POLITECNICO DI TORINO

Master Degree Thesis

Voice analysis: from speaker identification to speaker verification using Siamese Neural Network

Supervisors
Prof.ssa Silvia Chiusano

Candidates
Antonio Falabella
matricola: s261834

Internship Tutor
Dott. Ing. Emanuele Gallo

Accademic year 2020-2021

Summary

Nowadays one of the most tedious tasks of our online life is the verification of our identity through several passwords, passphrases, PINs and cards. In this work, we want to analyse the possibility of using our recorded voices to automatically identify ourselves. Identifying a person through his voice is an important human ability that, most of the time, is taken for granted in face-to-face interactions, but when the visual verification fails, like in telephone calls, it becomes crucial to correctly identify and verify who is speaking.

This thesis is about the identification and verification of a person's identity through his voice imprint using at most two raw audio inputs in the test phase. The first step of this work is to identify a person's identity among a small group of people using a Deep Learning approach. In this phase, the Neural Network will learn to create an internal representation of each person's almost-unique voice imprint belonging to the before-mentioned group. Then a bigger group of voices taken from the popular LibriSpeech dataset will be considered to evaluate the capacity of the network to generalise the problem. Several architectures, like Resnet and SincNet among the others, will be observed and their result on different datasets will be discussed. The second objective is a more general task: the focus will be shifted to the verification of the identity of a person among a virtually illimited number of people. In this step, a Siamese Neural Network will be proposed and several architectures will be presented.

Acknowledgements

I want to thank my family that has supported me during all my academic formation. Without them, I would not have reached this moment, or worse, I would have pursued a degree in architecture. Thanks to my father Nicola, my mother Teresa and my brother Raffaele for the support and confidence that they gave me during these years.
Thanks to my relatives for the support despite the thousands of kilometres that divide us in our everyday life.
Thanks to the Ing. Emanuele Gallo and Ing Julien Genovese for giving me the opportunity to work on this case study, for helping me through my last year of university and the first year of my job in Reply. Thanks to the Reply company for the resources employed.
Thanks to the Ing. Silvia Chiusano for helping me in the thought time of the last month of the thesis: without her, I would not be discussing the thesis in December.
Thanks to all my friends that shared with me the thousands of hours in Biblio studying, to all that made me live moments of leisure and happiness.

Contents

List of Figures

Part I

Problem

Chapter 1

Introduction

Identifying a person through his voice is an important human ability that, most of the time, is taken for granted in face-to-face interactions, but when the visual verification fails, like in telephone calls, it becomes crucial to correctly identify and verify who is speaking. Speaker recognition systems have emerged in later years as a way to verify identity in many e-commerce applications. Experts trained in forensic speaker recognition can perform this task even better by identifying a set of acoustic and linguistic characteristics of speech in so-called structured listening. Experienced researchers in machine learning and signal processing continue to develop automatic algorithms to effectively perform speaker recognition to the point where automatic systems start to perform on par with human listeners.
In the Second Chapter of this work, an overview of the State of Art of signal processing and Machine learning is proposed. Starting from the Shallow Learning approach with the Gaussian Mixture Model, which is a probabilistic model that assumes that the data belong to a mixture of a limited number of Gaussian distributions with unknown parameters. Then this approach is further developed with the Universal Background Model, where a giant GMM is trained to describe a speaker-independent distribution of the speech features. The evolution of this strategy is the creation of Supervector obtained from the concatenation of the parameters of the Gaussian Mixture Model on which Super Vector Machine is applied.
Another interesting approach examined is the Factor Analysis that is a model that tries to describe the variability of these high dimensional data using a

low amount of not observable variables. The idea is to describe the independent features, as the speaker, the channel and the environment components, the speaker-dependent elements and the channel/environment dependent segments, as separate elements of the entire signal. This path is further developed into the Join Factor Analysis that combines the properties of the GMM with the benefits of the Factor Analysis. This section ends with the presentation of the i-vector with PLDA that is one of the best performing approaches in the text-independent speaker recognition task. This model does not make a distinction between speaker and channel, but it is, instead, just a dimensionality reduction of the GMM super vector method.
In section 3 of Chapter 2 advanced techniques of audio analysis based on deep learning are presented. The section begins with the presentation of a basic Deep Neural Network composed of Fully connected layers and used as feature extractors for what so-called D-vector. In this network, after the training phase, the last layer is removed and the output of the previous layer is taken into consideration for the creation of D-vectors. These vectors are then used to create a model of the speaker from the audio. The natural evolution of this approach is the X-vector in which, instead of feeding the network with punctual information of the audio signal only in a given instant, more instants are taken into consideration. In this way, the network has a neighbourhood, a context to work with.
In the latter year, the literature has shown a growing use of networks ideally designed to work with images on this task, like Convolutional Neural Network and Residual Neural Network. These networks have shown huge potential in this field reaching the performance of the i-vector with PLDA. The third chapter begins with the definition of the identification problem, and in what it differs from the verification problem. Identification is the task of finding the unknown identity of a voice between a restricted group of people. It is a one to N match, where N is the number of people belonging to the group taken into consideration. On the other hand, Speaker verification is a match between only one voice and only one identity.
In this chapter is present the analysis of the datasets used. The first dataset is the 5 voices dataset, which is a collection of audios of famous speeches taken from 3 men and 2 women: Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Tacher and Nelson Mandela, enriched with a selection of background noise such as the audience laughing or clapping. In section 3 of this chapter, a Convolutional Neural Network and a Residual Neural Network are trained on the 5 voice dataset.
The Convolutional Neural Network is a feed-forward neural network in which

the connectivity patterns between neurons are inspired by how the brain process the images. The more characteristic layer of this network is the Convolutional Layer that creates an abstracted feature map applying different filter functions to the input. This Network reached 98.4% accuracy on the evaluation set.
The second network presented is a Residual Neural network, which is a network that utilizes skip connections, also called shortcuts, to jump over some layers. Double and triple layer skip are implemented. The main reason to add skip connection is to avoid the problem of the vanishing gradients and to mitigate the accuracy saturation. In the first steps of the training phase, the network prefers using the skipping layer. This leads to a minor required time for the network to train. On the evaluation set, the ResNet reached an accuracy of 99.46%.
Since the dataset used is nearly perfect and far from reality, every sample was dirted with the noise present in the dataset. This passage took down the accuracy of the ResNet from the previous 99.46% to 97.9%. In the end, the ResNet comes out on top with a small margin over the CNN and, even if the ResNet has more parameters, takes less time to train thanks to the skip connections.
The second dataset used is the LibriSpeech one. This dataset is composed of approximately 1000 hours of English speech. The data is derived by reading audiobooks from the LibriVox project. Given the limitation of Google Colab Pro, only the subset "clean" was used. The ResNet was trained again on this, compared with the 5 voices dataset, bigger and less clean dataset, and reached an accuracy score of 85.76%. The third network presented is the SincNet that is a special Convolutional Network that can ingest raw audio signals as inputs before applying standard Convolutional or Dense Layers. The first layer of this network is the most critical part: it has to deal with very high dimensional inputs and is also the layer that is affected by vanishing gradients the most. Usually, the filters learned by the CNN take shapes of noisy multi-band filters. To avoid this behaviour some constraints on their shapes are set, forcing them to have a passband filter shape. This network shows an accuracy of 86.3%. In chapter 4 the speaker verification problem is presented. Usually, speaker verification is used as a "sentry" to provide access to a secure system, for example, it can be used to control the entry to a restricted area or to access privileged information. This process is conceptually really different from the identification problem since, here, the system has to correctly verify if two speeches are from the same person even if the system has never heard that person before. This leads the network to focus

the attention on the characteristics of each audio and on what can differ between two voices of different people. To solve this problem a Siamese Neural Network was developed. The Siamese Neural Networks have been used in the past for recognizing handwritten checks. This network works on two different input vectors at the same time, while using the same weights for each branch, to compute comparable output vectors. The first layer used in each branch of the network is the aforementioned SincLayer and the top-performing distance layers found were the Euclidean Distance layer and the Cosine Distance Layer. This network reached an accuracy score of 78.20% with a relatively big gap between the train and test set even after the implementation of the dropout layer.

Part II

Audio Signal Analysis

Chapter 2

State of Art

2.1 Characteristics of audio signals

Unlike other forms of biometrics (e.g., facial features, fingerprints), the human voice is a performance biometric. To put it in simple words the identity information of the speaker is embedded in how the speech is spoken and not, instead, on what is being said. This leads voice signals to have a high degree of variability.

It is important, in fact, to note that even the same person will say the same words in different ways at different times. This is known as style-shifting. For example, if the subject is performing other tasks while speaking, such as writing, driving a vehicle, the voice will be affected by the Situational task stress. Another example can be the emotion that the subject is feeling while communicating (e.g. anger, sadness, happiness, etc.) [23] or physiological like the subject has some illness like a cold or is under the influence of medication, this can include ageing too.

Every speaker has some personal traits in his voice that are unique even if they may not be easily discernible but are different due to the talker vocal anatomy and learned habits of articulation. Also, identical twins have some differences in their voices even if hard to distinguish [15] [21].

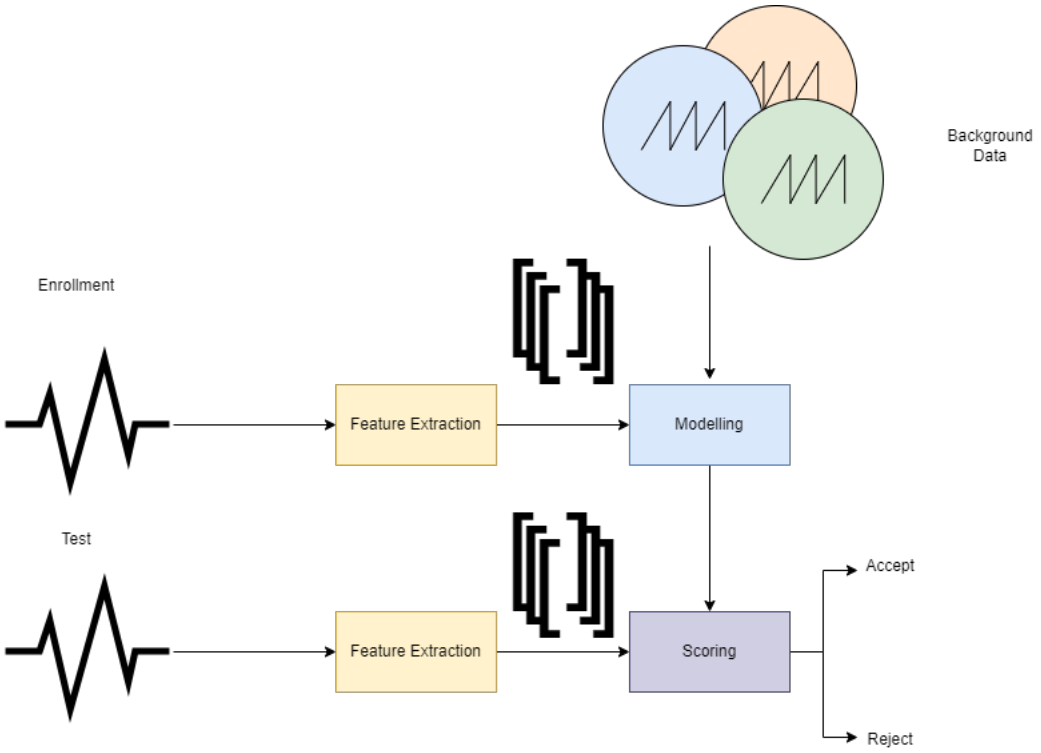

Figure 2.1. Basic speaker-verification system

A simple diagram in Fig. 2.1 represents an automatic speaker verification system. Predefined feature parameters are first extracted from the audio registrations and are meant to catch the characteristics of a voice in mathematical parameters. These features inherited from the previously taken audio of the speaker are used to build and train mathematical representations that abstract their speaker properties. For an unseen test segment, the same features are extracted and they are compared against the model recreated in the enrollment phase. The model is designed so that this comparison returns a score indicating if the two audios are from the same speaker or not. If this score is higher than a given threshold the system will accept the test speaker and, otherwise, it will be rejected. In some automatic systems, the creation of a model to abstract a speaker voice can start from background noise data as shown in Fig. 2.1.

Feature parameters are extracted from an entire utterance. This becomes more important in the automatic speaker recognition context because many common algorithms that recognise patterns operate on vectors of fixed dimension. These features are extracted from utterances of the duration of around 20-25 milliseconds. The most popular short-term acoustic features are the Mel-frequency cepstral coefficients [3]. To obtain these coefficients from a recording the audio must be divided into short overlapping segments. Typically 25-millisecond segments are used. The signal obtained in these frames has to pass through a window function (Hamming is the most used), and the Fourier power spectrum is generated. Then the logarithm of this spectrum is taken into consideration and the nonlinearly spaced Mel-space filter-bank analysis is performed. A tipical 24-channel filter bank used. The filter bank is designed in a way that is more sensitive to frequencies in the lower end of the spectrum. The same characteristic is present in the human ear. In the end, MFCCs are obtained by applying cosine transformation on the filter bank energy parameters. One of the desirable properties of an acoustic feature is the robustness to degradation and noise. In reality, it is not possible to design a feature that will remain unchanged if the acoustic conditions will differ and meanwhile providing meaningful speaker-dependent information. To minimize these changes the cepstral mean subtraction is used. It is important to notice that the normalization techniques are not designed to improve the ability of the features to discriminate, but they aim to adjust them so they are more harmonious among several different expressions. With the audio segment converted to feature parameters, the new task of the recognition of the speaker is the modelling. The model must provide means of its comparison with an unknown utterance. Such model is called robust when its characterizing process of describing the feature properties is not heavily affected by unwanted distortions even if the features themself are.

Most speaker-modelling techniques, like the Gaussian distributed, make various assumptions on the features that not always are met. If that is the case imperfections will be introduced during the modelling phase.

However, from the speaker-recognition research trend in the latter years, it appears that increasing feature robustness beyond a given level is very challenging.

2.2 Shallow Learning

2.2.1 Gaussian Mixture Model

A gaussian mixture model (GMM) is a combination of the Gaussian probability functions typically used to model multivariate data. This method clusters the data in an unsupervised way, but it provides a probability density function of the data. Using the GMM to model a person's features will result in a speaker-dependent probability density function [6]. Evaluating this function at different data points will provide a score that can be used to compute the similarity between the GMMs of two different speakers. This has been found as one of the most effective ways to model short-term features in a text-independent speaker-recognition task, where there is no prior knowledge about the content of the audio.

2.2.2 Gaussian Mixture Model with UBM

For speaker verification an alternate general speaker model that will represent speakers other than the target is needed so these two models can be compared and the more likely model can be chosen. The other speaker model, known as the Background model, is basically a giant GMM trained to describe the speaker-independent distribution of the speech features for all talkers in general [17]. The UBM is assumed to be a universal model. In contrast to performing maximum likelihood training of the GMM for a speaker, this model will update the well-trained UBM parameters to fit the speaker features. This relation between the UBM and the model that represent the speaker will provide better performance than the crude GMMs.

2.2.3 Gaussian Mixture Model with Supervectors and SVM

One of the main issues with speaker verification is that the training data differs in duration from test data. So one of the problems of speaker recognition is to obtain a fixed-dimensional representation of a single utterance. This is extremely valuable because several different classifiers can work on these utterance-level features from the machine learning literature. The solution to obtain a vector of fixed length from an utterance of variable duration is the formation of a GMM super vector, which is a large vector obtained by concatenating the parameters of a GMM model. The term super vector was first

used in this context for eigenvoice speaker adaptation in speech recognition applications [10]. For speaker recognition, super vectors were first introduced in [13], motivating new model adaptation strategies involving eigenvoice and MAP adaptation.

SVM [2] is one of the most popular classifiers in machine learning. In the work "Support vector machines using GMM supervectors for speaker verification" [1], it was pointed out that GMM super vectors could be used for speaker recognition and verification effectively using SVM. The vectors obtained from the training utterances were used as examples while a set of impostor utterances were used as negative examples. Using GMM supervectors with SVM and NAP [20] provided the most effective solution.

2.2.4 JFA and i-vectors

Factor Analysis (FA) aims to describe the variability in high dimensional observable data, using a lower number of not observable variables. For this task, the idea of describing the speaker and the channel-dependent variability using factor analysis was brought to attention in [9]. The current state of the art of this approach is the i-vector. A speaker-dependent GMM super vector is generally a linear combination of four components:

- speaker/channel/environment independent component
- speaker-dependent component
- channel/ environment dependent component
- residual

The first component is a constant obtained from the UBM. The others are casual vectors and are accountable for variability in the super vectors due to various aspects.

The first FA-related model used in speaker recognition was the eigenvoice method [10]. This method was proposed for speaker adaptation. In short, this method restricts the speaker model parameters to belong in a lower-dimensional subspace, which is defined by the eigenvoice matrix. The GMM mean super vector is composed by:

- the speaker-independent super vector obtained starting from the UBM
- the matrix that spans the speaker subspace and the standard normal hidden variables known as speaker factors.

The speaker factors need to be determined for an enrollment talker.

The join FA (JFA) is formulated starting by the eigenvoice with the MAP adaptation for a single model is applied. This model assumes that both speaker and channel variability lie in lower-dimensional subspaces of the GMM super vector space. This is one of the few models that consider all four components of the linear distortion model. JFA was shown to outperform the other contemporary methods [8] [11].

Both JFA and GMM SVM were among the state of the art systems and in the attempt to combine the strengths of each approach JFA was used as feature extractor for SVM [5]. In the first attempt, the speaker factors extracted by JFA were used in an SVM classifier. Since channel factors contain speaker-dependent information, those two factors were combined in a total variability space [5]. As with the other FA methods, the hidden variables are not observable but can be estimated by their posterior expectation. The estimates of the factors, that can be used as features in a classifier, are called i-vector that is short for "identity vector". The i-vector strategy does not make a distinction between speaker and channel, but instead, it is just a dimensionality reduction of the GMM super vector method. In the end, it is similar to a PCA model on the GMM super vectors.

It has been proved [4] [7] that PLDA on top of i-vector is one of the best performing approaches to text-independent speaker recognition tasks. The Probabilistic Linear Discriminant Analysis on the i-vectors can decompose the total variability into two variabilities: the speaker and the session. These 2 variabilities can be easily compared to the JFA ones

2.3 Advanced techniques of audio analysis and Deep Learning

2.3.1 D-vectors and X-vectors

As in the i-vector approach, the DNN aims to give a compact representation of the speaker acoustic frames using a Deep Neural Network instead of a generative Factor Analysis model [22]. In Fig 2.2 is shown the architecture of this DNN [22]. Once the model has been trained successfully the last layer, the output layer, is removed and the output of the last hidden layer is taken into consideration as the new speaker representation for that given utterance. For every frame of the audio belonging to a new speaker, the output activations of the last hidden layer are computed and then accumulated to form a new compact representation of that speaker: the D-vector. An improvement

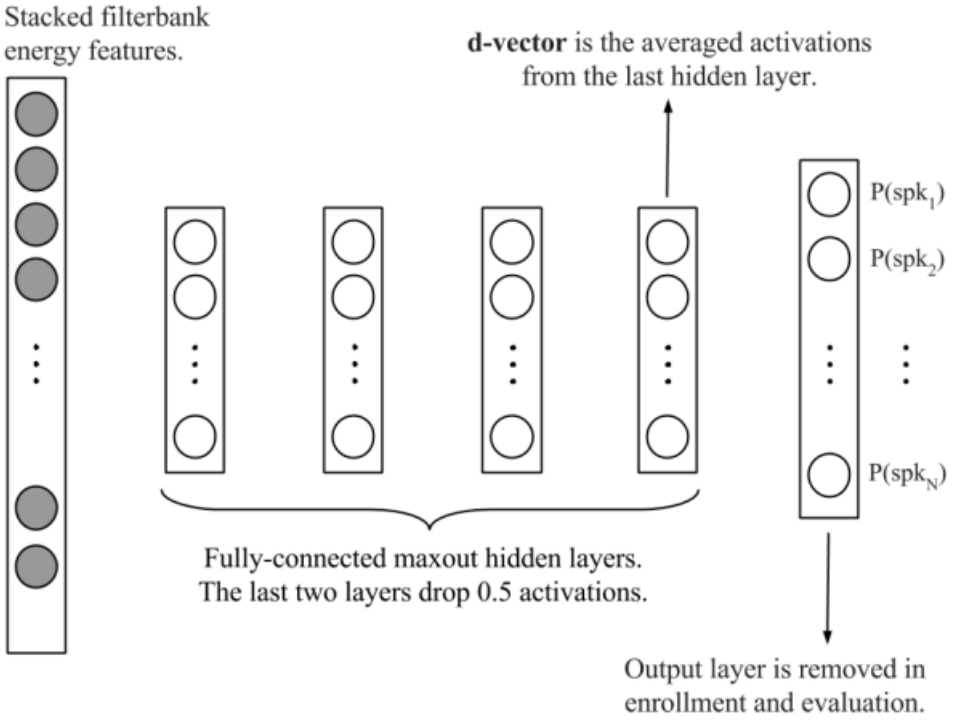

Figure 2.2. DNN architecture

of this method is to consider a sliding window of the signal up to 3 seconds instead of a fixed length utterance. This gives the network some information

about the context of the utterance evaluated and improve its performance as demonstrated in literature [19]. These new models create a strong contender for the representations of speaker recognition and it is called X-vector.

2.3.2 CNN and ResNet

A Convolutional Neural Network is a class of artificial neural networks designed to analyze images but they have applications also in video recognition, image classification, medical image analysis, recommender systems, natural language processing and much more. The characteristic of this network is that it takes benefit of the hierarchical patterns in data and it shapes filters that model patterns of growing complexity using simple filters. Convolutional Neural Network [14] in the latter year have been used for speech recognition too [18] with good results. CNNs are a type of neural network that can be used to register spatial or temporal correlation while decreasing translational fluctuations in signal. They can capture translational invariance with few parameters by replicating weights across time and frequency in opposite to Dense Neural Networks that need sufficient deep architecture and a lot of training examples.

A Residual Neural Network is an artificial neural network that uses double or triple layer skips. These skips are added to mitigate the problem of vanishing gradient and to try to avoid the saturation of the accuracy. The saturation of the accuracy is a problem that occurs in deep models when adding more layers results in higher training errors. Introducing the residual connections to the CNN and normalising the residual blocks, ResNets are capable of training deep networks to deliver better performance than normal CNN [24]. More about CNNs and ResNets in Chapter 3.3.

Part III

Speaker Identification and Verification

Chapter 3

Speaker identification

3.1 The identification problem

3.1.1 Definition of the problem

Speaker Identification is the identification of a person by the characteristics of his voice. It is the task that needs to be completed to answer the "Who is speaking?" question.

Recognizing the speaker is crucial in several tasks, as translating speech systems that have been trained on specific voices, in order to simplify them. Speaker recognition uses the acoustic features of speech that have been found to differ between individuals. These acoustic patterns are derived from both anatomy and learned behaviour.

There are two major applications of speaker recognition: one is Speaker Identification and the other is Speaker Verification. If a speaker claims to be a certain person and audio containing a voice is used to test this claim, this is called verification or authentication. Identification is the task of finding the unknown identity of a voice between a restricted group of people, so it is a one to N match where N is the number of persons belonging to the group. On the other hand, Speaker verification is a one to one match between one voice and one identity.

3.2 5 voices dataset

3.2.1 Presentation

This dataset is composed of 7 folders, divided into 2 groups. Voice samples are in the first 5 folders, each folder for each different speaker. Each folder contains around 1500 audio files of one second long and sampled at 16000 hertz with PCM encoding. The other 2 folders contain general noise and background noise. The files in these last 2 folders are longer than 1 second and not sampled ad 16000 hertz so they need to be preprocessed. The dataset is composed of voice samples of 3 men and 2 women. They are Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Tacher and Nelson Mandela which also represents the folder names. The background noise instead does not present audios that are speeches but is composed of sounds that can be found inside the speaker environment, for example, the audience laughing or clapping.

3.2.2 Preprocessing

The two categories inside the dataset are sorted into 2 folders:

- an audio folder that contains all the per-speaker speech sample folders
- a noise folder that will contain all the noise samples

Through the execution of a bash file, using the library ffmpeg, the samples are converted from flac to wave with the following command inside the bash file:

```
ffmpeg -y -f flac -i $flacfile -ab 64k -ac 1 -ar 16000 -f wav "
   ${flacfile%.*}.wav"
```

Where FLAC is the format, 64k stands for 64kb per second and -ac 1 is the channel of the original file; while -ar 16000 is the desired sample rate of the output file which will be saved in the wave format file. Then a list of paths of each file of each folder is created and given as input to a function that creates the label from the name of the folder and decodes the wave file into a tensor. In the decoding phase, only the first second of each audio is taken into consideration and the others remain unused. It follows a Fast Fourier Transformation and a generation of a label for each signal. Then the dataset is shuffled and is split into three parts: a test, a validation and a train set.

3.3 Neural Networks

3.3.1 Convolutional Neural Network

The first Neural Network used is a very straightforward Convolutional Neural Network. In automatic learning, the Convolutional Neural Network (CNN or ConvNet) is a feed-forward neural network in which the connectivity patterns between neurons are inspired by how the brain process the images. Each neuron is placed in order to react to the region of overlap that dowel the visual field. There are several layers that can compose a CNN. The most used ones are the following:

- Convolutional Layer: create an abstracted feature map, also called activation map, with shape number of inputs x feature map height x feature map width x feature map channels
- Pooling layers: can be local or global and they reduce the dimensions of data by combining the outputs of the previous layer.
- Fully connected layers: each neuron of the previous layer is connected to each neuron of the fully connected layer. It is the same as a traditional multi-connected layer perceptron neural network
- Activation layer: an activation function is performed on each output for example a ReLU function that is a non-saturating activation function that effectively removes negative values from the outputs by setting them to zero. It also introduces nonlinearities in the overall network without affecting the convolution layers.

In the following Fig. 3.1, the used CNN is represented.

The neural network is fed with one sample at a time and it is composed of the following layers:

- The input layer is the layer that receives the top half of the data obtained by the FTT
- Convolutional layer in one dimension that creates a convolution kernel that is convolved with the layer input over a single temporal dimension to produce a tensor of outputs
- ReLU function is used as activation layer

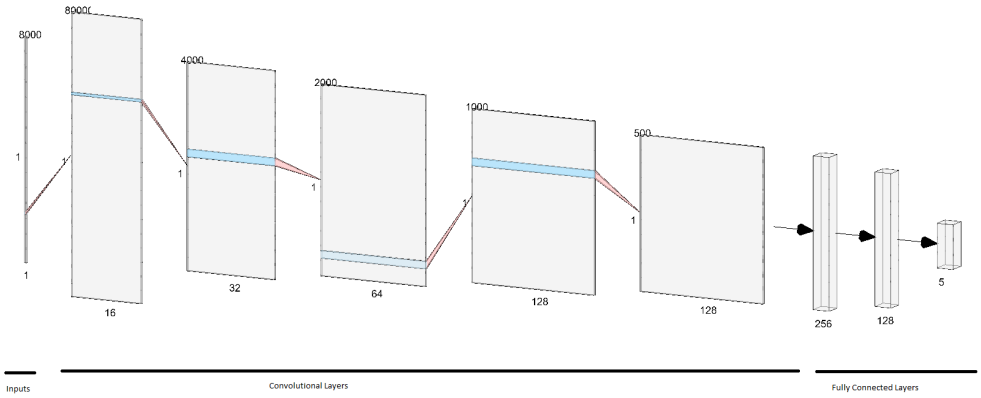

Figure 3.1. CNN

- Max Pool in one dimension is particularly useful temporal data. It downsamples the input representation by taking the maximum value over a spatial window of size 2 in our case with stride 2.

- Average pooling in one dimension differs from the previous one by taking the average value over the window defined by the pool size that in this context is 3 with stride 3.

- Flatten layer flattens the input and does not affect the batch size. it is used after the convolutional layers and before the fully connected layer

- Dense layers will terminate the neural networks.

In Fig. 3.1 each convolution layers is in reality composed of two convolutions followed by one activation function each and a maximum pooling layer. The last convolution instead uses the average pooling layer in one dimension described before. The first fully connected layer hides in itself a flatten layer that is used to flatten all the 128 output layers of the previous convolutional layer. Each fully connected layer use the ReLU activation function while the last one uses the softmax activation function instead.

In the training phase, the Adam optimizer and the sparse categorical cross-entropy were used. The Adam optimization is a stochastic gradient descent

method that is based on the adaptive estimation of first-order and second-order moments. According to Kingma et al., 2014, the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters" [12].

Categorical cross-entropy is a loss function that well suits multi-class classification tasks. These are tasks where a sample can only belong to one out of many possible categories, and the model must decide which one. Formally, it is designed to quantify the difference between two probability distributions. The categorical crossentropy loss function calculates the loss of a sample by computing the following sum:

$$Loss = -\sum_{i=1}^{outputsize} y_i \cdot \log \hat{y}_i \tag{3.1}$$

The model was trained for 25 epochs with a batch size of 128.

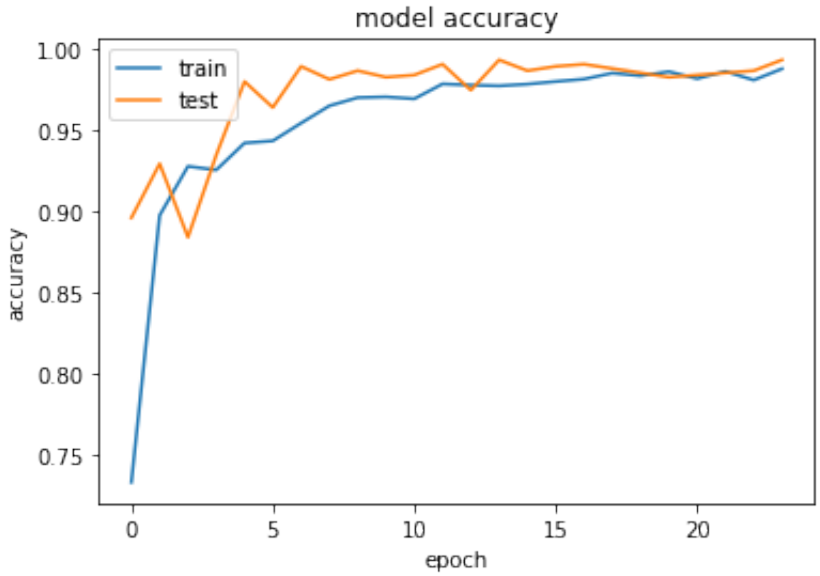

Figure 3.2. CNN accuracy

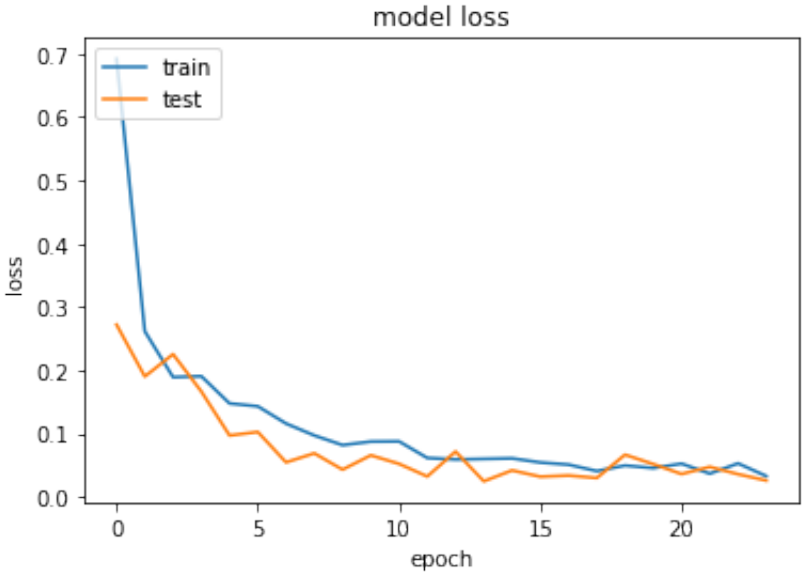

Figure 3.3. CNN loss

Fig. 3.2 and Fig. 3.3 show respectively the accuracy and the loss of the CNN on both the train and test set. As shown in the first figure the accuracy reached is 99.33% and the loss is 0.024 on the test set. In the evaluation phase, so on data that the CNN have never seen, the CNN showed an accuracy of 98.41%.

3.3.2 Residual Network

The second Neural Network proposed is the Residual Neural Network (ResNet). This is an artificial neural network that utilizes skip connections, or shortcuts to jump over some layers. In this ResNet, it is implemented a double and triple layer skip that contains nonlinearities in form of ReLU activation layer. There are two main reasons to add skip connections: to avoid the problem of vanishing gradients and to mitigate the accuracy saturation in which adding more layers leads to higher training error. In the training phase, the weights

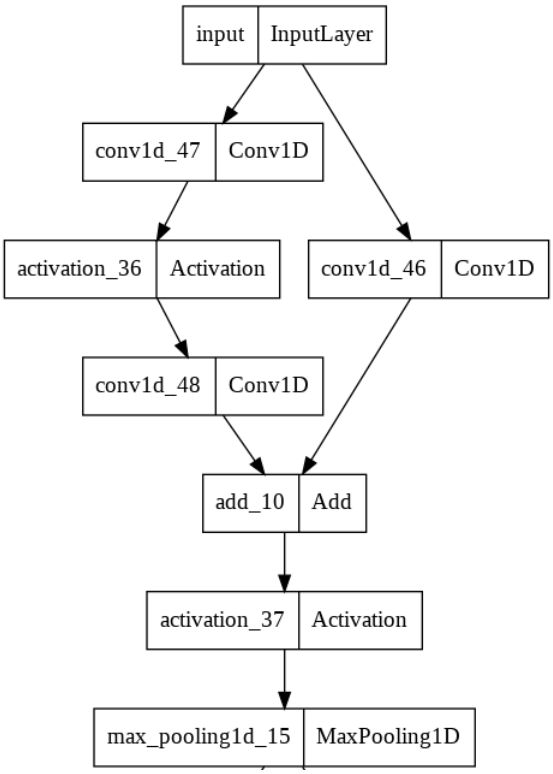

Figure 3.4. Double skip schema

adapt to prefer the skipping path over the more complex path. Skipping simplifies the network using fewer layers in the initial training stages and reduces the time required for the learning process. The layers used in this network are the same as the previous network but they are rearranged to fit the ResNet design. The schema of the double layer skip is shown in Fig. 3.4, while the schema of the triple layer skip is shown in Fig. 3.5.

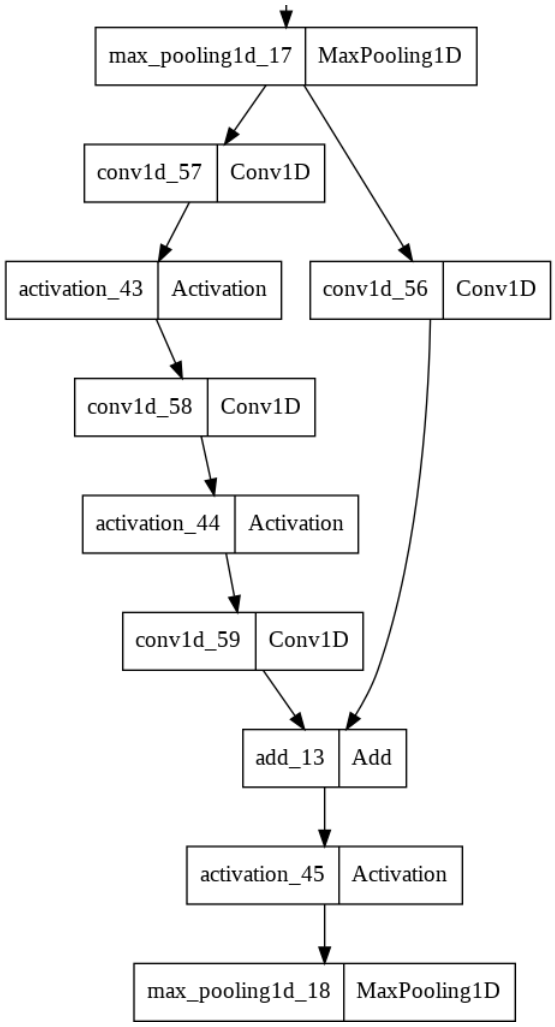

Figure 3.5. Triple skip schema

The architecture of this network is composed of two double skip schema followed by three triple skip schema, then an average pooling in one dimension, a flatten layer and terminated by three fully connected layers.

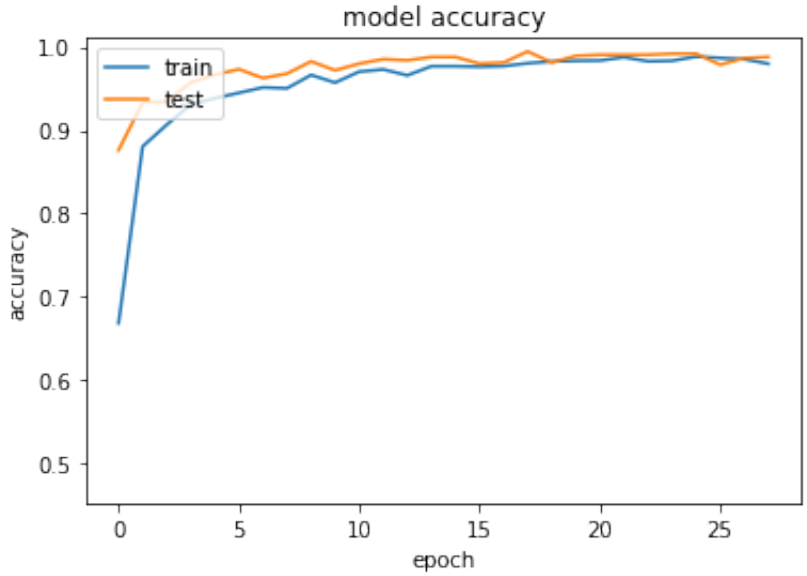

Figure 3.6. ResNet accuracy

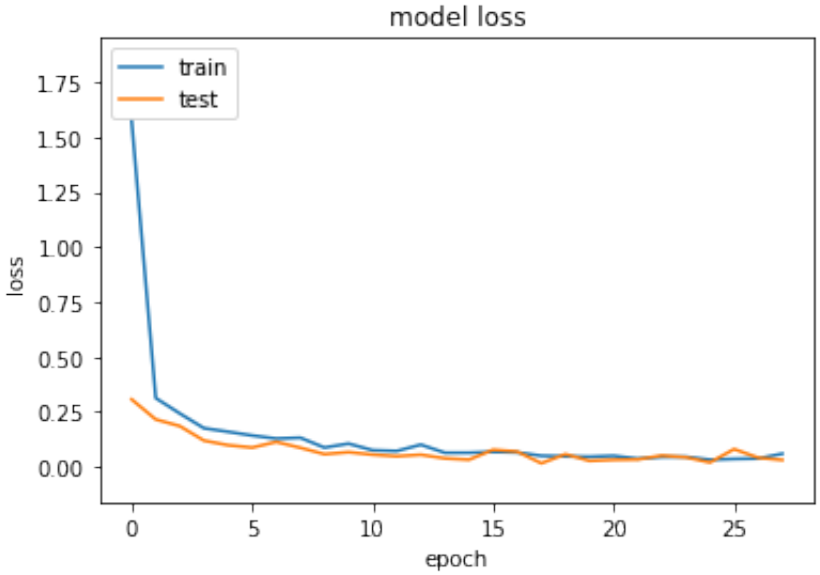

Figure 3.7. ResNet loss

The model was trained for 25 epochs with a batch size of 128. Fig. 3.6 and Fig. 3.7 show respectively the accuracy and the loss of the ResNet on both the train and test set. As shown in the first figure the accuracy reached is 99.2% and the loss is 0.043 on the test set. In the evaluation phase, so on data that the ResNet have never seen, the ResNet showed an accuracy of 99.46% and a loss of 0.016.

3.3.3 Adding noise

Since the results on both the network are really high, in this section, noise was added to the samples to try to generalize the problem. Inside the database, there are present some background noises within 2 folders and a total of 6 files.

These files are longer than one second and were originally not sampled at 16000Hz. Those six files were used to recreate 354 files of 1-second-long noise samples to be used for training. The noise needs also to be resampled to a sampling rate of 16000Hz and, in order to do this, the ffmpeg command is used again as shown in the next snippet of code:

```
command = (
    "for␣dir␣in␣‘ls␣-1␣" + DATASET_NOISE_PATH + "‘;␣do␣"
    "for␣file␣in␣‘ls␣-1␣" + DATASET_NOISE_PATH + "/$dir/*.wav‘;
       ␣do␣"
    "sample_rate=‘ffprobe␣-hide_banner␣-loglevel␣panic␣-
       show_streams␣"
    "$file␣|␣grep␣sample_rate␣|␣cut␣-f2␣-d=‘;␣"
    "if␣[␣$sample_rate␣-ne␣16000␣];␣then␣"
    "ffmpeg␣-hide_banner␣-loglevel␣panic␣-y␣"
    "-i␣$file␣-ar␣16000␣temp.wav;␣"
    "mv␣temp.wav␣$file;␣"
    "fi;␣done;␣done"
)
os.system(command)
```

Each audio sample is then merged with a random noise sample of the same lenght in a given amplitude proportion so the noise does not cover the audio completely. This step is preformed before the Fast Fourier Transformation.

The accuracy reached in this phase for the ResNet is 98.5% and the loss is 0.049 on the test set. In the evaluation phase, so on data that the ResNet have never seen, the ResNet showed an accuracy of 97.9% and a loss of 0.035.

3.3.4 Considerations

In this comparison, the ResNet comes out on top with a margin of just 0.13% over the CNN. The ResNet present 3'088'597 trainable parameters and 99.33% accuracy while the CNN has 2'950'165 trainable parameters and 99.46% but thanks to the skip connection the ResNet takes less time to train. The results show how good both CNN and ResNet perform in the identification task when the dataset contains few people and a huge number of samples. This lets the networks deeply learn the features of each person of the group. In the next chapter the ability of these networks, along with others, to work with a significantly larger group of people will be tested.

3.4 LibriSpeech: a 40 voices dataset

3.4.1 Presentation

LibriSpeech is a corpus of approximately 1000 hours of 16'000 Hz English speech, provided by Vassil Panayotov with the assistance of Daniel Povey. The data is derived by reading audiobooks from the LibriVox project and has been carefully segmented and aligned. The corpus is freely available under the very permissive CC BY 4.0 license. The data is split into 3 partitions of 100hr, 360hr, and 500hr while the dev and test data are split into the 'clean' and 'other' categories, each depending upon how well Automatic Speech Recognition systems would perform against. The test clean

Each of the dev and test audio is around 5hr in audio length. Each sample is saved as a FLAC file that stands for Free Lossless Audio Codec. This is a free audio codec with lossless compression. This means that the audio is compressed without losing quality opposite to lossy compressing such as MP3 or the AAC. This process does not remove information in the audio flow. FLAC is designed for the compression of audio data in fact it can perform compression between 30 to 50 % in contrast to the 10 to 20 % that the generic compression algorithm can archive.

The audios will be packed in the Waveform Audio File Format that is an audio file format standard revealed by IBM and Microsoft in August 1991 for storing audio on computers. This is the main format used on Windows systems for uncompressed audio even if it can archive compressed ones.

3.4.2 Preprocessing

The subsets with "clean" in their name are supposedly cleaner (at least on average) than the rest of the audio and US English accented. The data in the dev set clean that is used in the train part is divided into 40 folders, each of them named with the code of the speaker that made the audio inside the folder. The number of audios present in each folder can vary from 36 to 90 entries, for a total of 2703 audio. Along with the aforementioned folders are present some textual files that contain information about who is reading and what is reading inside each audio. In the Chapter.txt file are present:

- chapter_id: the ID of the chapter in the LibriVox's database
- reader_id: the ID of the reader in the LibriVox's database
- duration: how many minutes of this chapter are used in the corpus

- subset: the corpus subset to which this chapter is assigned
- project_id: the LibriVox project ID
- book_id: the Project Gutenberg's ID for the book on which the LibriVox project is based
- chapter_title: the title of the chapter on LibriVox
- project_title: the title of the LibriVox project

While in the Speakers.txt file:

- reader_id: the ID of the reader in the LibriVox's database
- gender: 'F' for female, 'M' for male
- subset: the corpus subset to which the reader's audio is assigned
- duration: total number of minutes of speech by the reader, included in the corpus
- name: the name under which the reader is registered in LibriVox

As done with the previous dataset, only the first second of each audio is extracted, labelled and decoded from the audio signal to a list of floating points. Then the signal is transformed by the Fast Fourier Transformation. A Fast Fourier Transform is an algorithm that computes the Discrete Fourier Transform of a signal. The Fourier analysis converts a signal from its original domain of space or time to a function in the frequency domain. The Discrete Transform is obtained by decomposing a sequence of values into components of different frequencies. The labels are encoded with the sklearn LabelEncoder. Then all the samples with the corresponding labels are shuffled and divided in two sets: one for training with 1893 samples and one for validation with 810 samples.

3.5 Neural Networks

3.5.1 Residual Networks

The ResNet used in this part is the same used on the 5 person dataset in section 3.3.2. The model was trained for 20 epochs with a batch size of 128. Fig. 3.8 and Fig. 3.9 show respectively the accuracy and the loss of the ResNet on both the train and test set. As shown in the first figure the accuracy reached is 85.49% and the loss is 0.013 on the test set. In the evaluation phase, so on data that the ResNet have never seen, the ResNet showed an accuracy of 83.46% and a loss of 1.084.

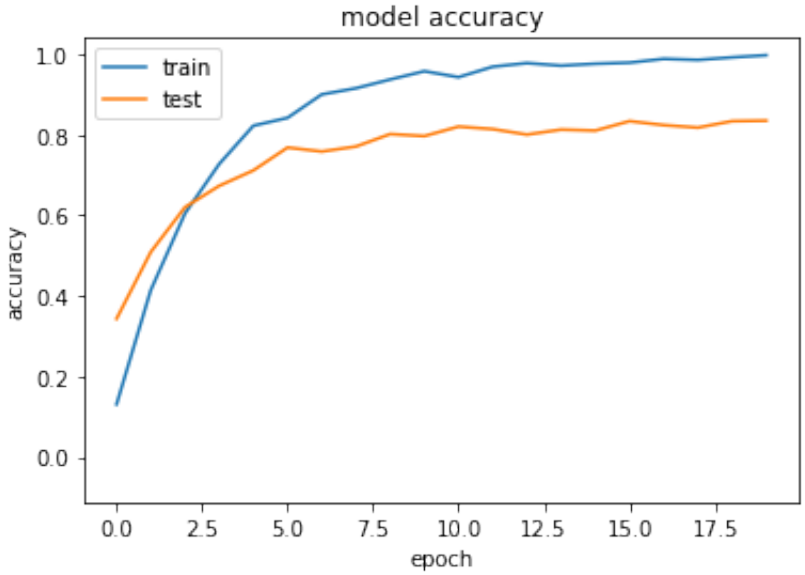

Figure 3.8. ResNet accuracy

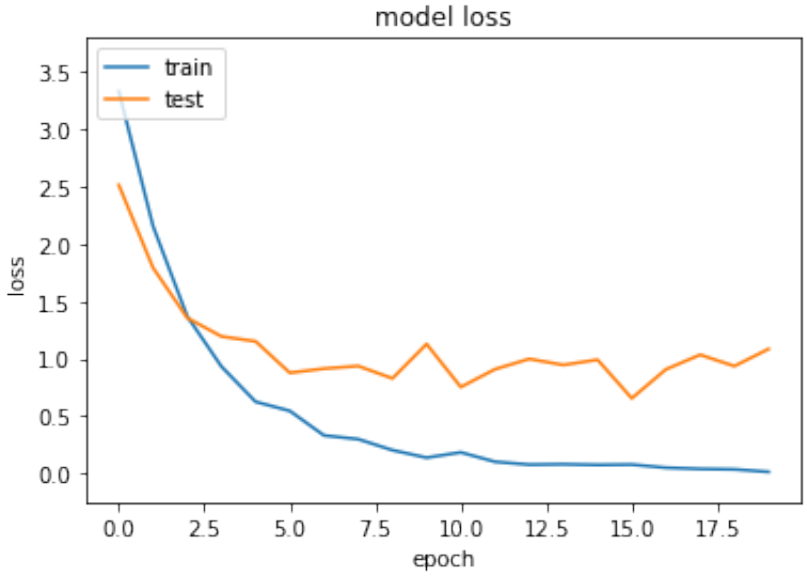

Figure 3.9. ResNet loss

3.5.2 Improved dataset

As said in section 3.4.2, even if the samples are longer, only the first second is taken into consideration in creating the dataset. This leads to an underutilization of the data. To fully utilize the dataset, in this section, each available sample will be taken into consideration in the creation of the dataset with the following function:

```
def fromFolderToAudioLabelSplitted(dataset):
    labels = []
    audio = []
    for folder in os.listdir(dataset):
        pointer = os.path.join(dataset, folder)
        if os.path.isdir(pointer):
            spid = folder;
            for folder1 in os.listdir(pointer):
                pointer1 = os.path.join(pointer, folder1)
                if os.path.isdir(pointer1):
                    for file in os.listdir(pointer1):
                        file_path = os.path.join(pointer1, file)
                        if os.path.isfile(file_path) & file_path.
                            endswith(".wav"):
```

```
                        file0 = tf.io.read_file(file_path)
                        wave, _ = tf.audio.decode_wav(file0,
                            1)
                        slices = int(wave.shape[0] /
                            SAMPLING_RATE)
                        sample = tf.split(wave[: slices *
                            SAMPLING_RATE], slices)
                        for i in sample:
                            labels.append(spid)
                        audio.extend(sample)
    return audio, labels
```

Moreover, from now on, the Fast Fourier Transformation will not be performed since the aim of this work is to develop an architecture that can elaborate raw audio input.
This leads to having around 450 entries for each person speaking for a total of 18024 samples. The model was trained for 25 epochs with a batch size of 128. Fig. 3.10 and Fig. 3.11 show respectively the accuracy and the loss of the ResNet on both the train and test set. As shown in the first figure the accuracy reached is 89.77% and the loss is 0.566 on the test set. In the evaluation phase, so on data that the ResNet have never seen, the ResNet showed an accuracy of 85.76% and a loss of 1.003.

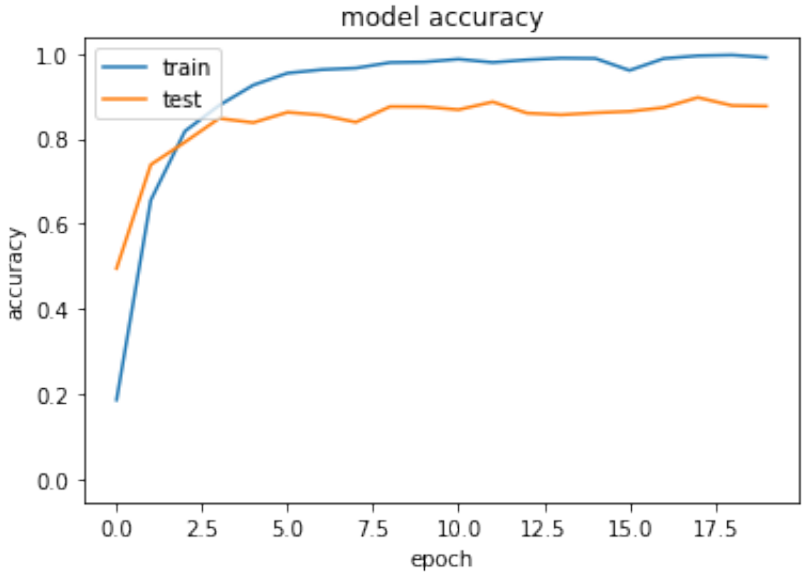

Figure 3.10. ResNet accuracy

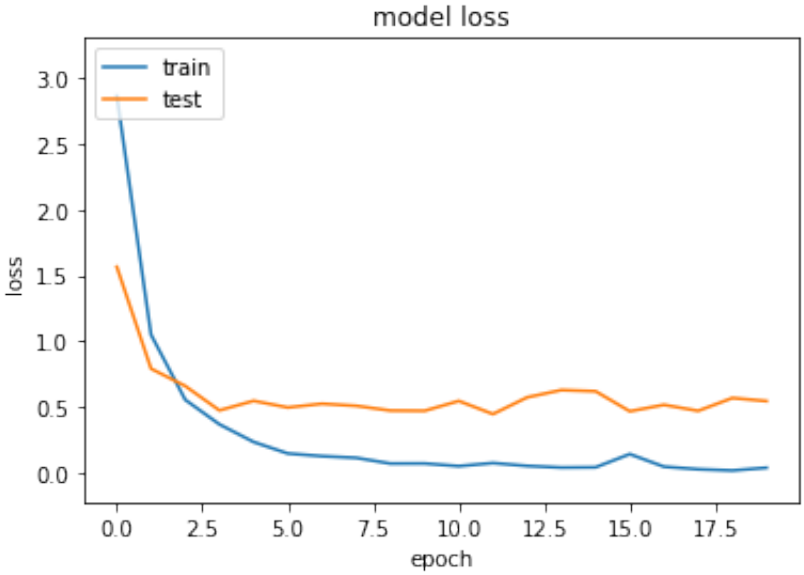

Figure 3.11. ResNet loss

Thanks to the new data the gap between the training accuracy and the test accuracy has thinned because the model has more information to train on in respect to before.

3.5.3 SincNet

SincNet [16] is a special Convolutional Network that can have raw audio signals as inputs before applying standard CNN or dense layers. This is because the first layer of a CNN is the most critical part. It not only has to deal with high-dimensional inputs but is also more affected by vanishing gradient problems especially when employed in very deep architectures like the ones presented in this work. The filters learned by CNN often take noisy multi-band shapes, as shown in Fig. 3.12, taken from [16], especially when only a few samples are available for training. To help the Convolutional

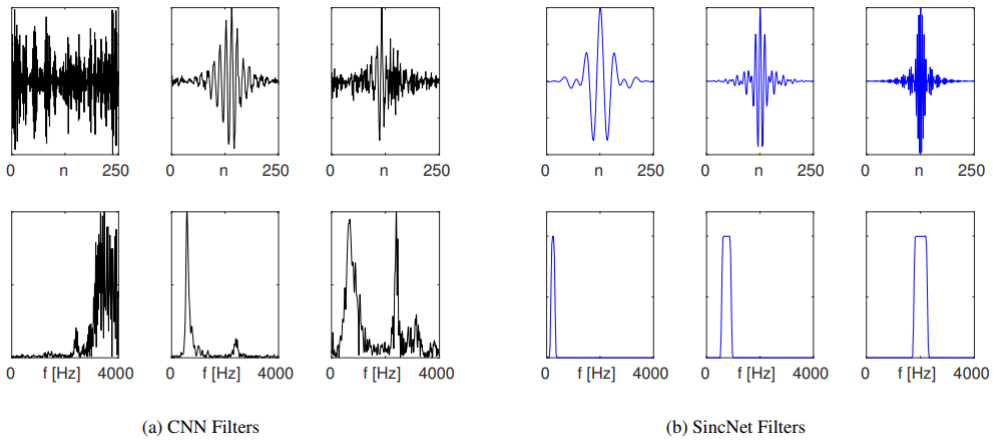

Figure 3.12. Sinc Layer

Neural Network more meaningful passband filters the SincLayer put some constraints on their shapes. This layer will only work on two parameters of the filter that are the low and high-cut frequency. This will force the network to focus on high-level parameters with a broad impact on the resulting shape and bandwidth of the filter. The SincNet layer can learn high and low cut-off frequencies of band-pass filters by a convolutional layer as shown in Fig. 3.13 [16].

The SincNet architecture (Fig. 3.14) used in the next part is the following one:

- The first rectangle is the SincLayer that can be interpreted as an advanced convolutional layer
- The second, third, fourth and fifth rectangles represent the combinations of max-pooling layer, batch normalization, leaky ReLU as activation

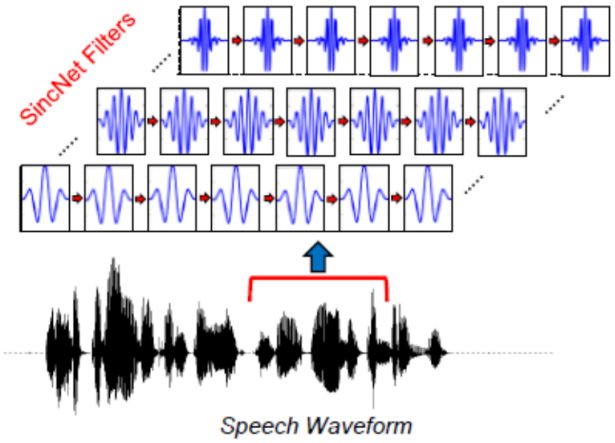

Figure 3.13. Sinc Layer

function and convolutional layer

- The last three rectangles represent the three-time repetition of dense layer, batch normalization, leaky ReLU and the dropout layer
- Between the dense layers and the convolutional layers is present a flattening layer

The purpose of the flattening layer is to squeeze the depth of the output of the previous layers to a one-dimension vector in order to feed it to the following dense layer. Above each macro-layer is presented dimension of the output of that layer.

The model was trained for 20 epochs with a batch size of 128. Fig. 3.15 shows an accuracy of 86.3% on the test set and an accuracy of 96.67% for the training set. To reduce the gap between the train and test set accuracy, dropouts layers have been introduced after CNN with really small dropout probability around 5%, and after dense layers with probability of 30%. In the evaluation phase the SincNet achieved an accuracy of 84.75% and a loss of 0.924.

3.5.4 Considerations

In the end, for the speaker identification task, there is not a clear winner between the ResNet and the SincNet architecture, since both of them register the approximately the same result. It is important to point out that the

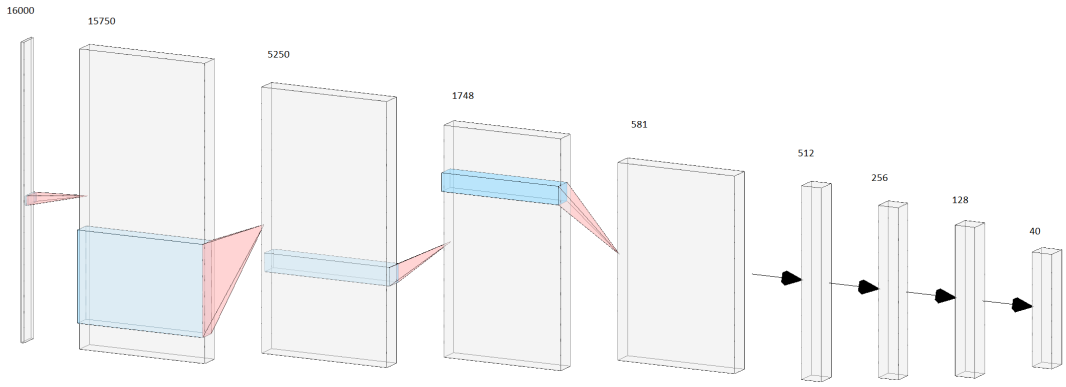

Figure 3.14. SincNet architecture

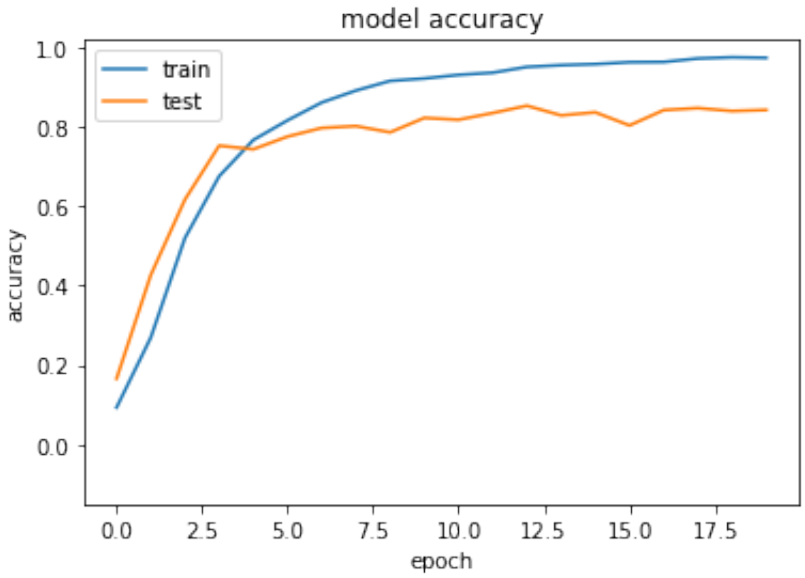

Figure 3.15. SincNet accuracy

number of trainable parameters for the ResNet is much higher than the number of the trainable parameters of the SincNet thank to the SincLayer of the latter one. This leads to a quicker training phase for the SincNet.

Chapter 4

Speaker verification

4.1 The verification problem

4.1.1 Definition of the problem

From a security perspective, verification is different from identification. Speaker verification is usually applied as a "sentry" to provide access to a secure system. These systems operate with the users' knowledge and usually expect their cooperation. Speaker identification systems can also be implemented covertly without the user's awareness to identify talkers in a discussion, alert automated systems of speaker changes, check if a user is already enrolled in a system, etc. In forensic applications, it is common to first perform a speaker identification process to create a list of "best matches" and then conduct a series of verification to define a convincing match. Working to match the samples from the speaker to the list of best matches figures out if they are the same person based on the number of similarities or differences. The prosecution and defence can use this as proof to settle if the suspect is truly the offender or not. Other applications of speaker verification may include entry control to a restricted area, access to privileged information, credit card authorizations, funds transfer and similar transactions.

4.2 Dataset

4.2.1 Preprocessing phase

The dataset used for this part is the LibriSpeech: a 40 voices dataset. For training, the dev set clean is used, while the validation is performed with the test set clean. For the Speaker Verification task the train, the test and the validation set generation follow a different path. The start of preprocessing phase is the same described in section 3.5.2:

- the audio are resampled in 16kHz
- saved in .wav file format
- decoded in vectors of fixed lenght (each second of each audio is considered)

Then each sample is paired one time with a sample from the same speaker and one time with a sample from a different random speaker of the same dataset division. To the pair of audio taken from the same speaker, a boolean flag set to True is used as label and for the others the flag is set to False since the audios are not from the same person with the following code:

```
def make_pairs(sounds, labels):
    pairSounds = []
    pairLabels = []
    numClasses = len(np.unique(labels))
    idx = [np.where(labels == i)[0] for i in range(0,
       numClasses)]
    #same class pairs
    for idxA in range(len(sounds)):
                # grab the current sounds and label belonging
                   to the current
                # iteration
        currentSounds = sounds[idxA]
        label = labels[idxA]
                # randomly pick an sounds that belongs to the *
                   same* class
                # label
        idxB = np.random.choice(idx[label])
        posSounds = sounds[idxB]
```

```
            # prepare a positive pair and update the sounds
                and labels
            # lists, respectively
        pairSounds.append([currentSounds, posSounds])
        pairLabels.append([1])

        #different class pairs
        negIdx = np.where(labels != label)[0]
        negSounds = sounds[np.random.choice(negIdx)]
        pairSounds.append([currentSounds, negSounds])
        pairLabels.append([0])
    return (np.array(pairSounds), np.array(pairLabels))
```

The pairs taken into consideration are 16'000 for train and 4'000 for test due to computational limit of Colab Pro. Higher numbers of pairs quickly saturated the ram leading to the crash of the whole system.

4.3 Siamese Neural Network

4.3.1 Presentation

A Siamese Neural Network (SNN) is an artificial neural network that works on two different input vectors at the same time, while using the same weights for each branch, to compute comparable output vectors. This is similar to comparing fingerprints, working with as a distance function. Siamese Neural Network has been used for recognizing handwritten checks and automatic detection of faces in camera pictures. In this section, the SNN will be applied to speaker verification.

4.3.2 Architecture

The Siamese Neural Network is composed of two branches that work in tandem on 2 different inputs and share weights of each layer. The architecture proposed for each branch of the siamese network is presented in Fig. 4.2 and can be schematized as follow:

1. Input is described in 4.2.1
2. SincLayer as presented in 3.5.3
3. 5 times the combination of Max Pooling, Batch Normalization, Leaky ReLU and Convolutional layer already described in 3.3.1
4. Dense Layer 3.3.1
5. Distance Layer
6. Dense Layer
7. Output Layer

A visual representation of the network is in Fig. 4.2 figure with the legend in Fig. 4.1 figure. The distances applied in the distance layer are the Euclidean Distance and the Cosine Distance.

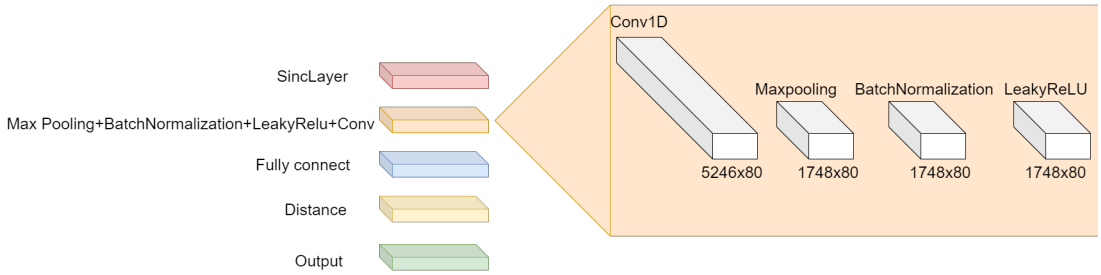

Figure 4.1. Siamese legend

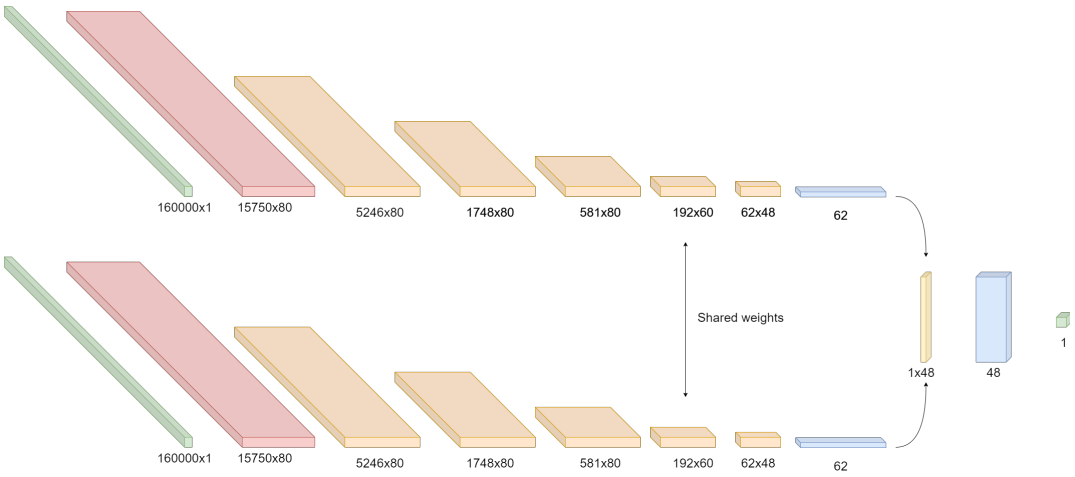

Figure 4.2. Siamese architecture

The loss used in this architecture is the Binary Cross-entropy loss function that is the negative average of the log of corrected predicted probabilities. It can be written as:

$$Loss = (Y)\,(-\log\,(Y_{pred})) + (1-Y)\,(-\log\,(1-Y_{pred})$$

It is important to notice that the first term is 0 when the Y is 0 and only the second term will have importance. Otherwise if the Y is 1 the second term will be zero.

4.3.3 Discussion of the results

The Siamese Neural Network has been trained for 100 epochs even if the maximum accuracy on the training set was reached around epoch 60 and the accuracy on the test set stopped to grow around the 30th epoch. The accuracy reached on the test set is 80.1% while the accuracy reached for the train set is 98.61%. The accuracy on the validation set is 78.20%. As shown

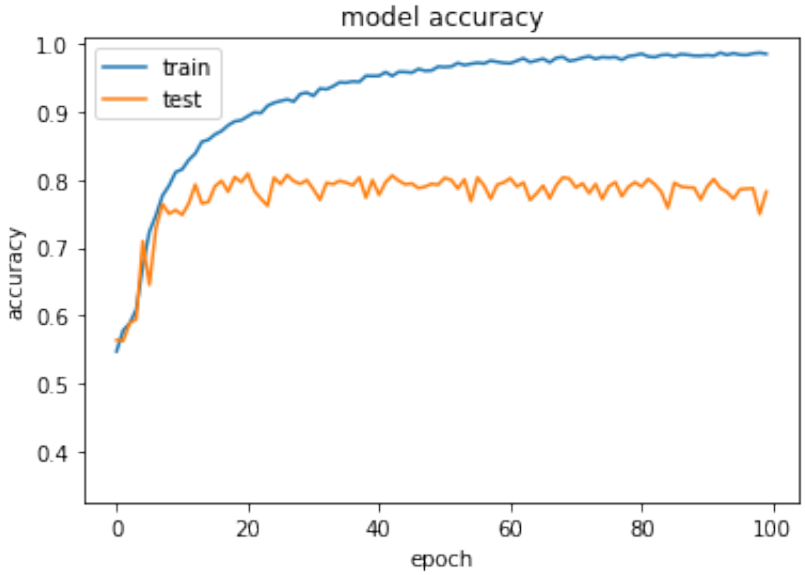

Figure 4.3. Siamese accuracy

in Fig. 4.3 the gap between the train accuracy and the test accuracy is around 15%. In order to thin the gap dropouts has been applied on both the convolutional layers and the dense layers. Also the different distance misures applied did not bring significant changes in the accuracy score.

Part IV

Conclusion

Chapter 5

Future work

The results obtained show the accuracy about the 80% for the verification task and about the 85% for the identification task. As said in the previous chapter the Colab Pro could not work on the whole dataset in the case of the verification task so a consideration to the future work can be to use more computational power and RAM capacity. In this work only a small part of the LibriSpeech dataset, the dev set, around 100 hours, was used. Using the whole dataset of more than 960 hours of clean and noisy audio can improve the performance. Also testing these networks on different dataset like VoxCeleb2 could lead to interesting results.
Another starting point for future work can be the evaluation of different lengths of the audio input. In this work, only segments 1-second length were used as inputs. The problem with this path is that every network developed must be adapted to the new input's length.
A different area of research, once reached higher accuracy scores, is the consideration of other metrics than just the accuracy, more valuable for the verification task as the recall, the precision and the F1-score.

Bibliography

[1] William Campbell, Douglas Sturim, and Douglas Reynolds. «Support vector machines using GMM supervectors for speaker verification». In: *Signal Processing Letters, IEEE* 13 (June 2006), pp. 308 –311. DOI: 10.1109/LSP.2006.870086 (cit. on p. 21).

[2] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Chem. Biol. Drug Des.* 297 (Jan. 2009), pp. 273–297. DOI: 10.1007/\%2FBF00994018 (cit. on p. 21).

[3] S. Davis and P. Mermelstein. «Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences». In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4 (1980), pp. 357–366. DOI: 10.1109/TASSP.1980.1163420 (cit. on p. 19).

[4] Najim Dehak et al. «Front-End Factor Analysis for Speaker Verification». In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4 (2011), pp. 788–798. DOI: 10.1109/TASL.2010.2064307 (cit. on p. 22).

[5] Najim Dehak et al. «Support vector machines and joint factor analysis for speaker verification». In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 4237–4240 (cit. on p. 22).

[6] J-L Gauvain and Chin-Hui Lee. «Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains». In: *IEEE transactions on speech and audio processing* 2.2 (1994), pp. 291–298 (cit. on p. 20).

[7] Patrick Kenny. «Bayesian Speaker Verification with Heavy-Tailed Priors». In: *Odyssey*. 2010 (cit. on p. 22).

[8] Patrick Kenny. «Joint factor analysis of speaker and session variability: Theory and algorithms». In: (Jan. 2006) (cit. on p. 22).

[9] Patrick Kenny and Pierre Dumouchel. «Disentangling speaker and channel effects in speaker verification». In: vol. 1. June 2004, pp. I –37. ISBN: 0-7803-8484-9. DOI: 10.1109/ICASSP.2004.1325916 (cit. on p. 21).

[10] Patrick Kenny, Mohamed Mihoubi, and Pierre Dumouchel. «New MAP estimators for speaker recognition.» In: Jan. 2003 (cit. on p. 21).

[11] Patrick Kenny et al. «Joint Factor Analysis Versus Eigenchannels in Speaker Recognition». In: *Audio, Speech, and Language Processing, IEEE Transactions on* 15 (June 2007), pp. 1435 –1447. DOI: 10.1109/TASL.2006.881693 (cit. on p. 22).

[12] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 31).

[13] Roland Kuhn et al. «Eigenvoices for speaker adaptation». In: *Proc. 5th International Conference on Spoken Language Processing (ICSLP 1998)*. 1998, paper 0303 (cit. on p. 21).

[14] Yann Lecun et al. «Gradient-Based Learning Applied to Document Recognition». In: *Proceedings of the IEEE* 86 (Dec. 1998), pp. 2278 –2324. DOI: 10.1109/5.726791 (cit. on p. 24).

[15] Francis Nolan and Tomasina Oh. «Identical twins, different voices». In: *The International Journal of Speech, Language and the Law* 3.1 (1996), pp. 39–49 (cit. on p. 17).

[16] Mirco Ravanelli and Yoshua Bengio. *Speaker Recognition from Raw Waveform with SincNet*. 2019. arXiv: 1808.00158 [eess.AS] (cit. on p. 44).

[17] Douglas A Reynolds. «Gaussian mixture models.» In: *Encyclopedia of biometrics* 741 (2009), pp. 659–663 (cit. on p. 20).

[18] Tara N. Sainath et al. «Deep convolutional neural networks for LVCSR». In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 8614–8618. DOI: 10.1109/ICASSP.2013.6639347 (cit. on p. 24).

[19] David Snyder et al. «X-Vectors: Robust DNN Embeddings for Speaker Recognition». In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 5329–5333. DOI: 10.1109/ICASSP.2018.8461375 (cit. on p. 24).

[20] A. Solomonoff, W.M. Campbell, and I. Boardman. «Advances in channel compensation for SVM speaker recognition». In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 1. 2005, I/629–I/632 Vol. 1. DOI: 10.1109/ICASSP.2005.1415192 (cit. on p. 21).

[21] WD Van Gysel, J Vercammen, and F Debruyne. «Voice similarity in identical twins.» In: *Acta oto-rhino-laryngologica Belgica* 55.1 (2001), pp. 49–55 (cit. on p. 17).

[22] Ehsan Variani et al. «Deep Neural Networks for Small Footprint Text-dependent Speaker Verification». In: *Proc. ICASSP*. 2014 (cit. on p. 23).

[23] Claude Vloeberghs et al. *The Impact of Speech Under" Stress" on Military Speech Technology.(l'Impact de la parole en condition de" stress" sur less technologies vocales militaries).* Tech. rep. NATO RESEARCH and TECHNOLOGY ORGANIZATION NEUILLY-SUR-SEINE (FRANCE), 2000 (cit. on p. 17).

[24] Qinghua Zhong et al. «Text-independent speaker recognition based on adaptive course learning loss and deep residual network». In: *EURASIP Journal on Advances in Signal Processing* 2021.1 (2021), pp. 1–16 (cit. on p. 24).