# Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System

*Weicheng Cai[2], Jinkun Chen[2] and Ming Li[1]*

[1]Data Science Research Center, Duke Kunshan University, Kunshan, China
[2]School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China

ming.li369@dukekunshan.edu.cn

## Abstract

In this paper, we explore the encoding/pooling layer and loss function in the end-to-end speaker and language recognition system. First, a unified and interpretable end-to-end system for both speaker and language recognition is developed. It accepts variable-length input and produces an utterance level result. In the end-to-end system, the encoding layer plays a role in aggregating the variable-length input sequence into an utterance level representation. Besides the basic temporal average pooling, we introduce a self-attentive pooling layer and a learnable dictionary encoding layer to get the utterance level representation. In terms of loss function for open-set speaker verification, to get more discriminative speaker embedding, center loss and angular softmax loss is introduced in the end-to-end system. Experimental results on Voxceleb and NIST LRE 07 datasets show that the performance of end-to-end learning system could be significantly improved by the proposed encoding layer and loss function.

## 1. Introduction

Language recognition (LR) , text-independent speaker recognition (SR) and many other paralinguistic speech attribute recognition tasks can be defined as an utterance level "sequence-to-one" learning issue, compared with automatic speech recognition, which is a "sequence-to-sequence" tagging task. They are problems in that we are trying to retrieve information about an entire utterance rather than specific word content [1]. Moreover, there is no constraint on the lexicon words thus the training utterances and testing segments may have completely different contents [2]. The goal, therefore, may boil down to find a robust and time-invariant utterance level vector representation describing the distribution of the given input data sequence with variable-length.

In recent decades, the classical GMM i-vector approach and its variants have dominated multiple kinds of paralinguistic speech attribute recognition fields for its superior performance, simplicity and efficiency [3, 4]. As shown in Fig. 1, the conventional processing pipeline contains four main steps as follows:

- Local feature descriptors, which manifest as a variable-length feature sequence, include hand-crafted acoustic level features, such as log mel-filterbank energies (Fbank), mel-frequency cepstral coefficients (MFCC), perceptual linear prediction (PLP), shifted delta coefficients (SDC) features [2, 5], and automatically learned phoneme discriminant features from deep neural networks (DNN), such as bottleneck features [6, 7, 8], phoneme posterior probability (PPP) features [9], and tandem features [10, 11].

- Dictionary, which contains several temporal orderless center components (or units, words, clusters, etc.), includes vector quantization (VQ) codebooks learned by K-means [12], a universal background model (UBM) learned by Gaussian Mixture Model (GMM) GMM [13, 14] or a supervised phonetically-aware acoustic model learned by DNN [10, 15].

- Vector encoding. This procedure aggregates the variable-length feature sequence into an utterance level vector representation, based on the statistics learned on the dictionaries mentioned above. Typical examples are the GMM Supervector/i-vector [1, 3] or the recently proposed DNN i-vector [15, 16].

- Decision generator, includes logistic regression (LogReg), support vector machine (SVM), and neural network for closed-set identification, cosine similarity or probabilistic linear discriminant analysis (PLDA) [17, 18] for open-set verification.

The GMM i-vector based approaches comprise a series of hand-crafted or ad-hoc algorithmic components, and they show strong generalization ability and robustness when data and computational resource are limited. In recent years, with the merit of large labeled datasets, enormous computation capability, and effective network architectures, emerging progress towards end-to-end learning opens up a new area for exploration [19, 20, 21, 22, 23]. In our previous works [24, 25], we proposed a learnable dictionary encoding (LDE) layer, which connects the conventional GMM Supervector procedure and state-of-the-art end-to-end neural network together. In the end-to-end learning scheme, a general encoding layer is employed on top of the front-end convolutional neural network (CNN), so that it can encode the variable-length input sequence into an utterance level representation automatically. We have shown its success for closed-set LR task. However, when we move forward to SR task, the situation becomes much more complicated.

Typically, SR can be categorized as speaker identification and speaker verification. The former classifies a speaker to a specific identity, while the latter determines whether a pair of utterances belongs to the same person. In terms of the testing protocol, SR can be evaluated under closed-set or open-set settings, as illustrated in Fig. 2. For closed-set protocol, all testing identities are enrolled in the training set. It is natural to classify a testing utterance to a given identity. Therefore, closed-set language or speaker identification can be well addressed as a classification problem. For the open-set protocol, speaker identities in testing set are usually disjoint from the ones in training set, which makes the speaker verification more challenging yet closer to practice. Since it is impossible to classify testing utterances to known identities in training set, we need to map speak-
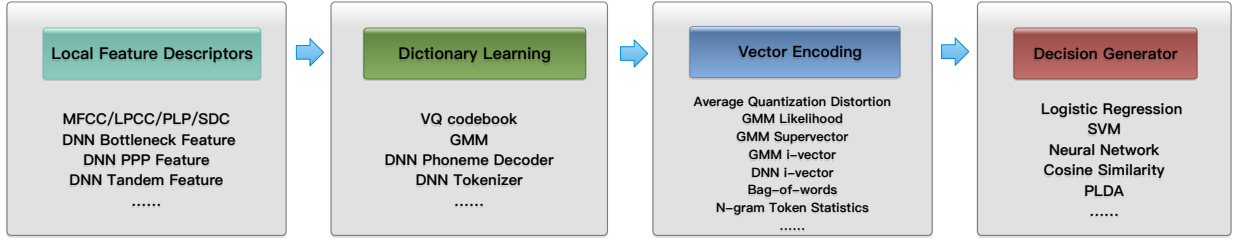
Figure 1: Four main steps in the conventional processing pipeline



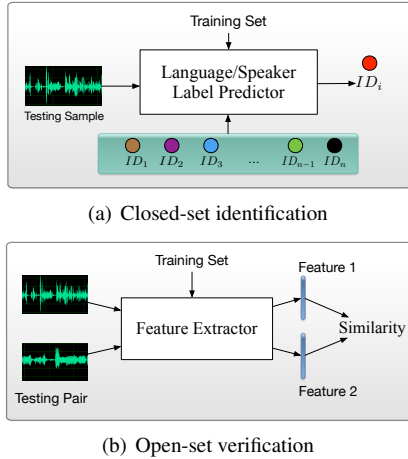(a) Closed-set identification



(b) Open-set verification

Figure 2: Comparison of closed-set identification and open-set verification problem. The closed-set identification is equivalent to classification task, while the open-set verification can be considered as a metric learning task

ers to a discriminative feature space. In this scenario, open-set speaker verification is essentially a metric learning problem, where the key is to learn discriminative large-margin features.

Considering the aforementioned challenges, we generalize the learning scheme for closed-set LR in [24], and build a unified end-to-end system for both LR and SR. The whole pipeline contains five key modules: input data sequence, frame-level feature extractor, encoding layer, loss function, and similarity metric. In this paper, We focus on investigating how to enhance the system performance by exploring different kinds of encoding layers and loss functions.

## 2. End-to-End System Overview

The speech signal is naturally with variable length, and we usually don't know exactly how long the testing speech segment will be. Therefore, a flexible processing method should have the ability to accept speech segments with arbitrary duration. Motivated by [21, 22, 24], the whole end-to-end framework in this paper is shown in Fig. 3. It accepts variable-length input and produces an utterance level result. The additional similarity metric module is specifically designated for the open-set verification task.

Given input data feature sequence such as log mel-filterbank energies (Fbank), we employ a deep convolutional neural network (CNN) as our frame-level feature extractor. It can learn high-level abstract local patterns from the raw input

automatically. The frame-level representation after the front-end convolutional layers is still in a temporal order. The remaining issue is to aggregate them together over the entire sequence. In this way, the encoding layer plays a role in extracting a fixed-dimensional utterance level representation from a variable-length input sequence. The utterance level representation is further processed through a fully-connected (FC) layer and finally connected with an output layer. Each unit in the output layer is represented as a target speaker/language label. All the components in the pipeline are jointly learned in an end-to-end manner with a unified loss function.

## 3. Encoding layer

### 3.1. Temporal average pooling layer

Recently, in both [21, 22], similar temporal average pooling (TAP) layer is adopted in their neural network architectures. As shown in Fig. 5, the TAP layer is inherently designated in the end-to-end network, and it equally pools the front-end learned features over time.

### 3.2. Self-attentive pooling layer

The TAP layer equally pools the CNN extracted features over time. However, not all frame of features contribute equally to the utterance level representation, We introduce a self-attentive pooling (SAP) layer to pay attention to such frames that are important to the classification and aggregate those informative frames to form a utterance level representation.

In [26], attention-based recurrent neural network (RNN) is introduced to get utterance level representation for closed-set LR task . However, the work in [26] relies on a non-trivial pre-training procedure to get the language category embedding, and the authors only report results on 3s short duration task. Different from [26] , the attention mechanism in our network architecture is self-contained, with no need for extra guiding source information.

We implement the SAP layer similar to [27, 28, 29]. That is, we first feed the utterance level feature maps $\{x_1, x_2, \cdots, x_L\}$ into a multi-layer perceptron (MLP) to get $\{h_1, h_2, \cdots, h_L\}$ as a hidden representation. In this paer, we simply adopt a one-layer perceptron,

$$h_t = \tanh(W x_t + b) \tag{1}$$

Then we measure the importance of each frame as the similarity of $h_t$ with a learnable context vector $\mu$ and get a normalized importance weight $w_t$ through a softmax function.

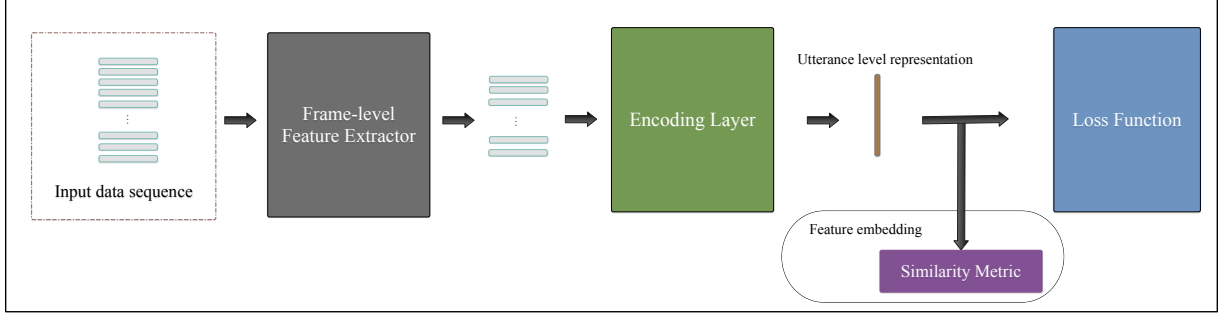$$w_t = \frac{\exp(h_t^T u)}{\sum_{t=1}^{T} \exp(h_t^T u)} \tag{2}$$

Figure 3: End-to-end framework for both LR and SR. It accepts input data sequence with variable length, and produces an utterance level result. The whole pipeline contains five key modules: input data sequence, frame-level feature extractor, encoding layer, loss function, and similarity metric. The additional similarity metric module is specifically designated for the open-set verification task.

The context vector $\mu$ can be seen as a high level representation of a fixed query "what is the informative frame over the whole frames [27]. It is randomly initialized and jointly learned during the training process.

After that, the utterance level representation $e$ can be generated as a weighted sum of the frame level CNN feature maps based on the learned weights.

$$e = \sum_{t=1}^{T} w_t \boldsymbol{x_t} \tag{3}$$

### 3.3. Learnable dictionary encoding layer

In conventional speaker verification system, we always rely on a dictionary learning procedure like K-means/GMM/DNN, to accumulate statistics. Inspired by this, we introduce a novel LDE Layer to accumulate statistics on more detailed units. It combines the dictionary learning and vector encoding steps into a single layer for end-to-end learning.

As demonstrated in Fig. 5, given an input temporal ordered feature sequence with the size of $D \times L$ (where $D$ denotes the feature coefficients dimension, and $L$ denotes the temporal duration length), LDE layer aggregates them over time. More specifically, it transforms them into an utterance level temporal orderless $D \times C$ vector representation, which is independent of length $L$. The LDE Layer imitates the mechanism of GMM Supervector, but learned directly from the loss function.

The LDE layer is a directed acyclic graph and all the components are differentiable $w.r.t$ the input $\boldsymbol{X}$ and the learnable parameters. Therefore, the LDE layer can be trained in an end-to-end manner by standard stochastic gradient descent with backward propagation. Fig. 4 illustrates the forward diagram of LDE layer. Here, we introduce two groups of learnable parameters. One is the dictionary component center, noted as $\mu = \{\mu_1, \mu_2 \cdots \mu_c\}$. The other one is assigned weights, noted as $w$.

Consider assigning weights from the features to the dictionary components. Similar as soft-weight assignment in GMM, the features are independently assigned to each dictionary component and the non-negative assigning weight is given by a softmax function,

$$w_{tc} = \frac{\exp(-s_c \|\boldsymbol{r_{tc}}\|^2)}{\sum_{m=1}^{C} \exp(-s_m \|\boldsymbol{r_{tm}}\|^2)} \tag{4}$$

where the smoothing factor $s_c$ for each dictionary center $\boldsymbol{u_c}$ is learnable.
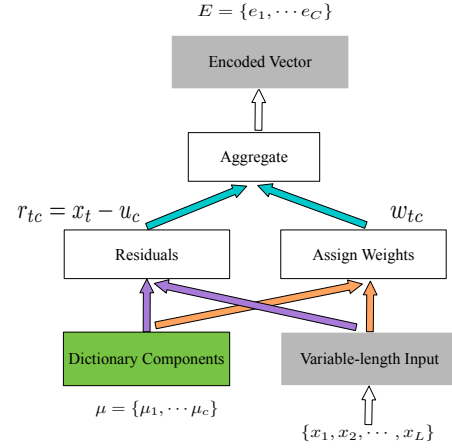


Figure 4: The forward diagram within the LDE layer

Given a set of $L$ frames feature sequence $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_L}\}$ and a learned dictionary center $\mu = \{\mu_1, \mu_2 \cdots \mu_c\}$, each frame of feature $\boldsymbol{x_t}$ can be assigned with a weight $w_{tc}$ to each component $\mu_c$ and the corresponding residual vector is denoted by $\boldsymbol{r_{tc}} = \boldsymbol{x_t} - \boldsymbol{u_c}$, where $t = 1, 2 \cdots L$ and $c = 1, 2 \cdots C$. Given the assignments and the residual vector, similar to conventional GMM Supervector, the residual encoding model applies an aggregation operation for every dictionary component center $\mu_c$:

$$\boldsymbol{e_c} = \sum_{t=1}^{L} e_{tc} = \frac{\sum_{t=1}^{L}(w_{tc} \cdot \boldsymbol{r_{tc}})}{\sum_{t=1}^{L} w_{tc}} \tag{5}$$

In order to facilitate the derivation we simplified it as

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

The LDE layer concatenates the aggregated residual vectors with assigned weights. The resulted encoder outputs a fixed dimensional representation $\boldsymbol{E} = \{\boldsymbol{e_1}, \boldsymbol{e_2} \cdots \boldsymbol{e_C}\}$.

## 4. Loss function

### 4.1. Loss function for closed-set identification

In conventional LR or SR problem, the processing stream is explicitly separated into front-end and back-end. The i-vector

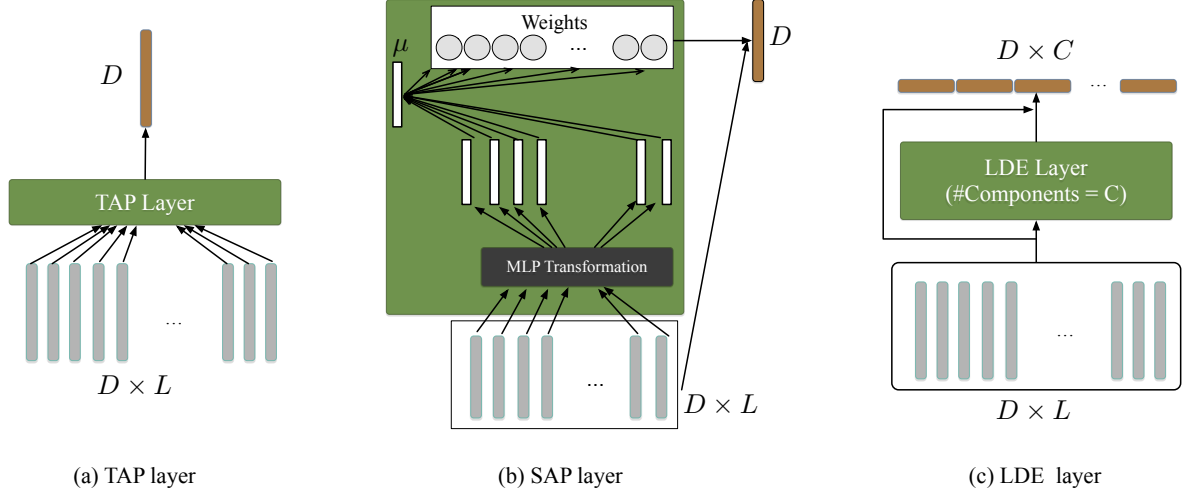(a) TAP layer     (b) SAP layer     (c) LDE layer

Figure 5: Comparison of different encoding procedures

extracting front-end is comprised of multiple unsupervised generative models. They are optimized through Expectation Maximum (EM) algorithm under a negative complete-data log-likelihood loss. Since they are all generative models, we refer their loss functions as a kind of generative negative log-likelihood (GNLL) loss for simplicity. Once front-end model is trained and i-vector is extracted, a back-end LogReg or SVM is commonly adopted to do the back-end classification. Their loss function is softmax or hinge loss.

As illustrated in Fig. 6, for an end-to-end closed-set identification system, the front-end feature extractor and back-end classifier could be jointly learned. In this way, the whole identification system could be optimized within a unified softmax loss:

$$\ell_s = -\frac{1}{M} \sum_{i=1}^{M} \log \frac{e^{\boldsymbol{W}_{y_i}^T f(\boldsymbol{x}_i) + \boldsymbol{b}_{y_i}}}{\sum_{j=1}^{C} e^{\boldsymbol{W}_j^T f(\boldsymbol{x}_j) + \boldsymbol{b}_j}} \qquad (7)$$

where $M$ is the training batch size, $\boldsymbol{x_i}$ is the $i^{th}$ input data sequence in the batch, $f(\boldsymbol{x_i})$ is the corresponding output of the penultimate layer of the end-to-end neural network, $y_i$ is the corresponding target label, and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights and bias for the last layer of the network which acts as a classifier.

### 4.2. Loss function for open-set verification

Once front-end model is trained and i-vector is extracted, PLDA is commonly adopted in the state-of-the-art open-set speaker verification system. PLDA is a Bayesian generative model. Thus its loss function is still GNLL.

We believe that PLDA is not necessary, and a completely end-to-end system should have ability to learn this kind of open-set problem with a unified loss function. However, for open-set speaker verification task, the learned feature embedding need to be not only separable but also discriminative. Since it is impractical to pre-collect all the possible testing identities for training, the label prediction goal and corresponding basic softmax loss is not always applicable. Therefore, as illustrated in Fig. 6, a unified discriminative loss function is needed to have better generalization than closed-set identification.

In [21, 22], similar pairwise loss such as contrastive loss [30, 31] or triplet loss [32] is adopted for open-set speaker ver-



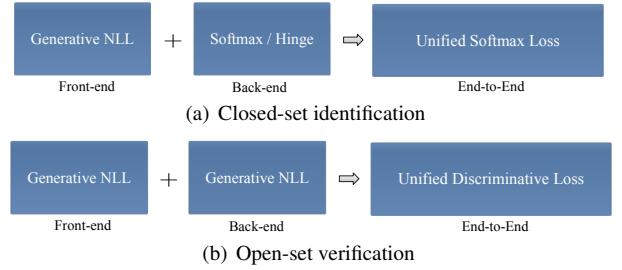(a) Closed-set identification

(b) Open-set verification

Figure 6: Conventional explicitly separated front-end and back-end loss are proceeded into a unified end-to-end loss

ification. They all explicitly treat the open-set speaker verification problem as metric learning problem. However, a neural network trained with pairwise loss requires carefully designed pair/triplet mining procedure. This procedure is non-trivial, both time-consuming and performance-sensitive [33]. In this paper, we focus on the general classification network. This means the units in the output layer are equal to the speaker numbers in the training set. Here we introduce two discriminative loss which is first proposed in computer vision community.

#### 4.2.1. Center loss

The basic softmax loss encourages the separability of features only. In [34], the authors propose a center loss simultaneously learning a center for deep features of each class and penalizing the distances between the deep features and their corresponding class centers. The learning goal is to minimize the within-class variations while keeping the features of different classes separable. The joint supervision of softmax loss and center loss is adopted for discriminative feature learning:

$$\ell = \ell + \lambda \ell_C$$
$$= -\frac{1}{M} \sum_{i=1}^{M} \log \frac{e^{\boldsymbol{W}_{y_i}^T f(\boldsymbol{x}_i) + \boldsymbol{b}_{y_i}}}{\sum_{j=1}^{C} e^{\boldsymbol{W}_j^T f(\boldsymbol{x}_j) + \boldsymbol{b}_j}} + \frac{\lambda}{2} \sum_{i=1}^{M} \|f(\boldsymbol{x}_i) - \boldsymbol{c}_{y_i}\|_2^2 \quad (8)$$

The $\boldsymbol{c}_{y_i} \in \mathbb{R}^d$ denotes the $y_i$th class center of deep features. The formulation effectively characterizes the intra-class varia-

tions. A scalar $\lambda$ is used for balancing the two loss functions.

The conventional softmax loss can be considered as a special case of this joint supervision, if $\lambda$ is set to 0. With proper $\lambda$, the discriminative power of deep features can be significantly enhanced [34].

### 4.2.2. Angular Softmax loss

In [33], the authors propose a natural way to learn angular margin. The angular softmax (A-Softmax) loss is defined as

$$\ell = \frac{1}{M} \sum_{i=1}^{M} - \log\left( \frac{e^{\|f(\boldsymbol{x}_i)\|\phi(\theta_{y_i},i)}}{e^{\|f(\boldsymbol{x}_i)\|\phi(\theta_{y_i},i)} + \sum_{j \neq y_i} e^{\|f(\boldsymbol{x}_i)\|cos(\theta_j,i)}} \right) \quad (9)$$

where $\phi(\theta_{y_i}, i) = (-1)^k cos(m\theta_{y_i,i}) - 2k$, $\theta(y_i, i) \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$ and $k \in [0, m-1]$. $m \geq 1$ is an integer that controls the size of angular margin. When $m = 1$, it becomes the modified softmax loss.

A-Softmax loss has clear geometric interpretation. Supervised by A-Softmax loss, the learned features construct a discriminative angular distance metric that is equivalent to geodesic distance on a hypersphere manifold, which intrinsically matches the prior that speakers also lie on a manifold. A-Softmax loss has stronger requirements for a correct classification when $m \geq 2$, which generates an angular classification margin between learned features of different classes [33].

## 5. Experiments

### 5.1. Data description

#### 5.1.1. Voxceleb

Voxceleb is a large scale text-independent SR dataset collected "in the wild", which contains over 100,000 utterances from 1251 celebrities. It can be used for both speaker identification and verification [35]. We pool the official split training and validation set together as our development dataset.

For speaker verification task, there are totally 1211 celebrities in the development dataset. The testing dataset contains 4715 utterances from the rest 40 celebrities. There are totally 37720 pairs of trials including 18860 pairs of true trials. Two key performance metrics $C_{det}$ [36] and EER are used to evaluate the system performance for the verification task as shown in Table 2.

For speaker identification task, there are totally 1251 celebrities in the development dataset. The testing dataset contains 8251 utterances from these 1251 celebrities. We report top-1 and top-5 accuracies as in Table 3.

#### 5.1.2. NIST LRE07

The whole training corpus including Callfriend datasets, LRE 2003, LRE 2005, SRE 2008 datasets and development data for LRE07. The total training data is about 37000 utterances. The task of interest is the closed-set language detection. There are totally 14 target languages in testing corpus, which included 7530 utterances split among three nominal durations: 30, 10 and 3 seconds. Two key performance metrics Average Detection Cost $C_{avg}$ [37] and Equal Error Rate (EER) are used to evaluate system performance as shown in Table 4.

### 5.2. i-vector system

For general usage, we focus on the comparison on those systems that do not require additional transcribed speech data and extra

Table 1: Our end-to-end baseline network configuration

| Layer | Output size | Downsample | Channels | Blocks |
|-------|-------------|------------|----------|--------|
| Conv1 | $64 \times L_{in}$ | False | 16 | - |
| Res1 | $64 \times L_{in}$ | False | 16 | 3 |
| Res2 | $32 \times \frac{L_{in}}{2}$ | True | 32 | 4 |
| Res3 | $16 \times \frac{L_{in}}{4}$ | True | 64 | 6 |
| Res4 | $8 \times \frac{L_{in}}{8}$ | True | 128 | 3 |
| Avgpool | $1 \times \frac{L_{in}}{8}$ | - | 128 | - |
| Reshape | $128 \times L_{out}, L_{out} = \frac{L_{in}}{8}$ | - | - | - |

DNN acoustic model.

As for the baseline i-vector system, raw audio is converted to 7-1-3-7 based 56 dimensional SDC feature for LR task. For SR task, 20 dimensional MFCC is augmented with their delta and double delta coefficients, making 60 dimensional MFCC feature vectors. A frame-level energy-based voice activity detection (VAD) selects features corresponding to speech frames. A 2048 components full covariance GMM UBM is trained, along with a 600 dimensional i-vector extractor.

For closed-set speaker/language identification, a multi-class LogReg is adopted as the back-end classifier. For open-set verification, cosine similarity or PLDA with full rank is adopted.

### 5.3. End-to-end system

Audio is converted to 64-dimensional Fbank with a frame-length of 25 ms, mean-normalized over a sliding window of up to 3 seconds. The same VAD processing as in i-vector baseline system is used here. We fix the front-end deep CNN module based on the well known ResNet-34 architecture [38]. The detail architecture is described in Table 1. The total parameters of the front-end feature extractor is about 1.35 million.

In CNN-TAP system, a simple average pooling layer is built on top of the front-end CNN. In CNN-LDE system, the TAP layer is replaced with a LDE layer. The number of dictionary components in CNN-LDE system is 64.

The lose weight parameter $\lambda$ of center loss is set to 0.001 in our experiments. For A-Softmax loss, we use the angular margin $m = 4$.

The model is trained with a mini-batch, whose size varies from 96 to 256 considering different datasets and model parameters. The network is trained using typical stochastic gradient descent with momentum 0.9 and weight decay 1e-4. The learning rate is set to 0.1, 0.01, 0.001 and is switched when the training loss plateaus. The training is finished at 40 epochs for Voxceleb dataset and 90 epochs for LRE 07 dataset. Since we have no separated validation set, the converged model after the last optimization step is used for evaluation. For each training step, an integer $L$ within [300,800] interval is randomly generated, and each data in the mini-batch is cropped or extended to $L$ frames.

For open-set speaker verification, the 128-dimensional speaker embedding is extracted after the penultimate layer of neural network. Additional similarity metric like cosine similarity or PLDA is adopted to generate the final pairwise score.

In the testing stage, all the testing utterances with different duration are tested on the same model. Since the duration is arbitrary, we feed the testing speech utterance to the trained neural network one by one.

Table 2: Results for verification on VoxCeleb (lower is better)

| System ID | System Description | Encoding Procedure | Loss Function | Similarity Metric | $C_{det}$ | $EER(\%)$ |
|---|---|---|---|---|---|---|
| 1 | i-vector + cosine | Supervector | GNLL | cosine | 0.829 | 20.63 |
| 2 | i-vector + PLDA | Supervector | GNLL + GNLL | PLDA | 0.639 | 7.95 |
| 3 | TAP-Softmax | TAP | softmax | cosine | 0.553 | 5.48 |
| 4 | TAP-Softmax | TAP | softmax + GNLL | PLDA | 0.545 | 5.21 |
| 5 | TAP-CenterLoss | TAP | center loss | cosine | 0.522 | 4.75 |
| 6 | TAP-CenterLoss | TAP | center loss+ GNLL | PLDA | 0.5155 | 4.59 |
| 7 | TAP-ASoftmax | TAP | A-Softmax | cosine | **0.439** | 5.27 |
| 8 | TAP-ASoftmax | TAP | A-Softmax + GNLL | PLDA | 0.577 | **4.46** |
| 9 | SAP-Softmax | SAP | softmax | cosine | 0.522 | 5.51 |
| 10 | SAP-Softmax | SAP | softmax + GNLL | PLDA | 0.545 | 5.08 |
| 11 | SAP-CenterLoss | SAP | center loss | cosine | 0.540 | 4.98 |
| 12 | SAP-CenterLoss | SAP | center loss+ GNLL | PLDA | 0.571 | 4.89 |
| 13 | SAP-ASoftmax | SAP | A-Softmax | cosine | 0.509 | 4.90 |
| 14 | SAP-ASoftmax | SAP | A-Softmax + GNLL | PLDA | 0.622 | 4.40 |
| 15 | LDE-Softmax | LDE | softmax | cosine | 0.516 | 5.21 |
| 16 | LDE-Softmax | LDE | softmax + GNLL | PLDA | 0.519 | 5.07 |
| 17 | LDE-CenterLoss | LDE | center loss | cosine | 0.496 | 4.98 |
| 18 | LDE-CenterLoss | LDE | center loss + GNLL | PLDA | 0.632 | 4.87 |
| 19 | LDE-ASoftmax | LDE | A-Softmax | cosine | **0.441** | **4.56** |
| 20 | LDE-ASoftmax | LDE | A-Softmax + GNLL | PLDA | 0.576 | 4.48 |

Table 3: Results for identification on VoxCeleb (higher is better)

| System ID | System Description | Top-1(%) | Top-5(%) |
|---|---|---|---|
| 1 | i-vector + LogReg | 65.8 | 81.4 |
| 2 | CNN-TAP | 88.5 | 94.9 |
| 3 | **CNN-SAP** | 89.2 | 94.1 |
| 4 | **CNN-LDE** | **89.9** | **95.7** |

Table 4: Performance on the 2007 NIST LRE closed-set task (lower is better)

| System ID | System Description | $C_{avg}(\%)/EER(\%)$ | | |
|---|---|---|---|---|
| | | 3s | 10s | 30s |
| 1 | i-vector + LogReg | 20.46/17.71 | 8.29/7.00 | 3.02/2.27 |
| 2 | CNN-TAP | 9.98/11.28 | 3.24/5.76 | 1.73/3.96 |
| 3 | **CNN-SAP** | 8.59/9.89 | **2.49**/4.27 | **1.09**/2.38 |
| 4 | **CNN-LDE** | **8.25/7.75** | 2.61/**2.31** | 1.13/**0.96** |

**5.4. Evaluation**

As expected, the end-to-end learning systems outperform the conventional i-vector approach significantly for both SR and LR tasks (see Table 2-4).

For encoding layer, as can be observed in Table 1-3, both SAP layer and LDE layer outperform the baseline TAP layer. Besides, the LDE layer system also show superior performance compared with SAP layer. Considering loss functions in Table 1, in most cases, systems trained with discriminative loss function like center loss or A-Softmax loss achieve better results than softmax loss. In terms of similarity metric, we can find that PLDA gets significant error reduction in conventional i-vector approach. However, when it turns into end-to-end system, especially for those system trained with discriminative loss funtion, PLDA achieves little gain and sometimes makes the result worse.

Finally, CNN-LDE based end-to-end systems achieve best result in speaker/language identification task. Compared with CNN-TAP baseline system, the CNN-LDE system achieve

25%, 45%, 63% relative error reduction for corresponding NIST LRE 07 3s, 10s, 30s duration task. For Voxeceleb speaker identification task, system trained with LDE layer get relative 12% error reduction compared with CNN-TAP system.

In speaker verification task, the speaker embeddings extracted from neural network trained in LDE-ASoftmax system perform best. In the testing stage, a simple cosine similarity achieves the result of $C_{det}$ 0.441 and EER 4.56%, which achieves relative 20% error reduction compared with TAP-Softmax baseline system.

## 6. Conclusions

In this paper, a unified and interpretable end-to-end system is developed for both SR and LR. It accepts variable-length input and produces an utterance level result. We investigate how to enhance the system by exploring different kinds of encoding layers and loss function. Besides the basic TAP layer, we introduce a SAP layer and a LDE layer to get the utterance level representation. In terms of loss function for open-set speaker verification, center loss and A-Softmax loss is introduced to get more discriminative speaker embedding. Experimental results show that the performance of end-to-end learning system could be significantly improved by designing suitable encoding layer and loss function.

## 7. Acknowledgement

# 8. References

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

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

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[4] N. Dehak, P.A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *INTERSPEECH 2016*, pp. 857–860.

[5] H. Li, B. Ma, and K. Lee, "Spoken language recognition: From fundamentals to practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[6] P. Matejka, L. Zhang, T. Ng, H. Mallidi, O. Glembek, J. Ma, and B. Zhang, "Neural network bottleneck features for language identification," *Proc. IEEE Odyssey*, pp. 299–304, 2014.

[7] Y. Song, X. Hong, B. Jiang, R. Cui, I. Mcloughlin, and L. Dai, "Deep bottleneck network based i-vector representation for language identification," 2015.

[8] Fred Richardson, Douglas Reynolds, and Najim Dehak, "A unified deep neural network for speaker and language recognition," *arXiv preprint arXiv:1504.00923*, 2015.

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

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

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

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

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

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

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

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

[17] S.J.D. Prince and J.H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *ICCV 2007*, pp. 1–8.

[18] P. Kenny, "Bayesian speaker verification with heavy tailed priors," *ODYSSEY 2010*.

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

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

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

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

[23] M. Jin, Y. Song, I. McLoughlin, and L. Dai, "Lid-senones and their statistics for language identification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 171–183, 2018.

[24] W. Cai, Z. Cai, W. Liu, X. Wang, and M. Li, "Insights into end-to-end learning scheme for language identification," in *ICASSP 2018*.

[25] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," in *ICASSP 2018*.

[26] W. Geng, W. Wang, Y. Zhao, X. Cai, and B. Xu, "End-to-end language identification using attention-based recurrent neural networks," in *INTERSPEECH*, 2016, pp. 2944–2948.

[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[28] G. Bhattacharya, J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification," in *Proc. Interspeech 2017*, pp. 1517–1521. 2017.

[29] FA Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," *arXiv preprint arXiv:1710.10470*, 2017.

[30] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, pp. 1735–1742.

[31] Y. Chen, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," vol. 27, pp. 1988–1996, 2014.

[32] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[33] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 1.

[34] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 499–515.

[35] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[36] C.S. Greenberg, "The NIST 2012 Speaker Recognition Evaluation Plan," *NIST Technical Report*, 2012.

[37] NIST, "The 2007 NIST Language Recognition Evaluation Plan," *NIST Technical Report*, 2007.

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