Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

Neural i-vectors

Ville Vestman[1], *Kong Aik Lee*[2], *Tomi H. Kinnunen*[1]

[1]Computational Speech Group, University of Eastern Finland, Finland
[2]Biometrics Research Laboratories, NEC Corporation, Japan
vvestman@cs.uef.fi, kongaik.lee@nec.com, tkinnu@cs.uef.fi

Abstract

Deep speaker embeddings have been demonstrated to outperform their generative counterparts, i-vectors, in recent speaker verification evaluations. To combine the benefits of high performance and generative interpretation, we investigate the use of deep embedding extractor and i-vector extractor in succession. To bundle the deep embedding extractor with an i-vector extractor, we adopt aggregation layers inspired by the Gaussian mixture model (GMM) to the embedding extractor networks. The inclusion of GMM-like layer allows the discriminatively trained network to be used as a provider of sufficient statistics for the i-vector extractor to extract what we call *neural i-vectors*. We compare the deep embeddings to the proposed neural i-vectors on the Speakers in the Wild (SITW) and the Speaker Recognition Evaluation (SRE) 2018 and 2019 datasets. On the core-core condition of SITW, our deep embeddings obtain performance comparative to the state-of-the-art. The neural i-vectors obtain about 50% worse performance than the deep embeddings, but on the other hand outperform the previous i-vector approaches reported in the literature by a clear margin.

1. Introduction

Automatic speaker verification (ASV) systems extract speaker-related information from a pair of speech recordings (enrollment and test) to decide whether the speakers in the two recordings are the same. This is done by computing similarity score between speaker-related features in the two recordings. While the base features have remained the same for decades [1], extraction and comparison of speaker traits from these features has coevolved with advances in machine learning. Much of ASV research has focused on modeling low-level speech feature distributions via *Gaussian mixture models* (GMMs) [2, 3, 4, 5]. Common to models such as GMM with *universal background model* (GMM-UBM) [3], *joint factor analysis* (JFA) [4] and *i-vector* [5] is the use of GMM to model acoustic features within recording(s).

What has changed throughout the years, however, is how speaker comparison is carried out. In the classic GMM pipelines [2, 3], features in the enrollment utterance(s) are used to train a speaker-dependent GMM, and comparison consists of evaluating the likelihoods of the target speaker model and the UBM to form an average log-likelihood ratio over all frames. In contrast to these *frame-based* approaches, the modern approach is to first represent the enrollment and test utterances as vectors of the same dimensionality. They can then be compared using a simple inner product, or a trainable classifier [6]. How these vectors are defined (and called) has changed throughout the years. The early approaches, driven by the success of GMMs, used high-dimensional GMM *supervectors* [7] with inner product scoring, typically implemented using *support vector machines* (SVMs). Through base work in [4], this was followed up by the highly-successful *i-vector* framework [5] where GMM supervectors are presented as points in a low-dimensional latent subspace. Following trends in deep learning, the focus has recently shifted towards deep neural network (DNN) based features [8], called nowadays *embeddings*. The idea to represent utterances as vectors, however, is the same as before, with the same back-end classifiers [9] used with GMM- and neural network based embeddings.

As the title suggests, we focus on i-vector extraction along the lines of classic GMM-based pipelines, but with a 'neural twist'. The general idea, of course, is not new. The three building blocks of any GMM-based method are (a) **a frame-level feature extractor** (*e.g.* MFCC extractor), (b) **a dictionary** (*e.g.* a UBM), and (c) **a posterior estimator** (*e.g.* feature vector alignment to dictionary components), each of which has been successfully replaced in prior work by their neural versions [10, 11]. In contrast to these studies that have focused either on replacing one or two of the components only, or using GMM-inspired components [12, 13, 14] to implement neural embedding extractors, we obtain all the three as 'side-products' from a neural network and proceed with conventional i-vector extractor training on top of them. Noting that (a), (b) and (c) are the only needed building blocks of *any* GMM-based embedding — be it a GMM-UBM [3], JFA [4], GMM-supervector [7], or i-vector [5] — this opens up a pathway to re-address any of the classic pipelines, still respecting the undeniable performance gains demonstrated by the recent neural approaches.

Our focus on i-vectors is arbitrary and the goal of our work is *not* to improve upon state-of-the-art in deep neural network based speaker embeddings. Instead, we aim to demonstrate that classic GMM-based ASV pipelines may not be inferior because of their model structures *per se*, but in the adoption of generic (nondiscriminative) elements. Classic frame-based GMM approaches have certain, nearly forgotten advantages, such as the ability to provide 'partial' scores at a fine temporal scale — the frame level. This might be particularly useful in speaker diarization (not addressed here) and speaker recognition from short utterances. Even if DNN embeddings appear to perform well in short duration ASV tasks [15], we argue that using GMMs retains all the benefits of generative modeling, such as the possibility to do sampling and obtaining uncertainty estimates for features and speaker embeddings. These add up to transparency and explainability demanded with increasing frequency from any machine learning system.

2. Modern speaker embedding extractors

Deep neural networks used for extracting speaker discriminative embeddings typically consist of three main parts (see Figure 1). The first part of the network operates on *frame-level*

10.21437/Odyssey.2020-10

features as an input in order to construct discriminative features from short time contexts, ranging from 100 milliseconds up to a few seconds. The frame-level layers are followed by the second main component, *temporal aggregation layer*, which converts the variable length input feature vector sequence to a fixed-dimensional representation. Finally, the last part of the network, which consists of one or two feedforward layers and the output layer, acts as a *classifier* for speaker identities. The speaker embeddings are usually extracted from the first fully connected layer after the aggregation layer [8].

Each of the three main parts can be implemented in multiple different ways. The frame-level component is often implemented as 1D convolutional neural network (CNN) [16], 2D CNN [17], or as some variant of time-delay neural network [18]. In 1D CNN, the convolution kernel slides over the temporal dimension (frames), whereas in 2D convolution, the kernel slides over both time and frequency dimensions.

There are two commonly used approaches for temporal aggregation. In the first approach [8], relatively *high-dimensional* features are obtained from CNN/TDNN, which are then aggregated by computing the (sample) mean and the standard deviation of the feature vectors over time. The output of the aggregation layer is then formed by concatenating the mean and standard deviation vectors. The second approach [12] of aggregation assumes relatively *low-dimensional* features (akin to conventional hand-crafted acoustic features) from CNN/TDNN but assigns them into multiple clusters (see Figure 2). Here, the aggregation is performed for each cluster separately, resulting in locally aggregated descriptor vectors. Finally, the locally aggregated descriptors are concatenated to form a higher-dimensional residual vector. This approach is analogous to the process how GMM mean supervectors are formed in the GMM-UBM framework.

3. Cluster-wise temporal aggregation

We focus on cluster-wise temporal aggregation methods as they offer a natural pathway to utilize GMM-based speaker verification approaches, such as the i-vector approach, together with discriminatively trained features. In the following, we consider two recent aggregation methods known as *learnable dictionary encoder* (LDE) [19, 12] and *NetVLAD* [20, 13], where VLAD is an acronym for "vector of locally aggregated descriptors". As we will show below, both can be regarded as discriminatively trained GMM-supervector [7] encoders with specific assumptions.

Let us first recall the formula for *posterior* computation of a Gaussian mixture component given a feature vector $\boldsymbol{x}_t$ (time index $t = 1, \ldots, T$). By letting $\boldsymbol{\theta} = \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, w_c\}_{c=1}^{C}$ be a GMM of C components with mean vectors $\boldsymbol{\mu}_c$, covariance matrices $\boldsymbol{\Sigma}_c$, and component weights w_c, we can compute the posteriors as follows:

$$\gamma_{c,t} = P(c|\boldsymbol{x}_t) = \frac{w_c \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}{\sum_{l=1}^{C} w_l \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}, \quad c = 1, \ldots, C. \tag{1}$$

By denoting

$$\beta_c = \log\left(\frac{w_c}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_c|}}\right), \tag{2}$$

where D is the dimension of feature vectors, we can expand (1) to form

$$\gamma_{c,t} = \frac{\exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_c)^\mathsf{T} \boldsymbol{\Sigma}_c^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}_c) + \beta_c\right]}{\sum_{l=1}^{C} \exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_l)^\mathsf{T} \boldsymbol{\Sigma}_l^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}_l) + \beta_l\right]}. \tag{3}$$

Table 1: *Comparison of LDE and NetVLAD.*

Computation step	LDE	NetVLAD
Posterior computation	Eq. (4)	Eq. (7)
Cluster-wise representations	Eq. (5)	Eq. (8)
Supervector normalization	—	Length-norm

3.1. Learnable dictionary encoder

Equation (3) holds for any GMM with unrestricted covariance matrices. In the following, we consider special cases, where the covariance matrices are restricted to have specific forms. First, by assuming *isotropic* covariance matrices (*i.e.*, $\boldsymbol{\Sigma}_c = s_c \mathbf{I}$, with $s_c > 0$), (3) becomes

$$\gamma_{c,t} = \frac{\exp\left[-\frac{1}{2} s_c \|\boldsymbol{x}_t - \boldsymbol{\mu}_c\|^2 + \beta_c\right]}{\sum_{l=1}^{C} \exp\left[-\frac{1}{2} s_l \|\boldsymbol{x}_t - \boldsymbol{\mu}_l\|^2 + \beta_l\right]}, \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm. This is the formulation used for posterior computation in [18] with LDE. Earlier works on LDE [19, 12], did not include the bias terms β_c. Both the parameters s_c that define the isotropic covariance matrices, and the bias terms β_c as well as the cluster centroids $\boldsymbol{\mu}_c$ are learnable parameters of the LDE layer.

After computing the posteriors, the construction of the output of LDE layer is a two step process. First, the input features are temporally aggregated with respect to each cluster. This is done by computing the weighted means $\boldsymbol{m}_c$ of residuals $\boldsymbol{\mu}_c - \boldsymbol{x}_t$ around the cluster centroids for each cluster c:

$$\boldsymbol{m}_c = \frac{\sum_{t=1}^{T} \gamma_{c,t} (\boldsymbol{\mu}_c - \boldsymbol{x}_t)}{\sum_{t=1}^{T} \gamma_{c,t}}. \tag{5}$$

The second step is to concatenate the cluster-wise representations to form a supervector $\boldsymbol{m} = (\boldsymbol{m}_1^\mathsf{T}, \boldsymbol{m}_2^\mathsf{T}, \ldots \boldsymbol{m}_T^\mathsf{T})^\mathsf{T}$, which is the output of the LDE layer.

While the original formulation of LDE uses separate *isotropic* covariance matrices for each component, it is straightforward to modify the LDE layer to operate with *diagonal* covariance matrices, or to use one *shared diagonal* or *spherical* covariance matrix for all components. In our experiments, we consider only the shared diagonal matrix formulation besides the original formulation with non-shared spherical covariances to limit the computational burden.

3.2. NetVLAD encoder

Let us next assume *shared full covariance matrices* (*i.e.*, $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}\ \forall c$), which will lead to the NetVLAD formulation of posterior computation. The shared covariance assumption simplifies (3) to

$$\gamma_{c,t} = \frac{\exp\left[\boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{x}_t + \log(w_c) - \frac{1}{2} \boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c\right]}{\sum_{l=1}^{C} \exp\left[\boldsymbol{\mu}_l^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{x}_t + \log(w_l) - \frac{1}{2} \boldsymbol{\mu}_l^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_l\right]}, \tag{6}$$

which allows us to write

$$\gamma_{c,t} = \frac{\exp\left[\boldsymbol{\omega}_c^\mathsf{T} \boldsymbol{x}_t + \psi_c\right]}{\sum_{l=1}^{C} \exp\left[\boldsymbol{\omega}_l^\mathsf{T} \boldsymbol{x}_t + \psi_l\right]}, \tag{7}$$

where

$$\boldsymbol{\omega}_c = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c \quad \text{and} \quad \psi_c = \log(w_c) - \frac{1}{2} \boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c.$$

Equation (7) can be implemented as an affine transform followed by softmax operation over clusters, which is exactly what is done in NetVLAD layer to compute the posteriors. The NetVLAD layer has $\boldsymbol{\omega}_c$, ψ_c, and $\boldsymbol{\mu}_c$ as its learnable parameters.

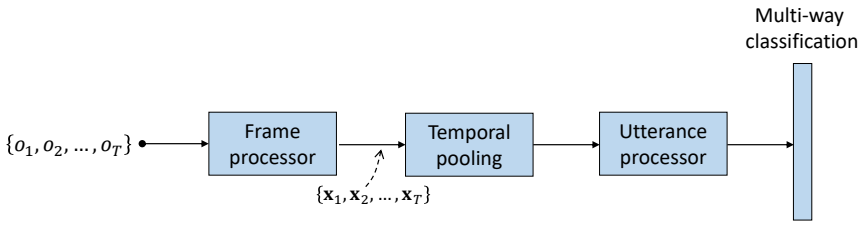

Figure 1: *An x-vector extractor consists of three functional blocks: a frame-level processor, a temporal pooling layer, and classifier. X-vector embeddings are derived from the affine transformation after the pooling layer.*

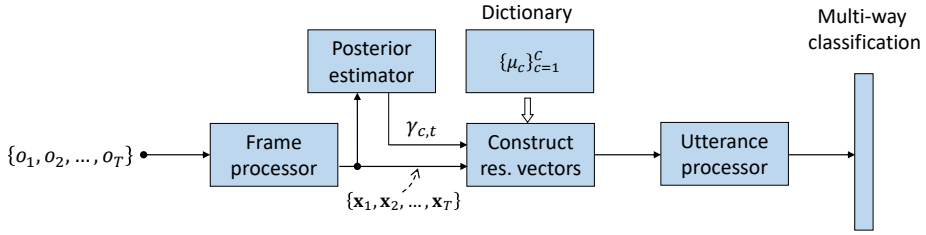

Figure 2: *The centrepiece of learnable dictionary encoder (LDE) and NetVLAD is the frame processor, frame posterior estimator and dictionary that are trained jointly to minimize a classification loss.*

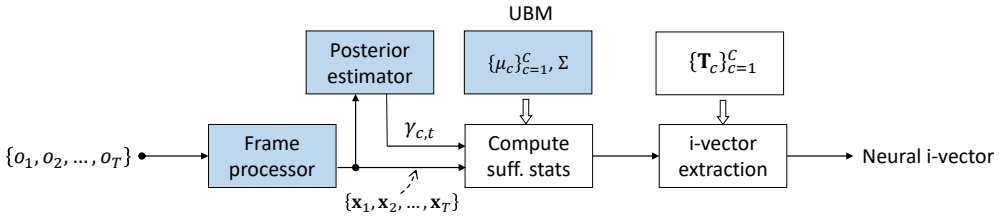

Figure 3: *The proposed neural i-vector relies on a deep structured front-end (shaded boxes) to extract sufficient statistics, which are then used for generative embedding.*

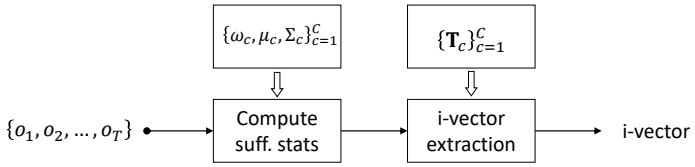

Figure 4: *An i-vector extractor is built upon a Universal Background Model (UBM) defined by the parameter set consists of weights, mean vectors, and covariance matrices.*

In terms of the number of learnable parameters, the correspondence between NetVLAD and GMM with shared covariances is not exact as covariance matrix $\mathbf{\Sigma}$ contains $D(D+1)/2$ free parameters, whereas a matrix containing all $\boldsymbol{\omega}_c$ vectors has CD parameters.

In NetVLAD, the construction of the output supervector differs from LDE in two ways. First, NetVLAD length-normalizes the component-wise outputs:

$$\boldsymbol{m}_c = \frac{\sum_{t=1}^{T} \gamma_{c,t}(\boldsymbol{\mu}_c - \boldsymbol{x}_t)}{\left\| \sum_{t=1}^{T} \gamma_{c,t}(\boldsymbol{\mu}_c - \boldsymbol{x}_t) \right\|}. \tag{8}$$

The second difference is that the supervector $\boldsymbol{m}$, obtained by concatenating the cluster-wise outputs $\boldsymbol{m}_c$, is further length-normalized to unit sphere.

The differences between LDE and NetVLAD are summarized in Table 1. As LDE and NetVLAD differ in the posterior computation as well as whether or not length-normalizations are applied, it is challenging to identify the potential causes of the performance difference between the two methods (if such difference is to exist). Therefore, we also study a hybrid approach (referred as NetVLAD/LDE), in which the posterior computation follows the NetVLAD approach, while the rest of the steps follow the LDE approach.

4. Utilizing aggregation statistics for i-vector extraction

Deep speaker embedding [8] has been demonstrated to outperform the i-vector representation shown in Figure 4. The enhanced performance is attained by (1) training the network using large amount of training data via *data augmentation*, and (2) *discriminative training* (e.g., multi-class cross entropy cost, angular margin cost [21]). The drawback is lack of generative interpretation. We propose to combine the benefits from both sides, leading to the so-called **neural i-vector** shown in Figure 3. In the neural i-vector, we utilize the features, posterior estimator, and the UBM that all have been trained discriminatively using speaker labels. This differs from the DNN i-vector

presented in [11] as it requires senone labels and does not utilize discriminatively trained features.

To extract neural i-vectors, we do not compute (5) or (8), but instead compute the suffiecient statistics in a standard way [22] as follows:

$$z_c = \sum_{t=1}^{T} \gamma_{c,t}, \tag{9}$$

$$\boldsymbol{f}_c = \sum_{t=1}^{T} \gamma_{c,t} \boldsymbol{x}_t, \tag{10}$$

$$\mathbf{S}_c = \sum_{t=1}^{T} \gamma_{c,t} \boldsymbol{x}_t \boldsymbol{x}_t^{\mathsf{T}}. \tag{11}$$

Here the features and posteriors are extracted from the embedding extractor network. The obtained statistics can be then easily used with any available i-vector code to train the i-vector extractor and to extract the i-vectors.

5. Speaker verification experiments

5.1. Network architectures and training procedure

The network architectures designed for this study are all derived from the standard x-vector architecture presented in [8]. Our most elementary architecture differs from [8] in the following ways:

- As in [16], we use *non-dilated* 1D CNN instead of TDNN with dilations used in [8].
- As in [16], we use *leaky* rectified linear unit (LReLU) activations (with slope of 0.01) instead of ReLUs.
- We have only one embedding layer (rather than two) after the aggregation layer. In our preliminary experiments, we did not find adding another layer to decrease the resulting speaker verification equal error rates.

We extend our default network (referred as TDNN) by adding *squeeze-and-excitation* (SE) modules [23] to the TDNN layers. The SE module aims to improve the representative power of hidden features by reweighting them using information from global temporal statistics of features. Using the terminology of [24], we adopt SE modules to perform *temporal squeeze* and *channel (feature) excitation*. That is, the output features of 1D CNN layer are weighted by factors computed from temporally pooled (non-weighted) features. Our implementation of the SE module is depicted in Figure 5, while Figure 6 illustrates how the SE module is added to the TDNN layer. The resulting TDNN-SE network architecture is presented in Table 2. Our SE module differs from the original in that it computes standard deviations in addition to means during the squeeze phase. In the excitation phase, we add batch normalization between the fully connected layers, as shown in Figure 5.

Inspired by the widely used ResNet architecture [25], our next network variant includes *residual* modules. Our implementation of a residual module (referred as TDNN-RES-SE) includes a fully connected layer, a 1D convolutional layer, and a SE module, as depicted in Figure 7. The network architecture is shown in Table 3. We replace neither the first nor the last TDNN-SE layer with the residual modules, as residual modules require the number of input and output features to be the same. The first layer has relatively low-dimensional MFCCs as its input, while the output size of the last layer depends on the aggregation method used. Networks with the mean and standard deviation pooling produce 1500-dimensional feature vectors at the output of the last TDNN layer. Networks with LDE or NetVLAD, in turn, produce TDNN outputs of 128-dimensions. With LDE and NetVLAD, we use 64 clusters, resulting in 8192-dimensional output vectors from the aggregation layer.

Table 2: *The architecture of TDNN-SE network.*

#	Layer type	CNN kernel size	Output dim.
1	TDNN-SE	5	512
2	TDNN-SE	5	512
3	TDNN-SE	7	512
4	TDNN-SE	1	512
5	TDNN-SE	1	1500
6	Aggregation	—	3000
7	FC-LReLU-BN	—	512
8	FC-softmax	—	#speakers

Table 3: *The architecture of TDNN-RES-SE network. The output sizes of the last TDNN-SE layer and the aggregation layer depend on the aggregation method. If aggregation using means and standard deviations is used, these sizes are* 1500 *and* 3000*, but if LDE or NetVLAD is used, the sizes are* 128 *and* 8192*.*

#	Layer type	CNN kernel size	Output dim.
1	TDNN-SE	5	512
2,3	TDNN-RES-SE	5	512
4,5	TDNN-RES-SE	7	512
6,7	TDNN-RES-SE	1	512
8	TDNN-SE	1	1500/128
9	Aggregation	—	3000/8192
10	FC-LReLU-BN	—	512
11	FC-softmax	—	#speakers

All our networks are implemented with PyTorch [26]. The Kaldi toolkit [27] is used to extract speech activity labels and 60-dimensional MFCCs (without delta features), used as the input features. PyKaldi [28] is used to load the features in Kaldi format in Python and to perform *cepstral mean normalization* (CMS) for the features.

For network training, we use four second long segments selected from random positions of the training utterances. During training, we feed about 14 000 short segments from each training speaker to the network in minibatches of size 64. Network weights are updated to minimize cross-entropy loss using stochastic gradient descend optimizer with weight decay parameter set to 0.001. We use a learning rate schedule that decreases the learning rate from 0.05 to 0.0002 during the training.

5.2. Neural i-vector training details

We utilized the *augmented* form of i-vector extractor as described in [29][1]. In the augmented form, the UBM mean vectors are augmented into the first column of the *total variability matrix* $\mathbf{T}$ and they are thus updated after the each iteration of extractor training, unlike in the standard formulation. For modeling residual covariances in the total variability model, we used a diagonal covariance matrix that was shared between all components. To initialize the first column of $\mathbf{T}$ and the residual covariance matrix, we used means and covariances computed from the sufficient statistics (9), (10), and (11) of the training data. We set the i-vector dimension to 512, which is the same as the dimension of the network embeddings.

[1]The PyTorch re-implementation of Kaldi's i-vector extractor used in our study is available at `https://github.com/vvestman/pytorch-ivectors`

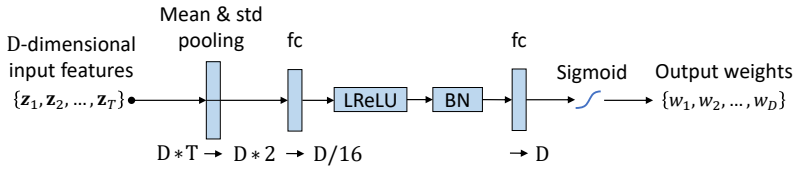

Figure 5: *The squeeze-and-excitation (SE) module. The output weights are used to weight the input features as shown in Figs. 6 and 7.*

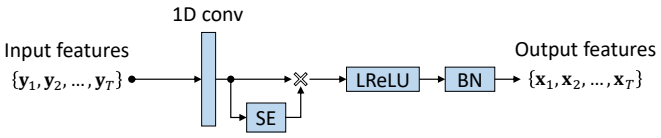

Figure 6: *A TDNN module with squeeze-and-excitation (SE). This module is used to build the frame processor of TDNN-SE network.*

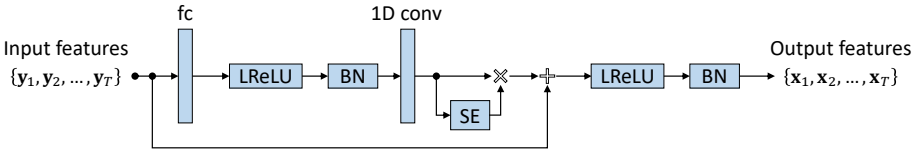

Figure 7: *A Residual module with squeeze-and-excitation (SE). This module is used to build the frame processor of TDNN-RES-SE network. Compared to the module in Figure 6, this module adds a fully connected layer (fc) and a residual connection. The residual connection adds input features to the features obtained from the SE operation.*

5.3. Training data

To train the neural networks, i-vector extractors, and scoring back-ends, we used 16 kHz speech data from VoxCeleb1 [30] and VoxCeleb2 [17]. VoxCeleb data has been collected from YouTube by automatic means. Like in [18], we concatenated all the segments that were extracted from the same YouTube source video, and used these concatenated segments as the training data. We excluded all concatenated segments less than six seconds long. After filtering out the short segments, we were left with data extracted from 149 754 unique YouTube videos, containing 7365 speakers. This data was then augmented five-fold using Kaldi's augmentation recipe, resulting in total of 748 770 concatenated segments. The augmentation creates copies of data by reverberating speech or by adding noise, babble, or music to the speech.

5.4. Evaluation data and metrics

We evaluated all the ASV systems on *Speakers in the Wild* (SITW) [31] and *NIST Speaker Recognition Evaluation* (SRE) 2018 [32] and 2019 [33] data. From SITW, we selected core-core (referred here as 'core') and core-multi (referred as 'multi') conditions. In both, only a single speaker appears in each of the enrollment segments, but in the multi condition, the test utterances may contain speech from multiple speakers (unlike in the core condition). The core condition evaluation contains 721 788 trials, out of which 3658 are target trials. The multi condition contains 2 010 683 trials, out of which 10 045 are target trials.

The SRE 2018 and the SRE 2019 both consist of two separate evaluations. One is based on telephone speech data in Call My Net 2 (CMN2) corpus, while the other one is based on Video Annotation for Speech Technology (VAST) corpus. In this study, we evaluated only the VAST portions of SREs as VAST data is a better match to our VoxCeleb training data. The SRE 2018 evaluation contains 31 815 trials, out of which 315 are target trials, while the SRE 2019 has 67 348 trials, out of which 452 are target trials.

With SREs, we used the diarization labels provided by NIST for the enrollment side to remove the unwanted portions of speech from the enrollment. We did not perform diarization of the test side for any of the datasets.

For each set of evaluation trials, we report *equal error rate* (EER) and normalized minimum detection cost (minDCF). See [34] for details of minDCF. We adopted the same minDCF parameters as used in SRE 2018 and 2019 evaluations. That is, we set the costs of miss and false alarm equal to one ($C_{\text{miss}} = C_{\text{fa}} = 1$), and the target prior P_{target} to 0.05.

5.5. Scoring back-end

We centered, whitened, and length-normalized (both discriminative and generative) speaker embeddings before simplified PLDA scoring [35]. We did not apply domain adaptation techniques, but simply used the training data (VoxCeleb) to compute centering vector and whitening matrix. Finally, we performed *adaptive symmetric score normalization* (AS-norm) [36]. For AS-norm, we randomly selected 2000 utterances from training data and chose 200 highest scoring utterances for each enrollment or test utterance to compute the normalization statistics.

5.6. Speaker verification results

Table 4 shows the results of our experiments with different systems on multiple speaker verification evaluations. The results for the core condition of SITW are the most representative of the basic accuracy of the ASV systems as it does not have multi-speaker utterances requiring diarization. The other evaluations provide supporting evidence, although the results may be impaired by the lack of diarization.

In general, we find that the differences between the results of different deep embedding extractors are small. For example, when migrating from TDNN to TDNN-SE and to TDNN-RES-SE architectures, the results slightly improve on some evaluations, but get slightly worse on others. Similarly, the differences between the different aggregation methods are relatively minor, which is quite intriguing considering the differences between

Table 4: *Speaker verification results for the systems evaluated in this study. In addition to the deep speaker embedding systems, the results are reported for four neural i-vector systems each of which are based on different variations of the aggregation layer.*

	SITW EVAL CORE		SITW EVAL MULTI		SRE18 EVAL VAST		SRE19 EVAL VAST	
	EER	Min Cost	EER	Min Cost	EER	Min Cost	EER	Min Cost
TDNN (mean & std)	2.21	0.135	**3.46**	**0.183**	12.69	**0.472**	5.97	0.223
TDNN-SE (mean & std)	2.02	0.125	4.03	0.188	12.70	0.473	5.97	0.212
TDNN-RES-SE (mean & std)	2.10	0.123	4.07	0.188	12.02	0.477	5.75	0.216
TDNN-RES-SE (LDE, isotropic)	2.02	0.122	4.04	0.185	12.70	0.497	5.53	0.212
↳ Neural i-vector	2.93	0.173	5.55	0.249	15.92	0.588	6.64	0.254
TDNN-RES-SE (LDE, shared diag.)	**1.83**	0.123	4.42	0.189	**11.75**	0.483	5.34	0.213
↳ Neural i-vector	2.81	0.168	5.40	0.246	15.87	0.522	6.43	0.256
TDNN-RES-SE (NetVLAD)	1.94	**0.117**	4.06	0.184	12.38	0.474	**5.31**	**0.208**
↳ Neural i-vector	3.09	0.175	5.73	0.261	16.51	0.588	6.00	0.290
TDNN-RES-SE (NetVLAD/LDE)	2.02	0.129	4.41	0.199	13.40	0.528	5.53	0.229
↳ Neural i-vector	3.06	0.188	5.65	0.262	15.56	0.596	5.97	0.253

Table 5: *Review of recent single system results for SITW core-core condition. Due to different experimental settings and implementations, the results from different approaches are not directly comparable. Out of the i-vector systems, the proposed neural i-vector obtains the lowest EER. The second lowest EER was obtained by an i-vector system using a dereverberation system (WPE) together with perceptual linear prediction (PLP) and stacked bottleneck features (SBN). Other two included i-vector systems use MFCCs and bottleneck features (BNF). Under the divider line are the systems based on deep speaker embeddings. All systems use either MFCCs or filterbank coefficients (FBANK) as input features. All the embedding networks use either TDNN, extended TDNN (E-TDNN), factorized TDNN (F-TDNN), or ResNet34 based architectures. One system uses additive angular margin (AAM) loss instead of standard cross-entropy. The performance differences between the deep embedding extractors are rather small, except for the last system utilizing tied mixture of factor analyzers (TMFA) layer that is trained on 8 kHz Switchboard and SRE data.*

System & study	EER (%)
Neural i-vector [this study]	2.81
WPE PLP+SBN i-vector [37]	3.38
MFCC i-vector [37]	4.40
BNF i-vector [18]	5.77
TDNN-RES-SE (LDE) [this study]	1.83
FBANK E-TDNN [37]	1.70
MFCC E-TDNN [15]	1.7
MFCC F-TDNN [18]	1.86
FBANK ResNet34+LDE (AAM-softmax) [18]	2.11
FBANK ResNet34+TMFA (8 kHz) [14]	5.74

the standard mean and standard deviation aggregation and the dictionary based methods.

Different variants of neural i-vectors perform almost equally well to each other. The performance of neural i-vectors is way behind the performance of their deep embedding counterparts. On the other hand, the neural i-vectors perform substantially better than the other i-vector systems reported in literature as can be observed from Table 5. The table also shows that our deep embeddings obtain a competitive results in comparison to the results reported in the other studies.

5.7. Visualizations of neural i-vectors

In Figure 8, we illustrate *sampled* neural i-vectors for 5 male speakers in the SITW corpus. From each speaker we selected six utterances and computed the posterior distributions [29, eqs. (3) and (4)] of i-vectors. These distributions were used to sample 50 i-vectors per utterance. From the figure, we can observe that different speakers are well separated and that the utterances with short durations have higher uncertainty (*i.e.*, more spread clusters) than the utterances with long durations, as expected.

Finally, in Figure 9, we depict traces of posterior covariances [29, eq. (3)] of i-vectors for SITW data. The traces reflect the uncertainty in the i-vector estimation [39]. As expected, the longer the duration, the less uncertain the i-vectors are.

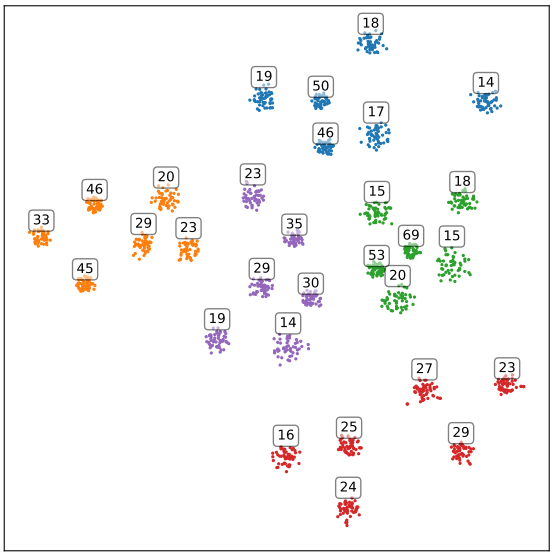

Figure 8: *T-SNE visualization [38] of random neural i-vectors drawn from i-vector posterior distributions of 30 utterances from 5 male speakers in SITW corpus. Different colors represent different speakers. Each of the 30 clusters consists of 50 random i-vectors drawn from the posterior distribution of one i-vector. The numbers show durations of the utterances in seconds after removing non-speech frames. The long utterances have less uncertainty than the short ones, which can be observed from the compactness of the clusters.*

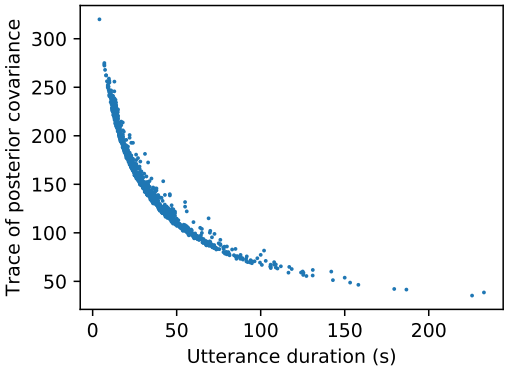

Figure 9: *Trace of i-vector posterior covariance matrix as a function of utterance duration for utterances in SITW core-core condition.*

6. Conclusion

At a broad outlook, the general developments in the field of speaker recognition have involved innovative (and often successful) re-use of previous generation tools to build up next generation recognizers: we have seen steady transition in state-of-the-art from individually trained GMMs to GMM-UBM, GMM supervectors, JFA and i-vectors (in this order). As a community, we have been working on multi-layered (deep) models, formed by stacking frame-level feature processors with utterance-level presentations and speaker latent variable models. Until the recent past, however, these pipelines have not been trained as a whole, but constructed from individually-optimized components. This is where the deep neural networks have come to a rescue, and we are witnessing transition towards the next generation deep models. Nonetheless, deep neural network models have seem to have interrupted the chain of GMM-based systems, particularly as they lack the concept of a universal background model. Some recent work has therefore looked into replacing the global temporal pooling operation of deep embedding extractors with learnable dictionaries, similar to the UBM, with demonstrated improvements.

In an attempt to bridge classic GMM-based technology and the modern deep learning era, we have provided a unified comparison of alternative i-vector extractors that use different variants of deep neural networks to optimize the frame-level features and the UBM. In particular, two recent deep neural network architectures, LDE and NetVLAD, *can be interpreted as GMMs with specific assumptions*. This interpretation enabled us to re-consider classic GMM-based systems using discriminatively obtained features and UBM. As a proof of concept, we decided to focus on the i-vector system, but similar construction is readily applicable to any ASV or diarization system that uses GMMs.

Our results indicate that 'neural i-vectors' outperform all the existing i-vector variants by a wide margin, indicating the importance of using speaker-informative short-term features and speaker-informative dictionary. Even if the corresponding 'purely neural' systems (used for obtaining the components of our i-vector system) outperform the neural i-vector approach, this was *not* the point of our study. The point, instead, is that it is possible to view certain neural architectures as if having a multi-modal aggregator (GMM) built in them. These identified connections may open up fresh ideas in revoking techniques such as uncertainty propagation, data augmentation (by sampling features or speaker embeddings). Potential applications that may benefit from fine-grained frame-by-frame speaker decisions, such as speaker diarization, provide another potential topic of future studies.

This work was partially supported by Academy of Finland (project 309629).

7. References

[1] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[2] Douglas A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Communication*, vol. 17, pp. 91–108, August 1995.

[3] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19 – 41, 2000.

[4] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio, Speech & Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[5] 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*, 2010.

[6] Pavel Matějka, Ondřej Glembek, Fabio Castaldo, Md Jahangir Alam, Oldřich Plchot, Patrick Kenny, Lukáš Burget, and Jan Černocky, "Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4828–4831.

[7] 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.

[8] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[9] Simon J. D. Prince and James H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, 2007, pp. 1–8.

[10] Yao Tian, Meng Cai, Liang He, and Jia Liu, "Investigation of bottleneck features and multilingual deep neural networks for speaker verification," in *Proc. Interspeech 2015*, 2015.

[11] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1695–1699.

[12] Weicheng Cai, Zexin Cai, Xiang Zhang, Xiaoqi Wang, and Ming Li, "A novel learnable dictionary encoding layer

for end-to-end language identification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5189–5193.

[13] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[14] Nanxin Chen, Jesús Villalba, and Najim Dehak, "Tied mixture of factor analyzers layer to combine frame level representations in neural speaker embeddings," *Proc. Interspeech 2019*, pp. 2948–2952, 2019.

[15] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.

[16] Hossein Zeinali, Lukas Burget, Johan Rohdin, Themos Stafylakis, and Jan Honza Cernocky, "How to improve your speaker embeddings extractor in generic toolkits," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6141–6145.

[17] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "VoxCeleb2: deep speaker recognition," in *Proc. Interspeech 2018*, 2018, pp. 1086–1090.

[18] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak, "State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and Speakers in the Wild evaluations," *Computer Speech & Language*, vol. 60, 2020.

[19] Hang Zhang, Jia Xue, and Kristin Dana, "Deep ten: Texture encoding network," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 708–717.

[20] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[21] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[22] Patrick Kenny, "A small footprint i-vector extractor," in *Odyssey*, 2012, vol. 2012, pp. 1–6.

[23] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[24] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger, "Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 421–429.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[27] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[28] Dogan Can, Victor R Martinez, Pavlos Papadopoulos, and Shrikanth S Narayanan, "Pykaldi: A python wrapper for Kaldi," in *International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5889–5893.

[29] Ville Vestman, Kong Aik Lee, Tomi H. Kinnunen, and Takafumi Koshinaka, "Unleashing the Unused Potential of i-Vectors Enabled by GPU Acceleration," in *Proc. Interspeech 2019*, 2019, pp. 351–355.

[30] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "VoxCeleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[31] Mitchell McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson, "The speakers in the wild (SITW) speaker recognition database.," in *Proc. Interspeech 2016*, 2016, pp. 818–822.

[32] *NIST 2018 Speaker Recognition Evaluation Plan*, 2018 (accessed January 24, 2020), `https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf`.

[33] *NIST 2019 Speaker Recognition Evaluation Plan*, 2019 (accessed January 24, 2020), `https://www.nist.gov/system/files/documents/2019/08/16/2019_nist_multimedia_speaker_recognition_evaluation_plan_v3.pdf`.

[34] Seyed Omid Sadjadi, Craig Greenberg, Elliot Singer, Douglas Reynolds, Lisa Mason, and Jaime Hernandez-Cordero, "The 2018 NIST Speaker Recognition Evaluation," in *Proc. Interspeech 2019*, 2019, pp. 1483–1487.

[35] Daniel Garcia-Romero and Carol Y Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech 2011*, 2011.

[36] Sandro Cumani, Pier Domenico Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis, "Comparison of speaker recognition approaches for real applications," in *Proc. Interspeech 2011*, 2011.

[37] Pavel Matějka, Oldřich Plchot, Hossein Zeinali, Ladislav Mošner, Anna Silnova, Lukáš Burget, Ondřej Novotný, and Ondřej Glembek, "Analysis of BUT Submission in Far-Field Scenarios of VOiCES 2019 Challenge," in *Proc. Interspeech 2019*, 2019, pp. 2448–2452.

[38] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[39] Amir Hossein Poorjam, Rahim Saeidi, Tomi Kinnunen, and Ville Hautamäki, "Incorporating uncertainty as a quality measure in i-vector based language recognition.," in *Odyssey*, 2016, pp. 74–80.