A CORRECTIVE LEARNING APPROACH FOR TEXT-INDEPENDENT SPEAKER VERIFICATION

Yandong Wen, Tianyan Zhou, Rita Singh, Bhiksha Raj*

Department of Electrical and Computer Engineering, Carnegie Mellon University
{yandongw,tianyanz,rsingh,bhikshar}@andrew.cmu.edu

ABSTRACT

We present a conceptually plausible approach for text-independent speaker verification (TISV) which treats speech recordings as a collection of segments providing incremental evidence. This approach, called corrective learning, gradually improves an initial prediction of speaker identity based on incoming speech and the latest prediction. Specifically, we propose deep corrective learning networks (CLNets) that explicitly learn a mapping from a new speech segment and the current predictions, to a correction. Intuitively, the predictions eventually converge to the ground truth after several corrections. Trained on NIST SRE datasets, CLNets outperform current CNN and the *i*-vector baselines. Moreover, CLNets and *i*-vectors are complementary, and their fusion leads to significant performance improvements compared to what can be achieved by each of them individually.

Index Terms— Speaker verification, deep corrective learning networks, universal background model, *i*-vectors

1. INTRODUCTION

Text-independent speaker verification (TISV) continues to be one of the most active research areas in speech processing [1], with a wide range of applications, such as access control [2], surveillance [3, 4], and forensics [5, 6, 7]. The challenge here is to determine if the speaker in a "test" recording is the same as that in available prior "enrollment" recording(s).

The key challenge in TISV is to derive an appropriate representation for the recording. Current state-of-art solutions are largely based on *Supervectors*, which are obtained from any recording by adapting a Universal background model (UBM) to fit it best, and concatenating the parameters of the resultant Gaussian mixture distribution [8]. These Supervectors are then projected into lower-dimensional representations such as *i*-vectors through factor analysis [9], or through probabilistic linear discriminant analysis (PLDA) [10], into its discriminative variant. TISV is then performed through direct comparisons of the *i*-vectors or PLDA vectors derived from the enrollment and test data.

*This work was funded by Schmidt Sciences, Palo Alto, CA.

A more recent approach to parametrizing entire speech recordings into feature vectors employs deep neural networks. The most effective method analyzes Mel-frequency spectrographic representations of recordings with a convolutional neural network (CNN) to derive their parametrizations [11, 12, 13, 14]. Since the size of the output of CNNs varies with the size (duration) of the input, the outputs of the final layer of the CNN are averaged across time to obtain the desired fixed-length representations. These neural-network derived representations are then used within a PLDA-based framework for verification, as described earlier. Unlike standard PLDA-based representations, which focus on the distribution of instantaneous spectral characteristics of the signal and ignore both temporal patterns and local frequency-patterns, CNNs capture the distribution of spectro-temporal patterns, effectively deriving more information from the signal.

However, CNNs too make a simplifying assumption about the data: longer recordings are implicitly assumed to be just longer samples of input, drawn from a stationary process. The features derived from the longer recordings are hence assumed to just be more robust estimates of the statistics accumulated from the filters of the network. While this is a perfectly reasonable assumption, we believe that an alternate perspective may yield better results.

We note that a long speech recording may be viewed as a collection of shorter segments, each individually providing evidence about the speaker. The standard approach to combining evidence from multiple independent inputs is to average their independent predictions. We however take a different view. When presented with multiple independently drawn instances of data, instead of assuming that each new instance makes an effectively independent prediction for the class (or feature) that must be averaged with prior predictions, we claim that each new instance may in fact be used to *build upon* the predictions that have already been made, in the form of incremental corrections to them.

In embodying this philosophy, we propose an alternative *recurrent* formalism to analyze independent data instances, where each new instance makes *corrective* predictions to *update* the predictions made from prior data. We will call this network formalism *deep corrective learning networks* (CLNets).

In the setting of TISV, this means that we can now view each recording as a collection of shorter segments, which can be analyzed by a CLNet to derive features for the recording. In the sections that follow we describe the proposed formalism. Section 2 provides the statistical justification for the model. Section 3 describes our model, and Section 4 explains how it is applied to the problem of speaker verification.

Experiments described in Section 5 show that our proposed network is able to provide significant gains over both the conventional *i*-vector/PLDA framework, and one that is based on CNN-based features. Further improvements are obtained by combining it with the *i*-vector based system. More importantly, as we emphasize in our discussions in Section 6, the new network formalism also provides auxiliary benefits such as the ability to identify information-bearing portions of a test recording.

2. AVERAGING VS. CORRECTIVE LEARNING

Consider a set of class-conditionally independent inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$, which are known to have been obtained from one of M classes $Y_1, \cdots, Y_M$. It is known that all inputs belong to the same class, but not which one. To determine this, we must perform Bayesian classification. Using the notation $\boldsymbol{x}_{1:N}$ to represent $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$:

$$\hat{Y} = \arg\max_Y P(Y|\boldsymbol{x}_{1:N})$$

Typically, this Bayes classifier is computed alternately as

$$\begin{aligned}\hat{Y} &= \arg\max_Y P(\boldsymbol{x}_{1:N}|Y)P(Y) \\ &= \arg\max_Y P(Y)\prod_i P(\boldsymbol{x}_i|Y) \\ &= \arg\max_Y \log P(Y) + \sum_i \log P(\boldsymbol{x}_i|Y) \end{aligned} \quad (1)$$

Both, the computation of *a posteriori* probabilities and the classification above may also be *incrementally* performed. Let $\hat{Y}_t$ be the classification after inputs $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ are observed. We recall the recursion:

$$P(Y|\boldsymbol{x}_{1:t}) = \frac{P(Y|\boldsymbol{x}_{1:t-1})P(\boldsymbol{x}_t|Y)}{P(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})} \quad (2)$$

Using the notation $L_t(Y) = \log P(Y|\boldsymbol{x}_{1:t})$ we can write

$$\hat{Y}_t = \arg\max_Y L_{t-1}(Y) + \Delta L(Y, \boldsymbol{x}_t) \quad (3)$$

where $\Delta L(Y, \boldsymbol{x}_t) = \log P(\boldsymbol{x}_t|Y)$.

Now consider how the above classification may be performed using a deep neural network in the conventional framework: a deep neural network directly computes the *a posteriori* probability $P(Y|\boldsymbol{x})$, or alternately, the logarithm $f(Y, \boldsymbol{x}) = \log P(Y|\boldsymbol{x})$. Given a set of independent inputs $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$, there is no direct mechanism for computing $\log P(Y|\boldsymbol{x}_{1:N})$, unless the network is *explicitly* trained to consume N inputs simultaneously and produce the posterior probability of Y given the combined input.

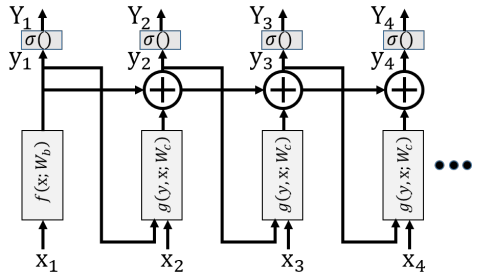

Fig. 1. Corrective networks

The usual approach then is to produce N separate estimates $f(Y, \boldsymbol{x}_1), f(Y, \boldsymbol{x}_2), \cdots, f(Y, \boldsymbol{x}_N)$ and sum them to compute $f(Y, \boldsymbol{x}_{1:N}) = \frac{1}{N}\sum_i f(Y, \boldsymbol{x}_i) = \frac{1}{N}\sum_i \log P(Y|\boldsymbol{x}_i)$. Classification is then usually performed as

$$\begin{aligned}\hat{Y} &= \arg\max_Y f(Y, \boldsymbol{x}_{1:N}) \\ &= \arg\max_Y \frac{1}{N}\sum_i f(Y, \boldsymbol{x}_i)\end{aligned}$$

However, the above estimate is at best only an approximation, since although $\boldsymbol{x}_i, \cdots, \boldsymbol{x}_N$ are class-conditionally independent, $P(Y|\boldsymbol{x}_{1:N}) \neq \prod_i P(Y|\boldsymbol{x}_i)^{1/N}$, as will be immediately apparent from inspection of Equation 2.

A more appropriate composition is derived from Equation 3. We may instead define *two* networks, the first to compute the *base* prediction $f_0(Y, \boldsymbol{x}) = L_0(Y)$, and the second to compute the incremental corrections $f(Y, \boldsymbol{x}_t) = \Delta L(Y, \boldsymbol{x}_t)$.

We can now more accurately perform incremental prediction as

$$\hat{Y}_t = \arg\max_Y f_0(Y, \boldsymbol{x}_1) + \sum_{i=2}^{t} f(Y, \boldsymbol{x}_i)$$

Our proposed deep corrective network (CLNet) architecture builds upon this rather simple principle, with the modification that the correction term $f(Y, \boldsymbol{x}_t)$ is composed instead as $f(Y, Y_{t-1}, \boldsymbol{x}_t)$, to explicitly represent the fact that the corrective network does not compute *a posteriori* probabilities for Y, but rather the log likelihood for Y conditioned on $\boldsymbol{x}_t$, and that it builds upon our prior $(t-1)$ beliefs about Y.

3. DEEP CORRECTIVE LEARNING NETWORKS

Figure 1 shows the basic form of the proposed CLNet. The network consists of two blocks: a first "base" block $f(x; W_b)$ with network parameters W_b, and a "correction" block $g(y, x; W_c)$ with parameters W_c. The specific architecture of $f()$ and $g()$ may vary with the problem.

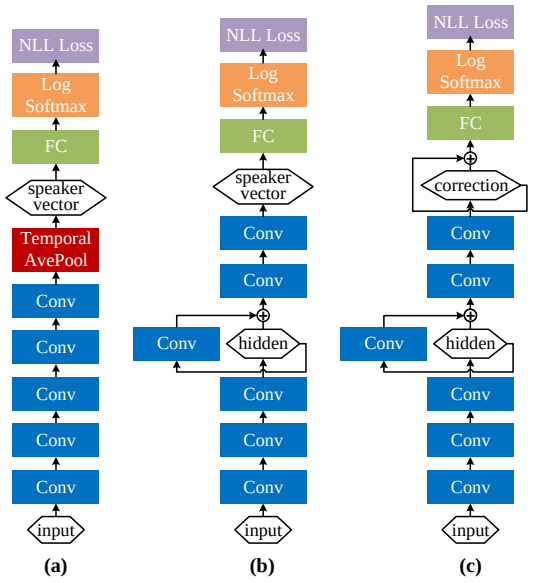

Fig. 2. Illustrations of three architectures for comparison. (a) CNN (b) RNN (c) CLNets. All networks had 5 convolutional layers, followed by a Batch Normalization (BN) [15] layer and rectified linear units (ReLU) [16]activation function. The filter size, stride, and padding are 3×3, 2, and 0, respectively. The number of filters are 4, 16, 64, 256, and 64 from bottom to top.

The base block takes in an initial input $\boldsymbol{x}_1$ to produce an initial output $y_1 = f(\boldsymbol{x}_1; W_b)$. Every subsequent input $\boldsymbol{x}_t$, $t = 2, \cdots$ is passed to a correction block which produces $\Delta y_t = g(y_{t-1}, \boldsymbol{x}_t; W_c)$. The corresponding corrected output is given by $y_t = y_{t-1} + \Delta y_t$.

The outputs y_i represent the *log* posteriors of the classes and are finally passed through a softmax layer, possibly with additional intervening layers, to produce the actual class posterior probabilities, Y_i.

For any set of inputs, all inputs are assumed to correspond to the same Y. To train the network we define the loss

$$L = \sum_t w_t Div(Y, Y_t)$$

where w_t is a weighting term that controls how quickly the estimate is required to converge with t. In our work we have set $w_t = 1$. W_b and W_c are trained by backpropagation.

4. CLNETS FOR SPEAKER VERIFICATION

In the context of TISV, we first note that the objective of the network is not to directly perform verification, but rather to compute *features* from the speech signal, which may subsequently be used within any other framework (e.g. PLDA) to perform the actual verification.

Since our approach fore speaker verification is similar to other CNN-based approaches [12, 13], we briefly outline a standard CNN-based approach here. We represent the speech signal as a Mel-frequency spectrogram. In the conventional CNN-based approach, each of the filters in the final layer of the CNN produces an output at every instant of time (*i.e.* for every frame of the spectrogram). These outputs are averaged across time to finally produce a single feature vector for the complete recording. The training data typically comprise training instances from many speakers. The CNN is followed by an additional classification subnet, which accepts the output of the CNN and attempts to distinguish between all the speakers in the training set. Once trained, the final classification subnet is removed to result in a CNN that produces features that are optimized to distinguish between speakers.

Our CLNet format too follows much of this structure. We work with Mel spectrographic representations of the speech signal. However, unlike the regular approach, the speech is segmented into many sections of K frames each (we set K=64 in our experiments, although this could be optimized). The first of these segments is used to produce the base feature, which is subsequently updated through corrections extracted from subsequent correction terms.

As in the CNN models, the base and corrective networks $f()$ and $g()$ are convolutional networks that operate on the segments. Within each segment the outputs of the final filters are averaged across the breadth of the segment to obtain either the base feature (in the case of $f()$) or the correction terms (in the case of $g()$). We extend the basic CLNet format described in Section 3 by also including the hidden representations in the correction nets in the recursion.

Thus, denoting the n-th input speech segment and n-th hidden state as $\boldsymbol{x}_n$ and $\boldsymbol{h}_n$, and the corresponding output feature as y_n, CLNets can be formulated as

$$\begin{aligned} \boldsymbol{h}_n &= f(\boldsymbol{x}_n, \boldsymbol{h}_{n-1}) \\ \Delta y_n &= g(y_n, y_{n-1}) \\ y_n &= \Delta y_n + y_{n-1} \end{aligned} \tag{4}$$

As in the case of CNNs, the feature is passed into a final classification subnet to train the network. Once the network is trained, the final classification subnet is removed.

The entire architecture is illustrated in Figure 2(c). Figure 2 also shows the conventional CNN architecture employed in TISV (Fig 2(a)), and a conventional LSTM-based recurrent architecture (Fig 2(b)) which has also been proposed for the problem of feature extraction for speaker verification.

5. EXPERIMENTS

Experiments were conducted on the NIST SRE corpora, obtained from the Linguistic Data Consortium. For training, we used the SRE 2004-2008 datasets with ~36,500 recordings, each of about 5 minutes, from 3801 speakers. The test set used was SRE 2010, with 11,959 recordings for enrollment and 767 recordings for testing. Experiments included evaluating 416,119 trial pairs of recordings to determine if

each pair was from the same speaker or not. The trial set included 7,169 *positive* pairs and 408,950 *negative* pairs of recordings. In each pair, one utterance was picked from the enrollment set, and the other from the test set. Further experimental setup-related details are given below.

***i*-vector baseline:** We implemented an *i*-vector baseline using Kaldi [11]. The front-end features consisted of 20-dimensional Mel Frequency Cepstral Coefficients (MFCCs), with delta and acceleration features. The MFCCs were derived from 25ms frames shifted by 10ms, and were mean-normalized over a sliding window of 3 seconds. An energy-based voice activity detector was used to remove unvoiced speech segments. The UBM was a 2048-component GMM with a full covariance matrix. The corresponding supervectors were transformed to 600-dimensional *i*-vectors that were length normalized.

CNN and RNN baseline: The CNN and RNN architectures used are shown in Fig. 2(a) and (b), respectively. The input comprised 64-component log Mel spectrograms. The networks were trained via stochastic gradient descent, in mini-batches of 128 samples, for 100 epochs. The momentum and weight decay values used were 0.9 and 0.0005 respectively. The learning rate was initialized at 0.1 and divided by 10 after 50 epochs and again after 75 epochs. While training the CNNs, all input spectrograms were restricted to 16,383 frames. For RNNs, the spectrograms were processed as a sequence of 510 inputs (sections), each 64 frames wide, with a stride of 32 frames. The test recordings were not restricted in size. Each recording resulted in a 64-dimensional feature vector.

CLNets: The architecture of the CLNet used is shown in Fig. 2(c). The number of parameters in the CLNet were the same as those in the RNN. The training and test settings were exactly the same as those used with RNNs, and the features extracted from the CLNet were also 64-dimensional. In addition, we also trained an ensemble of 4 more networks, fusing the features derived from them into a single 256-dimensional feature. Finally we fused the ensemble feature and the *i*-vectors to obtain a second fused 1,056-dimensional feature.

5.1. Evaluation and results

We evaluated performance on the SRE10 database under two conditions. In the first condition, entire utterances were used. The second condition evaluated the effect of changes in recoding duration. For this, the enrollment and testing utterances were truncated to durations of 10 to 80 seconds, with a granularity of 10 seconds. We computed the match between enrollment and test recordings using cosine similarity and PLDA. Fig. 3 shows the detection error tradeoff (DET) curves and equal error rates (EER) achieved for the first condition.

We note that the CLNet outperforms CNN and RNN, both of which have architectures that are similar and equivalent to the CLNet used, with both cosine similarity and PLDA measures, indicating that CLNets are better than both for the full-utterance scenario. Secondly, the ensemble of CLNets performs better than the *i*-vector system, which has long been the state of art. Moreover, features extracted from CLNets are complement the *i*-vectors, as evidenced by the improved performance on fusion of these features. The final EER of 1.79% improves on the *i*-vector baseline of 2.71% significantly, with a relative improvement of 34%.

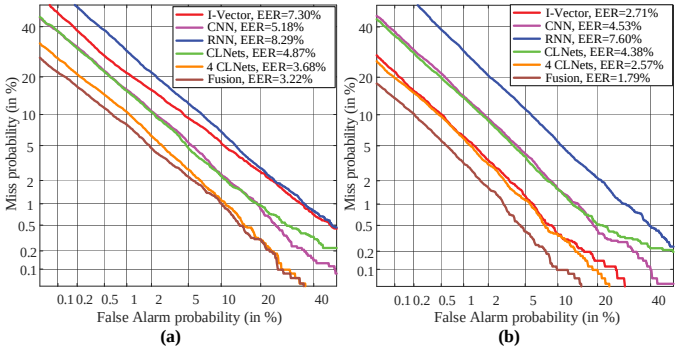

Fig. 3. DET curves and EERs of different approaches. (a) Score is computed by cosine similarity, (b) Score is computed by PLDA.

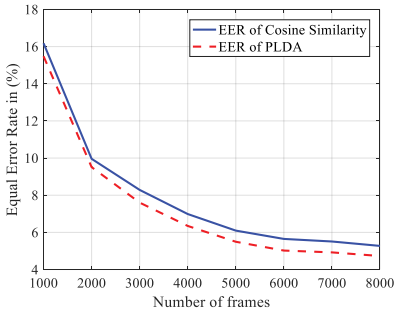

Fig. 4. EERs associated with increasing numbers of frames

The second condition, where we vary the duration of the recordings, yields additional insights. From Fig. 4, it is clear that with increasing numbers of speech segments (increasing duration), the CLNet is are able to make more corrections, leading to lower EERs. With only 10 seconds of speech, the EERs are 16.18% and 15.48% with cosine similarity and PLDA metrics, respectively. These errors quickly fall below 5% as the cumulative length of the segments extends to over 80 seconds, eventually saturating.

6. CONCLUSIONS

The proposed correction-based network outperforms all other approaches. It also converges faster than other methods. Segments that require zero correction during training are identifiable as noisy or non-contributing segments, so the CLNet can also be an effective automatic noise segmenter for speech signals. We expect additional improvements through segment-length optimization, improvements in the training loss function etc. These are areas of current investigation.

7. REFERENCES

[1] Douglas A Reynolds, "An overview of automatic speaker recognition technology," in *Acoustics, speech, and signal processing (ICASSP), 2002 IEEE international conference on*. IEEE, 2002, vol. 4, pp. IV–4072. 1

[2] Stephane Herman Maes, "Text independent speaker recognition for transparent command ambiguity resolution and continuous access control," June 6 2000, US Patent 6,073,101. 1

[3] Felix Burkhardt, Richard Huber, and Anton Batliner, "Application of speaker classification in human machine dialog systems," *Speaker Classification I*, pp. 174–179, 2007. 1

[4] Rita Singh, Abelino Jiménez, and Anders Øland, "Voice disguise by mimicry: deriving statistical articulometric evidence to evaluate claimed impersonation," *IET Biometrics*, vol. 6, no. 4, pp. 282–289, 2017. 1

[5] Amy Neustein and Hemant A Patil, *Forensic speaker recognition*, Springer, 2012. 1

[6] Rita Singh, Bhiksha Raj, and James Baker, "Short-term analysis for estimating physical parameters of speakers," in *Biometrics and Forensics (IWBF), 2016 4th International Workshop on*. IEEE, 2016, pp. 1–6. 1

[7] Rita Singh, Joseph Keshet, and Eduard Hovy, "Profiling hoax callers," in *Technologies for Homeland Security (HST), 2016 IEEE Symposium on*. IEEE, 2016, pp. 1–6. 1

[8] William M Campbell, Douglas E Sturim, and Douglas A Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006. 1

[9] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011. 1

[10] Simon J D Prince and James H Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8. 1

[11] David Snyder, Daniel Garcia-Romero, and Daniel Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 92–97. 1, 4

[12] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 165–170. 1, 3

[13] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification," *Proc. Interspeech 2017*, pp. 999–1003, 2017. 1, 3

[14] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017. 1

[15] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456. 3

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 3