

Text-independent voice recognition based on Siamese networks and fusion embeddings

R. De Prisco^{1,*}, C. Fusco¹, M. Iannucci¹, D. Malandrino¹ and R. Zaccagnino¹

¹University of Salerno, Fisciano (SA), Italy

Abstract

The problem of identifying people from their voices has been the subject of increasing research activities. Interest in this problem is fostered by the important practical applications that voice authentication has. Many solutions exploit neural networks based on *i-vectors* and, more recently, on *x-vectors*, which are computed from the input audio signal. In this paper we design and implement a novel voice recognition system based on the *fusion* of both *i-vectors* and *x-vectors*. The recognition is text-independent, that is, the user is recognized regardless of the actual words that are pronounced. We performed preliminary experiments to assess the effectiveness of the proposed solution. Results show that the proposed method achieves performance improvement compared with approaches based on only *i-vectors* or only *x-vectors*.

Keywords

Voice authentication, *i-vectors*, *x-vectors*, Siamese networks

1. Introduction

Identification of individuals through the voice relies on the existence of strictly personal traits [1] in the voice. Voice authentication can be useful in a wide range of applications in real-world scenarios. For example, it can be used for voice-based authentication of personal smart devices, and for guaranteeing the transaction security of bank trading and remote payment. In digital forensics, it has been widely applied for investigations [2, 3, 1], or surveillance and automatic identity tagging [4]. The research on this field dates back to at least 1960s [5]. During the years, a number of acoustic features, such as the mel-frequency cepstral coefficients, and template models have been applied [1]. Early approaches to the problem can be found in [6, 7, 8].


The development of *i-vectors* as fixed dimensional front-end features for speaker recognition tasks was introduced in [9, 8], and has provided the state-of-the-art performance for several years, until the era of Deep Learning. Approaches based on deep learning [10, 11, 12, 13] have significantly increased the performance, especially in noisy environments [14, 15], and are based on *x-vectors* [11, 12]. Systems based on *x-vectors* improve over those based on *i-vectors* [16]. Both *i-vectors* and *x-vectors* are a numerical representation, in the form of a fixed size array of numbers, of a voice audio signal.


ITASEC 2023: The Italian Conference on CyberSecurity, May 03–05, 2023, Bari, Italy

*Corresponding author.

✉ robdep@unisa.it (R. De Prisco); c.fusco19@studenti.unisa.it (C. Fusco); dmalandrino@unisa.it (D. Malandrino); rzaccagnino@unisa.it (R. Zaccagnino)

ORCID 0000-0003-0559-6897 (R. De Prisco); 0000-0003-2693-0196 (D. Malandrino); 0000-0002-9089-5957 (R. Zaccagnino)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Voice authentication can be classified into text-dependent and text-independent. Text-dependent voice authentication involves the use of a fixed sentence. The sentence doesn't need to be secret since the recognition is based on the voice and not on the sentence. The system is called text-dependent because the user is required to pronounce the fixed sentence in order to be recognized. With text-independent voice authentication, the user is recognized regardless of the actual words pronounced and thus there is no need to fix a text.

Contribution of this work. In this paper, we propose a novel text-independent voice recognition system. The novelty is in the combined use of i-vectors and x-vectors. The proposed system is based on a *fusion embedding vector*, obtained as a combination of i-vectors and x-vectors. For brevity, we will refer to fusion embedding vectors just as fusion embeddings. The motivation behind the use of fusion embeddings is that of bringing together the potential of both representations. Indeed, although x-vectors have better recognition performance especially on short speeches and an intrinsic ability to discriminate by definition, i-vectors seem to provide better results in recognizing the same speaker that records from different devices. Moreover, the idea of combining such embeddings is also motivated by the results obtained in [11], where the authors shown that the combined use of both approaches can lead to better performances. However, in [11] no vectorial combination has been investigated, but, instead two separate recognitions, one based only on i-vectors and the other based only on x-vectors, are first computed and a final one is obtained by averaging the scores. In our approach we first average the vectors and then apply the recognition. This is a fundamental difference.

The system uses a Siamese network which is trained on fusion embedding vectors from a database of recorded voice audio signals (recorded speeches). The Siamese network is then able to take as input two fusion embedding vectors and tell whether they are derived from speeches of the same person. The use of a Siamese network increases the discrimination strength because such category of Deep Learning Neural Networks are particularly suitable for the computation of similarity measures or determining relationships/discriminations between two comparable subjects [17].

Paper organization. The rest of the paper is organized as follows. In Section 2, we discuss some relevant works. In Section 3, we provide the needed background about i-vectors, x-vectors, and Siamese networks. In Section 4, we describe the proposed system. In Section 5, we report the results obtained from experiments carried out to assess the effectiveness of the proposed system. Finally, in Section 6 we provide conclusions and directions for future research.

2. Related Work

For several years, most voice recognition systems have been based on the i-vector and the Probabilistic Linear Discriminant Analysis (PLDA) [18]. With the advent of Deep Learning, deep speaker embedding has led to significant performance improvements [12, 19, 20, 21]. Deep speaker embedding uses a speaker identification network to create a speaker-embedding space. In [22], the i-vector extraction and the PLDA scoring have been jointly derived using a single deep neural network and the model is trained using a binary cross entropy criterion.

The use of triplet loss in end-to-end speaker recognition has shown improved performances for short utterances [23]. Wan et. al. [24] proposed a generalized end-to-end loss function inspired by minimizing the centroid mean of (same) speaker distances while maximizing the distances between clusters of different speakers. In this direction, many architectures based on *convolutional neural networks* have been used for frame-level processing, such as, x-vectors using *time delay neural networks* to extract the frame-level features [12]. Later, more advanced networks, such as ResNets [19], DenseNets [20], and Res2Nets [21], and different training loss functions besides the softmax loss function have been used. For example, *additive margin softmax* [25] loss and *additive angular margin softmax* loss [26] have been introduced. Another category of deep speaker embedding uses *metric learning* [27, 28, 29], which is characterized by distance measures used to guide the embedding network so that the speaker embeddings have simultaneously large inter-class distance and small intra-class distance. For example, triplet loss [29], prototypical network loss [28], and angular prototypical loss [27] have been investigated.

In [30], a novel text-independent method able to combine speaker feature extraction and speaker classification in only one step, was proposed. The two main aspects of such a method were the use of a speaker representation consisting in the Mel-frequency spectrogram extracted from the input audio, in order to benefit from the dependency of the adjacent spectro-temporal features, and the use of a Siamese convolutional network to perform feature extraction and speaker classification. Results obtained during experiments showed significant improvement over conventional classical and DL-based algorithms for forensic cross-device voice recognition.

The most interesting aspect that emerges from the literature is that the exclusive use of x-vectors or i-vectors does have advantages both it has also limitations. As we have already said in Section 1, in this preliminary work we investigate the combined usage of i-vectors and x-vectors with the goal of bringing their potential together and thus to build a novel feature speaker embedding. We show that such a combination can be effectively exploited for training a Siamese network to calculate a similarity score that can be used to recognize the voice of a speaker that can pronounce any pass-phrases to authenticate.

3. Background

In this section, we briefly recall the needed background to understand the proposed approach. Specifically, first we will provide details about i-vectors and x-vectors, by highlighting strengths and weaknesses, and then we will discuss Siamese networks.

3.1. Speaker feature extraction with i-vector

Joint Factor Analysis [8, 31, 32] has represented the state-of-the-art for text independent speaker detection tasks for several years, due to its powerful in modeling the inter-speaker variability and in compensating for channel/session variability in the context of the Gaussian Mixture Model. The first voice recognition system based on Joint Factor Analysis as a *feature extractor* was proposed in [33]. The idea was to represent a speaker utterance as a *speaker-dependent supervector* combining factors from two distinct spaces, i.e., the *speaker space* containing speaker

variabilities, and the *channel space* containing channel variabilities. Thus, the performance was essentially affected by the speaker and channel variations of utterances.

In [9], the authors proposed a kind of “speaker embedding”, by defining only a single *total variability space*, instead of two separate spaces, which contains the speaker and channel variabilities simultaneously. Given an utterance, the *speaker- and channel-dependent* Gaussian Mixture Model supervector, named *i-vector*, is defined as $M = m + Tw$ where m is the *speaker- and channel-independent* supervector, T is a rectangular matrix of low rank, and w is a random vector having a standard normal distribution. The components of the vector w are the total factors.

We remark that, the main advantage of the *i-vectors* is that they are not strictly dependent on the change of the transmission channel or on the variability of the speaker’s vocal characteristics (such as cadence or accent), since in the proposed model by Dehak et al. [9] based on the Joint Factor Analysis, both factors are taken into account as a whole, during the modeling phase.

3.2. Speaker feature extraction with deep embedding: x-vector

Similar to the *i-vector*, the *x-vector* is also a kind of speaker embedding, named *deep speaker embedding*, but that discriminatively embeds speakers into a vector space by using a Delay Neural Network trained in a supervised fashion [11]. Such a Delay Neural Network computes speaker embeddings from variable-length acoustic segments and was implemented using the `nnet3` neural network library in the *Kaldi Speech Recognition Toolkit*¹. The features are 30 dimensional Mel Frequency Cepstral Coefficients with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds. The Delay Neural Network (see Figure 1), is organized on 2 levels: *frame* and *segment* level. The nonlinearities are rectified linear units (ReLU). The *frame-level* consists of 5 layers, through which the audio frames are taken in input and split with a sliding window of 3 seconds, and given as input to a Time Delay Neural Network [34] which enable the network to learn the structural information of the signal and the relationship between the various frames. The *statistics pooling layer* receives the output of the final frame-level layer as input, aggregates over the input segment, and computes its mean and standard deviation. The segment level statistics are concatenated together, passed to two additional hidden layers with dimension 512 and 300, and finally the soft max output layer maps the *x-vector* obtained to the probability of the speaker.

The goal of the Delay Neural Network is to produce embeddings that capture speaker characteristics over the entire utterance, rather than at the frame-level. Although both layer a and b after the statistics pooling layer can be used to extract the embedding, usually a is used as the *x-vector*.

3.3. Siamese networks

Siamese networks were introduced in [35] to solve the problem of matching hand written signatures, and were subsequently adapted to other domains such as image and video processing [36, 37, 38], and Natural Language Processing tasks [39, 40]. A Siamese network is composed of two identical twins networks that share weights. Such networks pass their output

¹<https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2>

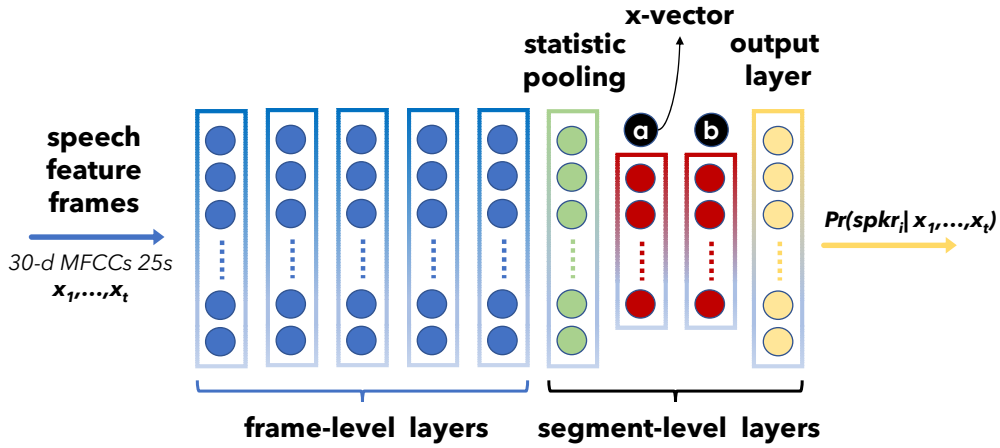


Figure 1: Delay Neural Network used to extract x-vectors. Segment-level embeddings a or b can be extracted from any layer of the network after the statistics pooling layer.

to a *similarity module*, which computes a “distance” between the two inputs. The distance is compared to the given target (i.e. whether or not the pair are similar), the loss is calculated, and the weights are then adjusted.

Several loss functions can be used to train Siamese networks. In this work we used the *triplet loss* function. During the training, instead of taking two inputs, this function takes three inputs: the *anchor*, the *positive*, and the *negative*. The anchor is the reference input, the positive is an input that is in the same class as the anchor, while the negative is an input with a different class from the anchor. The idea is to maximize (resp. minimize) the distance between the anchor and the negative (resp. positive). Formally, the triplet loss can be defined as:

$$L = \max(d(a, p) - d(a, n) + m, 0)$$

where d is some distance function, and m is a constant named *margin*; the constant m is used to decrease the probability that L be 0. The details of the architecture and building blocks of the twins networks used in this work are provided in Section 4.

4. A fusion embedding voice authentication system

4.1. Overview

In order to perform voice authentication, users of the system have to be registered. During the registration phase, each user is given a unique username and an *enrollment* audio file a_{username} is saved into a database. The enrollment audio file is obtained by recording the voice audio signal of the user while the user is reading a sufficiently long (random) text. In order to be recognized a user provides a username, and is asked to speak; the speech is recorded into a *test* audio file a_t . The goal of the voice authentication system is that of telling whether a_t and a_{username} are from the same speaker. Establishing whether the audio test file comes from the specified user will be denoted as

$$a_t \sim a_{\text{username}}$$

In order to solve the problem we compute both the i -vector and the x -vector of the audio file and from them we build a new vector, that we call *fusion embedding vector*. Such vector is then used with a Siamese network to tell whether $a_t \sim a_{\text{username}}$. The Siamese network is trained on the database of registered user.

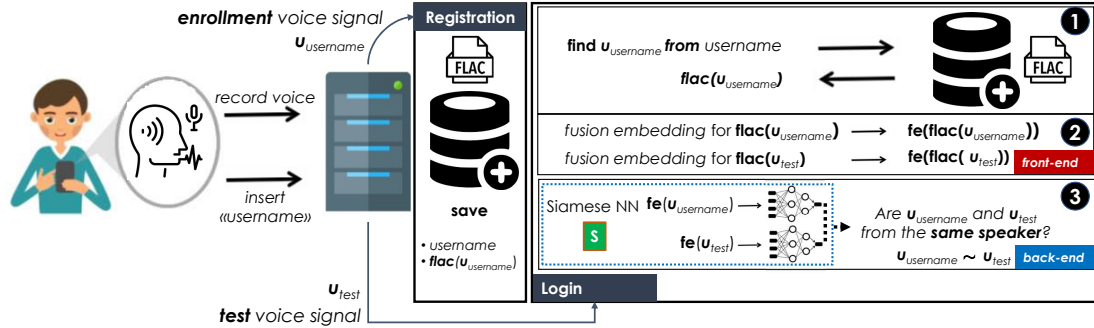


Figure 2: The overall scenario.

Figure 2 summarizes the overall system, which consists of two components: (i) a *front-end*, that, given a pair of audio files, the enrollment audio file a_{username} and the test audio file a_t , for each of them computes the fusion embeddings $fe(a_{\text{username}})$ and $fe(a_t)$; (ii) a *back-end*, which uses a Siamese Neural Network **S** to tell whether $a_t \sim a_{\text{username}}$, taking as input $fe(a_{\text{username}})$ and $fe(a_t)$. In the following we provide details about how the above is done.

4.2. Front-end: fusion speaker embedding generations

In this section, we describe the *front-end* of the proposed system focusing on computation of the fusion speaker vectors.

4.2.1. Speaker embeddings: i -vector and x -vector extraction

We used the `mnet3` Neural Network library in the *Kaldi Speech Recognition Toolkit* to extract from an audio file a , the corresponding i -vector $iv(a)$ and x -vector $xv(a)$. This library is one of the most used for the voice recognition problem. The cepstral features for the i -vectors are extracted using a 25 ms Hamming window; every 10 ms, 24 Mel Frequency Cepstral Coefficients were calculated; this 24-dimensional features vector was subjected to feature warping [41] using a 3-second sliding window; delta and delta-delta coefficients were then calculated using a 5-frame window to produce 60-dimensional features vectors; then, an energy-based speech activity detection system selects features corresponding to speech frames; finally, using gender-dependent Universal Background Models containing 2048 Gaussians and two gender-dependent joint factor analysis configurations the diagonal matrix is added in order to have speaker and common factors, obtaining one i -vector of 400 total factors. The `mnet3` library allows us to use vectors of size 100, 200 and 400. We experimented with all these 3 sizes and the best results were obtained with vectors of size 400.

For the x -vectors the features are 23 dimensional filterbanks with frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds; the speech activity detection used for the i -vectors filters out nonspeech frames; the Deep Neural Network configuration is outlined in Figure 1; it is trained to classify the N speakers in the training data; after training, x -vectors of size 512 are extracted.

4.2.2. Merging i -vector and x -vector: fusion speaker embedding

The proposed voice recognition system is based on a *fusion embedding vector* $fe(a)$ obtained by merging the i -vectors with the x -vectors. More in detail, let $iv(a)$ and $xv(a)$ be the i -vector and x -vector, respectively, for a given audio file a . By construction, we have $|iv(a)| = 400$ and $|xv(a)| = 512$. The idea is to define a new vector $fe(a)$ obtained as a sort of “mean vector” between $iv(a)$ and $xv(a)$. However, the difference in the sizes, 400 for the i -vectors and 512 for the x -vectors, is a problem. There are two natural solutions: adapt the x -vectors to the shape of the i -vectors cutting 112 entries, or adapt i -vectors to x -vectors, adding somehow 112 entries. Although the first solution seems more immediate and easy to apply, it actually has two serious drawbacks: first, it would distort the final mean, given that removing 112 entries would have roughly meant the loss of 20% of information; second, the Deep Neural Network provided by Kaldi and used for our experiments is configured for the native x -vector size, i.e., 512 entries, and so, cutting the x -vectors to obtain vectors of 400 entries would therefore have involved lengthy and expensive modifications to the entire network structure.

Hence we opt for the second alternative, that is, add 112 entries to the i -vector. The question that remains is how to add these 112 missing entries. We have tried the following alternatives, which assign to each new entry: (i) the average value of the i -vector elements, (ii) the most recurring value within the i -vector, (iii) the value 0 (*zero-padding*). From empirical observations during the experiments conducted we have found that zero-padding is the strategy that generally achieves the best results. We denote with $iv(a)_{zero}$ the vector obtained by zero-padding $iv(a)$ with additional 112 entries.

Notice that, having added zero entries, when calculating the mean vector $fe(a)$ between $iv(a)_{zero}$ and $xv(a)$, the first 400 entries correspond to an effective mean between the two vectors, while the remaining 112 entries are dominated by the value of x -vector. Since by doing this we somehow give more weight to the x -vector, we use a weighted mean in order to rebalance the contribution of the i -vector. Formally:

$$fe(a) = W_{iv} * iv(a)_{zero} + W_{xv} * xv(a)$$

where $W_{iv}, W_{xv} \in [0, 1]$ are the *weights* for $iv(a)$ and $xv(a)_{zero}$, respectively.

From empirical observations during the experiments carried out we have found that best results are obtained by setting $W_{iv} = 0.6$ and $W_{xv} = 0.4$. Thus, we build the fusion embedding vector as

$$fe(a)[i] = 0.6 * iv(a)_{zero}[i] + 0.4 * xv(a)[i]$$

for each $i = 0, \dots, 511$.

4.3. Back-end

4.3.1. Siamese architecture

The proposed *back-end* uses a Siamese network \mathcal{S} which, given a pair of fusion embeddings $fe(a_i)$ and $fe(a_j)$, computes a similarity score $\mathcal{S}(fe(a_i), fe(a_j))$. Then, to verify whether a_i and a_j are from the same speaker, the following rule is used by the system:

$$\mathcal{S}(fe(u_i), fe(u_j)) \geq \delta \implies u_i \sim u_j$$

where $\delta \in [0, 1]$ is the *system threshold* empirically estimated during the training of the network.

In the following, we will provide details about the architecture and the training of \mathcal{S} . We remark that the architecture has been modified during several experiments carried out to find the best setting. Figure 3 shows the one achieving the best performances. As explained in Section 3, a Siamese network consists of two identical subnetworks; in the following we will discuss the structure of such subnetworks.

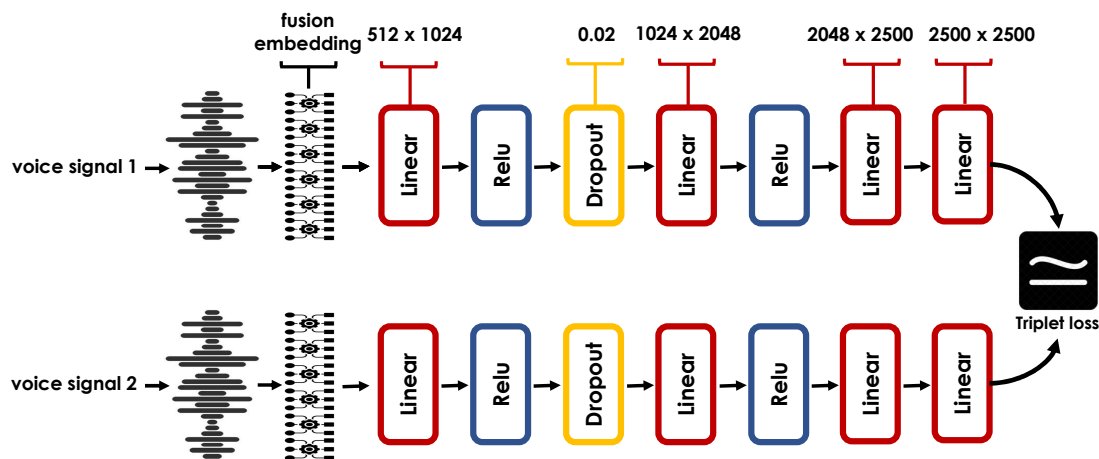


Figure 3: The Siamese network \mathcal{S} .

As can be seen in Figure 3, each subnetwork starts with a *Linear* layer which takes as input a fusion embedding, applies a linear transformation to the input data and then, through the application of a *ReLU* layer, outputs the input directly if it is positive, otherwise, it outputs zero. Such a technique is used to overcome the vanishing gradient problem, allowing models to learn faster and perform better. Then, a *Dropout* layer, during the training phase, randomly deactivates some of the elements (2% of the data taken as input), providing a series of advantages especially in the case where the dataset is small [42]. Following, a sequence of three *Linear* layers (the first one with *ReLU* activation function) expands the dimensionality of the initial fusion embedding until reaching one vector of 2500 features, which represents the peculiarities of the voice on which the network must learn to calculate the similarity.

One of the most interesting advantages of using Siamese networks is the ability to adopt the *One-Shot Learning* strategy, shown to be effective in identifying new classes based on one (or only a few) examples [43]. The idea is to extract rules on previously seen classes, i.e., to

learn patterns and similarities instead of fitting the ML model to fixed classes, in order to be able of classifying previously unseen classes using one instance. This strategy is very helpful in the scenario described in Section 4.1. Indeed, it allows us to define a system “calibrated” on a significant initial set of speakers, i.e, with a back-end exploiting a Siamese network trained on an initial set of voices provided by a “representative” sample of speakers; a new speaker can be added to the system without having to retrain the network, but simply by saving a reference enrollment audio signal, which will be used every time the user needs to be recognized.

4.3.2. One-Shot Learning.

The network **S** is trained using One-Shot learning. The performance on verifying a new speaker without to retrain **S** is evaluated by considering only the *enrollment fusion embedding* saved during the registration phase (see Figure 2). Here, we provide details about the process of establishing the proposed system based on One-Shot learning and about the methodology of assessing performance for new speaker classes without retraining **S**. Let $D = \{s_1, \dots, s_N\}$ be the dataset of voice audio signals provided by N speakers (N classes), where each s_k is the class containing voice audio signals by the k^{th} user, for $k = 1, \dots, N$. In order to train the network to recognize also speakers that were not included in the training phase, for each i , we train the network by excluding the samples from s_i . More in details, each of the remaining $N - 1$ classes is split into two balanced subsets. Then, the first one is used to generate the training set pairs, while the second one is used as the evaluation pool of instances. Then, we perform the training process using the Triplet Loss function (see Section 3).

For the evaluation, the instances of the excluded set s_i are split in two balanced subsets s_i^l and s_i^u where s_i^l represents the “labelled” samples for user i , while s_i^u represents the “unlabelled” samples for user i , i.e., recognition attempts by user i . A set of *evaluation pairs* P_i is generated as follows. Let $D' = \{s_1, \dots, s_{i-1}, s_i^l, s_{i+1}, \dots, s_N\}$ be the evaluation set. For each $\bar{e}_i \in s_i^u$ one element e_j is randomly chosen from each $s_j \in D'$, obtaining the N pairs $E_i = \{(\bar{e}_i, e_1), \dots, (\bar{e}_i, e_{i-1}), (\bar{e}_i, e_i), (\bar{e}_i, e_{i+1}), \dots, (\bar{e}_i, e_N)\}$. Then, we set $P_i = \bigcup_{\bar{e}_i \in s_i^u} E_i$. Observe that $|P_i| = N \times |s_i^u|$. The similarity is calculated for the pairs in P_i and the classification is based on the pair with the highest similarity (i.e. least distance). To determine the trade-off between the number of labelled instances of the new speaker class and accuracy, the process is repeated $|P_i|$ times, i.e., for each instance in P_i . Majority voting is then applied to deduce the instance label; the class that has been recognized more often, is used as instance label. Algorithm 1 reports the pseudo-code for the training and evaluation of **S** described above.

5. Preliminary experiments

Data collection. To assess the performance of the proposed voice recognition system, we collected voice samples from 100 speakers. Specifically, we have recruited 100 students at the University of Salerno. The sample was 65% male and 35% female, with a mean age of 22. Participants were informed that the information provided remain confidential. To collect data, each student was led to a room dedicated to the registration phase. Each speaker was asked to read one short text for a recording duration of 30 seconds. Then, we split the recorded audio in several overlapping 10-second fragments using a sliding window of 2 seconds, for a total of

Algorithm 1: Siamese NN \mathbb{S} One-Shot Learning

Input : $D = \{s_1, \dots, s_N\}$, threshold
Output: $\langle \mathbb{S}, \text{one-shot-accuracy} \rangle$

```
1  $\mathbb{S} \leftarrow \text{InitializeSiamese}(\mathbb{S});$ 
2 one-shot-accuracy  $\leftarrow [];$ 
3 for  $i = 1$  to  $N$  do
4   /* Select "new" speaker  $s_i$ 
5    $\langle s_i^l, s_i^u \rangle \leftarrow \text{SplitUtterancesBySpeaker}(s_i, 0.5);$ 
6   training_set $_i \leftarrow \emptyset;$ 
7   testing_set $_i \leftarrow \emptyset;$ 
8   /* Build training/testing sets without  $s_i$ 
9   for  $j = 1$  to  $N$  do
10    if  $j \neq i$  then
11       $\langle s_j^l, s_j^u \rangle \leftarrow \text{SplitUtterancesBySpeaker}(s_j, 0.5);$ 
12      training_set $_i \leftarrow \text{training\_set}_i \cup s_j^l;$ 
13      testing_set $_i \leftarrow \text{testing\_set}_i \cup s_j^u;$ 
14    $P_s \leftarrow \text{GetSimilarPairs}(\text{training\_set}_i);$ 
15    $P_d \leftarrow \text{GetDissimilarPairs}(\text{training\_set}_i);$ 
16   /* Train and Test Siamese NN
17    $\mathbb{S} \leftarrow \text{Train}(\mathbb{S}, P_s, P_d, \text{threshold}, \text{"Triplet Loss"});$ 
18   accuracy  $\leftarrow \text{Test}(\mathbb{S}, \text{testing\_set}_i, \text{SV\_threshold});$ 
19   /* One-Shot Evaluation
20    $P_i \leftarrow \text{GetPairsFromOtherSpeakers}(s_i^u, \{s_1, \dots, s_{i-1}, s_i^l, s_{i+1}, \dots, s_N\});$ 
21   correct  $\leftarrow 0;$ 
22   for  $k = 1$  to  $|P_i|$  do
23      $x \leftarrow \text{Voting}(P_i[k], \mathbb{S});$ 
24     if  $x == i$  then
25       /* Correct classification
26       correct  $\leftarrow \text{correct} + 1;$ 
27   accuracy $_i \leftarrow \frac{\text{correct}}{100};$ 
28   one-shot-accuracy.append(accuracy $_i$ );
29 return  $\langle \mathbb{S}, \text{Average}(\text{one-shot-accuracy}) \rangle;$ 
```

11 10-second fragments. The voices were registered in wav format then compressed using the flac compression algorithm to save space on the server.

At the end of the registration phase, we have collected $11 \times 100 = 1100$ voice audio files lasting each 10 seconds. Then, for each of them we used the Kaldi library to extract the corresponding i-vector and x-vector. Finally, for each audio file we generated the fusion embedding as described in Section 4.2.

Training and test results. We have trained the network \mathbb{S} , with several configurations for the parameters. Specifically, we tried $\text{batchsize} \in \{16, 32, 64, 128\}$, $\text{learning_rate} \in \{0.1, 0.001, 0.0001\}$, $\#\text{epochs} \in \{2, 4, 6, \dots, 18, 20\}$, and $\text{threshold} \in \{0.10, 0.15, 0.20, \dots, 0.90, 0.95, 1.00\}$. The best result, that is 99% of accuracy, was obtained by setting $\text{batchsize} = 32$, $\text{learning_rate} = 0.001$, $\#\text{epochs} = 4$, and $\text{threshold} = 0.15$. Thus, the network \mathbb{S} trained with such a configuration has been used the following *Testing Phase*. For the Testing phase, we have recruited 20 people not involved in the data collection phase.

The sample was 55% male and 45% female, with a mean age of 21. As done during the data collection phase, participants were informed that the information provided remain confidential. Each speaker was asked to provide a username and to read a text for a duration of 10 seconds. The server registered the voice in wav format, and compressed it using the flac compression algorithm.

Several experiments have been conducted in order to evaluate the *recognition performance* of the system in three different scenarios:

- *same-device-scenario*: the speaker uses the same device both for the registration and for the recognition.
- *different-device-scenario*: registration is made with a device, while recognition is attempted with a different device.
- *attack-scenario*: Let i be different from j ; a speaker s_i claims to be s_j (that is, provides $username_j$ as username).

The evaluation has been carried out as follows. Each speaker i was requested to try 10 times each of the three scenarios defined above. That is, the user attempts 10 times the recognition using his own username and recording the voice with the same device that has been used in the registration phase; 10 times with his own username but recording the voice through a different device and finally 10 times with the username of another user, without restriction on the device.

In addition to the similarity computed using [S](#), we have computed the similarity using the PLDA approach and the cosine distance. Table 1 reports the results obtained. For compactness, we indicate with S_1 the *same-device-scenario*, S_2 the *different-device-scenario*, and S_3 the *attack-scenario*. In general, as we can see, when the speaker attempts a recognition through the same device used during the registration phase (*same-device-scenario*), the use of the x-vectors tends to give better results than using i-vectors. Instead, when the speaker attempts a recognition through using a device other than the one used during the registration (*different-device-scenario*), or tries to be recognized as another user (*attack-scenario*), the use of the i-vectors tends to give better results than using x-vectors. The results obtained using fusion embeddings with Siamese networks are better than those obtained with the other configurations.

Back-end	Speaker embedding								
	i-vector			x-vector			fusion		
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
cosine	0.80	0.44	0.30	0.78	0.30	0.28	0.86	0.42	0.49
PLDA	0.82	0.64	0.53	0.86	0.41	0.61	0.86	0.64	0.52
Siamese	0.84	0.78	0.70	0.98	0.41	0.60	0.98	0.81	0.89

Table 1

Comparison of accuracy reached in the testing phase for each back-end (cosine, PLDA, and Siamese) model and speaker embedding (i-vector, x-vector, and fusion embedding).

6. Conclusion

In this paper we have described a study whose goal is that of investigating the advantages deriving from the combination of i-vectors and x-vectors. The results obtained show that the

approach combining the two types of embedding provide better recognition accuracy. The study is only a preliminary investigation and research in this direction can be extended in several ways. First, other more complex embedding merging techniques, for example based on DL models, can be investigated. Second, the size of the dataset used to train the Siamese network and to carry out our experiments is small; increasing the size of the dataset, and diversifying the type of speakers, would provide a better evaluation of the performance, also in terms of scalability in real-world scenarios. Finally, it is worth to investigate improvements of the model used in the back-end. For example, define an image-based combination of embeddings and explore the potential of a Siamese network based on convolution neural networks.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

- [1] T. Kinnunen, H. Li, An overview of text-independent speaker recognition: From features to supervectors, *Speech communication* 52 (2010) 12–40.
- [2] J. P. Campbell, W. Shen, W. M. Campbell, R. Schwartz, J.-F. Bonastre, D. Matrouf, Forensic speaker recognition, *IEEE Signal Processing Magazine* 26 (2009) 95–103.
- [3] C. Champod, D. Meuwly, The inference of identity in forensic speaker recognition, *Speech communication* 31 (2000) 193–203.
- [4] R. Togneri, D. Pullella, An overview of speaker identification: Accuracy and robustness issues, *IEEE circuits and systems magazine* 11 (2011) 23–61.
- [5] S. Pruzansky, M. V. Mathews, Talker-recognition procedure based on analysis of variance, *The Journal of the Acoustical Society of America* 36 (1964) 2041–2047.
- [6] D. A. Reynolds, T. F. Quatieri, R. B. Dunn, Speaker verification using adapted gaussian mixture models, *Digital signal processing* 10 (2000) 19–41.
- [7] W. M. Campbell, D. E. Sturim, D. A. Reynolds, Support vector machines using gmm supervectors for speaker verification, *IEEE signal processing letters* 13 (2006) 308–311.
- [8] P. Kenny, G. Boulianne, P. Ouellet, P. Dumouchel, Joint factor analysis versus eigenchannels in speaker recognition, *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007) 1435–1447.
- [9] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification, *IEEE Transactions on Audio, Speech, and Language Processing* 19 (2010) 788–798.
- [10] Y. Lei, N. Scheffer, L. Ferrer, M. 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.
- [11] D. Snyder, D. Garcia-Romero, D. Povey, S. Khudanpur, Deep neural network embeddings for text-independent speaker verification., in: *Interspeech*, volume 2017, 2017, pp. 999–1003.
- [12] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, S. 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.
- [13] E. Variani, X. Lei, E. McDermott, I. L. Moreno, J. Gonzalez-Dominguez, Deep neural networks for small footprint text-dependent speaker verification, in: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, 2014, pp. 4052–4056.
 - [14] M. McLaren, L. Ferrer, D. Castan, A. Lawson, The speakers in the wild (sitw) speaker recognition database., in: Interspeech, 2016, pp. 818–822.
 - [15] A. Nagrani, J. S. Chung, A. Zisserman, Voxceleb: a large-scale speaker identification dataset, arXiv preprint arXiv:1706.08612 (2017).
 - [16] M. McLaren, D. Castan, M. K. Nandwana, L. Ferrer, E. Yilmaz, How to train your speaker embeddings extractor (2018).
 - [17] S. Bell, K. Bala, Learning visual similarity for product design with convolutional neural networks, ACM transactions on graphics (TOG) 34 (2015) 1–10.
 - [18] P. Kenny, Bayesian speaker verification with, heavy tailed priors, Proc. Odyssey 2010 (2010).
 - [19] W. Xie, A. Nagrani, J. S. Chung, A. 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.
 - [20] W. Lin, M.-W. Mak, L. Yi, Learning mixture representation for deep speaker embedding using attention., in: Odyssey, 2020, pp. 210–214.
 - [21] B. Desplanques, J. Thienpondt, K. Demuynck, Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification, arXiv preprint arXiv:2005.07143 (2020).
 - [22] J. Rohdin, A. Silnova, M. Diez, O. Plchot, P. Matějka, L. Burget, End-to-end dnn based speaker recognition inspired by i-vector and plda, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 4874–4878.
 - [23] C. Zhang, K. Koishida, End-to-end text-independent speaker verification with triplet loss on short utterances., in: Interspeech, 2017, pp. 1487–1491.
 - [24] L. Wan, Q. Wang, A. Papir, I. L. Moreno, Generalized end-to-end loss for speaker verification, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 4879–4883.
 - [25] F. Wang, J. Cheng, W. Liu, H. Liu, Additive margin softmax for face verification, IEEE Signal Processing Letters 25 (2018) 926–930.
 - [26] J. Deng, J. Guo, N. Xue, S. Zafeiriou, Arcface: Additive angular margin loss for deep face recognition, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4690–4699.
 - [27] J. S. Chung, J. Huh, S. Mun, M. Lee, H. S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, I. Han, In defence of metric learning for speaker recognition, arXiv preprint arXiv:2003.11982 (2020).
 - [28] J. Wang, K.-C. Wang, M. T. Law, F. Rudzicz, M. Brudno, Centroid-based deep metric learning for speaker recognition, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 3652–3656.
 - [29] C. Zhang, K. Koishida, J. H. Hansen, Text-independent speaker verification based on triplet convolutional neural network embeddings, IEEE/ACM Transactions on Audio, Speech, and Language Processing 26 (2018) 1633–1644.

- [30] S. Soleymani, A. Dabouei, S. M. Iranmanesh, H. Kazemi, J. Dawson, N. M. Nasrabadi, Prosodic-enhanced siamese convolutional neural networks for cross-device text-independent speaker verification, in: 2018 IEEE 9th international conference on biometrics theory, applications and systems (BTAS), IEEE, 2018, pp. 1–7.
- [31] P. Kenny, G. Boulianne, P. Ouellet, P. Dumouchel, Speaker and session variability in gmm-based speaker verification, *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007) 1448–1460.
- [32] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, P. Dumouchel, A study of interspeaker variability in speaker verification, *IEEE Transactions on Audio, Speech, and Language Processing* 16 (2008) 980–988.
- [33] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, P. Dumouchel, Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification, in: Tenth Annual conference of the international speech communication association, 2009.
- [34] V. Peddinti, D. Povey, S. Khudanpur, A time delay neural network architecture for efficient modeling of long temporal contexts, in: Sixteenth annual conference of the international speech communication association, 2015.
- [35] J. Bromley, I. Guyon, Y. LeCun, E. Säcker, R. Shah, Signature verification using a "siamese" time delay neural network, *Advances in neural information processing systems* 6 (1993).
- [36] G. Koch, R. Zemel, R. Salakhutdinov, et al., Siamese neural networks for one-shot image recognition, in: *ICML deep learning workshop*, volume 2, Lille, 2015, p. 0.
- [37] Y. Yao, X. Wu, W. Zuo, D. Zhang, Learning siamese network with top-down modulation for visual tracking, in: *International Conference on Intelligent Science and Big Data Engineering*, Springer, 2018, pp. 378–388.
- [38] R. R. Varior, M. Haloi, G. Wang, Gated siamese convolutional neural network architecture for human re-identification, in: *European conference on computer vision*, Springer, 2016, pp. 791–808.
- [39] Y. Benajiba, J. Sun, Y. Zhang, L. Jiang, Z. Weng, O. Biran, Siamese networks for semantic pattern similarity, in: 2019 IEEE 13th International Conference on Semantic Computing (ICSC), IEEE, 2019, pp. 191–194.
- [40] W. Zhu, T. Yao, J. Ni, B. Wei, Z. Lu, Dependency-based siamese long short-term memory network for learning sentence representations, *PloS one* 13 (2018) e0193919.
- [41] J. Pelecanos, S. Sridharan, Feature warping for robust speaker verification, in: *Proceedings of 2001 A Speaker Odyssey: The Speaker Recognition Workshop*, European Speech Communication Association, 2001, pp. 213–218.
- [42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012. URL: <https://arxiv.org/abs/1207.0580>. doi:10.48550/ARXIV.1207.0580.
- [43] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, *Advances in neural information processing systems* 29 (2016).