Neural Predictive Coding Using Convolutional Neural Networks Toward Unsupervised Learning of Speaker Characteristics

Arindam Jati and Panayiotis Georgiou, *Senior Member, IEEE*

***Abstract*—Learning speaker-specific features is vital in many applications like speaker recognition, diarization, and speech recognition. This paper provides a novel approach, we term neural predictive coding (NPC), to learn speaker-specific characteristics in a completely unsupervised manner from large amounts of unlabeled training data that even contain many non-speech events and multi-speaker audio streams. The NPC framework exploits the proposed short-term active-speaker stationarity hypothesis which assumes two temporally close short speech segments belong to the same speaker, and thus a common representation that can encode the commonalities of both the segments, should capture the vocal characteristics of that speaker. We train a convolutional deep siamese network to produce "speaker embeddings" by learning to separate "same" versus "different" speaker pairs which are generated from an unlabeled data of audio streams. Two sets of experiments are done in different scenarios to evaluate the strength of NPC embeddings and compare with state-of-the-art in-domain supervised methods. First, two speaker identification experiments with different context lengths are performed in a scenario with comparatively limited within-speaker channel variability. NPC embeddings are found to perform the best at short duration experiment, and they provide complementary information to i-vectors for full utterance experiments. Second, a large-scale speaker verification task having a wide range of within-speaker channel variability is adopted as an upper-bound experiment where comparisons are drawn with in-domain supervised methods.**

***Index Terms*—Speaker-specific characteristics, unsupervised learning, Convolutional Neural Networks (CNN), siamese network, speaker recognition.**

I. Introduction

ACOUSTIC modeling of speaker characteristics is an important task for many speech-related applications. It is also a very challenging problem due to the highly complex information that the speech signal modulates, from lexical content to emotional and behavioral attributes [1], [2] and multi-rate encoding of this information. A major step towards speaker modeling is to identify features that focus only on the speaker-specific characteristics of the speech signal. Learning these characteristics has various applications in speaker segmentation [3], diarization [4], verification [5], and recognition [6]. State-of-the-art methods for most of these applications use short-term acoustic features [7] like MFCC [8] or PLP [9] for signal parameterization. In spite of the effectiveness of the algorithms used for building speaker models [6] or clustering speech segments [4], sometimes these features fail to produce high between-speaker variability and low within-speaker variability [7]. This is because MFCCs contain a lot of supplementary information like phoneme characteristics, and they are frequently deployed in speech recognition [10].

A. Prior Work

Significant research effort has gone into solving the above mentioned discrepancies of short-term features by incorporating long-term or prosodic features [11] into existing systems. These features can specifically be used in speaker recognition or verification systems since they are calculated at utterance-level [7]. Another way to tackle the problem is to calculate mathematical functionals or transformations on top of MFCC features to expand the context and project them on a "speaker space" which is supposed to capture speaker-specific characteristics. One popular method [12] is to build a GMM-UBM [7] on training data and utilize MAP adapted GMM supervectors [12] as fixed dimensional representations of variable length utterances. Along this line of research, there has been ample effort in exploring different factor analysis techniques on the high dimensional supervectors to estimate contributions of different latent factors like speaker- and channel-dependent variabilities [13]. Eigenvoice and eigenchannel methods were proposed by Kenny *et al.* [14] to separately determine the contributions of speaker and channel variabilities respectively. In 2007, Joint Factor Analysis (JFA) [15] was proposed to model speaker variabilities and compensate for channel variabilities, and it outperformed the former technique in capturing speaker characteristics.

Introduction of i-vectors: In 2011, Dehak *et al.* proposed i-vectors [16] for speaker verification. The i-vectors were inspired by JFA, but unlike JFA, the i-vector approach trains a unified model for speaker and channel variability. One inspiration for proposing the Total Variability Space [16] was

Manuscript received February 21, 2018; revised November 9, 2018 and April 4, 2019; accepted May 30, 2019. Date of publication June 10, 2019; date of current version July 12, 2019. This work was supported by the Office of the Assistant Secretary of Defense for Health Affairs in part through the Military Suicide Research Consortium under Award W81XWH-10-2-0181 and in part through the Psychological Health and Traumatic Brain Injury Research Program under Award W81XWH-15-1-0632. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Najim Dehak. *(Corresponding author: Panayiotis Georgiou.)*

The authors are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: jati@usc.edu; georgiou@sipi.usc.edu).

Digital Object Identifier 10.1109/TASLP.2019.2921890

from the observation that the channel effects obtained by JFA also had speaker factors. The i-vectors have been used by many researchers for numerous applications including speaker recognition [16], [17], diarization [18], [19] and speaker adaptation during speech recognition [20] due to their state-of-the-art performance. But, performance of i-vector systems tends to deteriorate as the utterance length decreases [21], especially when there is a mismatch between the lengths of training and test utterances. Also, the i-vector modeling, similar to most factor analysis methods, is constrained by the GMM assumption which might degrade the performance in some cases [7].

DNN-based methods in speaker characteristics learning: Recently, Deep Neural Network- (DNN) [22] derived "speaker embeddings" [23] or bottleneck features [24] have been found to be very powerful for capturing speaker characteristics. For example, in [25], [26] and [27], frame-level bottleneck features have been extracted using DNNs trained in a supervised fashion over a finite set of speakers; and some aggregation techniques like GMM-UBM [12] or i-vectors have been used on top of the frame-level features for utterance-level speaker verification. Chen *et al.* [28], [29] developed a deep neural architecture and trained it for frame-level speaker comparison task in a supervised way. They achieved promising results in speaker verification and segmentation tasks even when they evaluated their system on out-of-domain data [28]. In [30], the authors proposed an end-to-end text-independent speaker verification method using DNN embeddings. It uses the similar approach to generate the embeddings, but the utterance-level statistics are computed internally by a pooling layer in the DNN architecture. In more recent work [31], different combinations of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [22] have been exploited to find speaker embeddings using the triplet loss function which minimizes intra-speaker distance and maximizes inter-speaker distance [31]. The model also incorporates a pooling and normalization layer to produce utterance-level speaker embeddings.

Need for unsupervised methods and existing works: In spite of the wide range of DNN variants, all these need one or more annotated dataset(s) for supervised training. This limits the learning power of the methods, especially given the data-hungry needs of advanced neural network-based models. Supervised training can also limit robustness due to over-tuning to the specific training environment. This can cause degradation in performance if the testing condition is very different from that of the training. Moreover, transfer learning [32] of the supervised models to a new domain also needs labeled data. This points to a desire and opportunity in employing unlabeled data and unsupervised methods for learning speaker embeddings.

There have been a few efforts [33]–[35] in the past to employ neural networks for acoustic space division, but these works focused on speaker clustering and they did not exploit short-term stationarity towards embedding learning. In [36], an unsupervised training scheme using convolutional deep belief networks has been proposed for audio feature learning. They applied those features for phoneme, speaker, gender and music classification tasks. Although, the training employed there was unsupervised, the proposed system for speaker classification was trained on TIMIT dataset [37] where every utterance is guaranteed to come from a single speaker, and PCA whitening was applied on the spectrogram per utterance basis. Moreover, performance of the system on out-of-domain data was not evaluated.

B. Proposed Work

In this paper, we propose a completely unsupervised method for learning features having speaker-specific characteristics from unlabeled audio streams that contain many non-speech events (music, noise, and anything else available on YouTube). We term the general learning of signal characteristics via the short-term stationarity assumption *Neural Predictive Coding* (NPC) since it was inspired by the idea of predicting present value of signal from a past context as done in Linear Predictive Coding (LPC) [38]. The short-term stationarity assumption can take place, according to the frame size, along different characteristics. For example we can assume that the behaviors expressed in the signal will be mostly stationary within a window of a few seconds as we did in [39]. In this work we assume that any potentially active speaker will be mostly stationary within a short window: the active speaker is unlikely to change multiple times within a couple of seconds. LPC predicts future values from past data via a filter described by its coefficients. NPC can predict future values from past data via neural network. The embedding inside the NPC neural network can serve as a feature. Moreover, while predicting future values from past, the NPC model can incorporate knowledge learned from big, unlabeled datasets.

The short-term speaker stationarity assumption was exploited in our previous work [40] via an encoder-decoder model to predict future values from past through a bottleneck layer. The training involved in that work was able to see past and future values of the signal only from the 'same speaker', assuming speaker stationarity. In contrast, the currently employed siamese architecture [41], [42] helps the model to encounter and compare whether a pair of speech segments come from the same speaker or, two different speakers, based on unlabeled data via the short-term stationarity assumption.

We perform experiments under different scenarios and for different applications to explore the ability of the proposed method to learn speaker characteristics. Moreover, the NPC training is done on out-of-domain data, and its performance is compared with i-vectors and recently introduced x-vectors [43] trained on in-domain data.

Note that the NPC training needs no labels at all, not even speaker homogeneous regions. For that reason, we do not expect NPC-derived features to beat in-domain supervised algorithms, but rather present this as an upper-bound aim.

The comparison reveals interesting directions that can arise through further introduction of context. For example, if the algorithm employs longer same-speaker context (than 2 s assumed by this work) similar to i-vector systems then it can allow for variable length features and increased channel normalization.

Below are the major aims of the proposed work towards establishing a robust speaker embedding: 1) Training should require

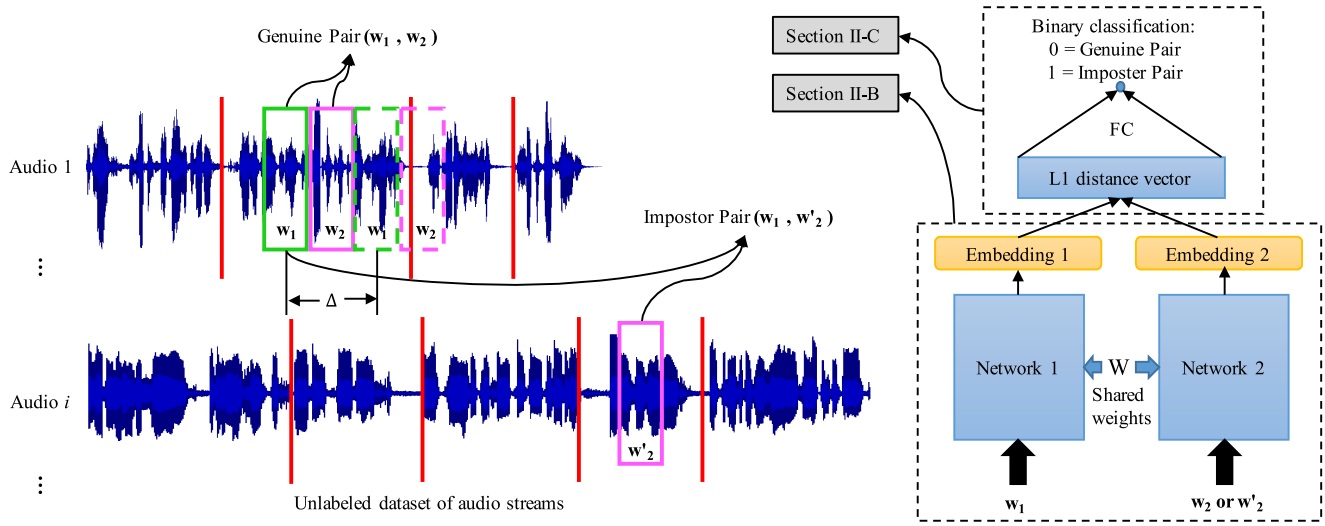


Fig. 1. NPC training scheme utilizing short-term speaker stationarity hypothesis. **Left:** Contrastive sample creation from unlabeled dataset of audio streams. Genuine and impostor pairs are created from unlabeled dataset as explained in Section II-A. **Right:** The siamese network training method. The genuine and impostors pairs are fed into it for binary classification. "FC" denotes Fully Connected hidden layer in the DNN. Note that the siamese convolutional layers have been discussed in Section II-B, and the derivation of the loss functions by comparing the siamese embeddings has been shown in Section II-C.

no labels of any kind (no speaker id labels, or speaker homogeneous utterances for training); 2) System should be highly scalable relying on plentiful availability of unlabeled data; 3) Embedding should represent short-term characteristics and be suitable as an alternative to MFCCs in an aggregation system like [25] or [27]; and, 4) The training scheme should be readily applicable for unsupervised transfer learning.

The rest of the paper is organized as follows. The NPC methodology is described in Section II. Section III provides details about evaluation methodology and required experimental setup. Results are tabulated and discussed in Section IV. A qualitative analysis and future scopes are provided in Section V. Finally conclusions are drawn in Section VI.

II. Neural Predictive Coding (NPC) of Speaker Characteristics

Our ultimate goal is to learn a non-linear mapping (the employed DNN or part of it) that can project a small window of speech from any speaker to a lower dimensional embedding space where it will retain the maximum possible speaker-specific characteristics and reject other information as much as possible.

We expect that based on the unsupervised training paradigm we employ the embedding may also capture additional information, mainly channel characteristics and we intend to address that in future work, as further discussed in Section V.

A. Contrastive Sample Creation

NPC learns to extract speaker characteristics in a contrastive way *i.e.* by distinguishing between different speakers. During training phase, it possesses no information about the actual speaker identities, but only learns whether two input audio chunks are generated from the same speaker or not. We provide the NPC model two kinds of samples [41]. The first kind consists of pairs of speech segments that come from the same speaker, called *genuine pairs*. The second type consists of speech segments from two different speakers, called *impostor pairs*. This approach has been used in the past for numerous applications [41], [42], [44], but all of them needed labeled datasets. The challenge is how we can create such samples if we do *not* have labeled acoustic data. We exploit the characteristics of speaker-turntaking that result in *short-term speaker stationarity* [40]. The hypothesis of short-term speaker stationarity is based on the notion that given a long observation of human interaction, the probability of fast speaker changes will be at the tails of the distribution. In short: it is very unlikely to have extremely fast speaker changes (for example every 1 second). So, if we take pairs of consecutive *short* segments from such a long audio stream, most of the pairs will contain two audio segments from the same speaker (genuine pairs). There will be definitely some pairs containing segments from two different speakers, but number of such pairs will probably be small compared to the total number of genuine pairs. To find the impostor pairs, we choose two random segments from two different audio streams in our unsupervised dataset. Again, intuitively the probability of finding the same speaker in an impostor pair is relatively lower than the probability of getting two different speakers in it, provided a sufficiently large unsupervised dataset. For example, sampling two random YouTube videos, the likelihood of getting the same speaker in both is very low.

The left part of Fig. 1 shows this contrastive sample creation process. Audio stream 1 and audio stream i (for any i between 2 to N, where N is the number of audio streams in the dataset) are shown here. Assume the vertical red lines denote (unknown) speaker change points. $(\mathbf{w_1}, \mathbf{w_2})$ is a window pair where each of the two windows has d feature frames. This window pair is moved over the audio streams with a shift of Δ to get the genuine pairs. For every $\mathbf{w_1}$, we randomly pick a window $\mathbf{w'_2}$ of same

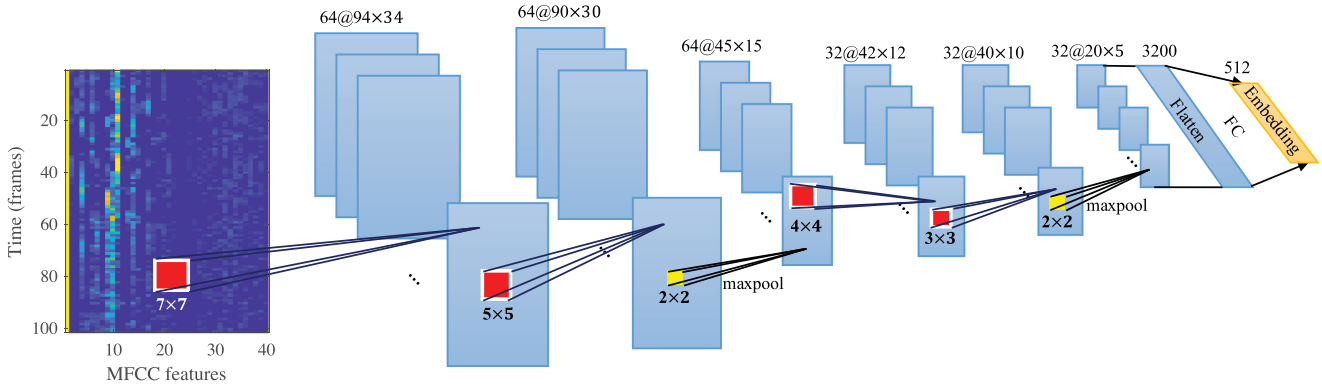


Fig. 2. The DNN architecture employed in each of the siamese twins. All the weights are shared between the twins. The kernel sizes are denoted under the red squares. 2×2 max-pooling is used as shown by yellow squares. All the feature maps are denoted as: $N@x \times y$, where N = number of feature maps, $x \times y$ = size of each feature map. Dimension of the speaker embedding is 512. "FC" = Fully Connected layer.

length from a different audio stream to get an impostor pair. All these samples are then fed into the siamese DNN network for binary classification of whether an input pair is genuine or impostor.

A siamese neural network (please see right part of Fig. 1), first introduced for signature verification [45], consists of two identical twin networks with shared weights that are joined at the top by some energy function [41], [42], [44]. Generally, the siamese networks are provided with two inputs and trained by minimizing the energy function which is a predefined metric between the highest level feature transformations of both the inputs. The weight sharing ensures similar inputs are transformed into embeddings in close proximity with each other. The inherent structure of a siamese network enables us to learn similar or dissimilar input pairs with discriminative energy functions [41], [42]. Similar to [44], we use L_1 distance loss between the highest level outputs of the siamese twin networks for the two inputs.

We will first describe in Section II-B about the CNN that processes the speech spectrogram to automatically learn features to generate the embeddings. Next, in Section II-C we will discuss about the top part of the neural network of Fig. 1 that involves comparing the two embeddings and deriving the final output and error for back-propagation.

B. Siamese Convolutional Layers

The amazing effectiveness of CNNs have been well established in computer vision field [46], [47]. Recently, speech scientists are also applying CNNs for different challenging tasks like speech recognition [48], [49], speaker recognition [31], [50], [51], large scale audio classification [52] *etc.* The general benefits of using CNNs can be found in [22] and in the above papers. In our work, the inspiration to use CNNs comes from the need of exploring spectral and temporal contexts together through 2D convolution over the mel-spectrogram features (please see Section III-D for more information). The benefits of such a 2D convolution have also been shown with more traditional signal processing feature sets such as Gabor features [53].

Our siamese network (one of the identical twins), built using multiple CNN layers and one dense layer at the highest level, is shown in Fig. 2. We gradually reduce the kernel size from 7×7 to 5×5, 4×4, and 3×3. We have used 2×2 max-pooling layers after every two convolutional layers. The size of stride for all convolution and max-pooling operations has been chosen to be 1. We have used Leaky ReLU nonlinearity [54] after every convolutional or fully connected layer (omitted from Fig. 2 for clearer visualization).

We have applied batch normalization [55] after every layer to reduce the "internal covariance shift" [55] of the network. It helped the network to avoid overfitting and converge faster without the need of using dropout layers [56]. After the last convolutional layer, we get 32 feature maps, each of size 20×5. We flatten these maps to get a 3200 dimensional vector which is connected to the final 512 dimensional NPC embedding through a fully connected layer. The embeddings are obtained before applying the Leaky ReLU non-linearity.[1]

C. Comparing Siamese Embeddings—Loss Functions

Let $\mathbf{f}(\mathbf{x_1})$ and $\mathbf{f}(\mathbf{x_2})$ be the highest level outputs of the siamese twin networks for inputs $\mathbf{x_1}$ and $\mathbf{x_2}$ (in other words, $(\mathbf{x_1}, \mathbf{x_2})$ is one contrastive sample obtained from the window pair $(\mathbf{w_1}, \mathbf{w_2})$ or $(\mathbf{w_1}, \mathbf{w_2'})$). We will use this transformation $\mathbf{f}(\mathbf{x})$ as our "embedding" for any input $\mathbf{x}$ (please see right part of Fig. 1). Here $\mathbf{x}$ is a matrix of size $d \times m$, and it denotes d frames of m dimensional MFCC feature vectors in window $\mathbf{w}$. Similarly, $\mathbf{x_i}$ denotes the feature frames in window $\mathbf{w_i}$ for $i = 1, 2$. We have deployed two different types of loss functions for training the NPC model. They are described below.

1) Cross Entropy Loss: Inspired from [44], the loss function is designed in a way such that it decreases the weighted L_1 distance between the embeddings $\mathbf{f}(\mathbf{x_1})$ and $\mathbf{f}(\mathbf{x_2})$ if $\mathbf{x_1}$ and $\mathbf{x_2}$ are from a genuine pair, and increases the same if they are from an impostor pair.

The "L_1 distance vector" (Fig. 1, right) is obtained by calculating element-wise absolute difference between the two

[1]Following standard convention for extracting embedding from DNNs, such as in [57].

embedding vectors $\mathbf{f}(\mathbf{x_1})$ and $\mathbf{f}(\mathbf{x_2})$ and is given by:

$$\mathbf{L}(\mathbf{x_1}, \mathbf{x_2}) = |\mathbf{f}(\mathbf{x_1}) - \mathbf{f}(\mathbf{x_2})|. \quad (1)$$

We connect $\mathbf{L}(\mathbf{x_1}, \mathbf{x_2})$ to two outputs $g_i(\mathbf{x_1}, \mathbf{x_2})$ using a fully connected layer:

$$g_i(\mathbf{x_1}, \mathbf{x_2}) = \sum_{k=1}^{D} w_{i,k} \times |f(\mathbf{x_1})_k - f(\mathbf{x_2})_k| + b_i \quad (2)$$

for $i = 1, 2$. Here, $f(\mathbf{x_1})_k$ and $f(\mathbf{x_2})_k$ are the kth elements of $\mathbf{f}(\mathbf{x_1})$ and $\mathbf{f}(\mathbf{x_2})$ vectors respectively, and D is the length of those vectors (so, D is the embedding dimension). $w_{i,k}$'s and b_i's are the weights and bias for the ith output. Note that these weights and biases are affecting only the binary classifier, and they are not part of the siamese network.

A softmax layer produces the final probabilities:

$$p_i(\mathbf{x_1}, \mathbf{x_2}) = s(g_i((\mathbf{x_1}, \mathbf{x_2}))) \qquad \text{for } i = 1, 2. \quad (3)$$

Here $s(.)$ is the softmax function given by

$$s(g_i(\mathbf{x_1}, \mathbf{x_2})) = \frac{e^{g_i(\mathbf{x_1}, \mathbf{x_2})}}{e^{g_1(\mathbf{x_1}, \mathbf{x_2})} + e^{g_2(\mathbf{x_1}, \mathbf{x_2})}} \quad \text{for } i = 1, 2. \quad (4)$$

The network in Fig. 1 is provided with the genuine and impostor pairs as explained in Section II-A. We use cross entropy loss here. It is given by

$$\begin{aligned} e(\mathbf{x_1}, \mathbf{x_2}) = &-\mathcal{I}(y(\mathbf{x_1}, \mathbf{x_2}) = 0) \log(p_1(\mathbf{x_1}, \mathbf{x_2})) \\ &-\mathcal{I}(y(\mathbf{x_1}, \mathbf{x_2}) = 1) \log(p_2(\mathbf{x_1}, \mathbf{x_2})) \end{aligned} \quad (5)$$

where $\mathcal{I}(.)$ is the indicator function defined as:

$$\mathcal{I}(t) = \begin{cases} 1, & \text{if } t \text{ is true} \\ 0, & \text{if } t \text{ is false} \end{cases}$$

and, $y(\mathbf{x_1}, \mathbf{x_2})$ is the true label for the sample $(\mathbf{x_1}, \mathbf{x_2})$, defined as:

$$y(\mathbf{x_1}, \mathbf{x_2}) = \begin{cases} 0, & \text{if } (\mathbf{x_1}, \mathbf{x_2}) \text{ is a genuine pair.} \\ 1, & \text{if } (\mathbf{x_1}, \mathbf{x_2}) \text{ is an impostor pair.} \end{cases} \quad (6)$$

Using Equation 6, we can write the error as

$$\begin{aligned} e(\mathbf{x_1}, \mathbf{x_2}) = &-(1 - y(\mathbf{x_1}, \mathbf{x_2})) \log(p_1(\mathbf{x_1}, \mathbf{x_2})) \\ &-y(\mathbf{x_1}, \mathbf{x_2}) \log(p_2(\mathbf{x_1}, \mathbf{x_2})) \end{aligned} \quad (7)$$

2) Cosine Embeddings Loss: We also analyze the performance of the network when we directly minimize a contrastive loss function between the embeddings. So, there is no need to add an extra fully connected layer at the end. The employed cosine embedding loss is defined below.

$$L_{cos}(\mathbf{x_1}, \mathbf{x_2}) = \begin{cases} 1 - C(\mathbf{f}(\mathbf{x_1}), \mathbf{f}(\mathbf{x_2})), & \text{if } y(\mathbf{x_1}, \mathbf{x_2}) = 0 \\ C(\mathbf{f}(\mathbf{x_1}), \mathbf{f}(\mathbf{x_2})), & \text{if } y(\mathbf{x_1}, \mathbf{x_2}) = 1 \end{cases}$$

Here $C(\mathbf{f}(\mathbf{x_1}), \mathbf{f}(\mathbf{x_2}))$ is the cosine similarity between $\mathbf{f}(\mathbf{x_1})$ and $\mathbf{f}(\mathbf{x_2})$ defined as

$$cos(\mathbf{x_1}, \mathbf{x_2}) = \frac{\mathbf{x_1}.\mathbf{x_2}}{||\mathbf{x_1}||_2 ||\mathbf{x_2}||_2}.$$

TABLE I
NPC TRAINING DATASETS

Name of the dataset	Size (hours)	Number of samples
Tedlium	100	358K
Tedlium-Mix	110	395K
YoUSCTube	584	2.1M

Here $||.||_2$ denotes the L_2 norm. In Section IV, performances of the two types of loss functions will be analyzed through experimental evidence.

D. Extracting NPC Embeddings for Test Audio

Once the DNN model is trained, we use it for extracting speaker embeddings from any test audio stream. As discussed in Section II-C, the transformation achieved by the siamese network on an input segment $\mathbf{x}$ of length d frames is given by $\mathbf{f}(\mathbf{x})$. We use only this siamese part of the network to transform a sequence of MFCC frames of any speech segment into NPC embeddings by using a sliding window $\mathbf{w}$ of d frames and shifting it by 1 frame along the entire sequence.

III. EVALUATION METHODOLOGY

The nature of the proposed method introduces a great challenge in its evaluation. All existing speaker identification methods employ some level of supervision. For example x-vector systems [43] employ data with complete speaker labels, while i-vector systems [16] require labeling of speaker-homogeneous regions.

In our proposed work we intend to establish a low-level speaker-specific feature, on which subsequent supervised methods or layers can operate.

Given the above evaluation challenge we perform two sets of comparisons with existing methods:

1) **Speaker identification evaluation**: Speaker identification (*i.e.,* closed set multi-class speaker classification) experiments are performed at different context lengths. In that case we compare with other low-level features such as MFCCs and statistics of MFCCs, as well as an i-vector system.
2) **Upper-bound comparison**: A large scale speaker verification experiment is done to set upper-bounds on performance by in-domain supervised methods. We present this to observe the margin of improvement of the proposed out-of-domain unsupervised method via additional higher level integration methods or layers. We note that our method only integrates 1 second level information ($d = 1\ s$) while the i-vector and x-vector upper-bound methods use all the available data in an utterance.

The experimental setting for NPC training and the above experiments is described below.

A. NPC Training Datasets

Table I shows the training datasets along with their approximate total durations and number of contrastive samples created from each dataset. We train three different models individually on these datasets, and we call every trained model by the name

of the dataset used for training along with the NPC prefix (for example, the NPC model trained on YoUSCTube data will be called as NPC YoUSCTube).

1) Tedlium Dataset: The Tedlium dataset is built from the Tedlium training corpora [58]. It originally contained 666 unique speakers, but we have removed the 19 speakers which are also present in the Tedlium development and test sets (since the Tedlium dataset was originally developed for speech recognition purposes, it has speaker overlap between train and dev/test sets). The contrastive samples created from the Tedlium dataset are less noisy (compared to the case for YoUSCTube data as will be discussed next), because most of the audio streams in the Tedlium data are from a single speaker talking in the same environment for long (although there is some noise, for example, speech of the audience, clapping *etc.*).

The reason for employing this dataset is two-fold: First, the model trained on the Tedlium data will provide a comparison with the models trained on the Tedlium-Mix and YoUSCTube datasets for a validation of the short-term speaker stationarity hypothesis. Second, since the test set of the speaker identification experiment will be from the Tedlium test data, this will help demonstrate the difference in performance for in-domain and out-of-domain evaluation.

2) Tedlium-Mix Dataset: The Tedlium-Mix dataset is created mainly to validate the short-term speaker stationarity hypothesis (please see Section IV-B). We create the Tedlium-Mix dataset by creating artificial dialogs through randomly concatenating utterances. Tedlium is annotated, so we know the utterance boundaries. We thus simulate a dialog that has a random speaker every other utterance of the main speaker. For every audio stream, we reject half of the total utterances, and between every two utterances we concatenate a randomly chosen utterance from a randomly chosen speaker (i.e. S, R_1, S, R_2, S, $R_3, \ldots$ where S's are the utterances of the main speaker and R_i (for $i = 1, 2, 3, \ldots$) is a random utterance from a randomly chosen speaker i.e. a random utterance from another Ted recording). In this way we create the Tedlium-Mix dataset having a speaker change after every utterance for every audio stream. It also has almost the same size as the Tedlium dataset.

3) YoUSCTube Dataset: A large amount of various types of audio data has been collected from YouTube to create the YoUSCTube dataset. We have chosen YouTube for this purpose because of virtually unlimited supply of data from diverse environments. The dataset has multilingual data including English, Spanish, Hindi and Telugu from heterogeneous acoustic conditions like monologues, multi-speaker conversations, movie clips with background music and noise, outdoor discussions *etc*.

4) Validation Data: The Tedlium development set (8 unique speakers) has been used as validation data for all training cases. We used the utterance start and end time stamps and the speaker IDs provided in the transcripts of the Tedlium dataset to create the validation set so that it does not have any noisy labels.

B. Data for Speaker Identification Experiment

The Tedlium test set (11 unique speakers from 11 different Ted recordings) has been employed for the speaker identification experiment. Similar to the development dataset, it has start and end time of every utterance for every speaker as well as the speaker IDs. We have extracted the utterances from every speaker, and all utterances of a particular speaker have been assigned the corresponding speaker ID. Those have been used for creating the experimental scenarios for speaker classification (Section IV-C2 and Section IV-C4). Similar to the validation set, the labels of this dataset are very clean since they are created using the human-labeled speaker IDs.

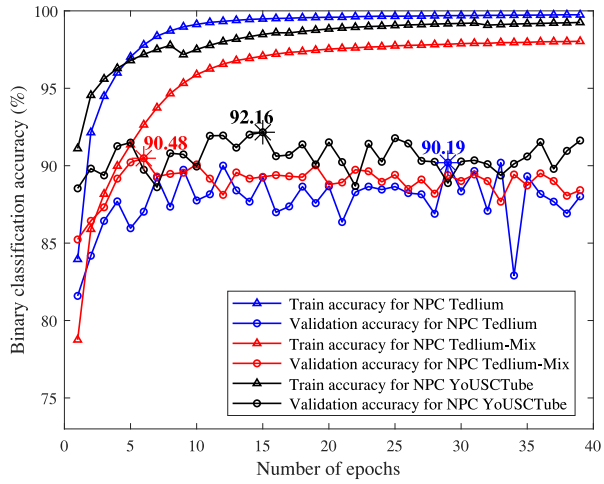


Fig. 3. Binary classification accuracies of classifying genuine or impostor pairs for NPC models trained on the Tedlium, Tedlium-Mix, and YoUSCTube datasets. Both training and validation accuracies are shown. The best validation accuracies for all the models are marked by big stars (*).

C. Data for Speaker Verification Experiment

A recently released large speaker verification corpus, VoxCeleb (version 1) [59] is employed for the speaker verification experiment. It has a total of 1251 unique speakers with $\sim$154 K utterances at 16 KHz sample rate. The average number of sessions per speaker is 18. We use the default development and test split provided with the dataset and mentioned in [59].

D. Feature and Model Parameters

We employ 40 dimensional MFCC features computed from 16KHz audio with 25 ms window and 10 ms shift using the Kaldi toolkit [60]. We choose $d = 100$ frames (1 s), and $\Delta = 200$ frames (2 s). Therefore, each window is a 100×40 matrix, and we feed this to the first CNN layer of our network (Fig. 2). The employed model has a total 1.8 M parameters and it has been trained using RMSProp optimizer [61] with a learning rate of 10^{-4} and a weight decay of 10^{-6}. The held out validation set (Section III-A4) has been used for model selection.

IV. Experimental Results

A. Convergence Curves

Fig. 3 shows the convergence curves in terms of binary classification accuracies of classifying genuine or impostor pairs for training the DNN model separately in different datasets along

with the corresponding validation accuracies. The development set for calculating the validation accuracy is same for all the training sets and it doesn't contain any noisy samples. In contrast, our training set is noisy since it's unsupervised and based on the short-term stationarity in assigning same/different class speaker pairs.

We can see from Fig. 3 that NPC Tedlium reaches almost 100% training accuracy, but NPC Tedlium-Mix converges at a lower training accuracy as expected. This is due to the larger portion of noisy samples present in the Tedlium-Mix dataset that arise from the artificially introduced fast speaker changes and the simultaneous hypothesis of short-term speaker stationarity.[2] However this doesn't hurt the validation accuracy on the development set, which is both distinct from training set and correctly labeled: we obtain 90.19% and 90.48% for NPC Tedlium and NPC Tedlium-Mix trained-models respectively. We believe this is because the model is correctly learning to *not* label some of the assumed same-speaker pairs as same-speaker when there is a speaker change that we introduced via our mixing, due to the large amounts of correct data that compensate for the smaller-amount of mislabeled pairs.

The NPC YoUSCTube model reaches much better training accuracy than the NPC Tedlium-Mix model even with fewer epochs. This points to both increased robustness due to the increased data variability and also that speaker-changes in real dialogs are not as fast as we simulated in the Tedlium-Mix dataset. It is interesting to see that the NPC YoUSCTube model achieves a little better validation accuracy (92.16%) than the other two models even when the training dataset had no explicit domain overlap with the validation data. We think, this is because of the huge size (approximately 6 times larger in size than the Tedlium dataset) and widely varying types of acoustic environments of the YoUSCTube dataset.

B. Validation of the Short-Term Speaker Stationarity Hypothesis

Here we analyze the validation accuracies obtained by the NPC models trained separately on the Tedlium and Tedlium-Mix datasets. From Fig. 3 it is quite clear that both models could achieve similar validation accuracies, although the Tedlium-Mix dataset has audio streams containing speaker changes at every utterance and the Tedlium dataset contains mostly single-speaker audio streams. The reason is that even though there are frequent speaker turns in the Tedlium-Mix dataset, the short length of context ($d = 100$ frames $= 1$ s) chosen to learn the speaker characteristics ensures that the total number of correct same-speaker pairs dominates the falsely-labeled same-speaker pairs. Therefore the sudden speaker changes are of little impact and do not deteriorate the performance of neural network on the development set. This result validates the short-term speaker stationarity hypothesis.

[2]The corpus is created by mixing turns. This means that there are 54,778 speaker change points in the 115 hours of audio. However in this case we assumed that there are no speaker changes in consecutive frames. If the change points were uniformly distributed then that would result in an upper-bound of 87%.

C. Experiments: *Speaker Identification Evaluation*

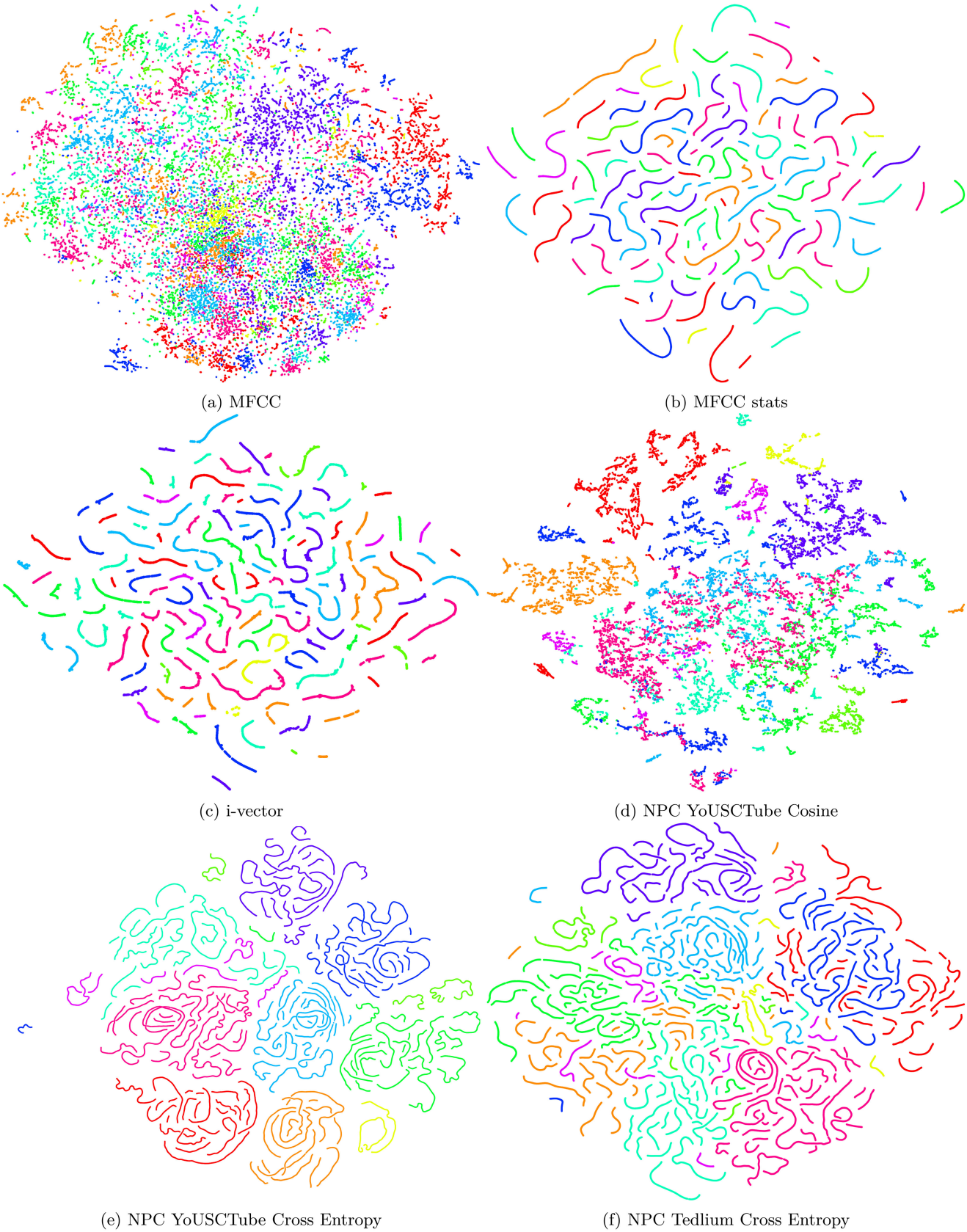
1) Frame-Level Embedding Visualization: Visualization of high dimensional data is vital in many scenarios because it can reveal the inherent structure of the data points. For speaker characteristics learning, visualizing the employed features can manifest the clusters formed around different speakers and thus demonstrate the efficacy of the features. We use t-SNE visualization [62] for this purpose. We compare the following features (the terms in **boldface** show the names we will use to call the features).

1) **MFCC**: Raw MFCC features.
2) **MFCC stats**: This is generated by moving a sliding window of 1 s along the raw MFCC features with a shift of 1 frame (10 ms) and taking the statistics (mean and standard deviation in the window) to produce a new feature stream. This is done for a fair comparison of MFCC and the embeddings (since the embeddings are generated using 1 s context).
3) **NPC YoUSCTube Cross Entropy**: Embeddings extracted with NPC YoUSCTube model using cross entropy loss.
4) **NPC Tedlium Cross Entropy**: Embeddings extracted with NPC Tedlium model using cross entropy loss.
5) **NPC YoUSCTube Cosine**: Embeddings extracted with NPC YoUSCTube model using cosine embedding loss.
6) **i-vector**: 400 dimensional i-vectors extracted independently every 1 s using a sliding window with 10 ms shift. The i-vector system (Kaldi VoxCeleb v1 recipe) was trained on the VoxCeleb dataset [59] (16 KHz audio). It is not possible to train an i-vector system on YoUSCTube since it contains no labels on speaker-homogeneous regions.

Fig. 4 shows the 2 dimensional t-SNE visualizations of the frames (at 10 ms resolution) of the above features extracted from the Tedlium test dataset containing 11 unique speakers. For better visualization of the data, we chose only 2 utterances from every speaker, and the feature frames from a total of 22 utterances become our input dataset for the t-SNE algorithm. From Fig. 4 we can see that the raw MFCC features are very noisy, but the inherent smoothing applied to compute MFCC stats features help the features of the same speaker to come closer. However we notice that although some same-speaker features cluster in lines, these lines are far apart in the space, which denotes that the MFCC features capture additional information. For example we see that the speaker denoted with Green occupies both the very left and very right parts of the t-SNE space.

The i-vector plot looks similar to the MFCC stats and does not cluster the speakers very well. This is consistent with existing literature [21] that showed that i-vectors do not perform well for short utterances especially when the training utterances are comparatively longer. In Section IV-C4, we will see that the utterance-level i-vectors perform much better for speaker classification.

The NPC YoUSCTube Cosine embeddings underperform the cross entropy-based methods possibly because of poorer convergence as we observed during training. They are also noisier than



(a) MFCC (b) MFCC stats

(c) i-vector (d) NPC YoUSCTube Cosine

(e) NPC YoUSCTube Cross Entropy (f) NPC Tedlium Cross Entropy

Fig. 4. t-SNE visualizations of the frames of different features for the Tedlium test data containing 11 speakers (2 utterances per speaker). Different colors represent different speakers.

TABLE II
FRAME-LEVEL SPEAKER CLASSIFICATION ACCURACIES OF DIFFERENT FEATURES WITH KNN CLASSIFIER (K = 1). ALL FEATURES BELOW ARE TRAINED ON UNLABELED DATA EXCEPT I-VECTOR WHICH REQUIRES SPEAKER-HOMOGENEOUS FILES.

# of Enrollment Utterances	Tedlium development set					
	MFCC	MFCC stats	NPC YoUSCTube Cross Entropy	NPC Tedlium Cross Entropy	NPC YoUSCTube Cosine	i-vector VoxCeleb
1	48.75	72.70	79.05	**80.25**	62.97	70.26
2	54.12	81.33	87.26	**88.32**	70.04	79.07
3	57.05	84.11	89.56	**89.62**	73.77	82.58
5	61.36	88.85	**92.34**	92.00	78.59	87.80
8	63.38	89.73	**91.62**	91.33	79.07	88.91
10	64.13	90.17	**92.42**	91.88	80.84	89.12
# of Enrollment Utterances	Tedlium test set					
	MFCC	MFCC stats	NPC YoUSCTube Cross Entropy	NPC Tedlium Cross Entropy	NPC YoUSCTube Cosine	i-vector VoxCeleb
1	38.02	70.45	75.62	**76.40**	56.30	64.02
2	44.08	79.43	**83.75**	83.21	58.18	74.50
3	46.39	81.98	**85.06**	84.79	59.05	76.76
5	50.24	86.20	**89.12**	88.65	62.18	81.65
8	51.56	87.70	**89.66**	89.07	64.33	84.21
10	52.65	88.46	**90.34**	89.94	65.79	88.13

MFCC stats and i-vectors, indicating that even a little change in the input (just 10 ms of extra audio) perturbs the embedding space, which might not be desirable.

The NPC YoUSCTube Cross Entropy and NPC Tedlium Cross Entropy embeddings provide much better distinction between different speaker clusters. Moreover, they also provide much better cluster compactness compared to the MFCC and i-vector features.

Among the NPC embeddings, NPC YoUSCTube Cross Entropy features provide possibly the best tSNE visualization. They even produce better clusters than NPC Tedlium Cross Entropy, although the latter one is trained on in-domain data. The larger size of YoUSCTube dataset might be the reason behind this observation.

2) Frame-Level Speaker Identification: We perform frame-level speaker identification experiments on the Tedlium development set (8 speakers) and the Tedlium test set (11 speakers). By frame-level classification we mean that every frame in the utterance is independently classified as to its speaker ID.

The reason for evaluating with frame-level speaker classification is that better frame-level performance conveys the inherent strength of the system to derive short-term features that carry speaker-specific characteristics. It also shows the possibility to replace MFCCs with the proposed embeddings by incorporating in systems such as [25] and [27].

Table II shows a detailed comparison between the 6 different features described in Section IV-C1 in terms of frame-wise speaker classification accuracies. We have tabulated the accuracies for different number of enrollment utterances (in other words training utterances for the speaker ID classifier) per speaker. We have used kNN classifier (with k = 1) for speaker classification. The reason for using such a naive classifier is to reveal true potential of the features, and not to harness strength of the classifier. We have repeated each experiment 5 times and the average accuracies have been reported here. Each time we have held out 5 random utterances from each speaker for testing. The same seen (enrollment) and test utterances have been used for all types of features and in all cases the test and enrollment sets are distinct.

From Table II we can see that MFCC stats perform much better than raw MFCC features. We think the reason is that the raw features are much noisier than the MFCC stats features because of the implicit smoothing performed during the statistics computation. The NPC YoUSCTube and NPC Tedlium models with cross entropy loss perform pretty similarly (for test data, NPC YoUSCTube even performs better) even though the former one is trained on out-of-domain data. This highlights the benefits and possibilities of employing out-of-domain unsupervised learning using publicly available data. NPC YoUSCTube Cosine doesn't perform well compared to other NPC embeddings. The i-vectors perform worse than NPC embeddings and MFCC stats for frame-level classification due to the reasons discussed in Section IV-C1 and as reported by [21].

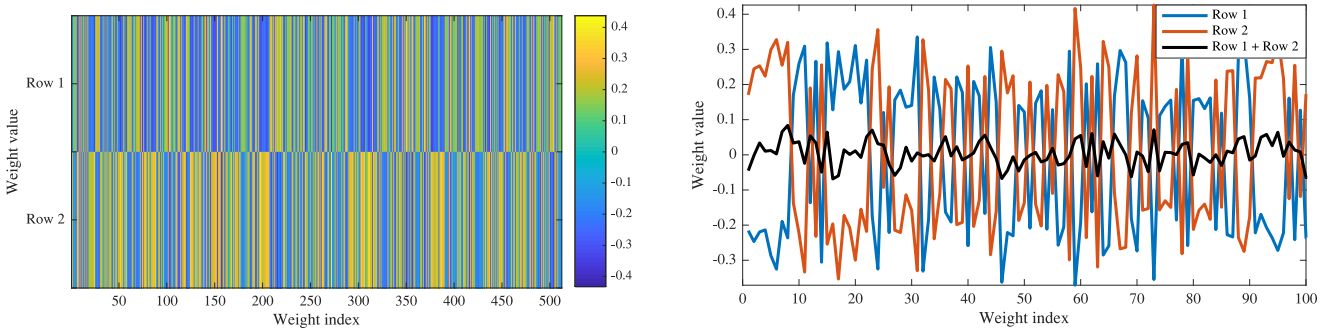
3) Analyzing Network Weights: We have seen in the previous experiments that the embeddings learned using the cross entropy loss performed better than those learned through minimizing the cosine loss. Here we analyze the learned weights in the last fully connected (FC) layer (size $= 512 \times 2$) in the network that uses cross entropy loss. From Fig. 5 we can see that the weights are learned in a way such that the weight value for a particular position of the first embedding, $w_{1,k}$, is approximately of same value and opposite sign of the weight value for that particular position in the second embedding, $w_{2,k}$ (please see Equation 2 for the notations). Experimentally, for NPC YoUSCTube Cross Entropy model, we found the mean of absolute value of $w_{1,k} + w_{2,k}$ to be 0.0284, with a standard deviation of 0.0206 (mean and standard deviations computed over all the embedding dimensions, *i.e.* k varying from 1 to $D = 512$). The two bias values we found are 1.0392 and -0.9876. In other words, the experimental evidence shows

$$w_{1,k} \approx -w_{2,k}$$
$$\text{and, } b_1 \approx -b_2.$$

One possible and intuitive explanation would be that the individual absolute weight values provide importance to different dimensions/features in the embedding (this has also been explained in [44]). The mirrored nature of weights and biases are possibly ensuring cancellation of same-speaker embeddings while ensuring maximization of impostor pair distance. For example, if $w_{1,k}$ is a high positive number then it ensures higher contribution of the kth dimension of $\mathbf{L}(\mathbf{x_1}, \mathbf{x_2})$ in the softmax output for the genuine pair class (since, $w_{1,k}|f(\mathbf{x_1})_k - f(\mathbf{x_2})_k|$ will be higher). On the other hand, $w_{2,k} \approx -w_{1,k}$ is ensuring "equally lower" contribution of $|f(\mathbf{x_1})_k - f(\mathbf{x_2})_k|$ to the probability of the input to be an impostor pair.

For the cosine embedding loss, these automatically learned importance weights are not present, which might be the reason for under performing the cross entropy embeddings; all embedding dimensions are equally contributing to the loss.

4) Utterance-Level Speaker Identification: Here we are interested in utterance-level speaker identification task.



(a) Two rows of the full 512×2 matrix shown as an image.

(b) Weights in the two rows plotted, along with their sum.

Fig. 5. Visualization of the learned weights in the last fully connected (FC) layer. (a) The matrix of the last FC layer. Note the opposite signs of the weight values in 'Row 1' and 'Row 2' for a particular weight index. (b) This figure shows a zoomed version of the figure in (a) for only 100 weights. Note the mirrored nature of the weights values in 'Row 1' and 'Row 2', and oscillation of their sum around zero.

TABLE III
UTTERANCE-LEVEL SPEAKER CLASSIFICATION ACCURACIES OF DIFFERENT FEATURES WITH KNN CLASSIFIER (K = 1). RED ITALICS INDICATES THE BEST PERFORMING SINGLE FEATURE CLASSIFICATION RESULT WHILE BOLD TEXT INDICATES THE BEST OVERALL PERFORMANCE.

# of Enrollment Utterances	Tedlium development set				Tedlium test set			
	MFCC stats	NPC YoUSCTube stats	i-vector VoxCeleb	NPC YoUSCTube + i-vector	MFCC stats	NPC YoUSCTube stats	i-vector VoxCeleb	NPC YoUSCTube + i-vector
1	75.12	82.12	***86.38***	85.62	80.00	83.27	*85.00*	**86.82**
2	83.00	87.88	*91.75*	**92.12**	87.64	92.36	*92.82*	**95.73**
3	84.88	89.88	***93.12***	**93.12**	92.18	*95.45*	94.82	**97.09**
5	91.25	94.50	*95.25*	**95.88**	92.09	*95.36*	93.91	**95.45**
8	92.12	95.00	*96.62*	**97.25**	95.27	***97.36***	95.82	**97.36**
10	92.50	*95.25*	95.12	**96.50**	96.54	*98.00*	96.73	**98.09**

We compare NPC YoUSCTube Cross Entropy (out-of-domain (OOD) YouTube), MFCC, and i-vector (OOD VoxCeleb) methods. For MFCC and NPC embeddings, we calculate the mean and standard deviation vectors over all frames in a particular utterance, and concatenate them to produce a single vector for every utterance. For i-vector, we calculate one i-vector for the whole utterance using the same i-vector system as mentioned in Section IV-C1.

We applied LDA (trained on development part of VoxCeleb) to project the 400 dimensional i-vectors to a 200 dimensional space. This gave better performance for i-vectors (also observed in literature [57]) and let us compare unsupervised NPC embeddings with the best possible i-vector configuration. We again classify using k-NN classifier with $k = 1$, as explained in Section IV-C2 to focus on the feature performance and not on the next-layer of trained classifiers.

Table III shows the classification accuracies for different features with increasing number of enrollment utterances.[3] In each enrollment scenario, 5 randomly held-out utterances from each of the 11 speakers have been used for testing, and the process has been repeated 20 times to report the average accuracies. Both i-vectors and NPC YoUSCTube embeddings perform similarly.

[3]Note that due to the small-size window for our feature, even two utterances provide significant information; hence we do not see significant change as the enrollment utterances increase.

TABLE IV
SPEAKER VERIFICATION ON VOXCELEB V1 DATA. I-VECTOR AND X-VECTOR USE THE FULL UTTERANCE IN A SUPERVISED MANNER FOR EVALUATION WHILE THE PROPOSED EMBEDDING OPERATES AT THE 1 SECOND WINDOW WITH A SIMPLE STATISTICS (MEAN+STD) OVER AN UTTERANCE.

Method	Training domain	Feature Context	Speaker labels	Speaker homogeneity	minDCF	EER(%)
i-vector [59]	ID	Full	No	Yes	0.73	8.80
x-vector	ID	Full	Full	Yes	**0.61**	**7.21**
NPC stats	OOD	1sec	No	No	0.87	15.54

It is interesting to note the complementarity of the concatenated i-vector-embedding feature. From Table III we can see that the NPC YoUSCTube Cross Entropy + i-vector perform the best for almost all the cases.

An additional important point is that the classifier used is the simple 1-Nearest Neighbor classifier. So, we believe that the highly non-Gaussian nature of the embeddings (as can be observed from Fig. 4) might *not* be captured well by the 1-NN since it is based on Euclidean distance which will under perform in complex manifolds as we observe with NPC embeddings. This motivates future work in higher-layer, utterance-based, neural network-derived features that build on top of these embeddings.

D. Experiments: Upper-Bound Comparison

Table IV compares performance of i-vector, x-vector [43], and the proposed NPC embeddings for the speaker verification task on Voxceleb v1 data using the default Dev and Test splits [59] distributed with the dataset.

We want to highlight that since the assumption for our system is that we have absolutely no labels during DNN training (in fact our YouTube downloaded data are not even guaranteed to be speech!), the comparison with x-vector or i-vector is highly asymmetric. To simplify this explanation:

- Our proposed method uses "some random audio": completely unsupervised and challenging data.
- i-vector uses "speech" with labels on "speaker homogeneous regions": unsupervised with a supervised step on clean data.

- x-vector uses "speech" with labels of "id of speaker": completely supervised on clean data.

Moreover, i-vector and x-vector here are trained on in-domain (ID) Dev part of the VoxCeleb dataset. On the other hand, the NPC model is trained out-of-domain (OOD) on unlabeled YouTube data. Please note that here "out-of-domain" refers to the generic characteristics of the YoUSCTube dataset compared to the Voxceleb dataset. For example, the Voxceleb dataset was mined using the keyword "interview" [59] along with the speaker name, and the active speaker detection [59] ensured active presence of that speaker in the video. On the other hand, the YoUSCTube dataset is mined without any constraints thus generalizing more to realistic acoustic conditions (see III-A3). Moreover, having only celebrities [59] in the Voxceleb dataset helped it to find multiple sessions of the same speaker, which subsequently helped the supervised DNN models to be more channel-invariant. However, such freedom is not available in the YoUSCTube dataset, thus paving a way to build unsupervised models that can be trained or adapted in such conditions.

Finally, the features employed by i-vector and x-vector employ the whole utterance of average length 8.2 s (min = 4 s, max = 145 s) [59] while the NPC model is only producing 1 second estimates. While we do intend to incorporate more contextual learning for longer sequences, in this work we are focusing on the low-level feature and hence employ statistics (mean and std) of the embeddings. This is suboptimal and creates an uninformed information bottleneck, however it is a necessary and easy way to establish an utterance-based feature, thus enabling comparison with the existing methods.

For all the above reasons we expect that any evaluation with i-vector and x-vector can only be seen as a very upper-bound and we dont expect to beat either of these two in performance.

The i-vector performance is as reported in [59]. No data augmentation is performed for x-vector for a fair comparison.

To maintain standard scoring mechanisms we employed LDA to project the embeddings on a lower dimensional space and, then PLDA scoring as in [43], [59]. The same VoxCeleb Dev data is utilized to train LDA and PLDA models for all methods for a fair comparison. The LDA dimension is 200 for x-vector and i-vector [59] systems, and 100 for NPC system. We report the minimum normalized detection cost function (minDCF) for $P_{target} = 0.01$ and Equal Error-Rate (EER). We can see that the best in-domain supervised method is 30% better than unsupervised NPC in terms of minDCF.

V. Discussion and Future Work

A. Discussion

Based on the visualization of Fig. 4 and the experiments of Section IV-C we have established that the resulting embedding is capturing significant information about the speakers' identity. The feature has shown to be quite better than using knowledge driven features such as MFCCs or statistics of MFCCs and even more robust than supervised features such as i-vector operating on 1 second windows. Importantly the proposed embedding showed extreme portability by operating better on the Tedlium dataset when trained on larger amounts of random audio from the collected YoUSCTube corpus than when trained in-domain on the Tedlium dataset itself.

Also importantly we have shown in Section IV-B that if we on purpose create a fast changing dialog by mixing the Tedlium utterances, the short-term stationarity hypotheses still holds. This encourages the use of unlabeled data.

Evaluating this embedding however is challenging as its use is not obvious until it is used for a full blown speaker identification framework. This requires several more stages of development that we will discuss further in this work, along with discussing the shortcomings of this embedding. However, we can, and we are, providing some early evidence that the embedding does indeed capture significant information about the speaker.

In Section IV-C4 we present results that compare an utterance-based classification system on the Tedlium data. We are comparing the i-vector system optimized for utterance-level classification, and which employs supervised data, with a very simple statistic (mean and std) of our proposed unsupervised embedding. We show that our embedding provides very robust results that are comparative to the i-vector system. The shortcoming of this comparison, is that the utterances are drawn from the Tedlium dataset, and they are likely also incorporating channel information. We provide some suggestions in overcoming this shortcoming further in this section.

We proceeded, in Section IV-D, to present results that compare an utterance-based verification system on the VoxCeleb v1 test. Here we wanted to provide an upper-bound comparison. We evaluated i-vector and supervised x-vector methods trained on VoxCeleb data. These methods are able to employ the full utterance as a single observation, while the proposed embedding only operates on a <1 second resolution, hence we again aggregate via an uninformed information bottleneck (mean and std). We see that despite the information bottleneck and complete unsupervised and out of domain nature of the experiment our proposed system still achieves an acceptable performance with a 30% worse minDCF than x-vector.

B. Future Work

The above observations and analysis provide many directions for future work.

Given that all our same-speaker examples come from the same channel, we believe that the proposed embedding captures both channel and speaker characteristics. This provides an opportunity for data augmentation, and hence reduction of the channel influence. In future work we intend to augment the near-by frames such that contextual pairs are coming from a range of different channels through augmentation.

This also provides another opportunity for joint channel and speaker learning. Through the above augmentation we can jointly learn same vs different speakers and same vs different channels, thus providing disentanglement and more robust speaker representations.

Further, triplet learning [31], especially with hard triplet mining, has been shown to provide improved performance and we intend to use such an architecture in future work to directly optimize intra- and inter-class distances in the manifold.

One additional opportunity for improvement is to employ a larger neural network. We employed a CNN with only 1.8 M parameters for our training (as an initial try to check the validity of the proposed method). But, recent CNN-based speaker verification systems employ much deeper networks (*e.g.,* VoxCeleb's baseline CNN comprises of 67 M parameters). We think utilizing recent state-of-the-art deep architectures will improve performance of the proposed technique for large scale speaker verification experiments.

Finally, and more applicable to the speaker ID task, we need embeddings that can capture information from longer sequences. As we see in Section IV-D the supervised speaker identification methods are able to exploit longer term context while the proposed embedding is only able to serve as a short-term feature. This requires either supervised methods, towards higher level information integration, or more in alignment with our interests of better unsupervised context exploitation. For example we can employ a better aggregation mechanism via unsupervised diarization using this embedding to identify speaker-homogeneous regions and then employ Recurrent Neural Networks (RNN) [22] towards longitudinal information integration.

VI. Conclusion

In this paper, we proposed an unsupervised technique to learn speaker-specific characteristics from unlabeled data that contain any kind of audio, including speech, environmental sounds, and multiple overlapping speakers of many languages.

The proposed system exploits the short-term active-speaker stationarity hypothesis to create contrastive samples from unlabeled data, and feed them into a deep convolutional siamese network which learns the NPC embeddings by learning to classify same vs different speaker pairs.

We trained the proposed siamese model on both the YoUSCTube and Tedlium training sets. We performed two sets of evaluation experiments: a closed set speaker identification experiment, and a large scale speaker verification experiment for upper-bound comparison. The NPC embeddings outperform i-vectors at frame-level speaker identification, and provide complementary information to i-vectors at the utterance-level speaker identification task.

As an upper-bound task we employed the VoxCeleb speaker verification set. As expected NPC embeddings underperform in-domain supervised x-vector and in-domain i-vector methods.

The analysis of the proposed out-of-domain unsupervised method with the in-domain supervised methods helps identify challenges and raises a range of opportunities for future work, including in longitudinal information integration and in introducing robustness to channel characteristics.

Acknowledgment

The U.S. Army Medical Research Acquisition Activity is the awarding and administering acquisition office. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the Department of Defense.

References

[1] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[2] S. Narayanan and P. G. Georgiou, "Behavioral signal processing: Deriving human behavioral informatics from speech and language," *Proc. IEEE*, vol. 101, no. 5, pp. 1203–1233, Sep. 2013.

[3] M. Kotti, V. Moschou, and C. Kotropoulos, "Speaker segmentation and clustering," *Signal Process.*, vol. 88, no. 5, pp. 1091–1124, 2008.

[4] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A review of recent research," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 2, pp. 356–370, Feb. 2012.

[5] K. Chen, "Towards better making a decision in speaker verification," *Pattern Recognit.*, vol. 36, no. 2, pp. 329–346, 2003.

[6] J. P. Campbell, "Speaker recognition: A tutorial," *Proc. IEEE*, vol. 85, no. 9, pp. 1437–1462, Sep. 1997.

[7] J. H. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 74–99, Nov. 2015.

[8] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[9] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoustical Soc. Amer.*, vol. 87, no. 4, pp. 1738–1752, 1990.

[10] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, N.J: PTR Prentice Hall, 1993.

[11] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling prosodic feature sequences for speaker recognition," *Speech Commun.*, vol. 46, no. 3, pp. 455–472, 2005.

[12] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Process.*, vol. 10, no. 1–3, pp. 19–41, 2000.

[13] P. Kenny and P. Dumouchel, "Disentangling speaker and channel effects in speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 1, 2004, pp. 1–37.

[14] P. Kenny, M. Mihoubi, and P. Dumouchel, "New map estimators for speaker recognition," in *Proc. Eurospeech*, Sep. 1–4, 2003, pp. 2691–2964.

[15] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Proc. Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 4, pp. 1435–1447, May 2007.

[16] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[17] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proc. 10th Annu. Conf. Int. Speech Commun. Assoc.*, 2009, pp. 1559–1562.

[18] G. Sell and D. Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *Proc. Spoken Lang. Technol. Workshop*, 2014, pp. 413–417.

[19] G. Dupuy, M. Rouvier, S. Meignier, and Y. Esteve, "I-vectors and ILP clustering adapted to cross-show speaker diarization," in *Proc. 13th Annu. Conf. Int. Speech Commun. Assoc.*, 2012, pp. 2174–2177.

[20] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 55–59.

[21] A. Kanagasundaram, R. Vogt, D. B. Dean, S. Sridharan, and M. W. Mason, "I-vector based speaker recognition on short utterances," in *Proc. 12th Annu. Conf. Int. Speech Commun. Assoc.*, 2011, pp. 2341–2344.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[23] M. Rouvier, P.-M. Bousquet, and B. Favre, "Speaker diarization through speaker embeddings," in *Proc. 23rd Eur. Signal Process. Conf.*, 2015, pp. 2082–2086.

[24] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[25] T. Yamada, L. Wang, and A. Kai, "Improvement of distant-talking speaker identification using bottleneck features of DNN," in *Proc. INTERSPEECH*, 2013, pp. 3661–3664.

[26] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 4052–4056.

[27] S. H. Ghalehjegh and R. C. Rose, "Deep bottleneck features for i-vector based text-independent speaker verification," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2015, pp. 555–560.
[28] K. Chen and A. Salman, "Learning speaker-specific characteristics with a deep neural architecture," *IEEE Trans. Neural Netw.*, vol. 22, no. 11, pp. 1744–1756, Dec. 2011.
[29] K. Chen and A. Salman, "Extracting speaker-specific information with a regularized siamese deep network," in *Proc. Advances Neural Inf. Process. Syst.*, 2011, pp. 298–306.
[30] 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 *Proc. IEEE Spoken Lang. Technol. Workshop*, 2016, pp. 165–170.
[31] C. Li *et al.*, "Deep speaker: An end-to-end neural speaker embedding system," 2017, *arXiv:1705.02304*.
[32] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[33] M. M. Saleem and J. H. Hansen, "A discriminative unsupervised method for speaker recognition using deep learning," in *Proc. IEEE 26th Int. Workshop Mach. Learn. Signal Process.*, 2016, pp. 1–5.
[34] X.-L. Zhang, "Multilayer bootstrap network for unsupervised speaker recognition," 2015, *arXiv:1509.06095*.
[35] I. Lapidot, H. Guterman, and A. Cohen, "Unsupervised speaker recognition based on competition between self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 877–887, Jul. 2002.
[36] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1096–1104.
[37] J. S. Garofolo *et al.*, "TIMIT acoustic-phonetic continuous speech corpus," 1993. [Online]. Available: https://catalog.ldc.upenn.edu/LDC93S1
[38] D. O'Shaughnessy, "Linear predictive coding," *IEEE Potentials*, vol. 7, no. 1, pp. 29–32, Feb. 1988.
[39] H. Li, B. Baucom, and P. Georgiou, "Unsupervised latent behavior manifold learning from acoustic features: Audio2behavior," in *Proc. IEEE Int. Conf. Audio, Speech, Signal Process.*, New Orleans, LA, USA, Mar. 2017, pp. 5620–5624.
[40] A. Jati and P. Georgiou, "Speaker2vec: Unsupervised learning and adaptation of a speaker manifold using deep neural networks with an evaluation on speaker segmentation," in *Proc. INTERSPEECH*, Aug. 2017, pp. 3567–3571.
[41] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, vol. 1, 2005, pp. 539–546.
[42] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, vol. 2, 2006, pp. 1735–1742.
[43] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2018, pp. 5329–5333.
[44] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. Int. Conf. Mach. Learn. Deep Learn. Workshop*, vol. 2, 2015.
[45] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proc. Advances Neural Inf. Process. Syst.*, 1994, pp. 737–744.
[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[48] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.
[49] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[50] M. McLaren, Y. Lei, N. Scheffer, and L. Ferrer, "Application of convolutional neural networks to speaker recognition in noisy conditions," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 686–690.
[51] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *Proc. IEEE 26th Int. Workshop Mach. Learn. Signal Process.*, 2016, pp. 1–6.
[52] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," *CoRR*, vol. abs/1609.09430, 2016. [Online]. Available: http://arxiv.org/abs/1609.09430
[53] S.-Y. Chang and N. Morgan, "Robust CNN-based speech recognition with Gabor filter kernels," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 905–909.
[54] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, vol. 30, 2013.
[55] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: Proceedings of Machine Learning Research, Jul. 07–09, 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html
[56] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[57] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," *Proc. INTERSPEECH*, 2017, pp. 999–1003.
[58] A. Rousseau, P. Deléglise, and Y. Esteve, "TED-LIUM: An automatic speech recognition dedicated corpus," in *Proc. Conf. Lang. Resour. Eval.*, 2012, pp. 125–129.
[59] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," 2017, *arXiv:1706.08612*.
[60] D. Povey *et al.*, "The kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, Dec. 2011.
[61] T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.
[62] L. Van Der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

Arindam Jati received the M.S. degree in electrical engineering from the University of Southern California (USC), Los Angeles, CA, USA, in 2017, and the B.E. degree in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 2013. He is currently working toward the Ph.D. degree in electrical engineering with USC.

He is broadly interested in machine learning and speech processing. His current research focuses on unsupervised learning and adaptation, speaker recognition, audio event detection, and human-centered signal and information processing. He was the recipient of the USC Annenberg Fellowship during 2015–2019, and the ISCA travel grant award for students and young scientists for InterSpeech 2017.

Panayiotis (Panos) Georgiou is currently an Assistant Professor in electrical engineering and computer science with the University of Southern California (USC), Los Angeles, CA, USA, the Director of the Signal Processing for Communication Understanding and Behavior Analysis (SCUBA), the Co-Director of the USC Behavioral Informatics Center, and an integral member of Signal Analysis and Interpretation Lab. He has authored/coauthored more than 200 papers and his co-authored papers won 3 best paper awards and an Interspeech Paralinguistics Challenge award and has 6 patents. His work has been featured in more than 100 national and international media outlets such as Washington Post, Telegraph U.K., US News and World report, etc. His current research interests include behavioral signal processing, speech processing, NLP, and machine learning. He is currently a member of IEEE-SLTC, an Editor for IEEE SIGNAL PROCESSING LETTERS, IEEE SIGNAL PROCESSING MAGAZINE, *EURASIP Journal on Audio, Speech, and Music Processing*, and *Advances in Artificial Intelligence* and was a Guest Editor for Computer Speech and Language. He has also served or serves as the Technical Chair of InterSpeech 2016; General Chair of ICMI 2018, Area Chair of InterSpeech 2015, 2017, and 2018; and a range of other speech and signal processing conferences.