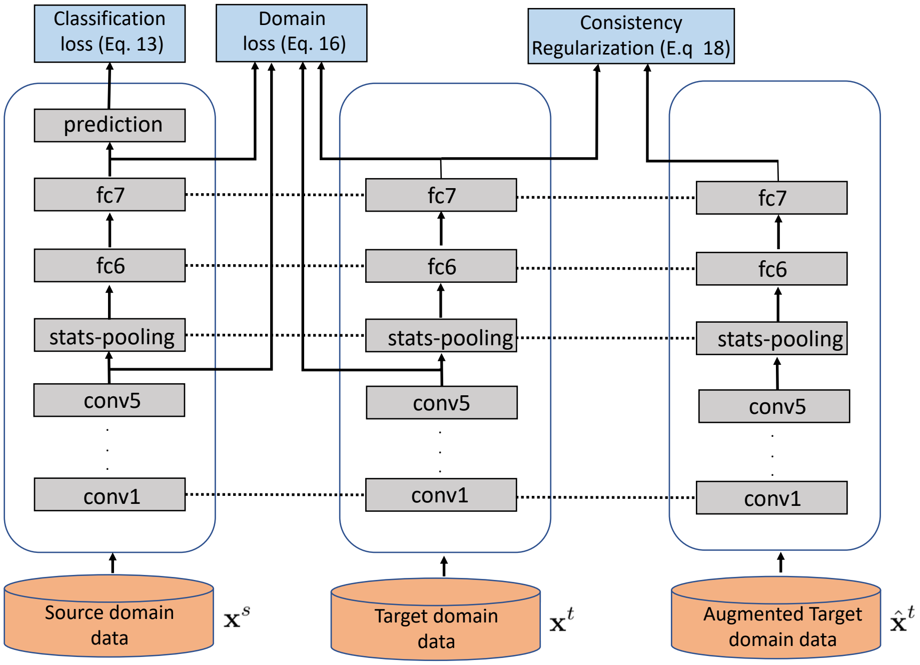

Figure 1. The architecture of our proposed framework. The network is trained to minimize the classification loss and the domain loss with consistency regularization (see Eq. 18). For target-domain data, no label is required. The dotted lines indicate weight-sharing within individual layers.

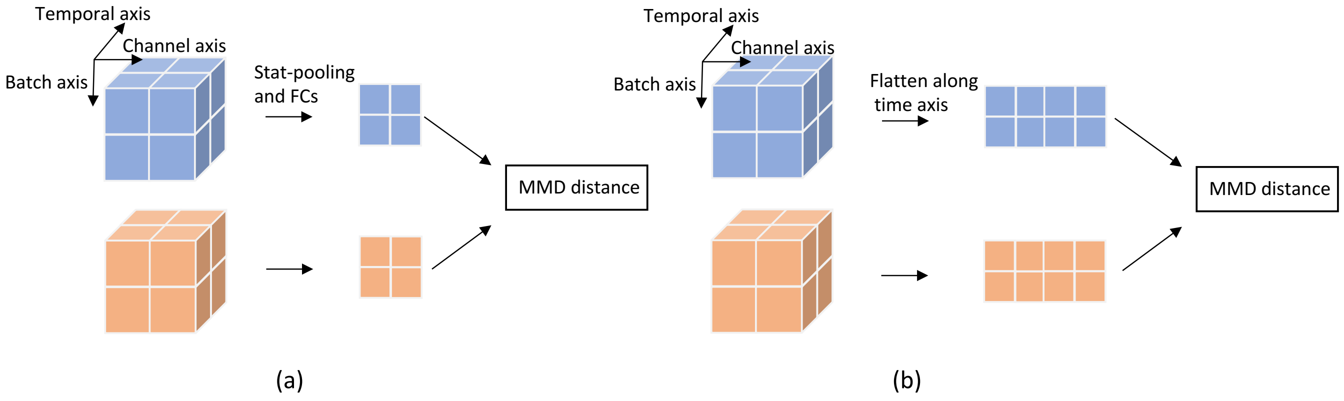

Figure 2. Diagrams demonstrate the difference between (a) utterance-level MMD distance $\mathcal{D}(\mathcal{H}_s^7, \mathcal{H}_t^7))$ and (b) frame-level MMD distance $\mathcal{D}(\text{FLAT}(\mathcal{H}_s^5), \text{FLAT}(\mathcal{H}_t^5)))$. Blue and orange cubes represent a batch of 3-dimensional data from two domains. In the case of utterance-level MMD distance, it is computed after the aggregation along the temporal axis in the stats-pooling layer. In the case of frame-level MMD distance, the temporal axis is flattened to transform a 3D array into a 2D array for computing the MMD distance.

network parameters. The network parameters can be found by solving the following optimization problem:

$$\min_{\Theta} \frac{1}{I} \sum_{i=1}^{I} J(p_{\Theta}(y|\mathbf{x}_i^s), y_i^s), \tag{13}$$

where J is the cross-entropy loss and $p_{\Theta}(y|\mathbf{x}_i^s)$ is the conditional probability that the network assigns $\mathbf{x}_i^s$ to class y_i^s. Minimizing this objective alone will not guarantee the generalization to the target-domain. To make generalization to the target-domain possible, we need to reduce the divergence between the marginal distribution of the source-domain and the target-domain. In neural networks, we typically reduce the divergence in the hidden activations. Let $\mathcal{H}_*^l = \{\mathbf{h}_i^l\}$ denotes the l-th layer hidden activations for source or target data. The cross-entropy loss together with MMD distance is

$$\min_{\Theta} \frac{1}{I} \sum_{i=1}^{I} J(p_{\Theta}(y|\mathbf{x}_i^s), y_i^s) + \lambda \cdot \mathcal{D}(\mathcal{H}_s^l, \mathcal{H}_t^l), \tag{14}$$

where λ is a constant controlling the trade-off between the two objectives. For how to compute the gradient of MMD loss, reader may refer to the Appendix.

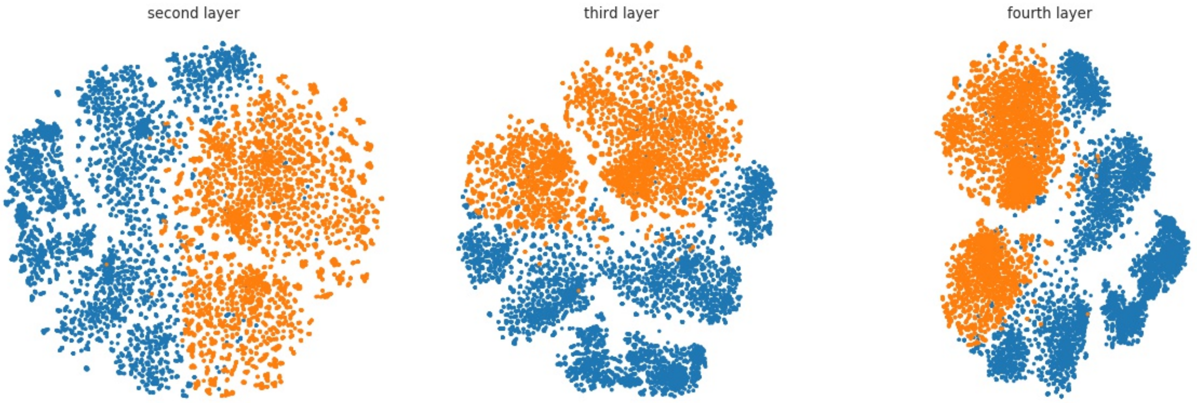

Figure 3. T-SNE plots of the hidden activations at the convolutional layers of the x-vector network in Table II. The orange dots correspond to the data from the source-domain (SRE04–SRE10). The blue dots correspond to the data from the target domain (SRE18 evaluation set.).

As mentioned in [38], deeper layers typically have larger domain discrepancy gaps. Therefore, it is very common to minimize the divergence at the network's last layer. In the x-vector network in Table II, it is the 7-th layer (i.e., $l = 7$). However, the current DNN training scheme typically uses very short speech segments (200 frames to 400 frames) for training and relies on the backend to compensate for the duration discrepancy. The embedding distribution shifts with speech duration, which results in an inaccurate distribution divergence estimate. Adapting frame-level activations, on the other hand, has no such problem. Therefore, we argue that it is important to adapt frame-level features as well. Here, we choose the last convolutional layer before statistics pooling, i.e., $l = 5$ in Table II, for adaptation. However, the MMD distance in Eq. 3 only works with vectors. Frame-level activations are represented by tensor with dimension $B \times T \times C$, where B is the batch size, T is the number of frames in the segment, and C is the number of channels (filters). We propose to flatten along the temporal axis to convert the frame-level activations of speech segments into a batch of 1D vectors and then compute the MMD distance between the two batches of flattened vectors:

$$\mathcal{D}(\text{FLAT}(\mathcal{H}_s^5), \text{FLAT}(\mathcal{H}_t^5))), \qquad (15)$$

where FLAT(.) denotes flattening a 3D tensor along the temporal axis. Figure 2(b) illustrates the flattening procedure for frame-level activations. Minimizing Eq. 15 can reduce domain mismatch even though utterances from the source- and target-domain are of different contexts. This is because the filters (feature detectors) in the lower CNN layers can extract the relevant features irrespective of the location of the features. The total domain loss is:

$$\lambda \cdot \mathcal{D}(\mathcal{H}_s^7, \mathcal{H}_t^7) + \alpha \cdot \mathcal{D}(\mathcal{H}_s^5, \mathcal{H}_t^5)), \qquad (16)$$

where α is a constant controlling the importance of Eq.15.

B. Consistency Regularization Using MMD

Data augmentation is the most important part of x-vector's success. However, how to use data augmentation on unlabelled data has not been explored in SV. Consistency training has been successfully explored in semi-supervised learning [39]. The idea is to enforce or regularize a network such that the network predictions are consistent even if the network's input is subject to noise perturbation for unlabelled data. In [39], the regularization is achieved by minimizing the following KL divergence:

$$\mathop{\mathbb{E}}_{q(\hat{\mathbf{x}}_{\text{unlab}}|\mathbf{x}_{\text{unlab}})}[\text{KL}(p_\Theta(y|\mathbf{x}_{\text{unlab}})||p_\Theta(y|\hat{\mathbf{x}}_{\text{unlab}}))], \qquad (17)$$

where KL(.,.) is the KL divergence, $\mathbf{x}_{\text{unlab}}$ denotes the clean unlabelled data, $\hat{\mathbf{x}}_{\text{unlab}}$ denotes the augmented unlabelled data, and $q(\hat{\mathbf{x}}_{\text{unlab}}|\mathbf{x}_{\text{unlab}})$ is a data augmentation transformation. For Eq. 17 to work, $p_\Theta(y|\mathbf{x}_{\text{unlab}})$ has to be very close to the true class distribution. This is typically achieved by training the network on a large number of labelled data point $\{\mathbf{x}_{\text{label}}, y\}$, where $\mathbf{x}_{\text{label}}$ denotes the labelled data. As a result, the network is discriminative for the class y and $p_\Theta(y|\mathbf{x}_{\text{unlab}})$ is very close to the true class distribution. However, if we do not have labelled data $\{\mathbf{x}_{\text{label}}, y\}$ to train the network, the softmax output $p_\Theta(y|\mathbf{x}_{\text{unlab}})$ is less useful.

We propose another form of consistency penalty that does not require labeled data to work. Instead of minimizing the KL divergence between the softmax output of the clean unlabelled data and augmented unlabelled data, we propose minimizing the discrepancy between the embeddings produced by the clean data and the embeddings produced by the augmented data. The motivation is that DNN embedding should be robust to input perturbation. After all, the goal of the DNN is to create speaker embedding instead of prediction. We use MMD to measure the consistency between the two sets of embeddings. Let $\mathcal{H}_t^7$ and $\hat{\mathcal{H}}_t^7$ denote the set of Layer-7's hidden activations obtained from the clean and the augmented target-domain data, respectively. The consistency regularization using MMD is

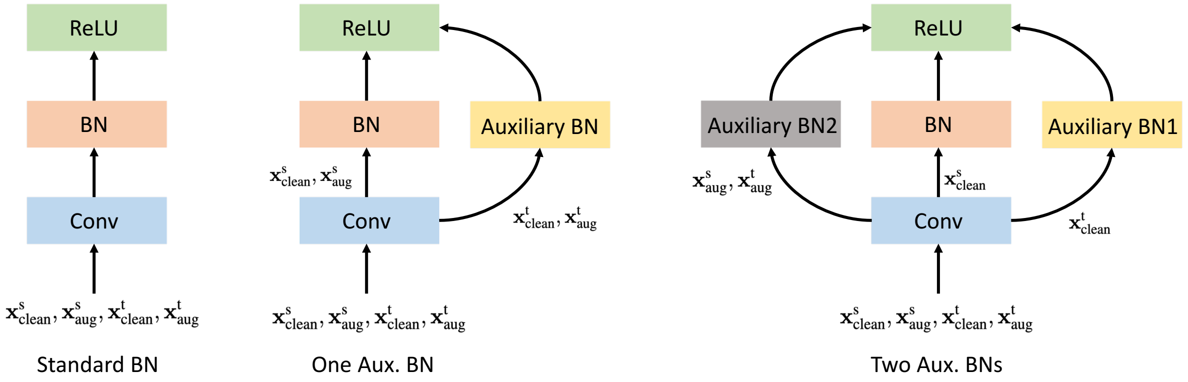
ReLU
BN
Conv
$\mathbf{x}^{\text{s}}_{\text{clean}}, \mathbf{x}^{\text{s}}_{\text{aug}}, \mathbf{x}^{\text{t}}_{\text{clean}}, \mathbf{x}^{\text{t}}_{\text{aug}}$
Standard BN
ReLU
BN
Auxiliary BN
Conv
$\mathbf{x}^{\text{s}}_{\text{clean}}, \mathbf{x}^{\text{s}}_{\text{aug}}$
$\mathbf{x}^{\text{t}}_{\text{clean}}, \mathbf{x}^{\text{t}}_{\text{aug}}$
$\mathbf{x}^{\text{s}}_{\text{clean}}, \mathbf{x}^{\text{s}}_{\text{aug}}, \mathbf{x}^{\text{t}}_{\text{clean}}, \mathbf{x}^{\text{t}}_{\text{aug}}$
One Aux. BN
ReLU
Auxiliary BN2
BN
Auxiliary BN1
$\mathbf{x}^{\text{s}}_{\text{aug}}, \mathbf{x}^{\text{t}}_{\text{aug}}$
$\mathbf{x}^{\text{s}}_{\text{clean}}$
Conv
$\mathbf{x}^{\text{t}}_{\text{clean}}$
$\mathbf{x}^{\text{s}}_{\text{clean}}, \mathbf{x}^{\text{s}}_{\text{aug}}, \mathbf{x}^{\text{t}}_{\text{clean}}, \mathbf{x}^{\text{t}}_{\text{aug}}$
Two Aux. BNs

Figure 4. “One Aux. BN” means that the batch statistics of the source- and target-domains are computed separately. In the case of “Two Aux. BNs”, we further divided the mini-batch into clean source-domain data, clean target-domain data, and augmented data when computing the mini-batch statistics. The superscripts “s” and “t” stand for source- and target-domains, respectively. The subscripts “clean“ and “aug” stand for clean original data and augmented data, respectively.

written as:[1]

$$\begin{aligned}\mathcal{D}(\mathcal{H}_t^7, \hat{\mathcal{H}}_t^7) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} k(\mathbf{h}_i^7, \mathbf{h}_{i'}^7) \\ &- \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} k(\mathbf{h}_i^7, \hat{\mathbf{h}}_j^7) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M} k(\hat{\mathbf{h}}_j^7, \hat{\mathbf{h}}_{j'}^7).\end{aligned} \tag{18}$$

Combining Eq. 18 with Eq. 13 and Eq. 16, we have the total loss function:

$$\begin{aligned}\min_{\Theta} \frac{1}{I}\sum_{i=1}^{I} & J(p_\Theta(y|\mathbf{x}_i^s), y_i^s) + \lambda \cdot \mathcal{D}(\mathcal{H}_s^7, \mathcal{H}_t^7) \\ & + \alpha \cdot \mathcal{D}(\mathcal{H}_s^5, \mathcal{H}_t^5)) + \beta \cdot \mathcal{D}(\mathcal{H}_t^7, \hat{\mathcal{H}}_t^7).\end{aligned} \tag{19}$$

The first term of Eq. 19 is the standard cross-entropy using speaker labels as supervised signals to minimize classification loss. The second and third terms are MMD distances using domain labels as supervised signals to minimize the mismatch between the source- and the target-domains ($\mathcal{H}_s^l$ and $\mathcal{H}_s^l$ represent the hidden activations from the source-domain data and the target-domain data, respectively). The last term is also an MMD distance but using augmentation information as supervised signals to enforce consistency between the origin and the augmented data ($\mathcal{H}_t^7$ and $\hat{\mathcal{H}}_t^7$). Figure 1 summarizes the architecture and objective functions.

C. Auxiliary BN

Batch normalization (BN) is an essential part of modern deep neural networks. The input features are normalized using the mean and variance computed from the mini-batch. The underlying assumption is that the input data are homogenous. However, if data are heterogeneous, the statistics used by BN is inaccurate. In domain adaptation, it is obvious that the source-domain data and the target-domain data come from two different distributions, as exemplified by the t-SNE plots in Figure 3. Thus we argue that we need two separate BNs to obtain more accurate batch statistics. A similar idea is proposed in [40], where the authors used two different BNs for clean and adversary inputs. The heterogeneity could also come from data augmentation [40]. Thus, we could also use three BNs per layer, i.e., one for the source-domain data, one for the target-domain data, and one for the augmentation data. Figure 4 illustrates the idea of auxiliary BN.

VI. Experiments

A. Data Preparation

The training data include NIST SRE 2004–2010 (SRE04–10 in short) and all of the Switchboard data. We follow the data augmentation strategy in the Kaldi SRE16 receipt. The training data were augmented by adding noise, music, reverb, and babble to the original speech files in the datasets. After filtering out utterances shorter than 500 frames and speakers with less than 8 utterances, we are left with 4,808 speakers. 23-dimensional Mel-frequency cepstral coefficients (MFCC) were computed from 8kHz speech files. Mean normalization was applied to the MFCC using a 3-second sliding window. Non-speech frames were removed using Kaldi's energy-based voice activity detector.

B. DNN and Backend Training

The hyper-parameters λ, β and α in Eq. 19 were all set to 1. For the models with vanilla BN and one auxiliary-BN, we sampled 32 speech segments from the source-domain and the target-domain, respectively. Therefore, the size of a mini-batch is 64. For the models with two auxiliary-BNs, we sampled 21 speech segments from the source-domain, the clean target-domain, and the augmented target-domain, respectively. As a result, the size of a mini-batch is 63. All samplings are with replacement. All DNNs were optimized by the Adam optimizer [41] with a learning rate of 0.001. The networks were implemented in PyTorch [42]. We used a standard backend comprised of LDA, length-normalization, and PLDA. Both LDA and PLDA were trained using embeddings extracted from

[1] To simplify notation, the subscript t is omitted in the right side of Eq. 18.

full-length utterances. We used correlation alignment [9] for domain adaptation in the PLDA backend. For the models with domain-dependent BNs, the embeddings were extracted using the BN that was trained on the same domain data.

C. Data Augmentation

In addition to data augmentation in [6], we also used speech conversion tools in FFmpeg. Every speech file was accompanied by one of the following augmentations:

- Reverberation: Speech files were reverberated through convolution with simulated RIRs [43].
- Music: A randomly selected music file from MUSAN was added to the original speech with SNR ranges from 5–15dB.
- Noise: Noise from MUSAN was added to the recording intermittently with SNR between 0dB and 10dB.
- Babble: Babble noise was added to the original speech files with an SNR of 0dB–10dB.
- Speed: the original speech was speeded up by 1.3 times using FFMpeg.[2]

Both the target-domain data and the source-domain data were augmented.

D. Evaluation

All systems were evaluated on the evaluation set of SRE 2016 and 2018. The SRE16 evaluation set is composed of Tagalog and Cantonese telephone conversations. For SRE18, we only used the CMN2 portion, which consists of Tunisian Arabic conversations. We report results in terms of equal error rate (EER) and minimum cost function (DCF). We used minDCF as defined in [4]. The minDCF reported in this paper is the average of the minDCFs with p-target set to 0.01 and 0.005.

VII. Results

A. Comparison with DNN Adaptations and Backend Adaptations

In this section, we compared the proposed framework (MSC) with the previously proposed DNN adaptation methods. The latter includes Wasserstein GAN (WGAN) adaptation, supervised WGAN adaptation in [27] and least square GAN (LSGAN) in [29]. The results and implementation details are presented in Table V. All the results in Table V are without additional backend adaptation. It is clear from the table that MSC performs significantly better than the existing methods. It is worth noting that MSC even performs better than the supervised adaptation in [27].

We also compared the proposed framework with two popular backend adaptation methods, namely, CORAL [9] and Kaldi's PLDA adaptation [6]. The potential for combining the proposed framework with backend adaptation was also investigated. The results are presented in Table VI. As can be seen from Table VI, in SRE16, CORAL is the most effective adaptation method. However, in SRE18, MSC has a clear advantage over backend adaptation methods. This, we believe, is due to the fact that SRE18 has more data for adaptation (over 4,000 utterances compared to 2,340 in SRE16). Besides, it seems that the proposed framework works well with backend adaptation methods, as combining them improves performance. More results for SRE16 subsets is presented in Table VII.

B. Ablation Study of Individual Components

To investigate the effect of individual components in our framework, we conducted an ablation study. Table VIII starts with only utterance-level adaptation (Utt. Adapt.) in the second row and incrementally adds frame-level adaptation (Frame Adapt.), consistency regularization (Consist. Reg.), and auxiliary BN (Aux. BN). Here, in the unadapted baseline, we did not use any in-domain data. A lot of papers, including the Kaldi SRE16 recipe, reported the unadapted baselines that use the target-domain data for centering the mean of the PLDA model.

Table VIII shows that utterance-level adaptation alone already gives a great improvement over no adaptation. Adding frame-level adaptation gives a considerable performance gain for both SRE16 and SRE18. Consistency regularization also further improves the performance in both SRE16 and SRE18. The last three rows of Table VIII show the results of the proposed method with three kinds of batch normalization schemes. The column "No. of Aux. BN" stands for the number of auxiliary BNs used in the models. "Zeros" refers to the vanilla BN (Figure 4 Standard BN) that conflates the source-domain data and the target-domain data in a mini-batch; "One" refers to two separate BNs (Figure 4 One Aux. BN) for the source-domain data and the target-domain data, respectively; "Two" refers to three BNs (Figure 4 Two Aux. BN) for clean source-domain data, clean target-domain data, and augmented data, respectively. We can see from Table VIII that separating the source-domain data and target-domain data improves the performance in both SRE16 and SRE18, while further separating the augmented data from clean data degrades the performance. One possible explanation is that when using three separate batch normalizations per layer, there are not enough data to compute the means and variances reliably. A similar phenomenon was also reported in [44].

C. Effect of Network Architectures

Generally speaking, large models and better architectures often improve DNN's performance. However, it is not clear whether domain adaptation will benefit from large models and better architectures. To investigate this, we used two DNN architectures, namely, the x-vector networks and DenseNets, with a different number of parameters. The configurations of the two x-vector networks are shown in Table II and the configurations of the two DenseNets are shown in Table IV. The results are presented in Table IX. It is clear

[2] The speed perturbation we used is different from the one used in the Kaldi's ASR recipe (.../master/egs/wsj/s5/utils/perturb_data_dir_speed.sh). The Kaldi's recipe uses the `speed` option of sox to change the speaking rate with the `speed` parameter ranged from 0.8 to 1.2, which also changes the speaker characteristics in the speech. We used FFmpeg with `atempo` set to 1.3. Perceptually, we notice that there is no loss in the speaker identities in the perturbed speech produced by FFmpeg.

Table V. COMPARISON WITH OTHER DNN ADAPTATION METHODS. SUP. WGAN [27] USED THE LABELS OF SRE16 AND SRE18 DEVELOPMENT DATA. THERE IS NO BACKEND ADAPTATION IN ALL OF THE SYSTEMS.

Adapt Method	Front-end	Back-end	Statistics Pooling	Training set	Loss	SRE16		SRE18	
						EER	minDCF	EER	minDCF
WGAN	X-vector	PLDA	Mean & STD	SRE & SWB	Softmax	13.25	0.899	10.35	0.658
Sup. WGAN	X-vector	PLDA	Mean & STD	SRE & SWB	Softmax	9.59	0.652	8.88	0.619
LSGAN	ResNet	Cosine	Attention	SRE & SWB	AM-Softmax	11.74	–	–	–
KALDI	X-vector	PLDA	Mean & STD	SRE & SWB	Softmax	8.27	0.604	9.6	0.575
MSC	X-vector	PLDA	Mean & STD	SRE & SWB	Softmax	8.69	0.556	7.95	0.500

Table VI. THE PERFORMANCES OF CORAL, PLDA ADAPTATION AND THE PROPOSED FRAMEWORK MSC.

		MSC	CORAL	PLDA Adapt	MSC+CORAL	MSC+PLDA Adapt
SRE16	EER	8.690	8.490	8.550	8.130	8.220
	minDCF	0.556	0.560	0.556	0.530	0.542
SRE18	EER	7.950	8.740	8.880	7.700	8.120
	minDCF	0.500	0.553	0.563	0.486	0.502

Table VII. BREAKDOWN OF DAS' PERFORMANCE ON SRE16.

	Cantonese				Tagalog			
	Male		Female		Male		Female	
	EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
CORAL	4.30	0.366	4.76	0.424	11.83	0.703	12.34	0.731
PLDA Adapt	4.35	0.371	4.80	0.437	11.97	0.714	12.33	0.735
MSC	4.39	0.373	4.95	0.440	11.98	0.718	12.54	0.741
MSC+CORAL	3.80	0.316	4.08	0.343	11.05	0.670	12.11	0.722
MSC+PLDA Adapt	3.88	0.317	4.03	0.349	11.14	0.671	12.31	0.724

Table VIII. ABLATION STUDY OF THE INDIVIDUAL COMPONENTS IN THE PROPOSED FRAMEWORK.

				SRE16		SRE18	
With Utt. Adapt.	With Frame Adapt	With Consist. Reg.	No. of Aux. BN	EER(%)	DCF	EER(%)	DCF
×	×	×	Zero	12.02	0.990	11.59	0.720
✓	×	×	Zero	9.79	0.621	9.08	0.580
✓	✓	×	Zero	9.63	0.606	8.77	0.555
✓	✓	✓	Zero	9.03	0.585	8.33	0.520
✓	✓	✓	One	8.69	0.556	7.95	0.500
✓	✓	✓	Two	9.28	0.566	8.33	0.511

from Table IX that increasing the model size improves the speaker embeddings' performance. We also observed, with the same amount of parameters, the DenseNets outperform the x-vector networks consistently. Besides, DNN domain adaptation does benefit from larger models. By doubling the number of channels in the x-vector networks, we achieved 3.68% and 4.02% improvement in SRE16 and SRE18, respectively. Using larger DenseNets, we achieved 6.01% and 8.23% improvement on SRE16 and SRE18, respectively. Clearly, DA benefits more from deep architectures than the swallow ones.

D. Effect of MMD Kernels

The most important component for MMD-based DA is the kernel. The Gaussian kernel is theoretically more powerful than the quadratic kernel in that it can match up to infinite moments of two distributions. However, it is much harder to find appropriate bandwidth parameters for the Gaussian kernels, which is still an on-going research area [46]. A heuristic is to use the median pair-wise distance computed from data [45]. Another way is to use multiple Gaussian kernels and hope that some of the kernels are close to the ideal ones. It is also possible to combine the median heuristic and the multi-kernel approach. Table X shows the effectiveness of different Gaussian kernels for DA in SRE16 and SRE18.

For the median heuristic, we randomly sampled 10,000 frames of MFCC vectors from the unlabelled part of SRE16 and SRE18 and used the sampled data to compute the pairwise median distance. For the multi-kernel approach, we chose $\sigma_q = 1$ in Eq. 12 and used a multiplicative step-size of 0.1. Specifically, with 5 Gaussian kernels, the bandwidth parameters range from $10^{-2}\sigma_q$ to $10^{2}\sigma_q$ with a multiplicative step-size of 0.1. With 19 Gaussian kernels, the bandwidth parameters range from $10^{-9}\sigma_q$ to $10^{9}\sigma_q$ with a multiplicative step-size of 0.1. For combining the median heuristic with multi-kernel,

Table IX. The performance of four DNN speaker embeddings with and without the proposed adaptation framework. The details of x-vector based networks and DenseNets are presented in table II and table IV, respectively.

Training Protocol	EER(%)				minDCF			
	x-vector	large-x-vector	Densenet61	Densenet121	x-vector	large-x-vector	Densenet61	Densenet121
SRE 16 w.o. adapt.	12.02	11.90	11.87	11.64	0.985	0.851	0.854	0.820
SRE 16 w. Adapt	8.69	8.37	8.31	7.81	0.556	0.534	0.520	0.515
SRE 18 w.o. adapt.	11.59	10.62	10.73	9.47	0.761	0.692	0.703	0.600
SRE 18 w. adapt.	7.95	7.63	7.65	7.02	0.500	0.482	0.482	0.460

Table X. Effect of the Gaussian kernel configuration on speaker embedding adaptation. MH means the bandwith parameters (σ_q in Eq .12) were set using median heuristics (MH) [45]

			No. of Gaussian Kernels					
		No Adapt	1 ($\sigma_q = 1$)	1 (MH)	5 ($\sigma_q = 1$)	9 ($\sigma_q = 1$)	19 ($\sigma_q = 1$)	19 (MH)
SRE16	EER(%)	12.02	14.93	10.85	9.69	9.17	8.99	8.69
	minDCF	0.990	0.841	0.660	0.569	0.559	0.547	0.556
SRE18	EER(%)	11.59	13.33	10.33	8.77	8.01	8.30	7.950
	minDCF	0.720	0.776	0.636	0.529	0.510	0.510	0.500

Table XI. Effect of target-domain data size on SRE18. The results are reported in EER (%).

No. Utt.	500	1000	2000	3000	4073
CORAL	10.54	9.32	9.12	8.81	8.74
PLDA	10.34	9.44	9.01	8.95	8.88
MSC	11.23	9.82	8.93	8.32	7.95

Table XII. Effect of the mismatch between the duration of the training and test utterances on the MMD-based domain adaptation. The results are reported in EER (%).

Adapt. Dur. / Test Dur.	400	800	1200	1600	Multi-level (400)
400	21.27	**20.80**	22.80	21.73	21.30
800	19.24	17.29	17.50	17.91	**17.11**
1200	16.71	15.13	14.55	**14.10**	14.67
1600	15.80	13.85	12.41	12.84	**12.21**

Table XIII. Performance of different DA methods on VoxCeleb1.

	No Adapt	CORAL	PLDA	MSC
EER(%)	8.400	8.750	9.230	8.280
minDCF	0.675	0.592	0.688	0.554

the bandwidth parameter σ_q was set to the median computed from data and a multiplicative step-size of 0.1 was used. We can see from Table X that the single kernel with arbitrarily chosen $\sigma_q = 1$ performs even worse than no adaptation, which shows that good kernel parameter is essential for MMD domain adaptation. On the other hand, the median heuristic performs much better than the single kernel with $\sigma_q = 1$. With multiple kernels, the performance of the adaptation improves significantly, even though σ_q was arbitrarily chosen. The multiple kernel approach performs significantly better than the median heuristic with single kernel. Finally, combining the multi-kernel approach with the median heuristic (19 MH) achieves the best results.

E. *Effect of Target-domain Data Size and Duration*

To investigate the effect of target-domain data size on DA performance, we reduced the number of adaptation utterances from the target-domain (SRE18) to 3,000, 2,000, 1,000, and 500, respectively. The adaptations were carried out using the reduced data. The results are presented in Table XI. Unsurprisingly, the performance of all DA methods degrades when the amount of adaptation data decreases. However, it seems that backend adaptation methods, such as CORAL and PLDA adaptation, are less affected by the amount of adaptation data than MSC.

The main assumption of our multi-level adaptation is that utterance-level adaptation suffers from the training–test duration mismatch. To verify this, we truncated the SRE18 unlabeled data and test data to 400, 800, 1200, and 1600 frames, respectively, and conducted experiments on how the duration mismatch between train–test affects DNN domain adaptation. This duration mismatch problem was also investigated thoroughly in [47] for PLDA adaptation. We trained the x-vector networks to minimize the cross-entropy and utterance-level MMD distance (Eq. 15) using the truncated target-domain data and evaluated on the truncated test data. The results are presented in Table XII. “Test Dur.” stands for the duration of test data (SRE18). “Adapt. Dur.” stands for the duration of target-domain data used for training the DNNs. The best result for each duration is highlighted in bold. We can see from the table that the duration of adaptation data is very important for the DA. The EER could increase from 12.41% to 15.80% when the adaptation data duration decreases from 1,200 frames to 400 frames for test data with 1600 frames. This shows that utterance-level adaptation tends to overfit to a specific duration range. Although the multi-level adaptation only uses 400 frames for adaptation, it produces consistently good results across all the test sets. This shows that adapting both frame-level and utterance-level is beneficial to DA.

F. DA for Cross-channel Problem

To investigate the proposed method's performance on the cross-channel problem, we conducted experiments on the VoxCeleb1 evaluation set. VoxCeleb audio mainly contains interviews recorded using microphones, while SRE and Switchboard data were recorded through telephones. The evaluation protocol of VoxCeleb adaptation is similar to that of SRE16 and SRE18. Ten thousand utterances were sampled from the VoxCeleb2 development set and were treated as unlabeled target-domain data. The models were trained using labeled data from SRE and Switchboard, while the target-domain data are only available in unlabeled form. All VoxCeleb audio was downsampled to 8kHz. The results are presented in Table XIII. All DA methods use the 10,000 unlabeled utterances from the VoxCeleb2 development set as target-domain data. We can see that in this scenario, the DA methods do not give significant improvement over models without any adaptation. The performance of PLDA adaptation is worse than the model without any adaptation. The results show that MSC works better than PLDA adaptation and CORAL. It is also worth noticing that MSC reduces minDCF significantly.

VIII. Conclusions

In this paper, we presented a framework for adapting DNN speaker embedding across languages. We studied all three individual components of our framework (multi-level adaptation, separate batch normalization, and consistency regularization) in detail and found that combining them achieves the best results. We also studied the effect of the kernel parameters in MMD and found that the multi-kernel approach together with the median heuristic give the best performance. What's more, we found that DNN adaptation also benefits from larger model size and better network architectures. As we did not make specific assumptions about the adaptation task in our framework, the proposed framework should be applicable to broader types of domain mismatch beyond the language mismatch studied in this work. It would be interesting to investigate other factors such as noise and channel induced domain differences in the future.

References

[1] S. B. David, T. Lu, T. Luu, and D. Pál, "Impossibility theorems for domain adaptation," in *Proc. the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 129–136, 2010.

[2] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *arXiv preprint arXiv:0902.3430*, 2009.

[3] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[4] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. Greenberg, D. Reynolds, E. Singer, L. Mason, and J. Hernandez-Cordero, "The 2016 NIST speaker recognition evaluation," in *Proc. Interspeech*, pp. 1353–1357, 2017.

[5] K. Jones, S. Strassel, K. Walker, D. Graff, and J. Wright, "Call My Net corpus: A multilingual corpus for evaluation of speaker recognition technology," in *Proc. Interspeech*, pp. 2621–2624, 2017.

[6] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, pp. 5329–5333, 2018.

[7] K. A. Lee and et al, "The I4U mega fusion and collaboration for NIST speaker recognition evaluation 2016," in *Proc. Interspeech*, pp. 1328–1332, 2017.

[8] O. Plchot, P. Matějka, A. Silnova, O. Novotný, M. D. Sánchez, J. Rohdin, O. Glembek, N. Brümmer, A. Swart, J. Jorrín-Prieto, P. García, L. Buera, P. Kenny, J. Alam, and G. Bhattacharya, "Analysis and description of ABC submission to NIST SRE 2016," in *Proc. Interspeech*, pp. 1348–1352, 2017.

[9] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," in *Domain Adaptation in Computer Vision Applications*, pp. 153–171, Springer, 2017.

[10] K. A. Lee, Q. Wang, and T. Koshinaka, "The CORAL+ algorithm for unsupervised domain adaptation of PLDA," in *Proc. ICASSP*, pp. 5821–5825, 2019.

[11] W. W. Lin, M. W. Mak, L. X. Li, and J. T. Chien, "Reducing domain mismatch by maximum mean discrepancy based autoencoders.," in *Odyssey*, pp. 162–167, 2018.

[12] W. W. Lin, M. W. Mak, and J. T. Chien, "Multisource i-vectors domain adaptation using maximum mean discrepancy based autoencoders," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 26, no. 12, pp. 2412–2422, 2018.

[13] W. W. Lin, M. W. Mak, N. Li, S. Dan, and Y. Do, "Multi-level deep neural network adaptation for speaker verification using MMD and consistency regularization," in *Proc. ICASSP*, pp. 6236–6240, 2020.

[14] Y. Tu, M. W. Mak, and J. T. Chien, "Variational domain adversarial learning for speaker verification," *Proc. Interspeech 2019*, pp. 4315–4319, 2019.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[17] Y. Z. Tu, M. W. Mak, and J. T. Chien, "Information maximized variational domain adversarial learning for speaker verification," *Proc. ICASSP*, pp. 6444–6448, 2020.

[18] S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan, "Auto-encoding total correlation explanation," *arXiv preprint arXiv:1802.05822*, 2018.

[19] M. J. Gales and P. C. Woodland, "Mean and variance adaptation within the mllr framework," *Computer speech & language*, vol. 10, no. 4, pp. 249–264, 1996.

[20] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer speech & language*, vol. 9, no. 2, pp. 171–185, 1995.

[21] Y. Wang and M. J. Gales, "Speaker and noise factorization for robust speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 7, pp. 2149–2158, 2012.

[22] M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[23] C. K. Un, N. S. Kim, *et al.*, "Speech recognition in noisy environments using first-order vector taylor series," *Speech Communication*, vol. 24, no. 1, pp. 39–49, 1998.

[24] D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION.* Springer, 2016.

[25] S. Sun, B. Zhang, L. Xie, and Y. Zhang, "An unsupervised deep domain adaptation approach for robust speech recognition," *Neurocomputing*, vol. 257, pp. 79–87, 2017.

[26] W. N. Hsu, Y. Zhang, and J. Glass, "Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation," in *Proc. ASRU*, pp. 16–23, IEEE, 2017.

[27] J. Rohdin, T. Stafylakis, A. Silnova, H. Zeinali, L. Burget, and O. Plchot, "Speaker verification using end-to-end adversarial language adaptation," in *Proc. ICASSP*, pp. 6006–6010, 2019.

[28] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[29] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "Generative adversarial speaker embedding networks for domain robust end-to-end speaker verification," in *Proc. ICASSP*, pp. 6226–6230, 2019.

[30] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, pp. 3214–3218, 2015.

[31] P. Li, Y. Fu, U. Mohammed, J. Elder, and S. Prince, "Probabilistic models for inference about identity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 144–157, 2012.

[32] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, 2017.

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

[34] M. J. Alam, G. Bhattacharya, and P. Kenny, "Speaker verification in mismatched conditions with frustratingly easy domain adaptation.," in *Odyssey*, pp. 176–180, 2018.

[35] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Proc. Advances in Neural Information Processing Systems*, pp. 513–520, 2007.

[36] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *European Conference on Computer Vision*, pp. 443–450, Springer, 2016.

[37] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proc. International Conference on Machine Learning*, pp. 1718–1727, 2015.

[38] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. International Conference on Machine Learning*, pp. 97–105, 2015.

[39] Q. Xie, Z. Dai, E. Hovy, M. T. Luong, and Q. V. Le, "Unsupervised data augmentation," *arXiv preprint arXiv:1904.12848*, 2019.

[40] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," *arXiv preprint arXiv:1911.09665*, 2019.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[43] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP*, pp. 5220–5224, 2017.

[44] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[45] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola, "A kernel statistical test of independence," in *Advances in Neural Information Processing Systems*, pp. 585–592, 2008.

[46] C. L. Li, W. C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD GAN: Towards deeper understanding of moment matching network," in *Proc. Advances in Neural Information Processing Systems*, pp. 2200–2210, 2017.

[47] Q. Hong, L. Li, J. Zhang, L. Wan, and H. Guo, "Transfer learning for plda-based speaker verification," *Speech Communication*, vol. 92, pp. 90–99, 2017.

Appendix A
Gradient of MMD

To make notation less cluttered, we will use $\mathcal{D}$ as a shorthand for $\mathcal{D}(\mathcal{X},\mathcal{Y})$ in Eq. 5 when deriving the derivative of the MMD loss. The gradient with respect to weight matrix $\mathbf{W}_l$ and bias vector $\mathbf{b}_l$ in layer l can be computed using the chain rule:

$$\begin{aligned}\frac{\partial\mathcal{D}}{\partial\mathbf{W}_l} &= \sum_q \frac{\partial\mathcal{D}}{\partial\mathbf{x}_q}\frac{\partial\mathbf{x}_q}{\partial\mathbf{W}_l}\\ \frac{\partial\mathcal{D}}{\partial\mathbf{b}_l} &= \sum_q \frac{\partial\mathcal{D}}{\partial\mathbf{x}_q}\frac{\partial\mathbf{x}_q}{\partial\mathbf{b}_l},\end{aligned} \tag{20}$$

where $\mathbf{x}_q$ is the hidden activation of the DNN and q is a sample index. Eq. 5 can be split into terms that involve $\mathbf{x}_q$ and terms that do not:

$$\begin{aligned}\mathcal{D} &= \frac{1}{N^2}\sum_{i'=1}^{N} k(\mathbf{x}_q,\mathbf{x}_{i'}) + \frac{1}{N^2}\sum_{i=1}^{N} k(\mathbf{x}_i,\mathbf{x}_q)\\ &+ \frac{1}{N^2}\sum_{i=1,i\neq q}^{N}\ \sum_{i'=1,i\neq q}^{N} k(\mathbf{x}_i,\mathbf{x}_{i'}) - \frac{2}{NM}\sum_{j=1}^{M} k(\mathbf{x}_q,\mathbf{y}_j)\\ &- \frac{2}{NM}\sum_{i=1,i\neq q}^{N}\sum_{j=1}^{M} k(\mathbf{x}_i,\mathbf{y}_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M} k(\mathbf{y}_j,\mathbf{y}_{j'}).\end{aligned} \tag{21}$$

Differentiating Eq. 21 with respect to $\mathbf{x}_q$, we have

$$\begin{aligned}\frac{\partial\mathcal{D}}{\partial\mathbf{x}_q} &= \frac{1}{N^2}\sum_{i'=1}^{N}\frac{\partial k(\mathbf{x}_q,\mathbf{x}_{i'})}{\partial\mathbf{x}_q} + \frac{1}{N^2}\sum_{i=1}^{N}\frac{\partial k(\mathbf{x}_i,\mathbf{x}_q)}{\partial\mathbf{x}_q}\\ &- \frac{2}{NM}\sum_{j=1}^{M}\frac{\partial k(\mathbf{x}_q,\mathbf{y}_j)}{\partial\mathbf{x}_q}.\end{aligned} \tag{22}$$

With a Gaussian kernel, Eq. 22 becomes

$$\begin{aligned}\frac{\partial\mathcal{D}}{\partial\mathbf{x}_q} &= \frac{1}{N^2}\sum_{i'=1}^{N}\left[-\frac{1}{\sigma^2}(\mathbf{x}_q-\mathbf{x}_{i'})\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_q-\mathbf{x}_{i'}\right\|^2\right)\right]\\ &+ \frac{1}{N^2}\sum_{i=1}^{N}\left[\frac{1}{\sigma^2}(\mathbf{x}_i-\mathbf{x}_q)\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_i-\mathbf{x}_q\right\|^2\right))\right]\\ &- \frac{2}{NM}\sum_{j=1}^{M}\left[-\frac{1}{\sigma^2}(\mathbf{x}_q-\mathbf{y}_j)\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_q-\mathbf{y}_j\right\|^2\right)\right].\end{aligned} \tag{23}$$

Substituting Eq. 23 in to Eq. 20, we have the gradient with respect to the network weights and bias.