Speaker recognition for Voice User Interface

Joonas Turpeinen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 25.05.2020

Thesis supervisor:

Prof. Mikko Kurimo

Thesis advisor:

M.Sc. Juho Leinonen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Joonas Turpeinen

Title: Speaker recognition for Voice User Interface

Date: 25.05.2020 Language: English Number of pages: 6+39

Department of Electrical Engineering and Automation

Professorship: Control, Robotics and Autonomous Systems

Supervisor: Prof. Mikko Kurimo

Advisor: M.Sc. Juho Leinonen

Speech processing has been a major interest for academic, commercial and military communities for centuries. It allows more natural human-machine interaction than mouse-keyboard systems and may even replace them in many applications in the future. Speaker recognition, one of many applications of speech processing, is in growing demand. Speaker recognition can be used to identify speaker based on model, or validate that speaker is the person claimed to be. It allows various applications tailored for individual users in for example in education and banking.

The goal of this thesis is to evaluate two different speaker recognition systems for Voice User Interface (VUI). For time and resource constrains, no VUI is developed during thesis, but focus is on speaker recognition when noise has corrupted speech samples. The speech samples are first evaluated with Voice Activity Detection (VAD) algorithm, after which noise is added and Mel Frequency Cepstral Coefficients (MFCC) are calculated and normalized. Systems evaluated in the thesis, have been developed during previous studies and MFCC coefficients are present as inputs for these systems. The first system is Gaussian Mixture Model - Universal Background Model (GMM-UBM) developed by Microsoft Research team, which is based on Gaussian mixtures trained into speaker independent model. The second algorithm is based on X-vectors build from neural networks and developed by Chau Luu.

The data-set used for training was Voxceleb2 collected from Youtube and evaluation was test-sets from Voxceleb1 and Voxceleb2 as well as kid voice samples collected by speech recognition research team from Aalto University. The results shows that training GMM-UBM was successful and handling resource management can be done with little training. However, the x-vector model training was not as successful as GMM-UBM model training and it does require more knowledge from neural networks and computer processing.

Keywords: ASR, Speaker Recognition, GMM-UBM, X-vectors

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Joonas Turpeinen

Työn nimi: Käyttäjän tunnistus osana puhekäyttöliittymää

Päivämäärä: 25.05.2020 Kieli: Englanti Sivumäärä: 6+39

Automaation ja Sähkötekniikan laitos

Professuuri: Control, Robotics and Autonomous Systems

Työn valvoja: Prof. Mikko Kurimo

Työn ohjaaja: DI Juho Leinonen

Puheen prosessointi on ollut merkittävä tutkimusalue jo useiden vuosikymmenien ajan ja se kiinnostaa niin tutkijoita kuin yrityksiä. Muun muassa se mahdollistaa luonnollisemman käyttöliittymän kuin mitä hiiri ja näppäimistö tarjoaa. Puheella toimivat käyttöjärjestelmät saattavat hyvinkin korvata etenevässä määrin perinteisiä käyttöliittymiä. Puhujantunnistus on yksi automaattisen puheen käsittelyn osa-alue ja tämä tutkielma keskittyy siihen. Puhujantunnistus voidaan toteuttaa rakentamalla puhenäytteestä malli, joka yksilöi puhujan. Puhujantunnistus mahdollistaa muun muassa yksilöidyn palvelun vaikkapa pankkialalla tai opetuksessa.

Tämän tutkielman tavoitteena on verrata kahta eri algoritmia puheella toimivan käyttöjärjestelmän tueksi. Itse käyttöjärjestelmää ei kehitetty ajan ja resurssien puutteen vuoksi. Tutkimuskysymyksenä on, miten taustamelu vaikuttaa puhujantunnistuksen onnistumiseen. Ääninäytteistä poistetaan hiljaiset hetket Voice Activity Detection- algoritmin avulla. Tämän jälkeen melu lisätään näytteisiin, ja Mel Frequency Cepstral Coefficient -kertoimet lasketaan ja keskiarvoistetaan. Nämä kertoimet syötetään tutkittaviin järjestelmiin, jotka on kehitetty aikaisemmissa tutkimuksissa. Ensimmäinen tutkittava järjestelmä on Gaussian Mixture Model - Universal Background Model (GMM-UBM), jonka on kehittänyt Microsoftin puheentunnistukseen keskittynyt tutkimusryhmä. Toinen järjestelmä perustuu neuroverkoilla laskettuihin X-vektoreihin, jonka on kehittänyt Chau Luu.

Algoritmien opetukseen käytettiin Voxceleb2:n development-datajoukkoa, joka on kerätty Youtubista. Algoritmit testattiin Voxceleb1:n ja Voxceleb2:n testijoukkojen avulla, sekä lapsidatalla, joka on kerätty Aalto Yliopiston puheenkäsittelyn tutkimusryhmä. Tuloksista huomaa, että GMM-UBM järjestelmä tunnistaa puhujat paremmin kuin x-vektoreihin perustuva algoritmi. X-vektoreihin perustuva algoritmi mahdollisesti vaatisi enemmän neuroverkkojen hallintaa.

Avainsanat: Automaattinen puheenkäsittely, Puhujan tunnistus, GMM-UBM, X-vektorit

Preface

I want to thank Professor Mikko Kurimo and my instructor Juho Leinonen for his guidance.

Otaniemi, May 29, 2020

Joonas Turpeinen

Contents

Abbreviations

ASR	Automatic speech recognition
VUI	Voice user interface
DNN	Deep neural network
TDNN	Time Delay neural network
GMM-UBM	Gaussian Mixture Model - Universal Backround Model
EM-algorithm	Expectation-Maximization algorithm
MAP adaptation	Maximum a Posteriori adaptation
X-vector	A feature convenient for identifying speakers

1 Introduction

1.1 Speaker recognition

Speech recognition has become a major interest for both academic and commercial communities. It allows more natural human-machine interaction than mouse-keyboard systems and may even replace them in many applications in the future. However, speech recognition algorithms are mostly designed for adults, who speak the chosen language, such as English, as their first language. Children and second language speakers are often neglected in speech recognition algorithms.

Speech processing has been a major interest for academic, commercial and military communities for centuries. It allows more natural human-machine interaction than mouse-keyboard systems and may even replace them in many applications in the future. Speaker recognition, one of many applications of speech processing, is in growing demand. Speaker recognition can be used to identify speaker based on model, or validate that speaker is the person claimed to be. It allows various applications tailored for individual users in for example in education and banking.

The goal of this thesis is to evaluate two different speaker recognition systems for Voice User Interface (VUI). For time and resource constrains, no VUI is developed during thesis, but focus is on speaker recognition when noise has corrupted speech samples.

1.2 Previous studies

There are number of speaker recognition alghorithms, but in here two system are discussed. These systems are:

- Gaussian Mixture Model - Universal Backround Model (GMM-UBM), which calculates speaker features based on Expectation-Maximization algorithm.
- Pytorch implementation based on work of Chau Luu, whos architecture is based on Kaldi.

UBM-GMM algorithm was first proposed by Douglas A. Reynolds and Richard C. Rose during mid-90s [1] and has remained since as a reference for comparing competing algorithms. It is based on number of Gaussian mixtures, which are first calculated for entire data set and then calculating maximum a posteriori (MAP) adaptation.

Chau Luu has developed a novel approach with neural networks. X-vectors are modern state-of-the-art algorithm for calculating speaker features and they are mostly developed for Kaldi toolbox. However, Chau Luu used Pytorch in order to create speaker features with more stable system than Kaldi.

1.3 Thesis objectives

The objective of this thesis is to train and evaluate GMM-UBM and Chau Luus X-vector models for clean and noisy data for adult and child speech samples. Best ways to allocate resources are also interest of this study.

1.4 Thesis structure

Thesis is structured in five (5) sections. The first section is Introduction and it sets context for the whole thesis. Automatic Speaker recognition is the second section and it explains overall feature extraction process from speech to Mel Frequency Cepstral Coefficients (MFCC), which are commonly used for calculation of speaker independent models. In here, the data structure is also explained. How speakers are modelled and how recognition is done, is covered in Chapter 3 before results in Chapter 4 and conclusion in Chapter 5.

2 Feature extraction process

The basic theory of how humans produce and perceive speech is covered in [2]. The speech apparatus is shown in Figure 1. The speech apparatus components are:

- Lungs: source of air during speech.
- Trachea, also known as windpipe, is the tube that carries air between lungs and larynx. Huang et. al. [2] does not go deeply in the role of windpipe in speech production, but acoustically it should operate similar to flute.
- Larynx, also known as vocal cords. When vocal cords are close together and oscillate during sound creation, the sound is said to be voiced. During unvoiced sounds, the vocal cord is either too slack or too tense to vibrate periodically.
- Nasal cavity, which is usually referred as nasal tract.
- Vocal tract, which consists of both oral and pharyngeal cavity, or the throat, from larynx to lips.

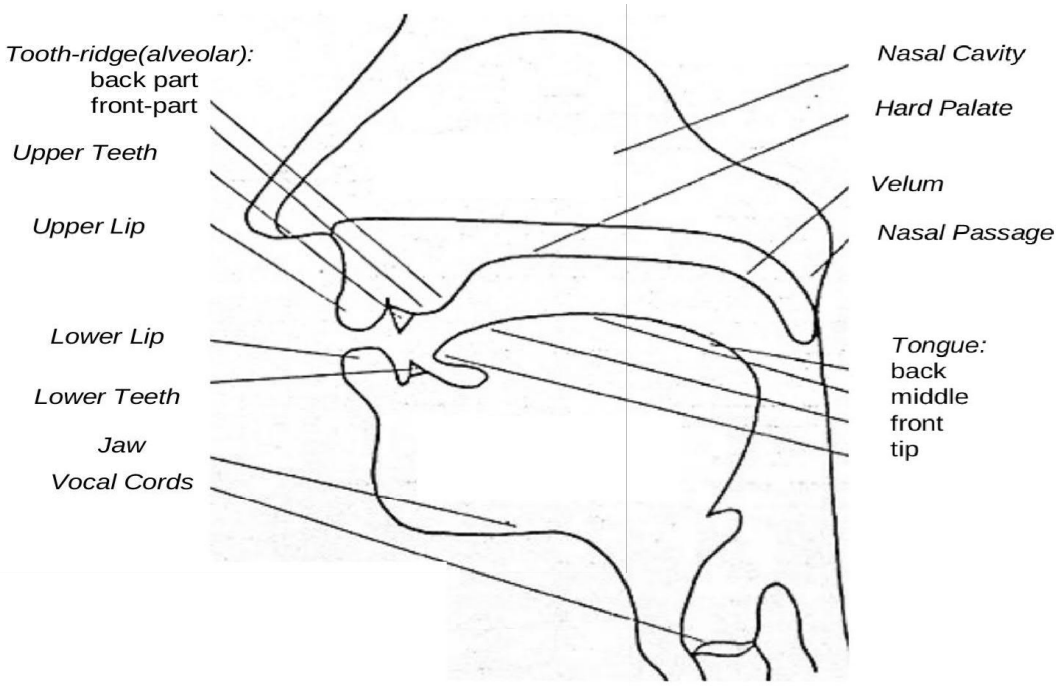

Figure 1: Human speech apparatus. [2]

Between nasal and vocal tract, there also exists hard and soft palate. The soft palate, the velum, opens to allow air passage through nasal cavity and closes to prevent it. Consonants, such as /n/ and /m/, are produces, when velum is open. The hard palate is the roof of mouth and most consonants are produces when tongue

is pressed against it.

In mouth, tongue, teeth and lips operates to articulate consonants and vowels. Tongue presses either hard palate or teeth at different locations to create different consonants. Vowels are produced when tongue does not touch hard palate but rather moves up and down, forward and backward for vowels and consonants. This alters sound frequency spectrum so that humans can differentiate phonemes from another, and construct words and sentences from speech. This also allows speaker recognition, as each speaker has unique speech apparatus with varying dimensions and speech styles.

The two most fundamental sound types speech and speaker recognition are interested in, are the voiced and voiceless sounds. The voiced sounds consists of vowels and some voiced consonants such as /z/ in English language. These sounds, and especially vowels, possess much more energy than voiceless sounds such as /s/ and thus appear more clearly in Fast Fourier Transform (FFT) and spectrograms.

The speaker identification process is Figure 2. The process begins with feature extraction following with pattern matching and decision subprocesses.

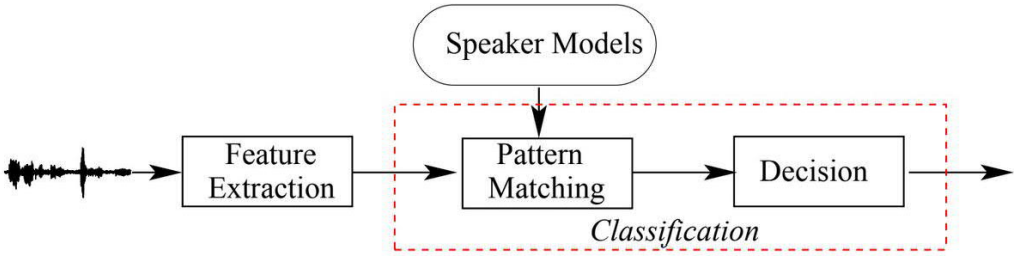

Figure 2: Speaker identification process. [3]

Automatic speaker recognition is often divided into two subcategories: speaker identification and validation. Speaker identification process aims to identify speaker from utterance. Speaker validation in turn aims to verify that either speaker is the person they claim to be or not. Both two categories are relevant in this thesis.

The generic speaker validation process is shown in Figure 3. The process begins with data acquisition, following with feature extraction and calculation of likelihoods before the claim is accepted or rejected. For continuous speaker recognition task, the speaker model can be updated with the most resent speaker data.

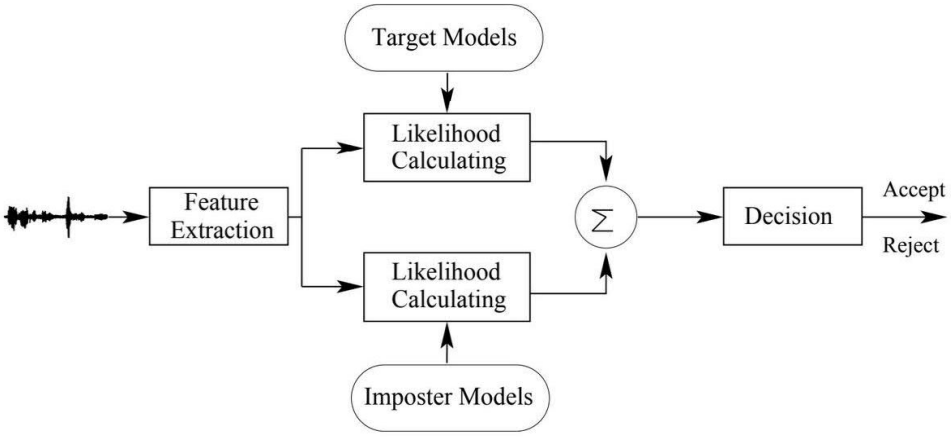

Figure 3: Generic speaker validation process. [3]

Speaker segmentation offers easy way of distinguishing speakers in a group as in Figure 4.

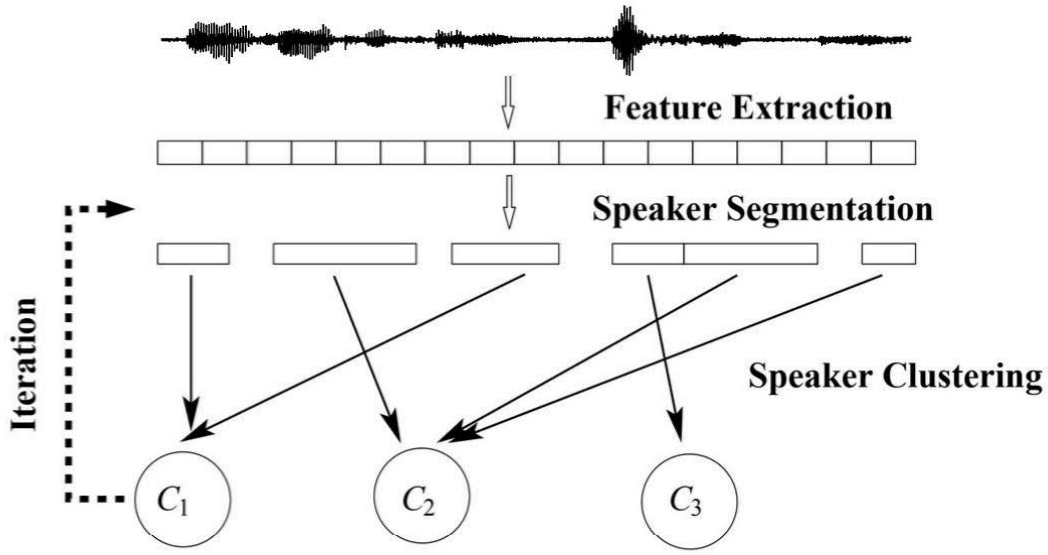

Figure 4: Speaker clustering algorithm. [3]

2.1 Feature extraction

Feature extraction is the first step in speaker recognition. The recognition model is trained from features, and accuracy of the model depends on the accuracy and usage

of these features. The sound consists of waves, which can be described as equation:

$$X = \sum_{k=0}^{\inf} A \sin(2\pi f_k t + \phi) \tag{1}$$

, where A is wave amplitude, f is frequency and ϕ is the phase of sound wave. As the voice consists of variety of frequencies, the it is natural to begin feature extraction by creating Fourier transform from speech sound. The Discrete Fourier Transform (DFT) is defined as:

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N}, \qquad 0 \leq k < N \tag{2}$$

, where $x_N[n]$ is a periodic signal with period of N, $X_N[k]$ is DFT of $x_N[n]$. Fast Fourier Transform (FFT) is often used in order to compute Fourier Transform in robust and accurate manner. FFT is a family of computational routines. In this thesis, spectrogram-function from Matlab is used with hamming window of length 400, which corresponds with 25ms time window. The spectrogram-function is set to produce 256 coefficients and it is used after high-pass filter with frequency respond shown in figure. 5 [2]

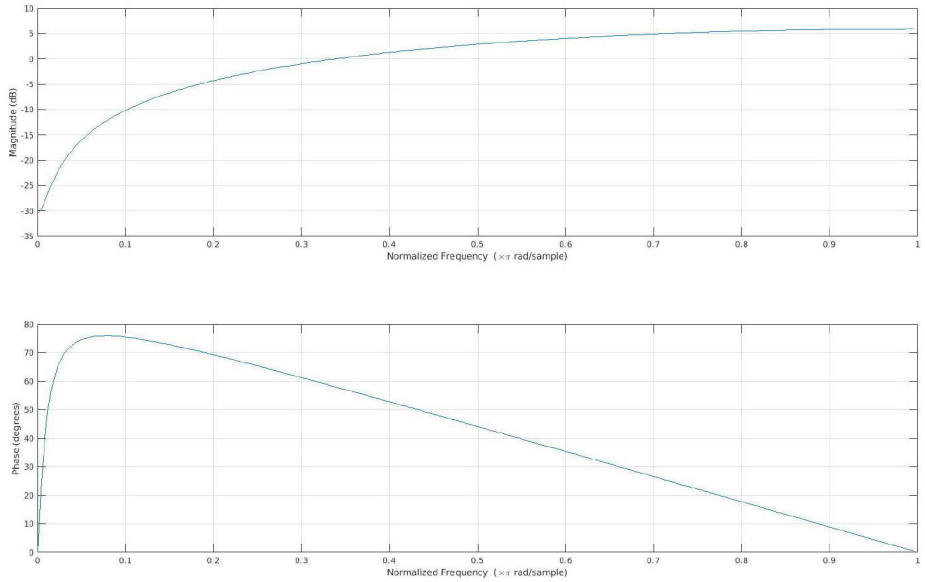

Figure 5: Frequency respond of filter used in this thesis.

The overall extraction process used in this thesis consists of additional Voice Activity Detection (VAD) , noise addition and filtering before Fourier transform process, as well as normalizing Mel Frequency Cepstral Coefficients (MFCC) before

processing. Both VAD and MFCC are explained in their respective subsections.

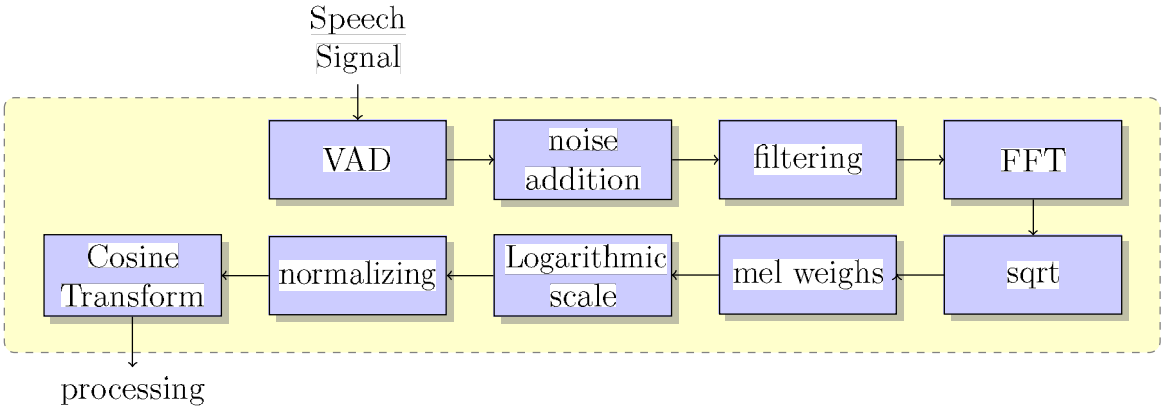

Figure 6: The overall feature extraction process

2.1.1 VAD

Speech processing is difficult without knowing, if speech is actually present in utterance sample. In addition, typical sample consists of series of speech and silence, and processing silence is only wasted use of Voice Activity Detection (VAD) is an algorithm, that detects presence and absence of speech. VAD algorithm used in this thesis is based on Matlab G.729 Voice Activity Detection-function. VAD is calculated on clean audio data. Example of G.729VAd is in Figure 7, which is from Matlab documentation [4].

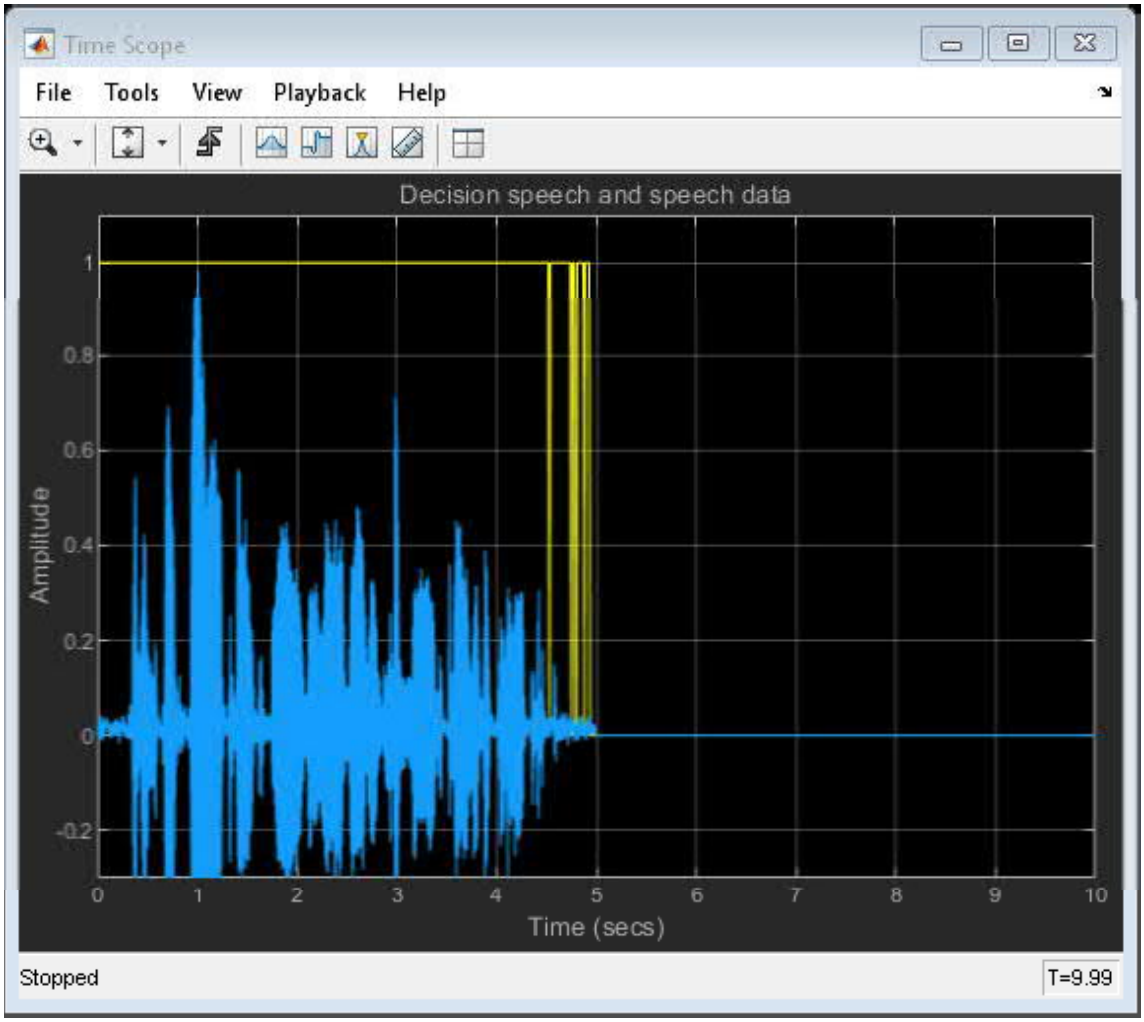

Figure 7: VAD decision corresponding to speech signal. [4]

2.1.2 Mel Frequency Cepstral Coefficients

Mel-Cepstral coefficients are defined in [2] as:

$$mel(f) = 1125 \ln(1 + \frac{f}{700}). \tag{3}$$

Mel-Cepstral coefficients are widely used speech processing as they emphasize lower frequencies similar to human ear. The process is shown in Figure 8 and it is the basis of feature extraction process in this thesis.

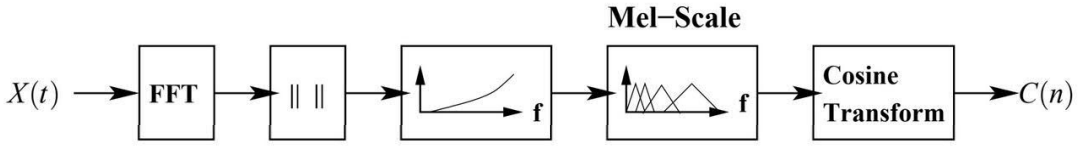

Figure 8: Extraction of Mel-Cepstral coefficient. [3]

The Figure 9 shows the varying difficulty of extracting traits usable in speaker recognition and how the cues can differ depending on acoustic aspects of speech, as well as psychological and sociological aspects of speaker.

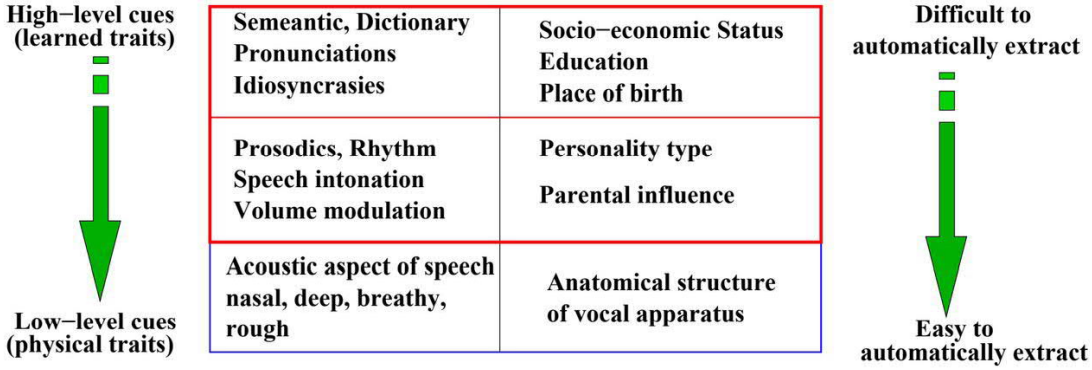

Figure 9: Hierarchy of Perceptual Cues. [3]

2.2 Data

Data consists of VoxCeleb1 and Voxceleb2 speech data from various YouTube celebrities [5] and [6]. In addition, Department of Signal Processing and Acoustics of Aalto University, has acquired some speech data from children [7], which was evaluated with same models as adults speech.

Model training was done with Voxceleb2 development set, which as 5,994 speakers and 1,092,009 utterances. Model evaluation was done with Voxceleb1 and Voxceleb2 test sets, where Voxceleb2 testset has 118 speakers and 36,237 utterances and Voxceleb1 testset has 40 speakers and 4,874 utterances. The atrkids dataset collected by Aalto ASR reserach team conists of 44 speakers and 881 utterances.

2.2.1 Noise addition

The noise is acquired form Musan Corpus [8] as well as from wgn-function from Matlab, which adds white noise to audio data. Noise-audio relationship is changed

during tests at between 0 and 90% at 10% intervals. [9]

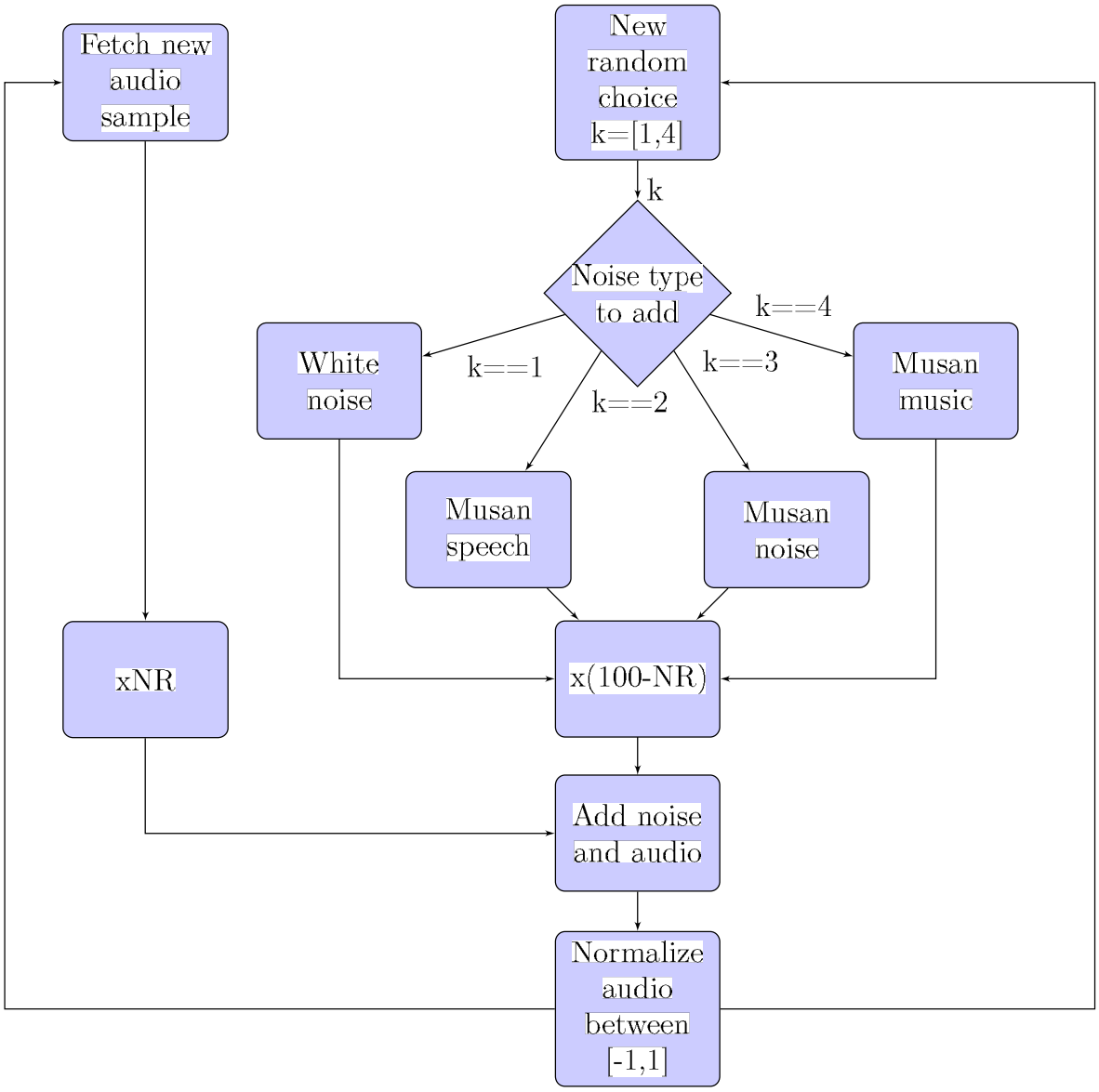

Figure 10: The overall noise addition process. NR=[0,10,20,30,40,50,60,70,80,90] and it defines relationship between audio and noise

3 Speaker modelling

3.1 UBM-GMM

3.1.1 GMM-UBM-model

D. A. Reynolds [1] described Gaussian Mixture Model-Universal Background Model (GMM-UBM) as algorithm based on Gaussian coefficients, which are calculated in two (2) step iteration Expectation-Maximization-algorithm (EM-algorithm). The Gaussian Mixture Density is a weighted sum of M component densities given by equation:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i p_i(\vec{x}). \tag{4}$$

, where $\boldsymbol{x}$ is a D-dimensional feature vector, such as MFCC, $p_i(\boldsymbol{x})$ are Gaussian densities , and w_i are the mixture weights. Each components density is a D-variate Gaussian function of form:

$$p_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2}|\sigma_i|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_i})'\sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu_i})\}. \tag{5}$$

, where $\boldsymbol{\mu_i}$ is the mean vector of $\boldsymbol{x}$ with size D x 1, and σ_i is covariance matrix, with size D x D. Mixture weights sum to 1 $\sum_{i=1}^{M} w_i = 1$. The Gaussian Mixture Model is described as:

$$\lambda = (p_i, \boldsymbol{\mu_i}, \sigma_i), i = 1, ..., M. \tag{6}$$

For speaker identification, each speaker has their own model λ, which is derived from global Universal Background Model (UBM), which represents speaker-independent distribution of GMM parameters. The parameters in λ, can be interpreted as modelling acoustic classes, which in turn arises from phonetic events such vowels, nasals and fricatives. These acoustic classes reflect general speaker-dependent vocal tract configurations, which can be used for speaker identification. Each ith class has then mean $\boldsymbol{\mu_i}$, variance σ_i and weight w_i, where the latter describes the significance of ith component to global UBM model and for each speaker model. [1]
Assuming feature vectors $\boldsymbol{X}$ are independent, then log-likelihood of model λ for feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_1 ... \boldsymbol{x}_T\}$ are:

$$\log p(\boldsymbol{X}|\lambda) = \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\lambda). \tag{7}$$

, where $p(\boldsymbol{x}_t|\lambda)$ is computed with equation 5.

For GMM-UBM model, a single GMM model is computed to represent speaker-independent background model, the Universal Background model (UBM). Different UBM models can be created to represent gender-dependent model, but in this paper gender-independent model is used as the data does not differentiate speakers by gender [5, 6].

3.1.2 EM algorithm

The EM-algorithm refers to Expectation-Maximization algorithm, which is divided into 2 different algorithm phases. The EM-algorithm of UBM begins by initialization of parameters. In [10], the initialization is done by calculating global mean and variance, which are set as initial $\boldsymbol{\mu_i}$ and σ_i respectively. All weights are set to 1. The reason is not mentioned in [1], which was the primary source in MSR Identity Toolbox GMM algorithm, but [11] does mention, that initialization method does not matter, as the EM-algorithm is guaranteed to converge.

After initialization, the UBM continues with the Expectation phase of the algorithm. For mixture i, posterior probability of

$$Pr(i|\boldsymbol{x}_t) = \frac{w_j p_j(\boldsymbol{x}_t)}{\sum_{j=1}^{M} w_j p_j(\boldsymbol{x}_t)}. \tag{8}$$

, which is then used to calculate weight, mean and variance parameters:

$$n_i = \sum_{t=1}^{T} Pr(i|\boldsymbol{x}_t). \tag{9}$$

$$E_i(\boldsymbol{x}) = \frac{1}{n_i} \sum_{t=1}^{T} Pr(i|\boldsymbol{x}_t). \tag{10}$$

$$E_i(\boldsymbol{x}^2) = \frac{1}{n_i} \sum_{t=1}^{T} Pr(i|\boldsymbol{x}_t)\boldsymbol{x}^{\boldsymbol{2}}. \tag{11}$$

In MSR toolkit [10], this are denoted as

$$N_i = n_i. \tag{12}$$

$$F_i = E_i(\boldsymbol{x}) * n_i. \tag{13}$$

$$S_i = E_i(\boldsymbol{x}^2) * n_i. \tag{14}$$

, as seen in alghorithm 1.

Algorithm 1: Expectation

Result: N, F, S, llk_i

C = $\sum_{t=1}^{T}(\frac{\mu^2}{\sigma}) + \sum_{t=1}^{T}(\log(\sigma))$;

D = $\frac{\boldsymbol{x}_t^2}{\sigma} - 2 \cdot \frac{\mu}{\sigma} \cdot \boldsymbol{x}_t + length(\boldsymbol{x}_t) \cdot log(2\pi)$;

$Pr(i|\boldsymbol{x}_t) = \log w - 0.5 * (C + D)$;

$llk_i = \max(Pr(i|\boldsymbol{x}_t)) + \log\{\sum_{t=1}^{T}[\exp(Pr(i|\boldsymbol{x}_t) - \max(Pr(i|\boldsymbol{x}_t)))]\}$;

$Pr(i|\boldsymbol{x}_t) = \exp\{Pr(i|\boldsymbol{x}_t) - llk_i\}$;

N = $\sum_{t=1}^{T} Pr(i|\boldsymbol{x}_t)$;

F = $\boldsymbol{x}_t \cdot Pr(i|\boldsymbol{x}_t)$;

S = $\boldsymbol{x}_t^2 \cdot Pr(i|\boldsymbol{x}_t)$;

During the Maximizing algorithm, the model is updated based on parameters from Expectation phase:

$$\hat{w}_i = [\alpha_i^w n_i/T + (1-\alpha_i^w)w_i]\gamma. \tag{15}$$

$$\hat{\mu}_i = \alpha_i^\mu E_i(\boldsymbol{x}) + (1-\alpha_i^\mu)\boldsymbol{\mu_i}. \tag{16}$$

$$\hat{\sigma}_i = \alpha_i^\sigma E_i(\boldsymbol{x}^2) + (1-\alpha_i^\sigma)(\boldsymbol{\sigma}_i^2 + \boldsymbol{\mu}_i^2) - \boldsymbol{\mu}_i^2. \tag{17}$$

, where adaptation coefficients $[\alpha_i^w, \alpha_i^\mu, \alpha_i^\sigma]$ creates balance between old and new weights, means and variances, respectively. During training these are set to 1, but when speakers are adapted to model, they are calculated based on

$$\alpha_i^\rho = \frac{n_i}{n_i + r^\tau}. \tag{18}$$

, where $\tau = w, \mu, \sigma$ and thus, defines adaptation coefficients for weights, means and variances. Reynolds [1] concluded, that model accuracy was indifferent with r^τ values between [8,20], so MSR toolbox prefers to use $r^\tau = r = 16$.

Algorithm 2: Maximization

Result: μ, σ, w

w = $\frac{N}{\sum(N)}$;

$\mu = \frac{F}{N}$;

$\sigma = \frac{S}{N}$ - μ^2;

vFloor = $\sigma \cdot w \cdot 0.1$;

$\sigma = \max(\sigma, vFloor)$;

EM-algorithm is from [1]

Algorithm 3: EM-algorithm

Result: Gaussian Mixture parameters for global UBM speaker model

Calculate mean and variance of data as $\overrightarrow{\mu}$ and σ.Set w=1;

factor = 10;

while *mix* <= *nmix* **do**

  if *mix* >= *nmix*/2 **then**

    factor = 1;

  end

  for *mix* <= *nmix* **do**

    [N, F, S,]=expectation(speaker_data(:, 1:factor:end), ubm);

    ubm = *maximization*(*N*, *F*, *S*);

    mix = *mix* ∗ 2;

  end

end

3.1.3 Speaker Scoring

Speaker Identification can be done, when speaker model has been constructed. This model is then evaluated against model of test data. In other words, speaker identification is about scoring speaker models and adapted model for test utterance, and then finding highest score. For test utterance X, this can be computed as

$$\Lambda(X) = F(\log\{p(X|\lambda_{hyp})\} - \log\{p(X|\lambda_{ubm})\}). \tag{19}$$

, where λ_{hyp} is parameters from adapted speaker model and λ_{ubm} from UBM model. F is some function such as mean, max or min, but in MSR toolbox, F is mean. Speaker identified for utterance X would then be speaker corresponding with highest score $\Lambda_(X)$.

$$\hat{S} = \underset{S}{\operatorname{argmax}}\, \Lambda_S(X). \tag{20}$$

, where $\hat{S}$ is the estimated speaker and S is group of speakers, each with corresponding score $\Lambda_S(X)$.

Algorithm 4: Evaluation of Speaker Models

Result: *Score* = *llr*, Speaker_guess=argmax(llr)
for *spk=1:length(test_data)* **do**
    [, , , llk_ubm]=expectation(speaker_data, ubm);
    [, , , llk_gmm]=expectation(speaker_data, gmm);
    llr(*tr*) = *mean*(llk_gmm − llk_ubm);
end

3.2 X-vector system

Artificial Neural Networks (ANN) has become the most modern approach to non-linear processing for artificial intelligence (AI). The model of Neural networks comes from neurons in the brain and neural pathways in the human body for example. Each neuron only processes inputs from previous neuron and processes these inputs for next neuron in the link. However, Artificial Neural Networks are not identical no their biological counterparts, as biological processes consists of various chemical processes. But just like their biological counterparts, ANN can learn models from inputs and goals set during training ANN models, and especially with modern Graphical Processing Units (GPU), these models can be trained and applied with easy to various use cases. [12]

Artificial Neural Networks consists of series of layers, with each layer containing different amount of neurons and instructions for each neurons. These connections and their training is a major interest of this thesis. [12]

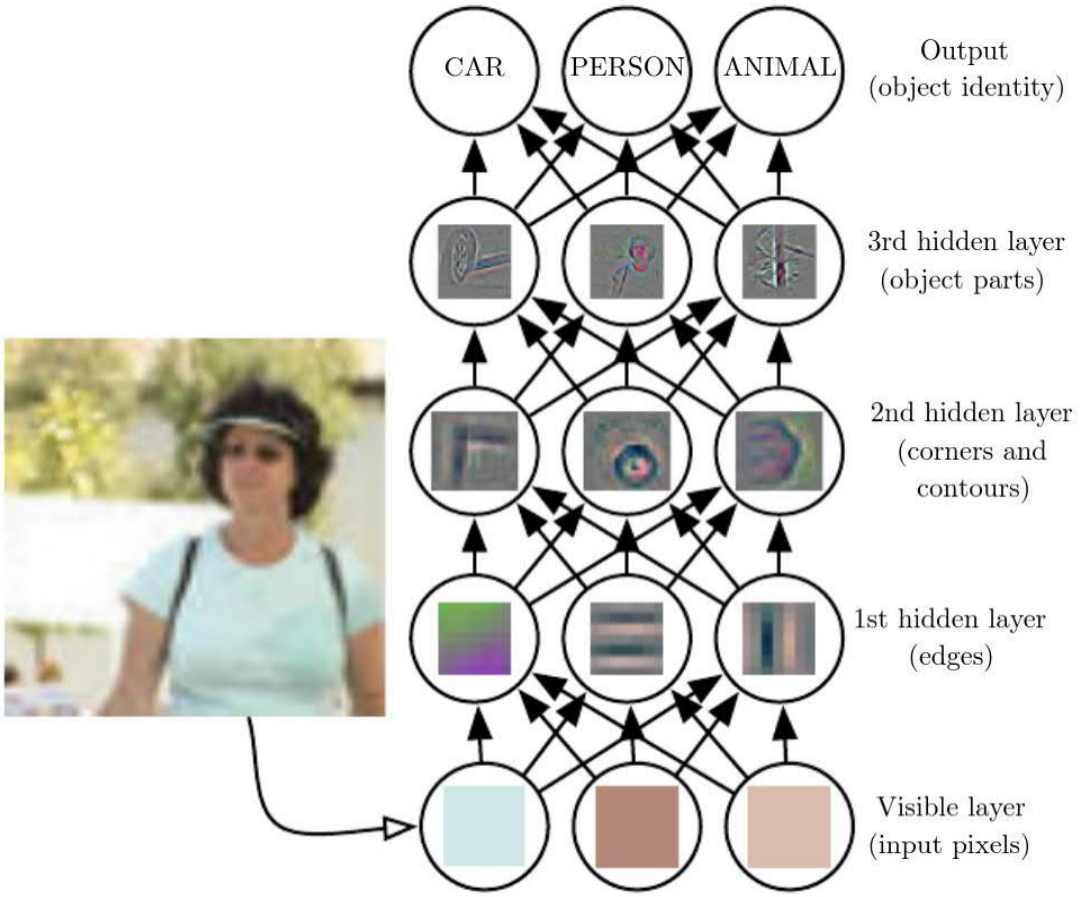

Figure 11: An example of Neural Network classification. . [12]

The X-vector system is based on Snyder et. al. work on Time Delay Neural Networks (TDNN) [13] and Factored Time Delay Neural Networks (TDNN-F) [14]. In both TDNN and TDNN-F architectures, the lowest layers would process speech frames from MFCC while top layers would process more abstract features based on training of layers.

Layer	Layer context	Total context	Input x output
frame1	$[t-2, t+2]$	5	(5*Nceps) x 512
frame2	$\{t-2, t, t+2\}$	9	1536 x 512
frame3	$\{t-3, t, t+3\}$	15	1536 x 512
frame4	$\{t\}$	15	512 x 512
frame5	$\{t\}$	15	512 x 1500
stats pooling	[0 , T)	T	1500T x 3000
segment6	{0}	T	3000 x 512
segment7	{0}	T	512 x 512
softmax	{0}	T	512 x N

Table 1: Embedding of TDNN architecture. *Nceps* are the number of MFCC filterbanks and N is the number of speakers in batch [13]

The total context in table 1 refers to number of speech frames centered around speech frame t. and the layer context refers to frame intervals each layer are processing. This is also present in figure 12. The layer context {0} and total context T both refer to the whole speech sample.

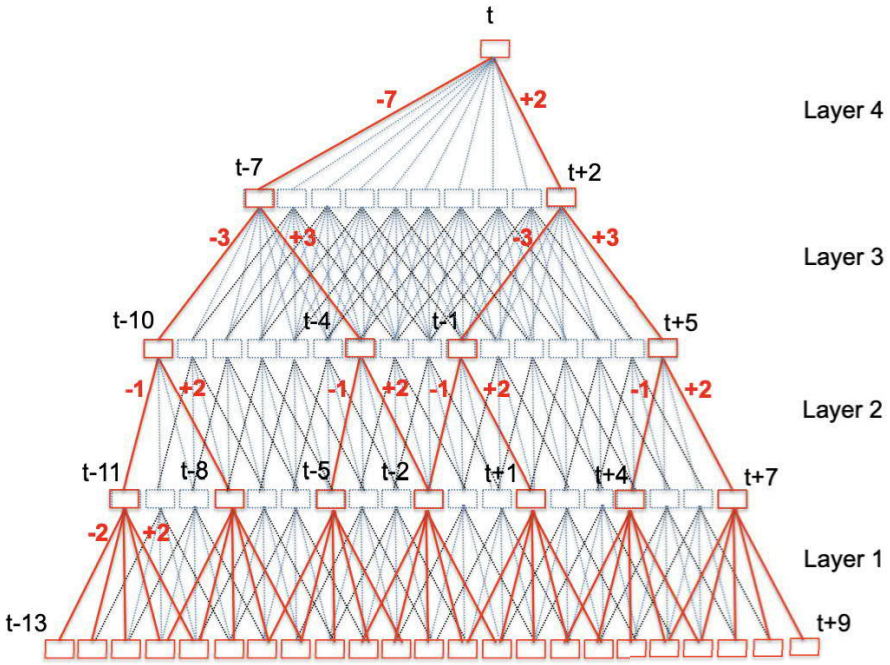

Figure 12: Each layer takes inputs from previous layer except for first layer, which processes speech frames. [15]

The statistics booling layer aggregates all T frame outputs from frame 5, and calculates means and standard deviation, and then concatenate them for segment 6. The output of segment 6 are referred as 'X-vectors'. The segment 7 and softmax layer then maximizes differences of speakers and they are only needed during training. The nonlinearities are rectified linear units.

In Kaldi implementation, the X-vectors are processed with probabilistic linear discriminant analysis (PLDA) for scoring. PLDA can increase accuracy of recognition task [13], but the focus in this thesis was robustness, PLDA system was not constructed in this thesis as well as for time and resource constraints.

3.2.1 Pytorch implementation

The Pytorch implementation is based on work of Chau Luu from University of Edinburgh, The Centre for Speech Technology Research, who published his Time Delay Neural Network (TDNN) in github [15]. Pytorch is well known as often used library for neural networks and it offers simple interface for GPU computing, which both were the reasons for utilizing speaker recognition task with it [16].

With small datasize of 10 speakers, Pytorch implementation worked without issue. However, the implementation reached NaN-values without regularization, when the number of speakers increased to 100. The likely error was due to batch size too huge and network gradients exploding above 1. This is a well known issue [17] and fixing it requires regularization. The chosen regularization technique was chosen as L2, which had the highest test accuracy in identifying hand written numbers [18]. The Pytorch implementation for L2 regularization was from M. Ossenis github [19].

TDNN takes features as inputs as form of $[batch\_size, nspeaker, T, Nceps]$, where Nceps is the number of MFCC fiterbanks, T is length of speech segment after MFCC calculation, nspeaker is number of speakers in batch, and batch_size tells, in how many batches data is processed through the network.

3.2.2 Training algorithm

3.2.3 Scoring

For simple vectors, such as X-vectors with 512 elements, the most convenient scoring method is cosine distance scoring. For each speaker model $\boldsymbol{x}_S$ and utterance model $\boldsymbol{x}_U$

$$score(S,U) = \frac{\boldsymbol{x}_S \cdot \boldsymbol{x}_U}{|\boldsymbol{x}_S| \cdot |\boldsymbol{x}_U|} \tag{21}$$

, where $\cdot$ is dot product, $|\boldsymbol{x}_S|$ and $|\boldsymbol{x}_U|$ are magnitudes of x-vector models for speaker and utterance respectively. The speaker, who acquired highest score, is the identified speaker.

$$\hat{S} = \underset{S}{\operatorname{argmax}}\, \Lambda_S(score(S,U)). \tag{22}$$

4 Results

This section presents results from GMM-UBM and X-vector systems. Each noise level from 0 to 70 percent is presented as its own Error rate-plot. False positive rate refers to rate, which system gives positive result when speaker is imposter. False negative rate is the opposite: correct speaker is identified as imposter. Equal error rate (EER) is the rate in which these two error rates is equal and for optimal system it should be 0. DCF08 and DCF10 are detection cost functions.

4.1 GMM-UBM model results

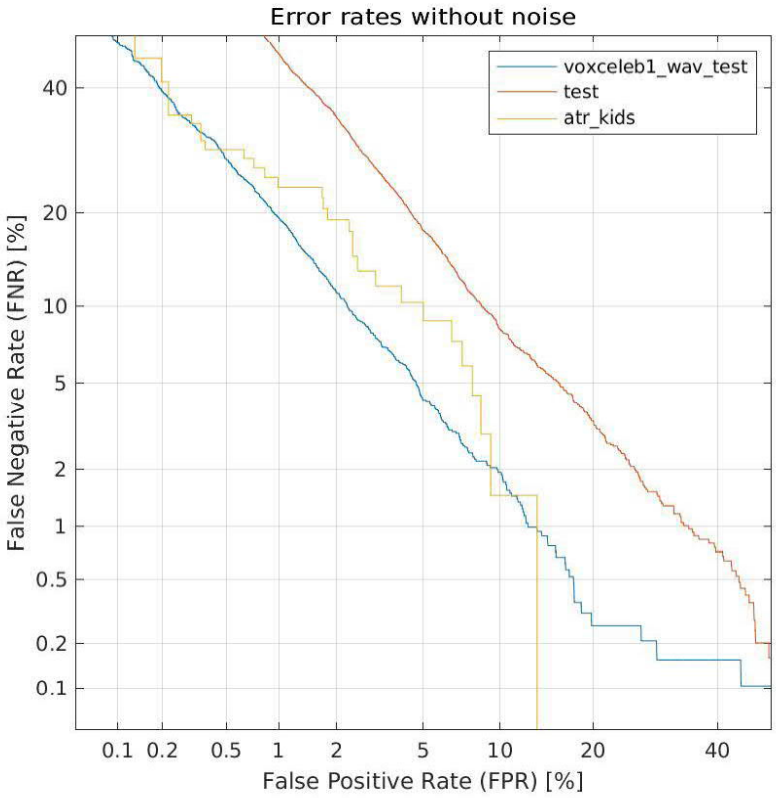

Figure 13: GMM-UBM without any noise.

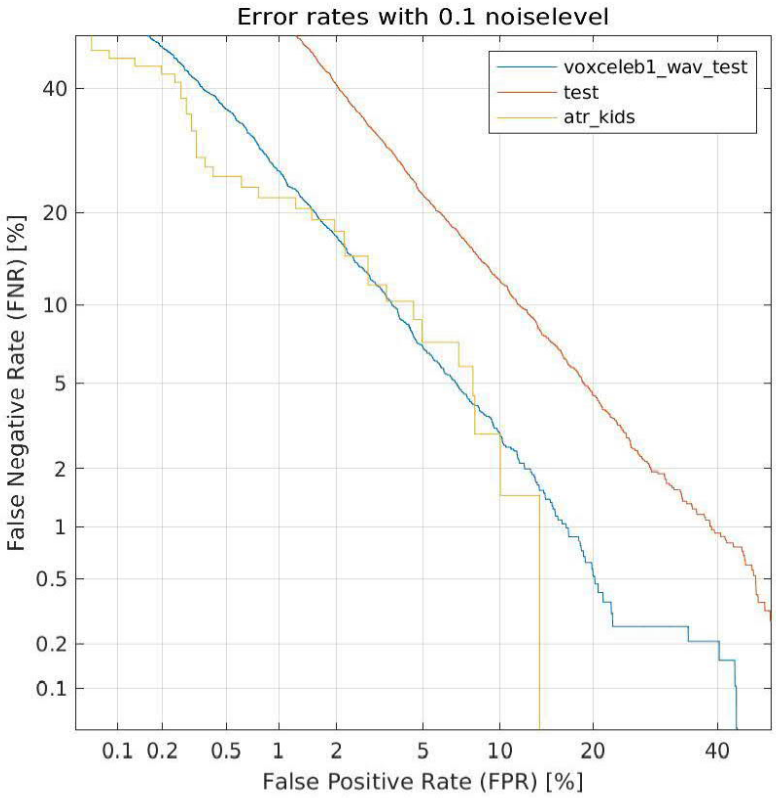

Figure 14: GMM-UBM with noise volume of 10 percent of audio volume.

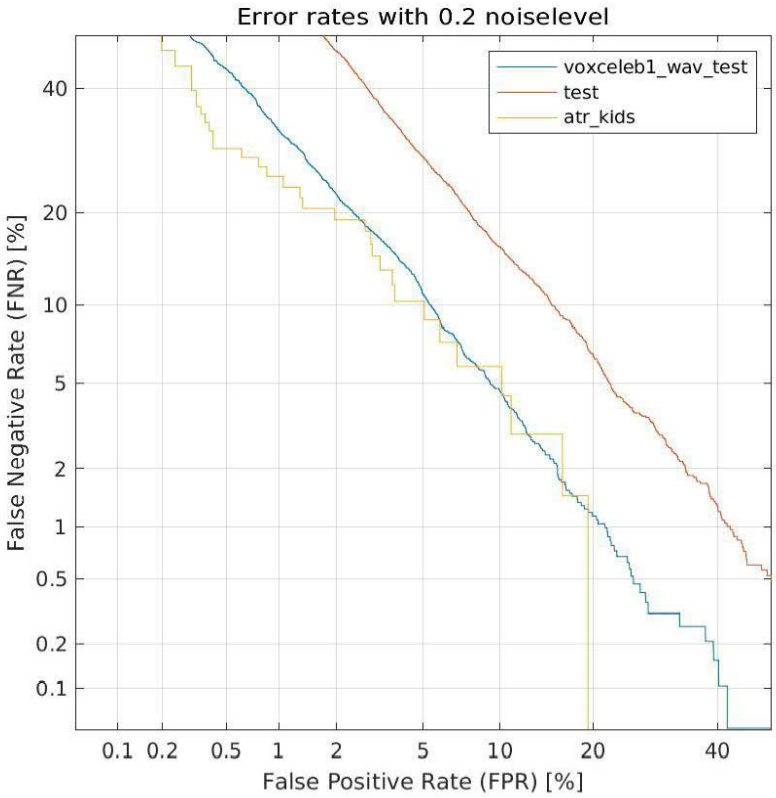

Figure 15: GMM-UBM with noise volume of 20 percent of audio volume.

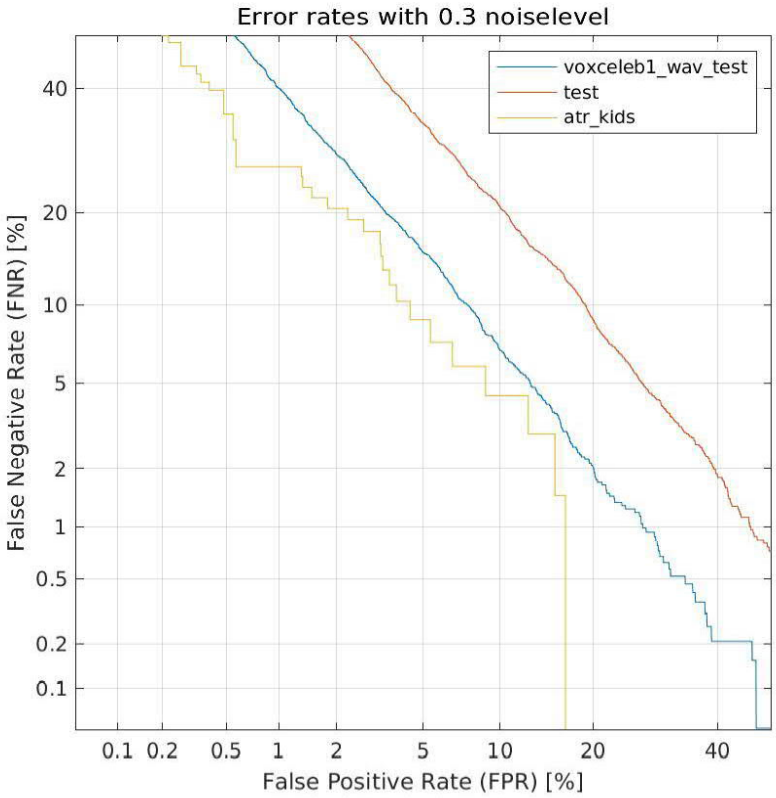

Figure 16: GMM-UBM with noise volume of 30 percent of audio volume.

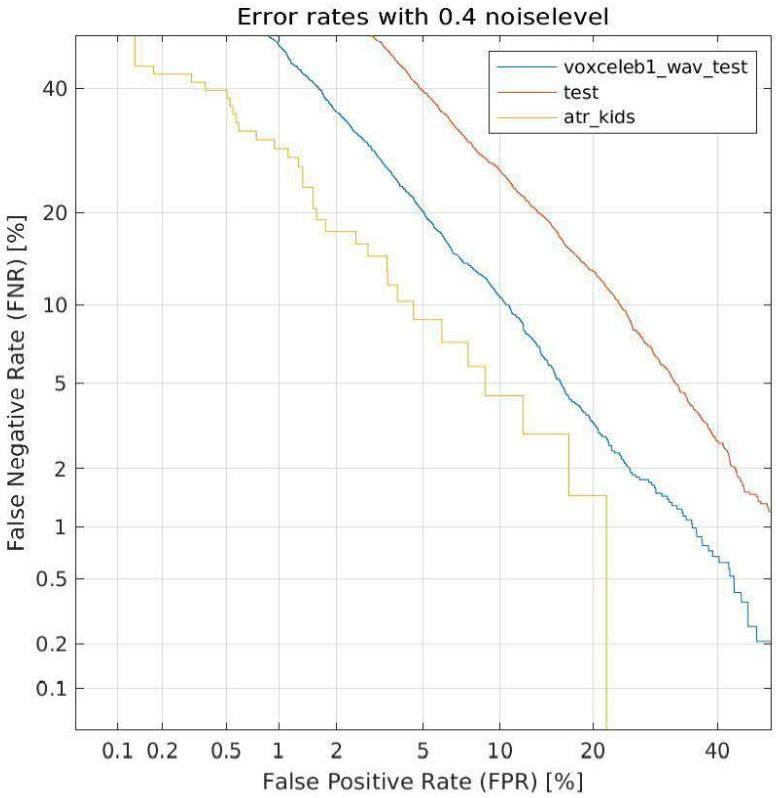

Figure 17: GMM-UBM with noise volume of 40 percent of audio volume.

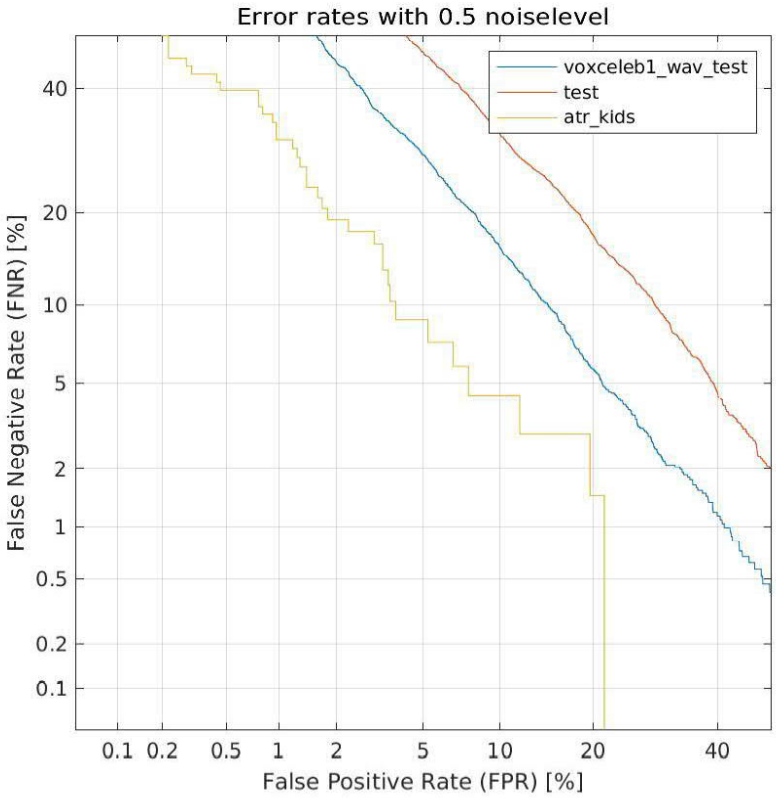

Figure 18: GMM-UBM with noise volume of 50 percent of audio volume.

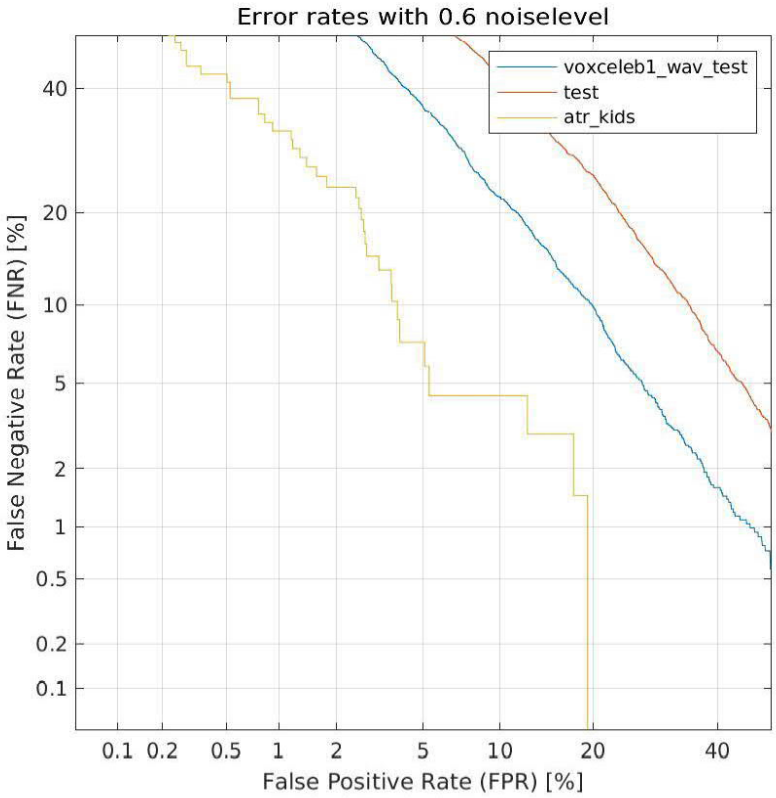

Figure 19: GMM-UBM with noise volume of 60 percent of audio volume.

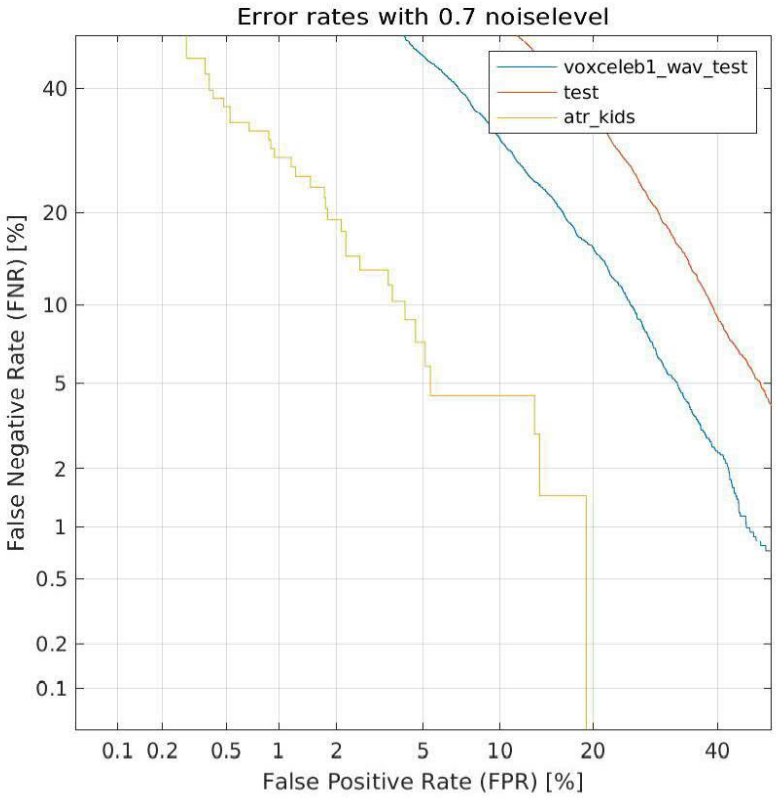

Figure 20: GMM-UBM with noise volume of 70 percent of audio volume.

Noise level	Dataset	Accuracy percentage	EER	dcf08	dcf10
0	Voxceleb 1 test	92.447917	4.778312	2.861579	0.075389
0	Voxceleb 2 test	65.980630	9.294786	5.411219	0.095632
0	ATR kids	98.529412	7.199298	3.310579	0.067647
10	Voxceleb 1 test	88.125000	5.885417	3.473758	0.089014
10	Voxceleb 2 test	59.402744	10.895884	6.013255	0.097545
10	ATR kids	98.529412	6.979807	2.912862	0.085294
20	Voxceleb 1 test	83.020833	7.131410	4.118870	0.093157
20	Voxceleb 2 test	52.138822	12.590799	6.640746	0.099596
20	ATR kids	97.058824	6.870061	3.347454	0.086765
30	Voxceleb 1 test	76.406250	8.489583	4.754647	0.099896
30	Voxceleb 2 test	45.883777	14.481626	7.124915	0.099960
30	ATR kids	94.117647	6.584723	3.212028	0.094118
40	Voxceleb 1 test	71.354167	10.271100	5.533974	0.099948
40	Voxceleb 2 test	42.332526	16.319337	7.555504	0.099960
40	ATR kids	94.117647	7.352941	3.476295	0.094118
50	Voxceleb 1 test	67.291667	12.321047	6.441587	0.099948
50	Voxceleb 2 test	38.579500	18.765133	8.118489	0.100000
50	ATR kids	89.705882	6.628622	3.693591	0.094118
60	Voxceleb 1 test	62.916667	14.829060	7.331571	0.099948
60	Voxceleb 2 test	33.817595	22.072184	8.680481	0.100000
60	ATR kids	88.235294	5.289728	4.037752	0.092515
70	Voxceleb 1 test	58.802083	17.708333	8.155649	0.099948
70	Voxceleb 2 test	58.802083	17.708333	8.155649	0.099948
70	ATR kids	85.294118	5.355575	3.665277	0.089706

4.2 X-vector model results

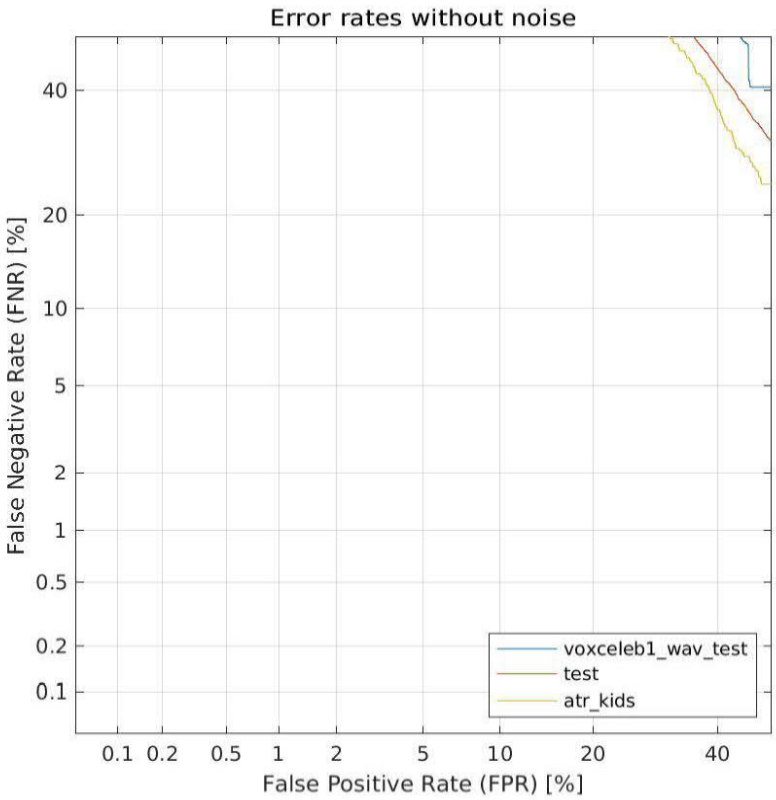

Figure 21: X-vector system results without noise.

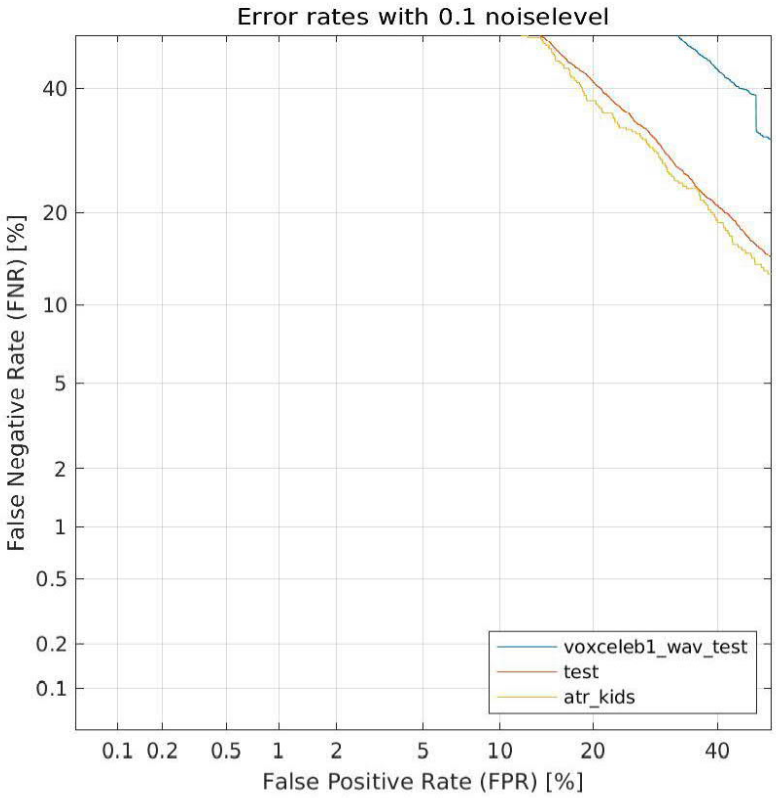

Figure 22: X-vector system results with 10 percent noise level.

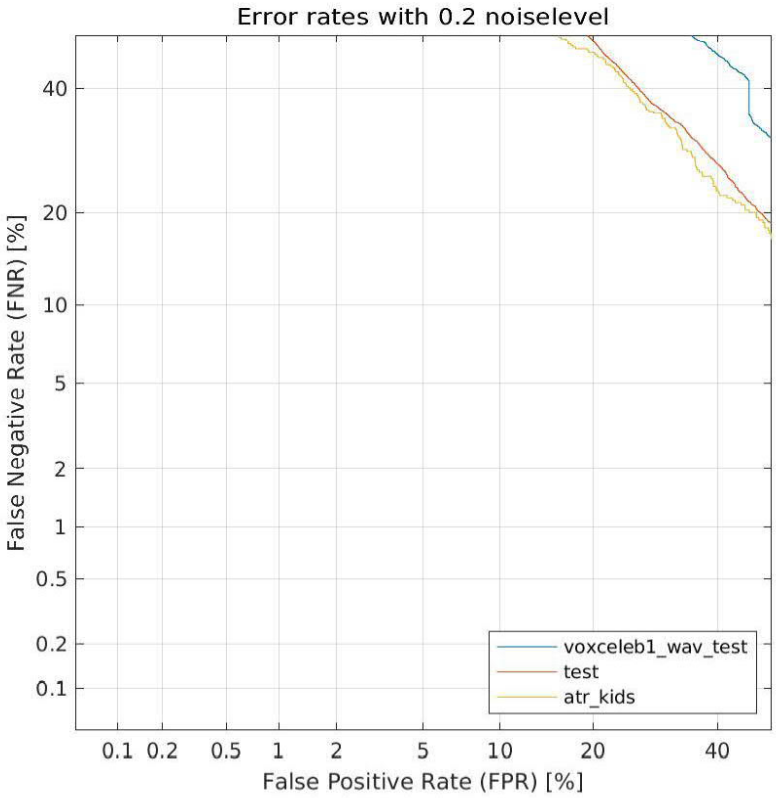

Figure 23: X-vector system results with 20 percent noise level.

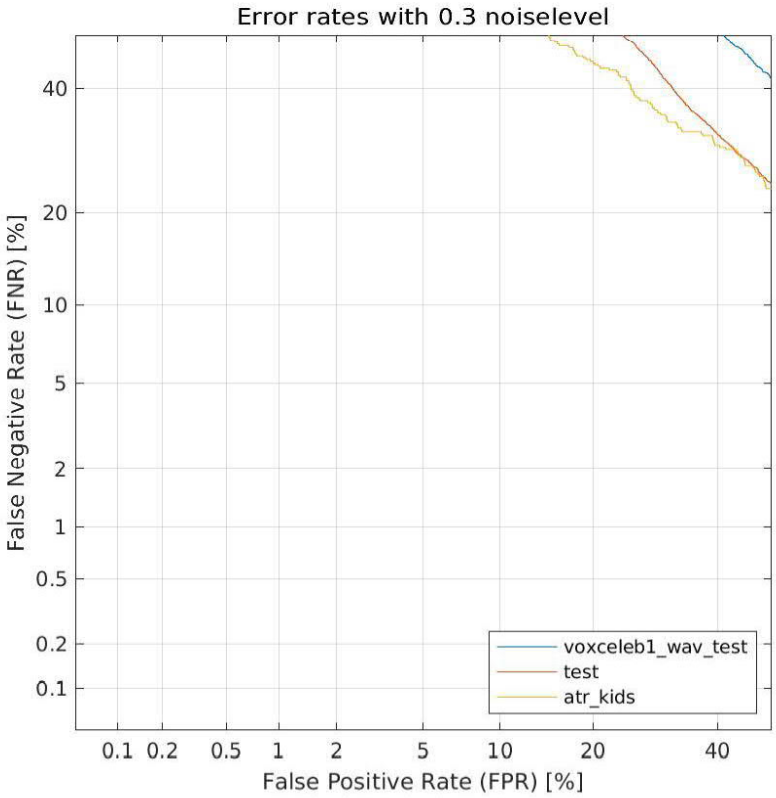

Figure 24: X-vector system results with 30 percent noise level.

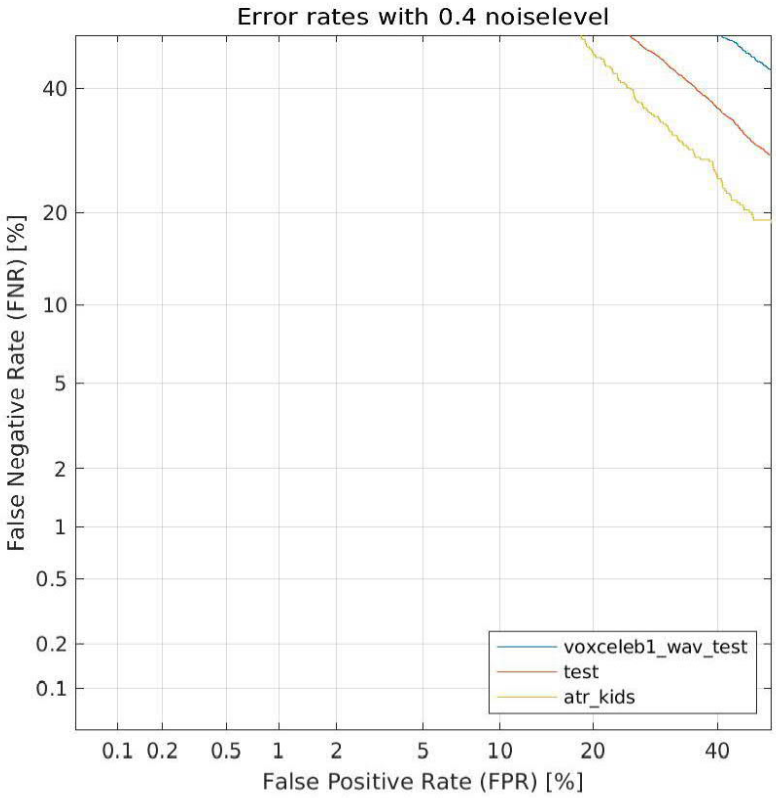

Figure 25: X-vector system results with 40 percent noise level.

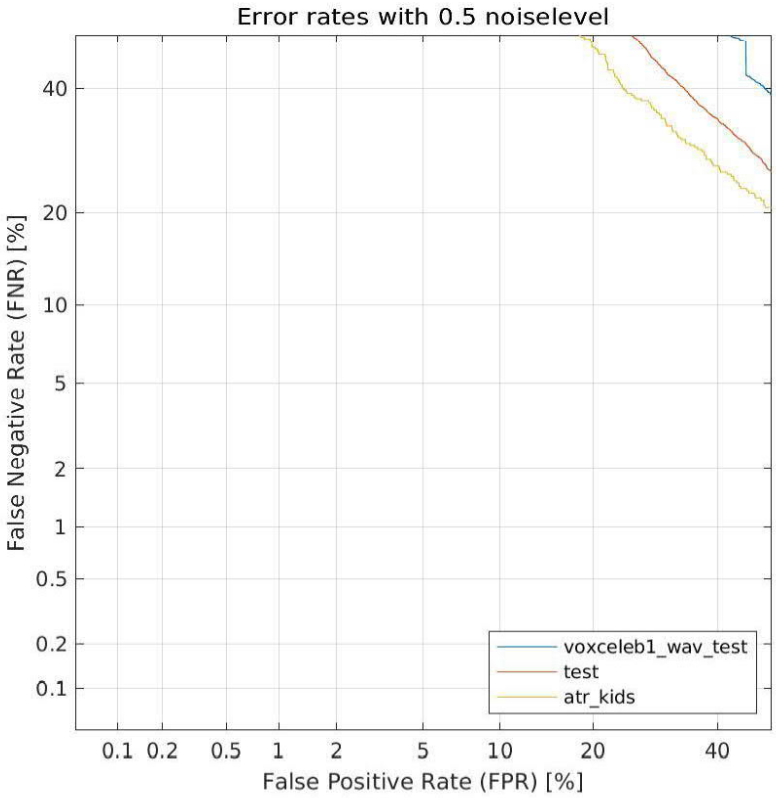

Figure 26: X-vector system results with 50 percent noise level.

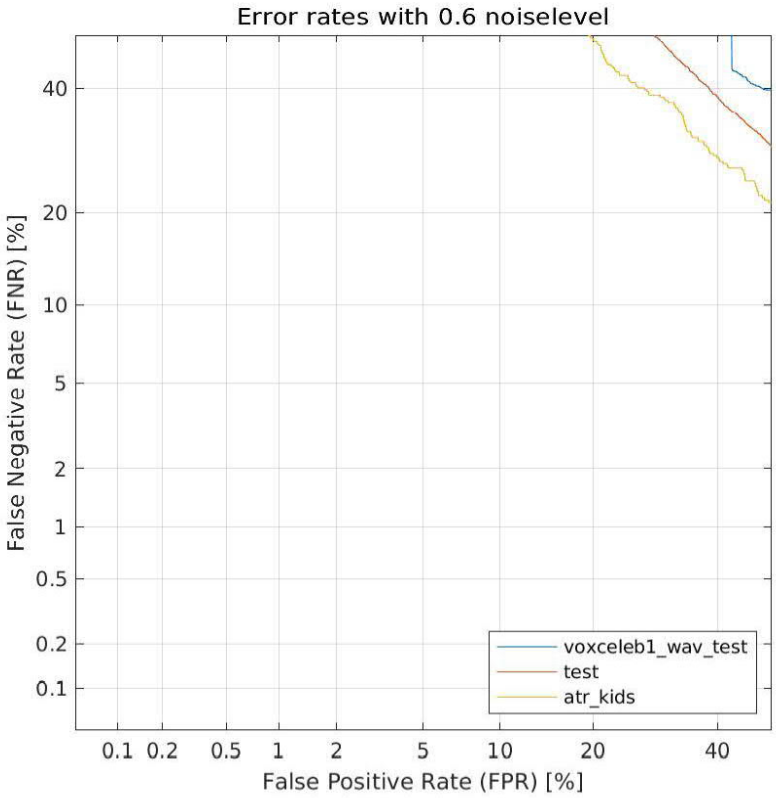

Figure 27: X-vector system results with 60 percent noise level.

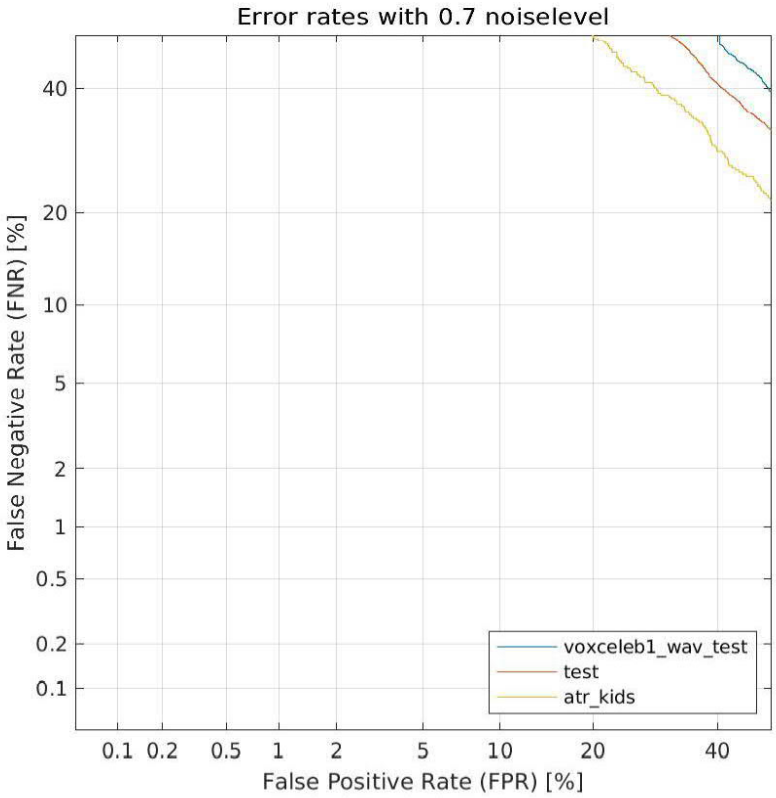

Figure 28: X-vector system results with 70 percent noise level.

Noise level	Dataset	Accuracy percentage	EER	dcf08	dcf10
0	Voxceleb 1 test	10.000000	45.694444	10.000000	0.100000
0	Voxceleb 2 test	0.000000	41.646489	10.000000	0.100000
0	ATR kids	5.263158	39.064723	10.000000	0.100000
10	Voxceleb 1 test	15.000000	41.875000	9.945593	0.099687
10	Voxceleb 2 test	23.728814	29.949366	9.680791	0.099839
10	ATR kids	42.105263	28.974040	8.779605	0.091118
20	Voxceleb 1 test	15.000000	43.437500	9.963542	0.099635
20	Voxceleb 2 test	16.101695	33.455778	9.509747	0.099919
20	ATR kids	31.578947	32.565789	8.762447	0.091118
30	Voxceleb 1 test	20.000000	45.781250	9.968750	0.099687
30	Voxceleb 2 test	15.254237	35.708767	9.627864	0.100000
30	ATR kids	34.210526	32.965861	9.053432	0.092105
40	Voxceleb 1 test	10.000000	46.140491	9.957252	0.099687
40	Voxceleb 2 test	11.864407	38.216303	9.649966	0.100000
40	ATR kids	31.578947	31.943457	9.078947	0.090789
50	Voxceleb 1 test	15.000000	45.188301	9.968750	0.099687
50	Voxceleb 2 test	8.474576	36.888378	9.547992	0.100000
50	ATR kids	31.578947	32.236842	9.162784	0.092434
60	Voxceleb 1 test	12.500000	43.177083	9.966186	0.099687
60	Voxceleb 2 test	6.779661	39.086526	9.581455	0.100000
60	ATR kids	26.315789	33.997155	9.197546	0.092105
70	Voxceleb 1 test	17.500000	44.659455	9.964864	0.099687
70	Voxceleb 2 test	11.864407	40.395480	9.575557	0.100000
70	ATR kids	26.315789	35.197368	9.204036	0.092434

5 Conclusion

5.1 UBM-GMM vs X-vector network

According to previous studies [13], x-vector system should produce higher results. The likely reason is that training failed especially with speech samples without noise. Training neural networks with pytorch proved to be difficult due to memory requirements and handling.

GMM-UBM system was handled with just Central Processing Unit (CPU), but neural networks has much more parameters and that is the reason why Graphical Processing Unit (GPU) had to be used. GPU has much more cores than CPU, but it is restricted by RAM-memory accessible by GPU. Since RAM-memory with GPU is restricted, the data had to be processed in small chunks of 100 speakers, while GMM-UBM-system could be trained with whole Voxceleb2 development dataset.

Better results could have gained if the whole dataset were processed by the whole network. This would require additional research about pytorch and GPU processing as well as memory handling.

5.2 Child vs Adult speech

Child speech samples produced much higher recognition accuracy than adult speech samples in nearly all noise levels. The likely reasons are:

1. Child speech is more distinguish than adult speech
2. Child dataset was more clear originally than Voxceleb
3. Child dataset had less speakers than Voxceleb, so if speaker sets were similar size, the results could be closer

Out of these three reasons, the more likely explanation is the second. Voxceleb is collected from Youtube and it has much more original noise than child speech samples collected by Aalto University ASR research group. Voxceleb has speakers, who speak at the same time, and various backround events, which are absent in child speech samples.

References

[1] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1):19 – 41, 2000.

[2] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[3] Q. Jin. *Robust Speaker Recognition*. Carnegie Mellon University, 2007.

[4] Mathworks. G.729 voice activity detection. https://se.mathworks.com/help/dsp/examples/g-729-voice-activity-detection.html.

[5] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[6] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[7] Hemant Kathania, Sudarsana Kadiri, Paavo Alku, and Mikko Kurimo. Study of formant modification for children asr. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, United States, 2020. IEEE.

[8] David Snyder, Guoguo Chen, and Daniel Povey. Musan: A music, speech, and noise corpus, 2015.

[9] K A. Al-Karawi, A H. Al-Noori, FF Li, and T Ritchings. Automatic speaker recognition system in adverse conditions – implication of noise and reverberation on system performance. *International Journal of Information and Electronics Engineering*, 5(6):423–427, November 2015.

[10] Seyed Omid Sadjadi, Malcolm Slaney, and Larry Heck. Msr identity toolbox v1.0: A matlab toolbox for speaker recognition research. Technical Report MSR-TR-2013-133, Microsoft Research, September 2013.

[11] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, Jan 1995.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, April 2018.

[14] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. pages 3743–3747, 09 2018.

[15] C. Luu. Time delay neural network (tdnn) implementation in pytorch using unfold method. https://github.com/cvqluu/TDNN.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. https://arxiv.org/pdf/1912.01703.pdf, 2019.

[17] Jason Brownlee. A gentle introduction to exploding gradients in neural networks. https://machinelearningmastery.com/exploding-gradients-in-neural-networks/, Dec 2017. [Online; Accessed 13 Dec. 2019].

[18] M. Osseni. Different types of regularization on neuronal network with pytorch - implementation of regularization tools in pytorch. https://towardsdatascience.com/different-types-of-regularization-on-neuronal-network-with-pytorch-a9d6faf47 2019.

[19] M. Osseni. Applied sparse regularization (l1), weight decay regularization (l2), elasticnet, grouplasso and groupsparselasso to neuronal network. https://github.com/dizam92/pyTorchReg.