Deep Speaker: an End-to-End Neural Speaker Embedding System

Chao Li*, Xiaokong Ma*, Bing Jiang*, Xiangang Li *
Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, Zhenyao Zhu

Baidu Inc.

Abstract

We present Deep Speaker, a neural speaker embedding system that maps utterances to a hypersphere where speaker similarity is measured by cosine similarity. The embeddings generated by Deep Speaker can be used for many tasks, including speaker identification, verification, and clustering. We experiment with ResCNN and GRU architectures to extract the acoustic features, then mean pool to produce utterance-level speaker embeddings, and train using triplet loss based on cosine similarity. Experiments on three distinct datasets suggest that Deep Speaker outperforms a DNN-based i-vector baseline. For example, Deep Speaker reduces the verification equal error rate by 50% (relatively) and improves the identification accuracy by 60% (relatively) on a text-independent dataset. We also present results that suggest adapting from a model trained with Mandarin can improve accuracy for English speaker recognition.

1 Introduction

Speaker recognition algorithms seek to determine the identity of a speaker from audio. Two common recognition tasks are speaker verification (determining whether a speaker's claimed identity is true or false) and speaker identification (classifying the identity of an unknown voice among a set of speakers). Verification and identification algorithms may require the speaker to utter a specific phrase (text-dependent recognition) or be agnostic to the audio transcript (text-independent recognition). In all these subtasks, embedding methods can be used to map utterances into a feature space where distances correspond to speaker similarity. Though many algorithms have pushed the state-of-the-art over the past couple years [1][2][3][4][5][6][7], speaker recognition is still a challenging task.

The traditional speaker recognition approach entails using i-vectors [3] and probabilistic linear discriminant analysis (PLDA) [5]. This framework can be decomposed into three stages [4]:

Step 1 Collect sufficient statistics

Step 2 Extract speaker embeddings (i-vector)

Step 3 Classify (PLDA)

Sufficient statistics (also known as Baum-Welch statistics) are computed from a Gaussian Mixture Model-Universal Background Model (GMM-UBM), which is optimized using a sequence of feature vectors (e.g., mel-frequency cepstral coefficients (MFCC) [3]. Recently, deep neural network (DNN) acoustic models have also been used to extract sufficient statistics [4]. The high-dimensional statistics are converted into a single low-dimensional i-vector that encodes speaker identity and other utterance-level variability. A PLDA model is then used to produce verification scores by comparing i-vectors from different utterances [5].

The three steps of an i-vector system are traditionally trained on subtasks independently, not jointly optimized. An alternative DNN-based approach uses a classification layer [8], combining both *Step 1* and *Step 2*. The intermediate bottleneck layer in the DNN provides a frame-level embedding, which can be used for speakers not included in the training set. During prediction, additional steps are required to aggregate frame-level representations and to perform verification. This approach suffers from at least two major issues: (1) *Step 1* and *Step 2* are not directly optimized with respect to speaker recognition, and (2) there's a mismatch between training and test. The training labels are given at the frame-level, while utterance-level predictions are made in testing.

[6] and [7] introduced end-to-end neural speaker verification systems, combining all three steps. [6] used the last frame output of a long short-term memory (LSTM) [9] model as an utterance-level speaker embedding, while [7] used a network-in-network (NIN) [10] nonlinearity followed by an utterance-level pooling layer to aggregate frame-level representations. Both [6] and [7] were trained using the same distance metric.

In this paper, we extend the end-to-end speaker embedding systems proposed in [6] and [7]. First, a deep neural network is used to extract frame-level features from utterances. Then, pooling and length normalization layers generate utterance-level speaker embeddings. The model is trained using triplet loss [11], which minimizes the distance between embedding pairs from the same speaker and maximizes the distance between pairs from different speakers. Pre-training using a softmax layer and cross-entropy over a fixed list of speakers improves model performance.

More specifically, we test convolutional neural network (CNN)-based and recurrent neural network (RNN)-based architectures for frame-level feature extraction, and present results both for speaker verification and speaker identification. CNNs are effective for reducing spectral variations and modeling spectral correlations in acoustic features [12]. CNNs have also recently been applied to speech recognition with good results [12][13][14][15]. Since deep networks can better represent long utterances than shallow networks [15], we propose a deep residual CNN (ResCNN), inspired by residual networks (ResNets) [16]. We also investigate stacked gated recurrent unit (GRU) [17] layers as an alternative for frame-level feature extraction, since they have proven to be effective for speech processing applications [18][19].

Like [7], we use a distance-based loss function to discriminate between same-speaker and different-speaker utterance pairs. However, unlike the PLDA-like loss function in [7], we train our networks so that cosine similarity in the embedding space directly corresponds to utterance similarity. We also select hard negative examples at each iteration by checking candidate utterances globally, not just in the same minibatch. This approach provides faster training convergence.

Finally, we evaluate our proposed Deep Speaker system on three different datasets, for text-independent and text-dependent speaker

*equally contributed to this work

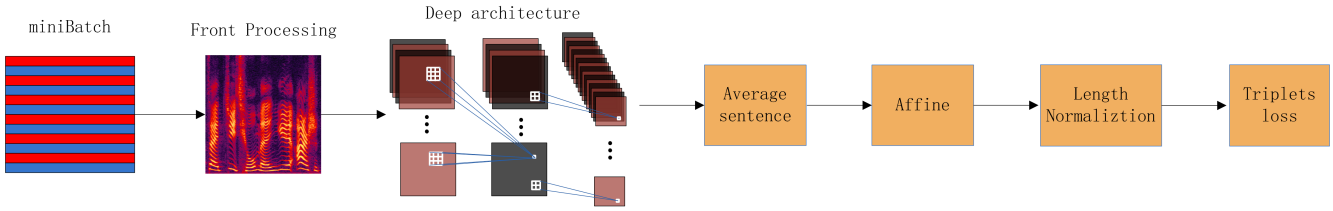


Figure 1: *Diagram of the Deep Speaker architecture*

recognition tasks in both Mandarin and English. We also investigate the effects of softmax pre-training, system combination, training dataset size, and enrollment utterance count. The experiments indicate Deep Speaker can significantly improve over DNN-based i-vector text-independent speaker recognition systems. In the text-dependent task, Deep Speaker can match a DNN i-vector baseline system, and improve upon it if fine-tuned after text-independent training. In particular, two interesting results are shown: (1) Deep Speaker leverages big data well (performance boosts when trained on as many as 250,000 speakers), and (2) Deep Speaker can transfer well across spoken languages that are vastly different, *i.e.*, Mandarin and English.

2 Related Work

Traditionally, i-vectors have been used to model inter-speaker variability [3]. i-vector-based speaker recognition models perform classification using cosine similarity between i-vectors or more advanced techniques such as PLDA [20], heavy-tailed PLDA [21], and Gauss-PLDA [5].

There have been several papers replacing pieces of the traditional speaker recognition system with DNNs. One approach is to train a GMM on bottleneck features extracted from a DNN, and then extract i-vectors [22]. Another DNN-based approach uses an acoustic speech recognition DNN instead of a UBM-GMM to produce frame posteriors for i-vector computation [4]. Ehsan Variani *et al.* [8] trained DNNs to classify speakers with frame-level acoustic features. The activations of the final hidden layer are averaged over the utterance to create a "d-vector" which replaces the i-vector. These approaches all show improvements upon the traditional i-vector baseline.

There have recently been end-to-end neural speaker recognition efforts as well. Georg Heigold *et al.* [6] trained an LSTM for text-dependent speaker verification, which acheived a 2% equal error rate (EER) on the "Ok Google" benchmark. The model maps a test utterance and a few reference utterances directly to a single score for verification and jointly optimizes the system's components using the same evaluation protocol as at test time. David Snyder *et al.* [7] also train an end-to-end text-independent speaker verification system. Like [6], the objective function separates same-speaker and different-speaker pairs, the same scoring done during verification. The model reduces EER by 13% on average, compared to the i-vector baseline.

Our paper uses different architectures than [6] and [7] that balance inference time with model depth and also draw from state-of-the-art speech recognition systems. We showcase our models' efficacy on both text-dependent and text-independent speaker recognition tasks. Lastly, we provide novel insight on the effect of dataset size, softmax pre-training, model fusion, and adaptation from one language to another.

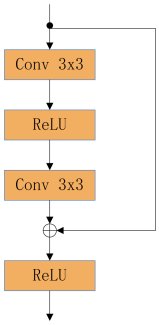


Figure 2: *Detailed view of ResBlock. A convolution block Conv* 3×3 *is parameterized by the filter size* 3×3*, the zero padding 1 in both directions and the consecutive striding* 1×1

3 Deep Speaker

Figure 1 illustrates the architecture of Deep Speaker. Raw audio is first preprocessed using the steps detailed in Section 4.3. Then, we use a feed-forward DNN to extract features over the preprocessed audio. We experiment with two different core architectures: a ResNet-style [16] deep CNN and the Deep Speech 2 (DS2)-style architecture consisting of convolutional layers followed by GRU layers. The details of these networks are described in Section 3.1. A sentence average layer converts frame-level input to an utterance-level speaker representation. Then, an affine layer and a length normalization layer map the temporally-pooled features to a speaker embedding, as presented in Section 3.2. Finally, the triplet loss layer operates on pairs of embeddings, by maximizing the cosine similarities of embedding pairs from the same speaker, and minimizing those from different speakers, as explained in Section 3.3.

3.1 Neural Network Architecture

As stated above, we use two types of deep architectures for frame-level audio feature extraction.

3.1.1 Residual CNN

Though deep networks have larger capacity than shallow networks, they tend to be more difficult to train. ResNet [16] was proposed to ease the training of very deep CNNs. ResNet is composed of a number of stacked residual blocks (ResBlocks). Each ResBlock contains direct links between the lower layer outputs and the higher

Table 1: *Architecture of ResCNN. "Average" denotes the temporal pooling layer and "ln" denotes the length normalization layer. A bracket describes the structure of a ResBlock as shown in Fig. 2*

layer name	structure	stride	dim	# params
conv64-s	$5 \times 5, 64$	2×2	2048	6K
res64	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	1×1	2048	41K×6
conv128-s	$5 \times 5, 128$	2×2	2048	209K
res128	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	1×1	2048	151K×6
conv256-s	$5 \times 5, 256$	2×2	2048	823K
res256	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	1×1	2048	594K×6
conv512-s	$5 \times 5, 512$	2×2	2048	3.3M
res512	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	1×1	2048	2.4M×6
average	-	-	2048	0
affine	2048×512	-	512	1M
ln	-	-	512	0
triplet	-	-	512	0
total				24M

layer inputs, as described in Figure 2. The ResBlock is defined as

$$h = \mathcal{F}(x, W_i) + x, \tag{1}$$

where x and h are the input and output of the layers considered, and $\mathcal{F}$ is the stacked nonlinear layer's mapping function. Note that identity shortcut connections of x do not add extra parameters and computational complexity.

Table 1 shows the details of the proposed ResCNN architecture. As described in Figure 2, the ResBlock contains two convolutional layers with 3×3 filters and 1×1 stride. Each block has an identical structure, and the skip connection is the identity mapping of x. Three ResBlocks are stacked in our architecture. When the number of channels increases, we use a single convolutional layer with filter size of 5×5 and a stride of 2×2. Thus, the the frequency dimension is kept constant in all convolution layers. We find that speaker recognition performance is not sensitive to the stride in the time dimension, contrary to [15]'s findings. Notably, when the number of channels increases, projection shortcuts are not used as in [16], because they increased the number of parameters without yielding significant improvement. We adopt sequence-wise batch normalization (BN) between the convolution and the nonlinearity, following [18]. We use the clipped rectified linear (ReLU) function [18],

$$\sigma(x) = \min\{\max\{x, 0\}, 20\} \tag{2}$$

as our nonlinearity for all of the network layers.

3.1.2 GRU Network

We also experiment with recurrent networks for frame-level feature extraction because they have worked well for speech recognition [19]. [23] showed that a GRU is comparable to an LSTM with a properly initialized forget gate bias, and their best variants are competitive with each other. We decided to use GRUs because previous speech recognition experiments [18] on smaller data sets showed the GRU and LSTM reach similar accuracy for the same number of parameters, but the GRUs were faster to train and less likely to diverge.

The details of the proposed GRU architecture are shown in Table 2. A 5×5 filter size, 2×2 stride convolution layer (like in the ResCNN architecture), reduces dimensionality in both the time and frequency domains, allowing for faster GRU layer computation. Following the convolutional layer are three forward-only GRU layers with 1024 units, recurrent in the time dimension. After the GRU layers, we apply the same average, affine, and length normalization layers as in the ResCNN model. We also use sequence-wise batch normalization and clipped-ReLU activation in the whole model.

Table 2: *Architecture of the GRU model. "Average" denotes the temporal pooling layer, and "ln" denotes the length normalization layer.*

layer name	struct	stride	dim	param
conv64-s	$5 \times 5, 64$	2×2	2048	6K
GRU	1024 cells	1	1024	9.4M
GRU	1024 cells	1	1024	6.3M
GRU	1024 cells	1	1024	6.3M
average	-	-	1024	0
affine	1024×512	-	512	500K
ln	-	-	512	0
triplet	-	-	512	0
total				23M

The ResCNN and GRU architectures have a similar number of parameters, 23M-24M, allowing us to better compare their performances.

3.2 Speaker Embedding

Frame-level activations are fed into a temporal average layer. Unlike in the pooling layer in [7], we do not use standard deviation of frame-level outputs. The layer activation h is computed as follows:

$$h = \frac{1}{T} \sum_{t=0}^{T-1} x(t) \tag{3}$$

where, T is the number of frames in the utterance. An affine layer then projects the utterance-level representation into a 512-dimensional embedding. We normalize embeddings to have unit norm and use cosine similarity between pairs in the objective function:

$$\cos(x_i, x_j) = x_i^{\mathrm{T}} x_j \tag{4}$$

where, x_i and x_j are two embeddings.

3.3 Triplet Loss and Selection

We model the probability of embeddings x_i and x_j belonging to the same speaker by their cosine similarity in Equation (4), allowing us to use the triplet loss function like in FaceNet [11].

As shown in Figure 3, triplet loss takes in as input three samples, an anchor (an utterance from a specific speaker), a positive example (another utterance from the same speaker), and a negative example (an utterance from another speaker). We seek to make updates such that the cosine similarity between the anchor and the positive example is larger than the cosine similarity between the anchor and the negative example [11]. Formally,

$$s_i^{ap} - \alpha > s_i^{an} \tag{5}$$

where, s_i^{ap} is the cosine similarity between the anchor a and the positive example p for triplet i, and s_i^{an} is the cosine similarity between

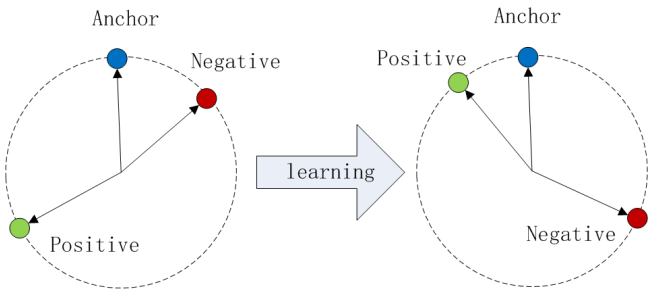


Figure 3: *The Triplet loss in cosine similarity.*

Table 3: *The effect of the number of minibatches scanned to pick a negative sample, measured by the probability of finding a hard sample (Equation 5 not met) and training time differential.*

#GPUs	1	4	8	16
Prob(hard)	29.06%	43.29%	45.54%	50.99%
Rel. time cost	-	+5.47%	+6.09%	+15.28%

the anchor a and the negative example n in triplet i. We impose a minimum margin α between those similarities. The cost function for N triplets can be written as

$$L = \sum_{i=0}^{N} [s_i^{an} - s_i^{ap} + \alpha]_+ \tag{6}$$

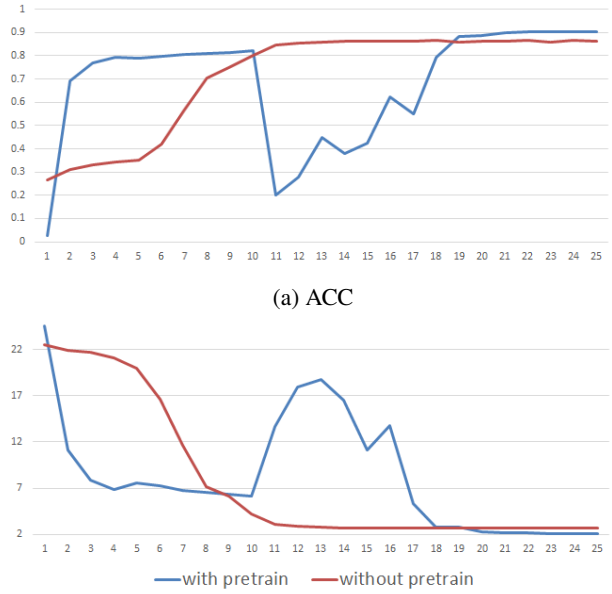
where the operator $[x]_+ = \max(x, 0)$. It is crucial to select "hard" triplets that do not fulfill the constraint in Equation (5).

Training examples are organized as anchor-positive (AP) pairs of same-speaker feature chunks. Mini-batches are formed by picking N pairs and splitting them onto M GPUs, so that each GPU has N/M pairs. All AP pairs in the mini-batch are used, and anchor negatives are selected from the same batch, though not necessarily the same minibatch. Why do we search across GPUs for negative examples? In the beginning of training, it may be easy to find hard negatives which cannot fulfill the constraint in Equation (5). However, finding hard negatives becomes harder and harder as training progresses. Thus, we search over the entire batch for negative examples, rather than in the same minibatch.

We design a simple experiment to investigate the effects of the number of GPUs scanned when picking negative samples. We trained the model for 6 epochs (about 1/4th of total training time), with $N/M = 64$ utterances per mini-batch. We see that the probability of finding an hard negative sample rises without too much additional time cost as the number of GPUs scanned increases (see Table 3). For example, when the number of GPUs increases from 1 to 4, the probability of finding an effective negative sample increases by 48.97%, while time costs only increase by 5.47%.

3.4 Softmax Pre-training

To avoid suboptimal local minima early-on in training, [11] proposed using *semi-hard* negative exemplars, as they are further away from the anchor than the positive exemplar, but still hard because the AN cosine similarity is close to the AP cosine similarity. That is to say $s_i^{an} + \alpha > s_i^{ap} > s_i^{an}$. In the beginning epochs, the model trains only using *semi-hard* negative exemplars, followed by the normal training with hard negative ones. However, properly scheduling "semi-hard" samples is not simple because of model and dataset variability. Instead of the *semi-hard* train stage, we use a softmax and cross entropy loss to pre-train the model. It uses a classification layer [8] to replace the length normalization and triplet loss layers in the standard Deep Speaker architecture described in Figure 1.



(a) ACC

(b) EER

Figure 4: *The effect of softmax pre-training, trained and measured in Train50k and Eva200 dataset, using ResCNN network architecture. (a) accuracy (ACC) vs epoch trained and (b) equal error rate (EER) vs epoch trained*

Using softmax pre-training to initialize the weights of the network has two main benefits. First, we notice that the cross entropy loss produces stabler convergence than triplet loss. We hypothesize that this is because softmax training is not impacted by the variable difficulty of pairs in triplet loss. Secondly, while triplet selection is faster with larger minibatches, smaller mini-batches usually yield better generalization in Stochastic Gradient Descent (SGD) [24].

We designed an experiment to investigate the effect of softmax pre-training for Deep Speaker. Figure 4 shows the validation set accuracy ratio (ACC) and equal error rate (EER) vs epoch index (details about experimental setup are described in Section 4). We pre-train using softmax for 10 epochs, followed by 15 epochs triplet loss training, which causes the spikes at epoch 11 in Figure 4a and Figure 4b. With the whole 25 epochs training, the pre-trained neural network can achieve lower EER and higher ACC than neural networks without pre-training. More detailed comparison will be described in Section 5).

4 Experimental Setup

4.1 Dataset and Evaluation

To investigate the performance of Deep Speaker, we run both speaker verification and identification experiments on three different datasets: UIDs (including Train250k, Train50k and Eva200), XiaoDu, and MTurk. The first two datasets consist of Mandarin speech

Table 4: Statistics of datasets used in the experiments. *UIDs* and *XiaoDu* consist of Mandarin speech from mobile queries, while *MTurk* is a English dataset from Amazon Mechanical Turk. *UIDs* and *Mturk* are text independent, while *XiaoDu* is a text-dependent dataset consisting of wake-words.

	#spkr	#utt	#utt/spkr	dur/utt
Train250k	249,312	12,202,181	48.94	3.72s
Train50k	46,835	2,236,379	47.75	3.66s
Eva200	200	3,800	19	4.25s

UIDs dataset.

	#spkr	#utt	#utt/spkr	dur/utt
train	11,558	89,227	7.72	1.56s
test	844	10,014	11.86	1.48s

XiaoDu dataset

.

	#spkr	#utt	#utt/spkr	dur/utt
train	2,174	543,840	250.16	4.16s
test	200	4,000	20	4.31s

MTurk dataset.

recorded by mobile phone apps, while the third is English speech collected on Amazon Mechanical Turk. UIDs and MTurk are text-independent datasets, while XiaoDu is text-dependent. The details of the three datasets are given as follows.

- *UIDs* is collected from anonymized voice search queries. Table 4 lists statistics for the utterances, the durations of which mostly range from 3 to 5 seconds. The full training partition, *Train250k*, comprises 249,312 speakers, 12,202,181 utterances, and more than 12,600 hours of speech. The subset *Train50k* comprises 46,835 speakers, 2,236,379 utterances, and more than 2,270 hours of speech. The evaluation partition, *Eva200*, consists of 200 speakers that do not overlap with the training speakers. 380,000 speaker recognition trials were run.
- *XiaoDu* contains Baidu wake-word utterances, "Xiaodu, xiaodu". The full training dataset comprises 11,558 speakers, 89,227 utterances, and more than 38 hours of speech. The evaluation dataset consists of 844 speakers that do not overlap with the training speakers. 1,001,400 speaker recognition trials were run.
- *MTurk* contains scripted English utterances collected on Amazon Mechanical Turk. The median utterance length is around 4 seconds, and the 25th and 75th percentile lengths are $\sim$3 seconds and $\sim$5 seconds. The full training dataset comprises 2,174 speakers, 543,840 utterances, and more than 620 hours of speech. The evaluation dataset consists of 200 speakers that do not overlap with the training speakers. 400,000 speaker recognition trials were run.

Speaker verification and identification trials were constructed by randomly picking one anchor positive sample (AP) and 99 anchor negative samples (AN) for each anchor utterance. Then, we computed the cosine similarity between the anchor sample and each of the non-anchor samples. EER and ACC are used for speaker verification and identification, respectively. Since gender labels are not available in all our datasets, the evaluation here is not split by gender, unlike the NIST speaker recognition evaluations (SREs) in [3].

4.2 Baseline DNN i-vector System

The baseline DNN i-vector model is built based on [4]. Raw audio is converted to 40-dimensional log mel-filter bank (Fbank) coefficients and 20-dimensional MFCC with a frame-length of 25ms. Delta and acceleration are appended to the input, and a frame-level energy-based VAD selects features corresponding to speech frames.

A seven-layer DNN that contains 600 input nodes, 1024 nodes in each hidden layer, and 4,682 output nodes is trained with cross entropy using the alignments from a HMM-GMM model. The input layer of the DNN is composed of 15 frames (7 frames on each side of the frame for which predictions are made) where each frame corresponds to 40 dimension Fbank coefficients. The DNN is used to provide the posterior probability in the proposed framework for the 4,682 senones defined by a decision tree.

A 1024 diagonal component UBM is trained in a gender-independent fashion, along with a 400 dimensional i-vector extractor, followed by length normalization and PLDA.

4.3 Training Methodology

Deep Speaker models are trained using the SpeechDL [18] distributed machine learning system with 16 K40 GPUs. Audio is converted to 64-dimensional Fbank coefficients, normalized to have zero mean and unit variance. The same VAD processing as DNN i-vector baseline system is used here.

As described in Section 3.4, Deep Speaker models are trained in two stages: softmax pre-training and triplet loss fine-tuning. In both stages, we use synchronous SGD with 0.99 momentum [24], with a linear decreasing learning rate schedule from 0.05 to 0.005. The model is pre-trained for 10 epochs with a minibatch size of 64 and fine-tuned with triplet loss for 15 epochs using a minibatch size of 128. Training pairs are re-shuffled in each epoch. The margin α is set to 0.1 in 5 A development dataset is used for hyper-parameter tuning and early stopping.

5 Experimental Results

5.1 Speaker-independent Experiments on UIDs

First, we compare the DNN-based i-vector system and Deep Speaker on the UIDs dataset, with the Train50k partition as the training set and Eva200 partition as the test set. Deep Speaker models are trained with different neural network architectures (GRU or ResCNN), and different strategies (softmax, triplet loss, or softmax + triplet loss, which is softmax pretraining followed by triplet loss fine-tuning). The results are listed in Table 5. All Deep Speaker models achieve notable improvements over the baseline, roughly 50 - 80% relative reduction on EER, and 60 - 70% relative improvement on ACC.

5.1.1 Softmax Pre-training

Training using "softmax + triplet" loss achieves the best performance, followed by triplet loss-only and softmax-only training, in decreasing performance order. In particular, the ResCNN with softmax + triplet loss achieves 63.62% and 17.10% relative reduction on EER and 47.53% and 31.33% on error (1−ACC) compared to softmax-only and triplet loss-only models. The pre-trained GRU architecture achieves 48.89% and 14.24% relative reduction on EER and 38.05% and 33.59% on identification error compared to the other

Table 5: *The performance of Mandarin text-independent speaker recognition task using Train50k as train set and Eva200 as test set. Results from different neural network architectures and training methodologies are reported.*

system	EER[%]	ACC[%]
DNN i-vector baseline	13.79	51.72
ResCNN, softmax	6.13	81.95
ResCNN, triplet	2.69	86.21
ResCNN, softmax (pre-train) + triplet	**2.23**	**90.53**
GRU, softmax	5.42	83.05
GRU, triplet	3.23	84.19
GRU, softmax (pre-train) + triplet	**2.77**	**89.50**

two GRU systems. The results confirm the advantages of end-to-end training and softmax pre-training.

5.1.2 Network Architecture

While the GRU architecture outperforms ResCNN with softmax-only training, ResCNN outperforms GRU layers after triplet loss fine-tuning. As shown in Table 5, GRU achieves a 11.58% lower EER and a 6.09% lower error rate compared to ResCNN after softmax training. After triplet loss training, ResCNNs had a 19.49% lower EER and a 10.88% lower error rate than GRUs. As "softmax + triplet loss" training strategy achieves best performance for both GRU and ResCNN, we will use it in the following experiments and also omit this label for brevity's sake.

Time cost is another important consideration for choosing network architectures. We measure the training speed of a model as the number of minibatches that can be processed per second. In this experiment, ResCNN can processes 0.23 minibatches per second, while GRU processes 0.44 minibatches per second. The time cost gap could be partially caused by using Deep Speech 2 HPC techniques for the GRU network [18] and not spending comparable effort speeding up the ResCNN.

5.1.3 System Combination

Individual Deep Speaker models perform well separately, but we anticipate that the ResCNN and GRU systems can benefit from fusion because of their contrasting architectures. To fuse ResCNN and GRU, we investigate two methods: embedding fusion and score fusion. In the first method, we add the speaker embedding from both models together, followed by length normalization and cosine score. In the second method, we first normalize the scores using mean and variance calculated from all scores and add them together. Table 6 indicates that relative to the best single system (ResCNN), both fused system improve the single-system baselines. Especially, the score fusion method gets the best performance, with 7.17% and 13.37% reductions in EER and error, respectively.

5.1.4 Amount of Training Data

Table 7 shows the impact of training dataset size on speaker recognition performance. We do not experiment with the i-vector baseline here, as it's too time consuming and computationally expensive. [1]

[1] For example, the total variance matrix T in the i-vector model is too hard to compute on a big dataset. In practice, people usually train the i-vectors using subsets of a large dataset. Indeed, we tried training i-vector systems on larger datasets and got no obvious improvements.

Table 6: *The system fusion performance of text-independent speaker recognition task using Train50k as train set, Eva200 as test set.*

system	EER[%]	ACC[%]
ResCNN	2.23	90.53
GRU	2.77	89.50
embedding fusion	2.17	90.95
score fusion	**2.07**	**91.83**

Table 7: *The performance of text-independent speaker recognition task using both Train250k and Train50k as training sets and Eva200 as the test set. All deep speaker models are trained with softmax pre-training.*

system	EER[%]	ACC[%]
ResCNN on Train50k	2.23	90.53
ResCNN on Train250k	**1.83**	**92.58**
GRU on Train50k	2.77	89.50
GRU on Train250k	**2.35**	**90.77**

It is clear that using tens of millions of samples results in a performance boost. Compared to using only around 1/5th the data, the using the full dataset reduces the identification error and EER by 17.94% and 21.65% for ResCNN and 15.16% and 13.88% for GRU.

5.1.5 Utterances Number for Enrollment

To investigate the effect of the enrollment utterance count on recognition tasks, we choose 1 to 5 utterances for each person's enrollment. The speaker embeddings are produced by averaging the enrollment utterance embeddings. As before, speaker verification and identification trials were constructed by randomly picking one AP and 99 AN speaker embeddings for each anchor utterance. In total, 280,000 trials were conducted.

Table 8 shows that the EER decreases and ACC increases as the enrollment utterance count increases, though with diminishing returns. These results have implications for production speech recognition system design choices. In particular, using too many enrollment utterances would provide minimal performance gains while increasing inference time, making new user enrollment more cumbersome, and increasing memory usage.

5.2 Text-dependent Experiments on XiaoDu

Table 9 shows the performance of both ResCNN and GRU models for text-dependent speaker recognition on the XiaoDu dataset. The flag "on Train50k" means the Deep Speaker models are only trained on Train50k, while the flag "finetuned" indicates that we first trained the model on Train50k, then used the XiaoDu dataset to fine-tune with triplet loss for Deep Speaker systems, and i-vector extraction for the DNN i-vector system.

Interestingly, the DNN i-vector baseline system achieves the best performance when only using XiaoDu to train the models. There are two possible reasons here. Firstly, the XiaoDu dataset is too small to train complex deep models like Deep Speaker, and we may be overfitting. Secondly, the text-dependent speaker verification

Table 8: *Effect of enrollment utterance count. The performance column is formatted as EER/ACC in each cell. All models are trained on Train50k dataset.*

#utt	i-vector	ResCNN	GRU
1	13.79 / 51.72	2.23 / 90.53	2.77 / 89.50
2	10.37 / 63.21	1.39 / 95.36	1.70 / 94.64
3	8.21 / 71.04	1.29 / 96.56	1.56 / 96.47
5	**7.57 / 75.02**	**1.13 / 96.83**	**1.37 / 97.07**

Table 9: *The text-dependent speaker recognition performance of different systems on XiaoDu dataset.*

system	EER[%]	ACC[%]
DNN i-vector	**3.50**	**95.05**
ResCNN	4.10	93.08
GRU	3.82	93.75
ResCNN on Train50k	3.62	93.25
GRU on Train50k	3.74	94.45
finetuned DNN i-vector	3.40	94.75
finetuned ResCNN	2.83	94.85
finetuned GRU	**2.78**	**95.75**

task constrains the lexicon and phonetic variability, so the i-vector extractor based on factor analysis can cover the speaker variability in the small dataset better.

To our surprise, we find that Deep Speaker models only trained on Train50k achieve slightly better performance than models only trained on XiaoDu. That is to say, models trained using text-independent datasets can work well in text-dependent tasks. We believe that the superior performance is a result of the amount of training data.

Fine-tuning the traditional DNN i-vector system does not significantly improve performance, while fine-tuned ResCNN and GRU networks outperform the DNN i-vector system, by 16.76% and 18.24% relative reduction on EER, and similar ACC. This shows that pre-training on large text-independent datasets can aid in data-constrained text-dependent scenarios. We speculate that the large datasets can cover a greater diversity of samples and encourage model generalization.

5.3 Text-independent Experiments on MTurk

The MTurk experimental results in Table 10 showcase that Deep Speaker works across languages. The flag "on Train50k" means the Deep Speaker models are only trained on Train50k, while the flag "finetuned" means models are first trained on Train50k, and then fine-tuned on the MTurk dataset using triplet loss. Note that this is a non-trivial task, since Mandarin and English speech sounds disparate. We don't report a fine-tuned result for DNN i-vector baseline here. Because Mandarin and English have different phone sets, the ASR-DNN model is difficult to adapt.

By comparing the different systems trained only on MTurk, the ResCNN and GRU system reduce EER by 12.11% and 9.79% and error by 15.02% and 19.38% compared to the DNN i-vector system. Interestingly, models trained solely on the Mandarin Train50k dataset perform fairly well on English speaker classification, even without fine-tuning. The "finetuned" models outperform "non-finetuned" models, reducing the EER by 25% and the error rate by 35%.

Table 10: *The text-independent speaker recognition performance of different systems on MTurk dataset.*

system	EER[%]	ACC[%]
DNN i-vector	3.88	89.68
ResCNN	**3.41**	91.23
GRU	3.50	**91.68**
ResCNN on Train50k	5.92	85.41
GRU on Train50k	5.57	88.22
finetuned ResCNN	2.68	94.53
finetuned GRU	**2.40**	**94.88**

Table 11: *Effect of time span on recognition performance across models. The performance column is formatted with EER/ACC in each cell. All models are trained on Train50k dataset*

#time span	baseline	ResCNN	GRU
1 week	12.73 / 55.33	**2.11 / 91.11**	2.63 / 90.66
1 month	14.39 / 46.96	**2.50 / 88.66**	3.33 / 87.57
3 months	15.31 / 44.37	**2.76 / 87.45**	3.42 / 85.80

These results indicate that Deep Speaker systems can work well not only on Mandarin, but also across other languages. In addition, the representations learned by Deep Speaker transfer well across different languages.

5.4 Time Span Experiments on UIDs

Speaker recognition systems usually struggle with robustness to time between enrollment and test time. People's voiceprints change over time, just like their appearances. We test the robustness of our model across a wide range of time spans using the Eva200 dataset. In Table 11, the first column divides the different time spans, "1 week" means the time span of registration and verification is less than 1 week, "1 month" means less than 1 month but longer than 1 week, and "3 months" means less than 3 months but longer than 1 month.

The performance of all systems decrease as the time span between enrollment and test increases, but ResCNN can still achieve the best performance with the same time span.

6 Conclusion

In this paper we present a novel end-to-end speaker embedding scheme, called Deep Speaker. The proposed system directly learns a mapping from speaker utterances to a hypersphere where cosine similarities directly correspond to a measure of speaker similarity. We experiment with two different neural network architectures (ResCNN and GRU) to extract the frame-level acoustic features. A triplet loss layer based on cosine similarities is proposed for metric learning, along with a batch-global negative selection across GPUs. Softmax pre-training is used for achieving better performance.

The experiments show that the Deep Speaker algorithm significantly improves the text-independent speaker recognition system as compared to the traditional DNN-based i-vector approach. In the Mandarin dataset UIDs, the EER decreases roughly 50% relatively,

and error decreases by 60%. In the English dataset MTurk, the equal error rate decreases by 30% relatively, and error decreases by 50%. Another strength of Deep Speaker is that it can take full advantage of transfer learning to solve the speaker recognition problems on small data sets, for both text-independent and text-dependent tasks.

Future work will focus on better understanding the error cases, reducing model size, and reducing CPU requirements. We will also look into ways of improving the long training times.

7 Acknowledgments

We would like to thank Liang Gao and Yuanqing Lin for their supports and great insights on speaker recognition. We would also like to thank Sanjeev Satheesh, Adam Coates, and Andrew Ng for useful discussions and thoughts. Also our work would not have been possible without the data support of Hongyun Zeng, Yue Pan, Jingwen Cao, and Limei Han.

References

[1] W.M. Campbell, D.E. Sturim, D.A. Reynolds, and A. Solomonoff, *SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, Toulouse, 2006, vol. 1, pp. 97100

[2] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, *Joint Factor Analysis versus Eigenchannels in Speaker Recognition*, IEEE Transaction on Audio Speech and Language Processing, vol. 15, no. 4, pp. 14351447, May 2007

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, *Front-end factor analysis for speaker verification*, IEEE Trans. ASLP, vol. 19, pp. 788798, May 2010.

[4] Lei, Y., Scheffer, N., Ferrer, L., McLaren, M. *A novel scheme for speaker recognition using a phonetically-aware deep neural network*. in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2014

[5] Cumani, S., Laface, P., Torino, P. *Probabilistic Linear Discriminant Analysis Of Ivector Posterior Distributions*. in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013

[6] Heigold, G., Moreno, I., Bengio, S., Shazeer, N. *End-to-end text-dependent speaker verification*. in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2016

[7] David S., Pegah G., Daniel P., Daniel G.R., Yishay C, Sanjeev K. *Neural Network-Based Speaker Embeddings for End-To-End Speaker Verification*. in IEEE Spoken Language Technology Workshop (SLT), 2016

[8] E. Variani, X. Lei, E. McDermott, I. Moreno, and Javier Gonzalez-Dominguez, *Deep neural networks for small footprint text-dependent speaker verification*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2014.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):17351780, 1997

[10] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, *Acoustic modelling from the signal domain using cnns*, Interspeech 2016.

[11] Schroff, F., Philbin, J. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823

[12] Zhang, Y., Pezeshki, M., Bengio, Y., Courville, A., Hc, Q. C. . *Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks*, Interspeech 2016.

[13] Yu, Dong, Xiong, Wayne, Droppo, Jasha, Stolcke, Andreas, Ye, Guoli, Li, Jinyu, and Zweig, Geoffrey. *Deep convolutional neural networks with layer-wise context expansion and attention*. Interspeech 2016

[14] Bi, M., Qian, Y., Yu, K. *Very Deep Convolutional Neural Networks for LVCSR*, Interspeech, 2015.

[15] Wang, Y., Deng, X., Pu, S., Huang, Z. *Residual Convolutional CTC Networks for Automatic Speech Recognition*. Retrieved from http://arxiv.org/abs/1702.07793

[16] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. *Deep residual learning for image recognition*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770 778, 2016.

[17] K. C ho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. *Learning phrase representations using rnn encoder-decoder for statistical machine translation*. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014

[18] Amodei, D., Anubhai, R., Battenberg, E., Carl, C., Casper, J., Catanzaro, B., Zhu, Z. *Deep-speech 2: End-to-end speech recognition in English and Mandarin*. https://arxiv.org/abs/1512.02595

[19] Sainath, Tara N, Weiss, Ron J, Senior, Andrew, Wilson, KevinW, and Vinyals, Oriol. *Learning the speech front-end with raw waveform cldnns*. Interspeech 2015.

[20] S. J. D. Prince, *Probabilistic linear discriminant analysis for inferences about identity*, in Proc. International Conference on Computer Vision (ICCV), Rio de Janeiro, Brazil, 2007.

[21] Castaldo, F., Alam, M. J., Cernocky, J. H. *Full-Covariance UBM and Heavy-tailed PLDA in I-vector Speaker Verification*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, 2011.

[22] S. Ghalehjegh and R. Rose, *Deep bottleneck features for i-vector based text-independent speaker verification*, in IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). IEEE, 2015, pp. 555 C560.

[23] R. Jozefowicz,W. Zaremba, and I. Sutskever. *An empirical exploration of recurrent network architectures*. In International Conference on Machine Learning (ICML), 2015

[24] D. R. Wilson and T. R. Martinez. *The general inefficiency of batch training for gradient descent learning*. Neural Networks, 16(10):14291451, 2003. 4