PENETRATION TESTS OF SPEAKER VERIFICATION SYSTEM

PENETRAČNÍ TESTY SYSTÉMU PRO VERIFIKACI ŘEČNÍKA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **FILIP WOJNAR**
AUTOR PRÁCE

SUPERVISOR **Ing. OLDŘICH PLCHOT, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

Department of Computer Graphics and Multimedia (DCGM) Academic year 2020/2021

Bachelor's Thesis Specification

23388

Student:	**Wojnar Filip**
Programme:	Information Technology
Title:	**Penetration Tests of Speaker Verification System**
Category:	Speech and Natural Language Processing

Assignment:

1. Get acquainted with the system for speaker recognition available at BUT Speech@FIT.
2. Study the problem of speech synthesis with a focus on imitation of a particular speaker.
3. Get acquainted with metrics used in speaker verification.
4. Using a suitable speech synthesis software and public dataset, design a set of penetration tests against a speaker recognition system.
5. Analyze the results of the test cases and suggest steps leading to better security of the target speaker recognition system.

Recommended literature:

- According to supervisor's reccommendation.

Requirements for the first semester:

- Items 1-3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Plchot Oldřich, Ing., Ph.D.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2020
Submission deadline:	July 30, 2021
Approval date:	July 27, 2021

Abstract

The aim of the thesis is to realize penetration tests of automatic speaker verification system with use of text-to-speech model. The thesis is focused on inner functioning of those systems and spoofing attacks against them. The thesis is also focused on speech synthesis. Later chapters are focused on realization of realized penetration tests and discussion about results they brought us.

Abstrakt

Cílem práce je provést penetrační testy na systému pro automatickou verifikace řečníka za použití syntézy řeči. Práce se zabývá fungování systému pro automatickou verifikaci řečníka a spoofing útoky na systémy, zabývající se touto problematikou. Práce se také podrobnější zabývá fungováním syntézy řeči. Pozdější kapitoly se zabývají realizací penetračních testů a výsledky, které nám tyto testy přinesly.

Keywords

speech recognition, speech verification, speech synthesis, text-to-speech, ASV spoofing, x-vectors

Klíčová slova

rozpoznávání řeči, ověřování řeči, syntéza řeči, text-to-speech, ASV spoofing, x-vektory

Reference

WOJNAR, Filip. *Penetration Tests of Speaker Verification System.* Brno, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Oldřich Plchot, Ph.D.

Penetration Tests of Speaker Verification System

Declaration

I hereby declare that this Bachelor's thesis was created as an original work by the author under the supervision of Ing. Oldřich Plchot Ph.D. I have listed all the literary sources, publications and other sources, which were used during the creation of this thesis.

. . . .
Filip Wojnar
July 28, 2021

Acknowledgements

I would like to give special thank you to Mr. Oldřich Plchot for always replying to my questions and putting hours into online consultations in these difficult times. I am also thankful for recommendations on dissertations which helped me understand ASV systems better, especially at the start of the thesis.

Contents

Chapter 1

Introduction

Informational technologies are one of the fastest growing technological branches in the world. Boom in computational power which has been happening last few decades allowed computers to solve complex problems like recognition of road lines, predicting stock market price behavior or recognizing human voice and further more verifying if it belongs to certain person. In this thesis we will focus on the case mentioned last. Biometrics are body measurements and calculations related to human characteristics. In computer science these characteristics are often used as authentication tools. Idea behind biometrics authentication is that every person's measurements are different and unique, therefore this technique should be strong against spoofing attacks. In this thesis we will take a look at this idea and investigate whether automatic speaker verification systems can be spoofed with text-to-speech and voice cloning software.

In the second chapter 2 we will describe basics of speech processing which we will need in order to understand automatic speech recognition and verification systems in chapter 3. Fourth chapter will focus on spoofing attacks against these systems 4. In the fifth chapter we will take closer look at text-to-speech systems and their potential for spoofing attacks 5. Sixth chapter will be focused on TTS model for our spoofing attacks 6. Then in seventh chapter we will be describe ASV system we will be trying to penetrate 7. And finally eight chapter will be discussing our experiments and their results 8.

1.1 Claims of the thesis

The focus of the thesis is to create our own dataset for penetration testing of automatic speaker verification system. Test some of the penetration techniques and of course analyze gathered data and compare them with existing data.

Chapter 2

Speech processing

In this chapter we will explain what speech processing is and describe what are it's most important aspects.

Speech processing is the study of speech signals and the processing methods of signals. In order to work with speech first thing we have to do is convert speech into form of data computers are able to use. This is achieved by A/D converters these usually have sampling rate from 8kHz to 20kHz and resolution of 12 to 16 bits [17].

2.1 Speech features

In order to work with speech data on a human level we have to identify useful speech signal information and characteristics. We can consider several levels of speech information which can be extracted from the speech signal [12].

- Acoustic (Spectral): spectral representation of speech produced by human vocal tract
- Prosodic: features like pitch, pauses, intonation, syllable lengths and rhythm
- Phonetic: analysis of sequences of phonemes specific to the speaker
- Idiolect: analysis of sequences of words specific to the speaker
- Linguistic (Dialogic and semantic): analysis of linguistic patterns characteristic to the speaker's conversation style

Figure comparing individual speech information 2.1

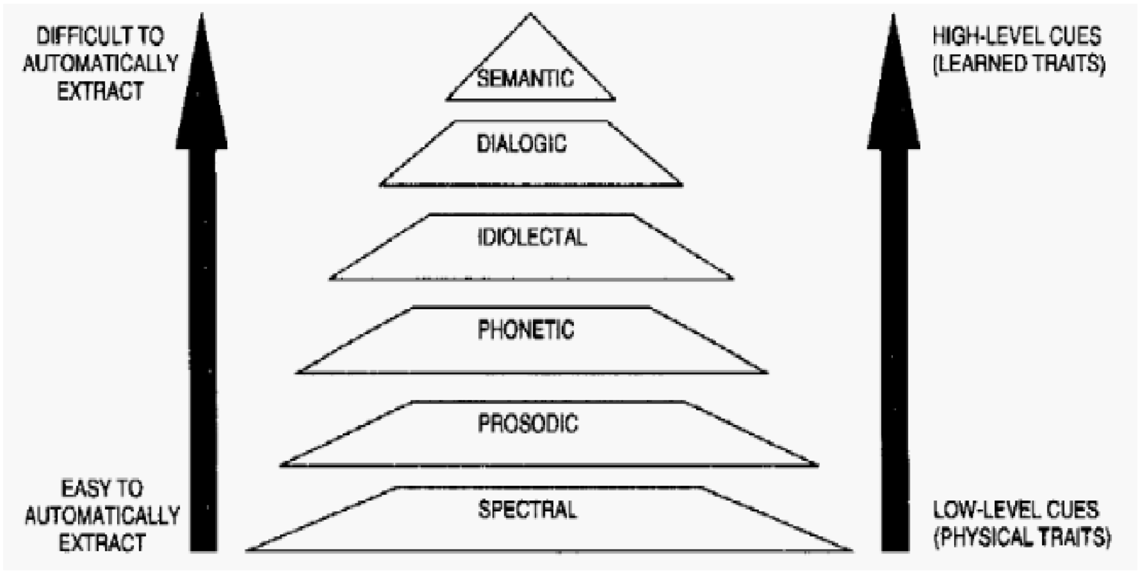

DIFFICULT TO AUTOMATICALLY EXTRACT
EASY TO AUTOMATICALLY EXTRACT
SEMANTIC
DIALOGIC
IDIOLECTAL
PHONETIC
PROSODIC
SPECTRAL
HIGH-LEVEL CUES (LEARNED TRAITS)
LOW-LEVEL CUES (PHYSICAL TRAITS)

Figure 2.1: Diagram ilustrating difficulty of extraction of individual features and information level they contain [4]

2.1.1 Phonetic

We will focus on phonetic a bit more as it will help us understand TTS systems in greater detail in chapter 5. As was said earlier main focus of phonetic is analysis of sequences of phonemes. Phonemes are basic units of human speech, individual phonemes are referred to as phones, in English individual phones are for example th,ch or h. When these are formed into phone sequences, they create words and give the speech meaning. Neural Networks have been emerged as an attractive acoustic modeling approach in automatic speech recognition and are being used in many aspects of speech recognition such as phoneme classification, isolated word recognition and speaker adaptation. It allows discriminative training in a natural and efficient manner. [8]. In this thesis we will be interested in this approach for it's speaker adaptation properties.

2.2 Mel spectograms

Mel-spectrogram is a low-level acoustic representation of speech waveform, which is commonly used for local conditioning of a WaveNet vocoder in current state-of-the-art text-to-speech (TTS) architectures [9]. Mel spectograms are also often used as form of speech signal compression. Another concept of mel spectograms is that we humans perceive frequencies logarithmically which can be problematic as vanilla waveforms represent speech signal in a linear way. In other words mel spectograms help us create perceptually relevant frequency representation of speech signals by converting frequencies into mel-scale. Example of synthesized mel-spectogram saying „this is a big red apple“ below, figure extracted from [6]

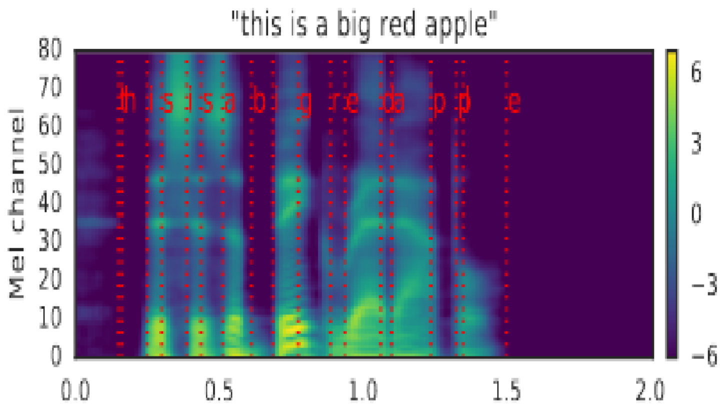


Figure 2.2: Example of mel-spectogram of synthesized sentence extracted from [6]

2.3 Mel-frequency cepstrum coeficients

For SRE, MFCCs have become a standard method of parametrization and they usually serve as a baseline for any newly proposed feature extraction method [12]. MFCCs are used for representation of speech features. Advantage of MFCCs is it can describe large structures of the spectrum and it ignores fine spectral structures which allows us to focus solely on information about phonemes ignoring noise and details. Disadvantage is that use of MFCCs require extensive engineering knowledge. Also even though they work well with speech recognition and speaker recognition, MFCCs aren't suitable for speech synthesis as we don't have suitable formula to go from MFCCs back to waveform.

2.3.1 Speech features extraction

Speech features extraction is process in which we convert speech signal into MFCCs. First thing which happens in this process is framing of speech samples. We divide whole signal into frames of fixed length. The length of frames is typically 25ms. These frames can be obtained by application of rectangular windowing function. However when we apply this function there are created high frequency distortions on the edges of frames due to sharp cuts. For this reason, a windowing function which weakens the signal on its borders have to be used. It is typically the bell-shaped Hamming-window function [12]. This method extends the length of the frames to 20-25ms which means constant shift corresponding to intended rate(usually 10ms) have to be applied. When the windowing is done every frame is used for calculation of low-dimensional representation in the form of MFCC feature vector.

First we apply Short Term DFT (Discrete Fourier Transform) on the frame and then we use power of 2 on its absolute value. Next step is application of Mel-filterbank to smoothen the spectrum. A vector of band energies is then computed as a weighted sum of squared values of the amplitude spectrum. Then a logarithm of the energies is used to match the human perception of sound loudness. Lastly we apply Discrete Cosine Transformation (DCT) to reduce feature vector's dimensionality. At the end we have low-dimensional MFCC feature vector. Visualization of MFCC extraction steps is depicted in Fig 2.3

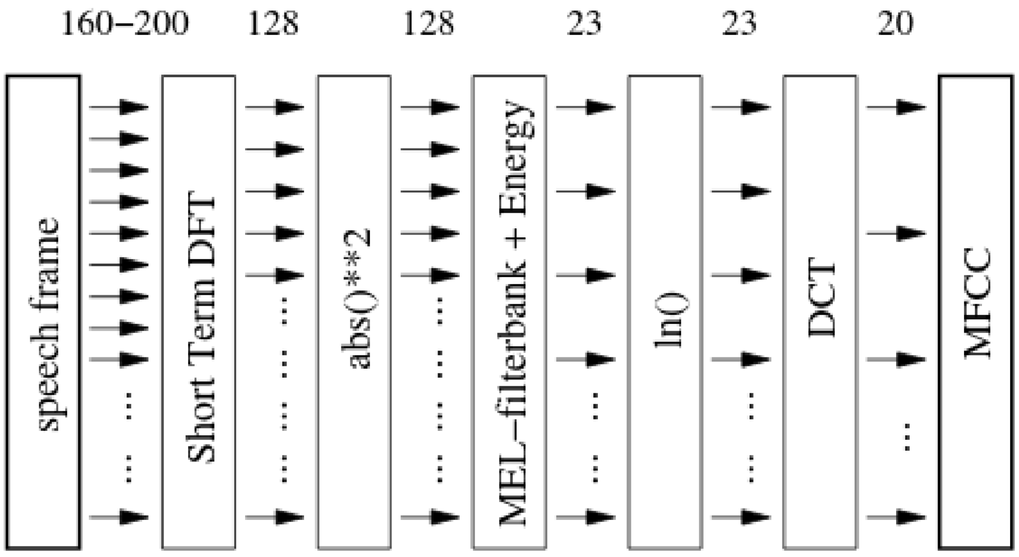


Figure 2.3: MFCC extraction steps — the numbers above the blocks show dimensionalities for frame lengths of 20 and 25 ms at sampling frequency fs= 8000 Hz. Figure was found in Ing. Oldřich Plchot's PhD thesis [12]

2.4 Voice activity detection

Voice activity detection (VAD) is important tool in speech processing related problems. Most often VAD is being used in pre-processing phase of speech recognition. It's purpose is to ignore noise and gaps in a signal that doesn't contain speech data. There exist well known noise suppression algorithms such as Wiener filtering (WF) or spectral subtraction, that are commonly being used for speech recognition. For these algorithms VAD is indispensible in order to maintain high level of performance [13]. There are also various approaches on how to detect speech ranging from simple energy thresholding, Gaussian Mixture Model (GMM) classifier or Neural Networks (NN) trained to discriminate between speech and the rest of the audio signal [12].

2.5 Mean-opinion-score (MOS)

Mean opinion score (MOS) is score given to a quality of speech assigned by people. As the people are deciding the MOS score of a recording it isn't given that recording with higher MOS score will be more successful in fooling automatic systems. MOS score is playing a role only in human perception of sound. The score is ranging from 0 to 5, but even the quality of human speech isn't scoring 5. Human speech is scored at around 4.6 to 4.8, when we take a look at the state-of-art TTS systems which are mostly using WaveNet vocoder and are based on Tacotron 2 neural network architecture, there are generated recordings which are getting score of 4.526 [14] which is very close to human generated speech quality and with human ear almost indistinguishable. We will take a closer look at these technologies in chapter 5.

Chapter 3

Automatic speech recognition and speaker verification systems

In this chapter we will focus on automatic speech recognition systems which from now on I will refer to as ASR systems and automatic speaker verification system or so called ASV systems. Difference between ASR and ASV is that ASR systems are designed to recognize speech and information it includes. Whereas ASV systems are designed to recognize if the speech belongs to so called target speaker. We can divide ASV systems into 2 categories. First category are text-dependent ASV systems which in order to verify speaker take into consideration not only speakers speech characteristics but also an semantic information in speech (text). This category is potentially safer as there is an addition of some kind of secret password to it, not just biometric component. Second category are text-independent systems these systems have no predetermined set of words and depend solely on speaker characteristics in order to verify speakers identity. This thesis will be focused towards text-independent systems. It is also important to say that everything we have included in chapter 2 is commonly used in ASV and ASR systems, but also in TTS systems which we will talk about in chapter 5.

3.1 EER

Equal-error-rate (EER) is commonly used metric for evaluation performance of ASV system. It usually uses probabilistic linear discriminant (PLDA) to compare pairs of embeddings. PLDA produces a comparison score, which is the log of the ratios of the probability that embeddings were produced by the same speaker, versus the probability that the speakers were different [17]. If the produced score is greater then threshold s, we assume that the embeddings belong to the same speaker. Otherwise we assume they belong to different speakers. There are two types of *trials* used for performance evaluation, first type is *target trial* in this type the system enrolled speaker and test speaker are same. The second type is *non target*. In this type the system enrolled speaker and test speaker are different. We treat ASV systems as binary problem as there are only 2 options for errors. More below.

1. **Misses:** also called false negatives, occur when the score for target trial is lesser then threshold s. In other words they occur when ASV system classifies target speaker as non-target speaker.

2. **False alarms:** also called false positives, occur when the score for a non-target trial is greater then or equal to threshold s. In other words they occur when ASV system classifies non-target speech as a target speaker.

Finally Equal error rate (EER) is obtained by tweaking thresholds until a value of **EER** is found, this value is where **false alarm probability** and **miss probability** are equal.

3.2 DET curves

The Detection Error Tradeoff (DET) is typical way to create visual representation of performance of a system. It is widely used among ASV community to visualize and compare performances of various systems on wider ranges of threshold points. DET curve allows as to approach both types of errors uniformly. It uses a scale for both axes, which spreads out the plot and better distinguishes different well designed systems. Is also usually produces plots that are close to linear [4]. The DET plot is derived from the Receiver Operating Characteristics (ROC) curve which is used in many other fields. ROC plots the detection probability as a function of false alarm probability. The DET plot is then created when both axes of ROC plot are transformed with a non-linear transformation. After the transformation, the x and y ranges are moved from (0,1) interval to $(-\infty, +\infty)$ interval [12].

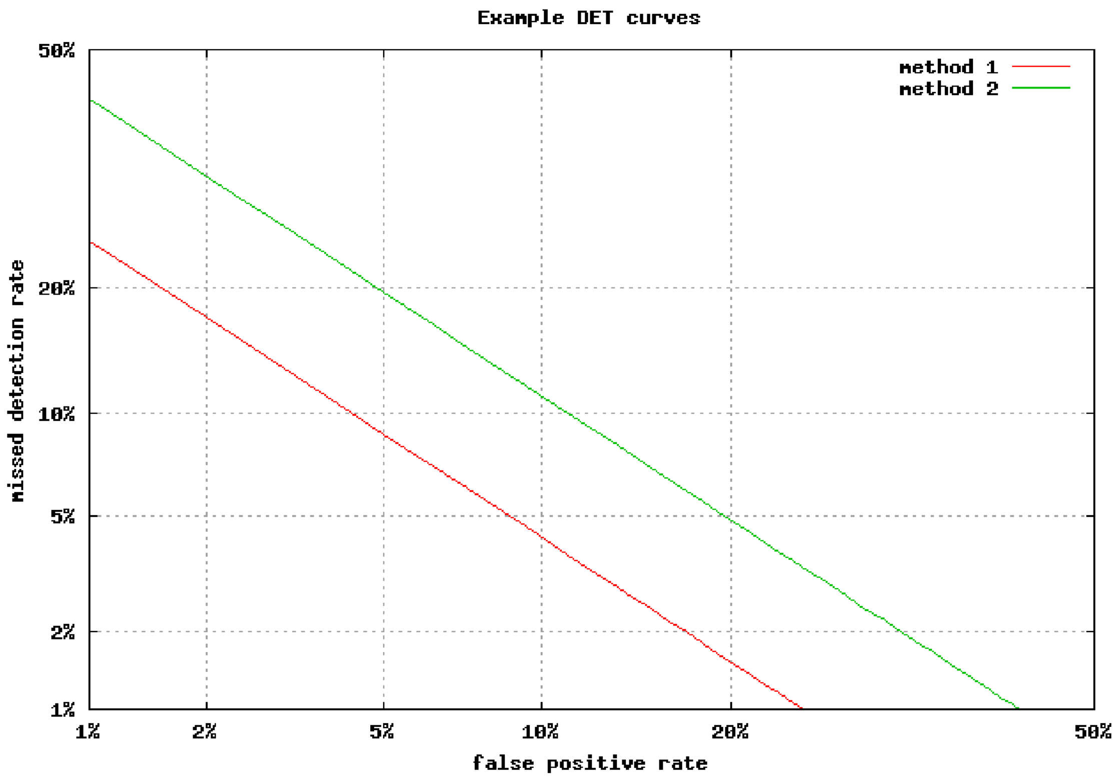


Figure 3.1: Example of two theoretical DET curves, the red curve would be considered more efficient then the green one. Figure extracted from https://en.wikipedia.org/wiki/Detection_error_tradeoff.

3.3 Score normalization

The goal of score normalization is to improve system performance by reducing variability in scores of individual trials. Normalization leads to better calibration and improved setting of thresholds. It was shown that for some methods, for example joint factor analysis (JFA), for example ZT-norm lead improvement in results up to 50% [1]. Normalization is typically linear operation which consists of global shift and scale. In general, normalization with shift μ and scale σ is performed as:

$$S_{\text{norm}} = \frac{s-\mu}{\sigma} \tag{3.1}$$

Some of used normalization formulas:

- Z-norm
- T-norm
- ZT-norm
- S-norm

Zero normalization:
The Z-norm is generally considered to be a means for compensating with respect to inter-speaker variability in the scores. It compensates for the biases and scales in the enrollment model scores evaluated against the test data [12].

$$p\left(\frac{S_{imp}-\mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}|\mathcal{M}\right) \approx \mathcal{N}\left(\frac{S_{imp}-\mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}};0,1\right) \tag{3.2}$$

Test normalization:
T-norm compensates for intersession variability between the tested utterance and a set of speaker models.

$$p\left(\frac{S_{imp}-\mu_{\mathcal{X}}}{\sigma_{\mathcal{X}}}|\mathcal{X}\right) \approx \mathcal{N}\left(\frac{S_{imp}-\mu_{\mathcal{X}}}{\sigma_{\mathcal{X}}};0,1\right) \tag{3.3}$$

ZT-norm: Is the combination of Z-norm and T-norm it is equivalent to performing Z and T norm sequentialy.

S-norm: In our case we will be using S-norm. It can achieve similar effect to ZT-norm, while performing less computations. Usually, a single held out cohort of speakers serves as a Z-norm cohortas well as segments for training T-norm models. Z-norm and T-norm are then independently applied in order to obtain two sets of normalized scores. Final scores are obtained by averaging of the corresponding scores from these sets [12].

3.4 Detection Cost Function

Detection Cost Function (DCF) has been introduced by NIST (National Institute of Standards and Technology) as an evaluation metric for the speaker verification systems, which focuses on a particular operation point of interest. It is defined as weighted sum of the false alarm probability and the miss detection probability:

$$\text{DCF} \quad = \quad C_{\text{miss}} p(\text{miss}|\mathcal{T}, \tau) p(H_{\text{s}}) + C_{\text{fa}} p(\text{fa}|\mathcal{T}, \tau) p(H_{\text{d}}) \tag{3.4}$$

where

$$p(H_d) \quad = \quad 1 - p(H_s), \tag{3.5}$$

where C_{miss} and C_{fa} are the relative costs of the detection errors, τ is the threshold, and $p(H_s$ and $p(H_p$ are the prior probabilities for the trial being the same or different speaker [12].

For evaluation of the system in our thesis we will be using minimal detection cost function which is modified version of function above. The minimal DCF computes a minimum possible DCF (min-DCF) by setting the optimal threshold for the given test set:

$$\min \text{DCF} \quad = \quad \min_{\tau} \quad [C_{\text{miss}} p(\text{miss}|\mathcal{T}, \tau) p(H_s) + C_{fa} p(fa|\mathcal{T}, \tau) p(H_d)] \tag{3.6}$$

3.5 ASV Spoof 2019

If we want to measure how reliable our ASV system is, we need a method which will specify how our system will be evaluated. On which characteristics of a system we will put emphasis, and on which datasets we will be using this method. One of the famous evaluation methods in context of ASV penetration testing is ASV spoof. There are many versions of ASV spoof challenge, there is 1 coming almost every year since 2015. In this thesis we will focus on ASV spoof 2019. Another widely used evaluation method for ASV systems is NIST (National Institute of Standards and Technology), which also has many versions correlating to a year it was introduced. NIST is by many considered ASV evaluation standard and has been around since 1996.

ASVspoof 2019 challenge is build upon ASV spoof 2015 where up to date text-to-speech (TTS) and voice conversion (VC) systems were introduced. Challenge from 2019 is first which focuses on countermeasures for all three major attack types, specifically these TTS, VC and replay spoofing attacks. Today TTS system are able to synthesize speech which is indistinguishable from bona fide speech with untrained human ear. ASVspoof 2019 introduces for the first time metric which is oriented towards ASV systems in the form of the tandem decision cost function (t-DCF) [2]. The task of the challenge was to create automatic system with the ability to distinguish between bona fide and spoofed speech.

Chapter 4

ASV Spoofing attacks

Spoofing refers to the presentation of a falsified or manipulated sample to the sensor of a biometric system in order to provoke a high score and thus illegitimate acceptance [5]

Spoofing in a context of ASV systems is when attacker tries to trick ASV system into believing that spoofed speech is authentic and comes from a known source. In other words the attacker is trying to create false acceptance in the system. The spoofed speech can be obtained in many different ways and the way of obtaining it is one of the things that define which attack we are dealing with. The most typical spoofing attack techniques are Voice conversion (VC), speech synthesis (SS), impersonation and replay attacks.

4.1 Replay attacks

Replay attack is the simplest ASV spoofing techniques. All it requires is pre-recorded speech signal of the target's voice. Attacker then simply plays the recording to an ASV system. This technique doesn't require any speech processing, ASV systems expertise or sophisticated equipment and thus they arguably present a great risk. Despite the lack of attention in the literature, experiments show that low-effort replay attacks provoke higher levels of false acceptance than comparatively higher-effort spoofing attacks such as voice conversion and speech synthesis. [5]

4.2 Impersonation

Impersonation is the oldest type of attack where attacker mimics person's voice with his voice tract. Attacker has to know person's speech patterns like prosody, phonetics, idiolect, pronunciation etc. Advantage of this method is it doesn't require any digital speech recording however it also can't really scale into attacking large amounts of systems. Disadvantage is it requires professional impersonator. It isn't considered a real thread for ASV systems. [7]

4.3 Speech synthesis

Speech sythesis which is also known as text-to-speech (TTS) is used to create speech signal from text. This speech is created by some kind of deep neural network (DNN). DNN type is most often recurrent neural network (RNN). Sometimes it can be convolutional neural network (CNN) as we can convert speech to spectogram and approach speech signal as an image. This method requires long DNN training on huge datasets but once the network is trained it can adapt to new speakers fairly quickly and with solid audio quality. We will take a deeper look at this method in the next chapter 5. We will also do our own experiments with this technique on the specific ASV system in chapter 5.

4.4 Voice conversion attack

In voice conversion attacks the natural voice of an attacker is converted to a speech utterance which sounds as it was spoken by the target. The mainstream voice conversion method is based on Gaussian mixture models (GMMs). ASV systems state-of-art technology against these attacks is joint factor analysis (JFA) recognizer. But even though JFA method is the most resilient one. Experiments on a subset of NIST 2006 SRE corpus indicate that even this method experiences more that 5 times increase in the false acceptance rate from 3.24 % to 17.33 % [3]. VC systems can also be useful to extend existing training datasets for DNN training purposes.

Diagram of voice conversion system 4.1

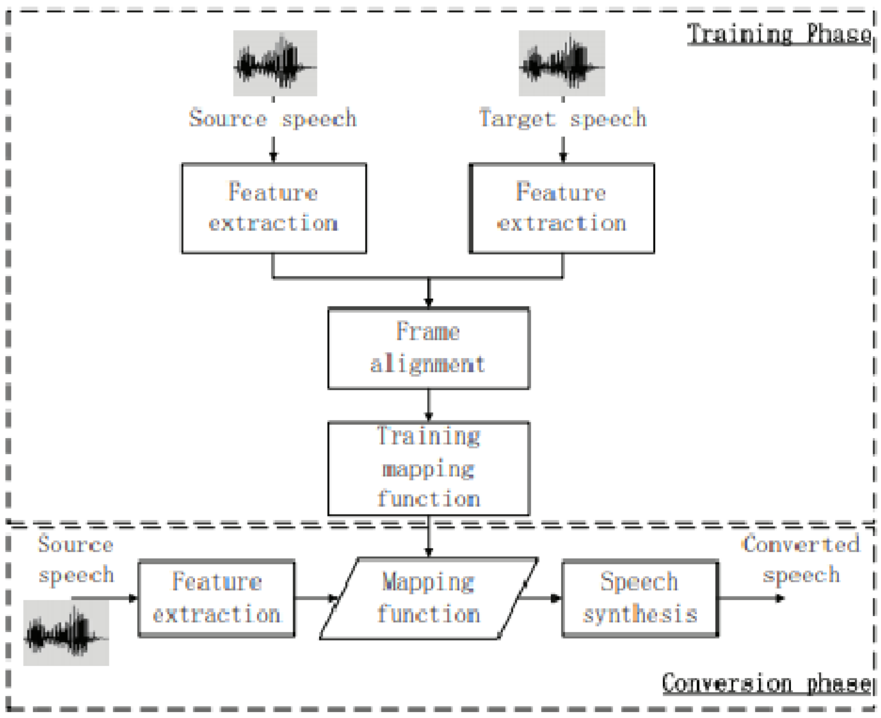


Figure 4.1: Diagram of voice conversion system [3]

Chapter 5

Text-to-speech systems

Text-to-speech are systems which are using speech synthesis technology to create human voice from text. These systems are used in platforms like Google's Alexa or Microsoft's Cortana to help users control their devices with voice or to help people who have some kind of speaking disability. However these technologies can also be used maliciously against ASV systems or on actual people in form of phishing attacks. In this chapter we will describe most commonly used techniques in speech synthesis and take a look at state-of-the-art text-to-speech (TTS) system architectures. We will be using this technology for ASV spoofing, therefore we will be focused towards data efficient solutions which require just a few minutes of target speaker recordings as there won't usually be hours of speech recordings of person in „the wild", unless it's a public figure.

5.1 TTS methods

Section explaining different TTS methods which could be divided into two categories. First category is parametric speech synthesis, in this category belongs Formant synthesis and Articulatory synthesis, with this approach voice can be modified with changes in parameters. It is old approach of doing speech synthesis. Second category is concatenative synthesis when we want to change voice in this case we have to come with a new voice database.

5.1.1 Formant synthesis

Formant synthesis is the oldest method for speech synthesis, it is based on source-filter model which means it is generating periodic and non-periodic signals which are then fed to resonator circuit or a filter. Idea of this method is based on emulation of human vocal tract. This method can theoretically create any sound but the sounds generated are unfortunately very unnatural sounding. The synthesizer sometimes contains filters that model lip radiation and anti-resonator to improve the quality of the sound [20].

5.1.2 Concatenative synthesis

Concatenative synthesis is sometimes also called „cut and paste" based on how it works. In concatenative synthesis there are selected sequences or individual phones from pre-recorded database and then they are concatenated one after another to create desired utterance of speech. Limitations of concatenative synthesis are mostly in the lack of human personality, affective tones etc [20]. In other words the outputs of this method are very dependent on the database it is coming from. Also if we want to select whole words and syllabels for this method, it creates problems with memory capacity, that's why this method usually sticks with phonemes as they are shorter. Another reason for use of phonemes is better flexibility.

5.1.3 Articulatory synthesis

Articulatory synthesis method is newest and by far most complicated with it's computational requirements and model structure. Articulatory synthesis as name suggests try to mimic human speech production as detailed as possible. The system is so complicated it's not being widely used yet. This system contains physical model of human vocal tract but also a vocal cords. Even though this system's successes are comparable with previous 2 methods, some argue [16] that potential of this method is far greater when enough research on it is done.

5.2 Generative TTS models

Generative TTS models are able to generate new data from the same distribution as given training data. Examples of few generative models are PixelCNN, PixelRNN, Tacotron/Tacotron2, Variational Autocoders (VAE) and Generative Adversarial Networks (GAN). This is the type of TTS system we will use using in this thesis to prepare spoofing data for our penetration testing task. More on this in next chapter 6.

Chapter 6

TTS model used in the thesis

This chapter will be focused towards TTS model used for ASV spoof task of the thesis. In a spoofing task our TTS system must be able to adapt to different speakers fairly quickly and with reasonably small amount of data, hence one of the best options will be speaker speech synthesis model.

6.1 Real-Time-Voice-Cloning by CorentinJ

As mentioned above my main parameters for choosing a model were amount of referral audio it requires to synthesis voice of a victim and also quality of it's prosody. That's why I decided to use Real-Time-Voice-Cloning implemented by CorentinJ which utilizes encoder and decoder architecture from [19]. The whole system is composed out of three independently trained neural networks. The first NN is recurrent speaker encoder which computes fixed dimensional vector from a reference speaker speech signal. The second is sequence to sequence synthesizer based on [15]. This NN is fed sequence of graphemes and phonemes on input and predicts from them Mel-spectograms. The last component is autoregressive WaveNet which converts Mel-spectograms into time-domain waveforms. This multispeaker speech synthesis model can be seen in 6.1.

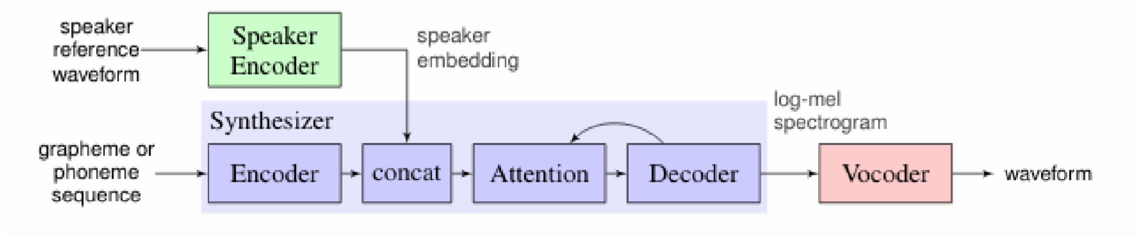


Figure 6.1: Multispeaker speech synthesis model utilized by Real-Time-Voice-Cloning by CorentinJ [19].

6.2 Speaker encoder

The speaker encoder is a component of TTS system responsible for conditioning of synthesis network for similarities in generated signal with desired target speaker speech characteristics. For achievement of good generalization it is necessary to select characteristics of different speakers and have the ability to identify these characteristics using short adaptation signal, independent of it's phonetic content. This can be achieved with speaker-discriminative model trained on a text-independent speaker verification task.

The network is consisting of a stack of 3 LSTM layers of 768 cells to which there are passed on the input 40-channel log-mel spectograms. Each followed by a projection to 256 dimensions. The final embedding is created by L_2-normalizing the output of the top layer at the final frame. During inference, the utterance is broken into 800ms windows, overlapping by 50%. The network runs independently on each window. The final utterance embedding is created by averaging and normalizing the outputs [6].

6.2.1 GE2E encoder

Model selected by us is utilizing GE2E encoder which uses new loss function called generalized end-to-end. This function is 60% more time effective and decreases speaker verification EER by more than 10% in comparison with older TE2T (tuple-based end-to-end) loss function [18].

6.3 WaveNet vocoder

Vocoder is used for transformation of synthesized mel-spectogram which was produced by synthesizer, back to time-domain waveform. Since 2016 the holder of state-of-the-art vocoder has been WaveNet. However WaveNet has autoregressive nature and therefore works quite slow. The architecture is composed of 30 dilated convolution layers. All of the relevant details needed for high quality synthesis of a variety of voices is contained in the mel spectogram predicted by the synthesizer. This allows a multispeaker vocoder to be constructed simply by training on data from big amount of speakers [6].

6.4 Tacotron synthesizer

Modern TTS systems are complex and are usually consisting of two parts. The first part is front-end which extracts various linguistic and acoustic features. Second is back-end in a form of signal-processing based vocoder. Tacotron is end-to-end generative TTS model based on sequence to sequence with attention paradigm. It works with transcripted audio data and can produce audio with MOS quality of 3.82 [19]. Tacotron tries to predict raw spectograms straight from the text hence it is considered to be end-to-end model. Later on these frames are converted into waveform.

Target spectrogram features are computed from 50ms windows computed with a 12.5ms step, passed through an 80-channel mel-scale filterbank followed by log dynamic range compression.

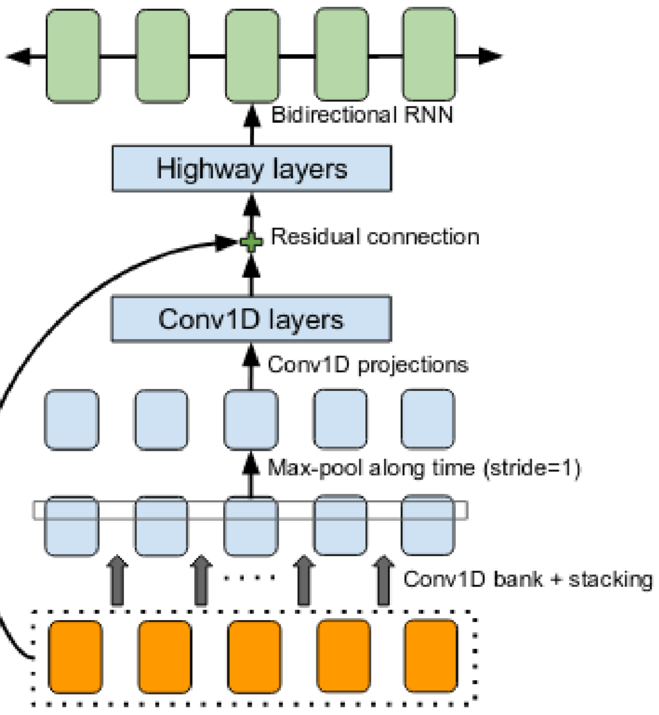


Figure 6.2: The CBHG (1-D convolution bank + highway network + bidirectional GRU) module [19].

The backbone of Tacotron is seq2seq model with attention. Speaking about our model on high-level, it takes characters as input and produces spectogram frames which are later converted into waveforms. The main building block of the Tacotron synthesizer is dubbed CBHG illustrated in figure above 6.2.

6.4.1 CBHG module

CBHG module consists of 3 main parts, bank of 1-D convolutional filters, highwat networks and a bidirectional gated recurrent unit (GRU) recurrent neural net (RNN). CBHG is a module for extracting representations from sequences. The input sequence is first convolved with K sets of 1-D convolutional filters. The convolution outputs are stacked together and further max pooled along time to increase local invariances. The processed sequence is then further passed to a few fixed width 1-D convolutions, whose outputs are added with the original input sequence via residual connections. On all convolutional layers there is used batch normalization. The convolution outputs are then fed to multi-layer highway network where extraction of high-level features happens. Finally there is bidirectional GRU RNN stacked on top for sequential features extraction [6].

Chapter 7

ASV system in our experiments

In this chapter we will take a look on how ASV systems based on x-vector extraction works, we will take a closer look at their architecture and lastly I will describe x-vector extractor used in our ASV penetration testing.

7.1 X-vectors

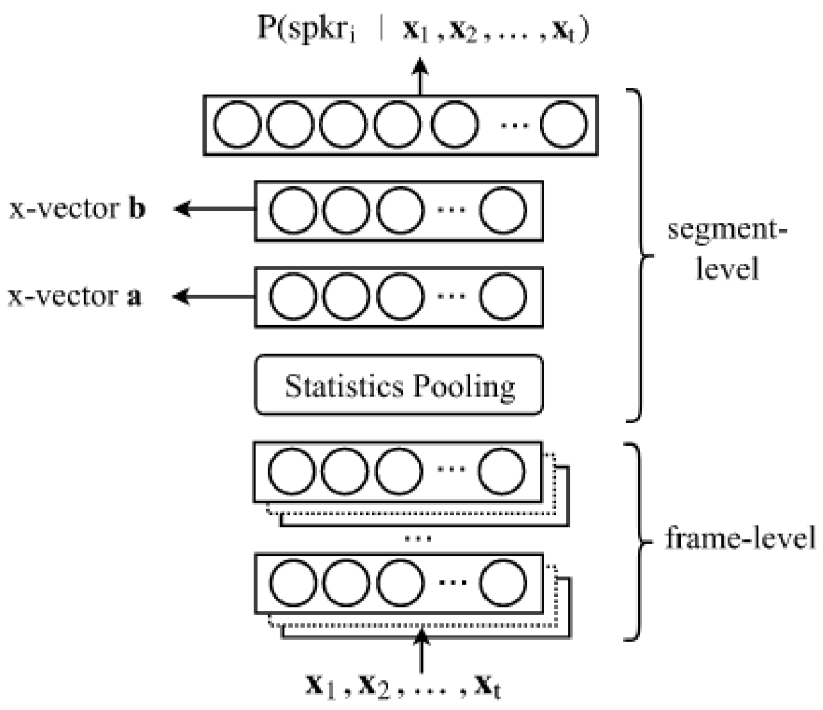
Speaker verification system against which we will be simulating spoofing attacks is x-vector system. It contains of two parts. The first part is DNN extracting embeddings, this part is called x-vector and the second part is seperately trained backend to compare those embeddings. X-vector system combines i-vector facilities like PLDA scoring, length-normalization and domain-adaptation techniques [17].

7.1.1 X-vector DNN architecture

The network of x-vector DNN architecture is illustrated in Figure 7.1 From the bottom of the figure network consists of several layers operating on speech frames, statistics pooling layer which aggregates on the frame-level representations and obtains a segment-level representation from them. There are also additional layers that operate at the segment-level and softmax output layer.

Frame-level

The first 5 layers of the network focus at the frame level using time-delay architecture. Let's suppose that the current time step is *t*. We splice frames together at the input at {t - 2, t - 1, t, t + 1, t + 2}. Output from these layers will be used by the two next layers which will splice these results {t - 2, t, t + 2} and {t - 3, t, t + 3}. Architecture has two more layers operating on frame-level, however these operate without any added temporal context. There are in total (t - 8, t + 8) frames of temporal context in the frame-level part. Based on splicing of context used layers are varying in sizes from 512 to 1536 [17].



$P(spkr_i \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$
x-vector **b**
x-vector **a**
Statistics Pooling
segment-level
frame-level
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$

Figure 7.1: Diagram of the DNN. Segment-level embeddings (e.g., x-vector **a** or **b**)can be extracted from any layer of the network after the statistics pooling layer. From [17]

Segment-level

Segment level starts with statistics pooling layer which receives on input the output of final frame-level layer. It aggregates whole frames input segment, then computes from it a segment-level mean and standard deviation. These statistics are then concatenated together and passed to additional hidden layers (either of these 2 layers can be used for embeddings computation). Last layer is softmax output layer which isn't anymore needed after training.

7.2 Experiments specific x-vector system

The x-vector extractor was trained on 1.2 million speech segments from 7,146 speakers from the VoxCeleb 1 and 2 development sets plus additional 5 million segments obtained with data augmentation. All training segments were 200 frames long. The model was evaluated on the original trials of the VOiCES challenge – model 14 in [10].

PLDA backend involves two pre-processing steps: which first reduces the x-vector dimension by LDA from 512 to 250, and then applies a length-normalization. the x-vector dimension is reduced from 512 to 250 by LDA, length-normalization of embeddings is applied. For the backend training, we concatenated all segments from each session of the VoxCeleb 1 and 2 development data. Including augmentations, this resulted in 830K files.

Table 7.1: x-vector topology from [10]. K in the first layer indicates different feature dimensionalities, T is the number of training segment frames and N is the number of speakers and t is the current time frame.

Layer	**Layer context**	**(Input) x output**
frame1	[t - 2, t - 1, t, t + 1, t + 2]	$(5 \times K) \times 512$
frame2	[t]	512×512
frame3	[t - 2, t, t + 2]	$(3 \times 512) \times 512$
frame4	[t]	512×512
frame5	[t - 3, t, t + 3]	$(3 \times 512) \times 512$
frame6	[t]	512×512
frame7	[t - 4, t, t + 4]	$(3 \times 512) \times 512$
frame8	[t]	512×512
frame9	[t]	512×1500
stats pooling	[0, T]	1500×3000
segment1	[0, T]	3000×512
segment2	[0, T]	512×512
softmax	[0, T]	$512 \times N$

Chapter 8

Tests and results

In this chapter we finally take a look on how we chose testing dataset, tests we have made and results those tests brought us. All the files used for preparation, realization and evaluation of the spoofing tests, can be found on school server pcburget in directory **/pub/users/xwojna03/CorentinJ**.

8.1 Preparing spoofing dataset

First thing needed in order to realize any kind of testing is to actually create synthesized recordings. To make a credible testing dataset it should be covering big enough number of speakers and also contain enough recordings belonging to every single speaker. Dataset should have reasonable distribution of male and female speakers and ideally not have speakers with significantly more or less recordings then other speakers.

8.1.1 LibriSpeech

LibriSpeech is a freely available corpus of approximately 1000 hours of audio sampled at 16kHz read English speech, prepared by Vassil Panayotov with the assistance of Daniel Povey[11]. The data is derived from read audiobooks from the LibriVox project, and has been carefully segmented and aligned.

Two main reasons I decided to choose reference recordings and create synthesized recordings from this dataset is that LibriSpeech provides not only audio but also transcript for it's audio. Second reason for this decision is our pretrained real-time-voice-cloning model 6.1 was trained on this dataset which allows us to do tests like comparing results of spoofing tasks synthesizing recordings with LibriSpeech transcript and with transcript unrelated to LibriSpeech.

8.1.2 Synthesizing spoofing data

We decided to create our test dataset from LibriSpeech's train-clean-360 dataset as it offers abundant number of male and female speakers and also enough recordings per speaker. We have concluded that 50 female speakers and 50 male speakers with 20 recordings per each speaker will be enough representative dataset with total of 2000 original recordings. Speakers and recordings were randomly chosen from train-clean-360 by adjusting Real-Time-Voice-Cloning's code 6.1. Our dataset will have 3 variations of synthesized recordings,

Table 8.1: Table visualizing LibriSpeech dataset

subset	hours	per-spkr minutes	female spkrs	male spkrs	total spkrs
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-clean-500	496.7	30	564	602	1166

first variations will use same text as the reference recording from original dataset. Second variation uses different text but the text used is belonging to different recording in original recordings from LibriSpeech. Third variation uses text unrelated to LibriSpeech dataset. Testing dataset has 2000 original recordings and 6000 (2000 x 3 variations) synthesized recordings in total. Table representing our testing dataset 8.1.2. The data is saved in folders orig, syn1, syn2 and syn3. Once data is synthesized we create list containing paths to all of our original and synthesized recordings which will be later needed for application of VAD(Voice activity detection) 2.4 and x-vector extraction 7.1. This was achieved with simple python script called *list_creator.py*. List needs to point to 16b .wav files as this format is needed for our VAD script. Generated files are already in 16b .wav but original files from LibriSpeech are by default .flac format. We had to convert those with a short bash script utilizing cross-platform tool **sox**.

Recordings were synthesizing on BUT's school server called pcburget with gpu NVIDIA GeForce GTX 1080 for period of 1 week. Adjusted script can be used to create another dataset similar to this with different number of male and female speakers and different number of recording per speaker required by the user. Amount of data created is controlled by arguments. For example this is how script was used for creation of our dataset *python demo_cli.py --number_of_speakers 100 --files_per_speaker 20.*

	F spkrs	M spkrs	Recs per spkr	Total recs
Original set	50	50	20	2000
Same text as in orig set	50	50	20	2000
Different text from orig set	50	50	20	2000
Text unrelated to LibriSpeech	50	50	20	2000

8.1.3 Extracting x-vector embeddings

After we have synthesized the recordings we will need to obtain their x-vector embeddings which will be later used to create target and non target trials for our evaluation script. This was achieved by running x-vector extractor script utilizing x-vector mentioned in 7.2. This script needs a list of the recordings we want it to extract x-vectors from and VAD from these files. Script obtaining VAD of the files also needs list of the recordings. The same list can be used in both cases. All three scripts mentioned were provided by thesis supervisor.

8.2 False alarm and miss rate

In order to obtain both false alarm and miss rate we have to choose **threshold s** for our system this threshold is chosen based on the requirements we have on our system. In general higher threshold means lower false alarm rate but higher miss rate.

We also have to know the **PLDA (Probabilistic Linear Discriminant Analysis) score** of the individual trials as comparison of these scores with threshold s dictates if the trial was accepted. ***Trial score < threshold s*** means trial was denied and ***Trial score > threshold s*** trial was accepted. In other words PLDA computes log-likelyhood that the pair of x-vector embeddings belongs to the same speaker and with threshold s we are specifying how much sure the system has to be in order to accept/deny the x-vector embedding pair.

False alarm rate: false alarm occurs when SRE system falsely accepts speaker who is not enrolled in the system and therefore should be declined by it. If we want to calculate false alarm rate we need to know how many false alarms occurred in a testing set and total number of non target trials in the set. Then we just use this simple formula:

$$\textbf{False alarm rate} = \frac{\text{total falsely accepted trials}}{\text{total non target trials}} * 100 \tag{8.1}$$

Miss rate: On the other hand miss occurs when system falsely rejects enrolled speaker who should be accepted. For this rate to be calculated we need to know total number of declined target trials and number of total target trials. Then we just use this simple formula:

$$\textbf{Miss rate} = \frac{\text{total falsely declined trials}}{\text{total target trials}} * 100 \tag{8.2}$$

8.3 Spoofing tests

This section will be dedicated to description of our individual tests and explanation of motives behind them. To begin with I would like to say that every trial was constructed with constriction of cross-gender trials as those would be too easy to detect and would disrupt experiment results. In total there was made 13 tests in 2 real world narratives (including test to obtain baseline data). Those tests and narratives are explained below. For creating of target and non target trials used in all narratives and tests I have created a python script *create_testing_list.py*. Target and non target trials in evaluation part are in a following format:

“enrollment x-vector audio file path” “x-vector from original audio file path” “tgt”
“enrollment x-vector audio file path” “x-vector from synthesized audio filepath” “imp”

Target trials are same in every test variation difference is in the imposter (non target) trials. In every variation of tests I will mention number of both target and non target trials. Results of every test variation can be later found in 8.2.

8.3.1 Narrative 1 - Voice controlled house assistant

In first narrative let's imagine we have an ASV system which isn't forced to put emphasize on security but instead it is focused to be reliable to accept our enrolled speakers therefore system like this prioritizes to have as low miss rate as possible even if it comes with a cost of accepting speakers that aren't enrolled in the system at higher rates. System wanting to achieve this behavior could be for example some kind of home assistant controlling devices like air-conditioner or blinds at the house. In scenario like this we are more concerned to not be falsely rejected as an owner then somebody coming into our house and shutting down our blinds. One could say we don't need ASV system for this scenario at all but let's suppose we have little kids and we don't want them to be able to control these devices. Let's set our goal to get miss rate at around 1% +-0.1%

Baseline: In order to successfully measure how individual tests performed we will need some kind of baseline performance. For this purpose we are running tests solely with our original non synthesized audio x-vector embeddings 8.1.2. In baseline test we have constructed in total 18961 target trials and 93100 non target trials. Non target trials in this case aren't between enrolled audio and synthesized x-vector but between enrolled audio x-vector and original x-vector of different speaker. By evaluating this test with bash script provided by thesis supervisor we can obtain PLDA scores for individual trials. These PLDA scores are then compared with threshold we have chosen. In our case we wanted threshold which will keep miss rate at around 1%. After little experimentation with threshold we found out that Equal error rate threshold equal to 8.453 will do just fine resulting in miss rate of 1.069% and same rate of false alarms at 1.069% by EER definition 3.1. The calculations of false alarm rate and miss rate were made with python script utilizing formulas mentioned in 8.2. We were also able to obtain EER of the system and minDCF of the system. All of the results can be found in table 8.2.

Same embedding same text: In first test we are trying to spoof the system with x-vector embedding of synthesized recording with reference recording same as original recording used for enrollment. We are also using same text as in the enrolled recording. These recordings were picked from 2. row of the table 8.1.2 This way the x-vector embedding on the right side of the non target trial should be very similar to x-vector embedding on the left side. In this scenario we are expecting highest false alarm rate as the attacking condition doesn't get much better from here supposing attacker has access to recording of target speaker used for enrollment in the system. It might be interesting to compare results of this test with results of simple replay attack 4.1. In total we have 18961 target trials and 1900 non target trials. I am again reminding that as I said in previous test we are going to use threshold used in baseline test for every other test in this testing narrative. After running the evaluation we have found out that false alarm rate jumped to 70.474% which is almost 70-fold increase from baseline.

Different embedding same text: X-vector embeddings for this test are also picked from 2. row of the table 8.1.2. But this time x-vector embedding on the right side of each trial (our spoofing embedding) comes from synthesized recording which had been given reference recording different from enrollment recording on the left side of the trial. This is slight complication for attacker as he doesn't have access to enrollment recording of the target speaker. Test consists of 18961 target trials and 18961 non target trials. As one would expect every complication for attacker should decrease false alarm rate of the system closer to FA rate in baseline test. Which turns of to be true if we look at the results 8.2. However even though false alarm rate dropped to almost half from previous test it is still at 36.168%. Similarly EER dropped almost by a half to 5.195%.

Same embedding different text: Third test is similarly to first test trying to spoof the ASV system with x-vector embedding extracted from synthesized recording which used reference recording utilized in enrollment side of the trial. However in this variation of the test the synthesized recording contains different text then the original one. With this diversification we are trying to get an idea how big of an impact does text has on the penetration of text-independent X-vector based ASV systems. There are 18961 target trials and 1900 non target trials in this variation. X-vector embeddings for the non target trials were selected from 3. row in 8.1.2. Test consists of 18961 target trials and 1900 non target trials.

Different embedding different text: Fourth test is similar to second test on the right side of non target trials we are using x-vector embeddings extracted from synthesized recording which has different reference recording than recording from which we extracted x-vector used for the left size of the non target trial. Again similarly to third test we are using different text then the original recording. X-vector embeddings used to construct non target trials were selected from 3. row in 8.1.2. Test has total of 18961 target trials and 18961 non target trials.

Same embedding unrelated text: In fifth test we are getting x-vector embeddings for our non target trials in similar fashion as in first and third test. There is another variation of text contained in the recordings from which x-vectors for right side of non target trials were extracted. This time it's text completely unrelated to whole LibriSpeech dataset. With this test we are trying to test how the FA rate will be affected when we start to introduce transcript which isn't from dataset which our tts-model used for creation of spoofing recordings was trained on 6.1. This test contains 18961 target trials and 1900 non target trial. X-vector embeddings for non target trials were selected from 4. row in 8.1.2.

Different embedding unrelated text: Sixth and last test is again similar to fifth test using text unrelated to LibriSpeech but once again with difference that the x-vector embeddings used for the right side of non target trials are extracted from synthesized message that used reference recording different from the original recording used for X-vector embedding on the left side of the non target trial. X-vectors used for construction of non target trials were selected from 4. row in 8.1.2. Test contains total of 18961 target trials and 18961 non target trials.

8.3.2 Narrative 1 results

We can see that results of the tests are ranging from 70.473% in scenario of same embedding and same text to 16.912% in the scenario of different embedding and unrelated text. The false alarm rate is decreasing with every test as attacker is going further and further away from enrollment embedding and as text of the recording is deviating from enrolled recording and from dataset our tts model was trained on. Tests with different text had almost twice as high false alarm rate as tests with unrelated text which confirms that using text which

Table 8.2: Results of first testing narrative - Voice controlled house assistant

Test	Threshold	EER [%]	minDCF	FA [%]	Miss [%]
baseline	8.453	1.069	0.043	1.069	1.069
same embedding same text	**8.453**	**9.102**	**0.236**	**70.473**	**1.069**
different embedding same text	8.453	5.194	0.154	36.167	1.069
same embedding different text	**8.453**	**5.952**	**0.169**	**44.368**	**1.069**
different embedding different text	8.453	5.015	0.137	33.533	1.069
same embedding unrelated text	**8.453**	**3.844**	**0.102**	**22.947**	**1.069**
different embedding unrelated text	8.453	3.282	0.101	16.912	1.069

isn't part of LibriSpeech in spoofing recording is playing significant role on the attack. It might be interesting to create testing dataset from different set than LibriSpeech and see how the Real-time-voice-cloning software will do in scenario where it uses dataset it wasn't trained on. Overall I think the results are showing that we have successfully penetrated the system.

8.3.3 Narrative 2 - Bank system with automatic speaker verification

In our second scenario let's imagine there is bank system with payment authorization based on ASV system. In scenario like this we can expect the security of the system to be a top priority as there is money involved. The bank has been breached in a past and bank ASV system administrators now have big enough dataset of synthesized messages attackers were using in order to break the system. Administrators were given orders that the false alarm rate of bank's ASV shouldn't surpass 5%. This can be achieved by adjusting systems threshold for trial acceptance. In this part of spoofing tests we will explore what threshold is needed in individual tests in order to keep false alarm rate under 5% and how this will affect miss rate of our system.

8.3.4 Narrative 2 results

Trials in individual tests of second narrative are same as in first narrative. With this in mind I will just skip describing how and why those were constructed as it was already described in 8.3.1. As said above the main point of second narrative is to find out system threshold where FA rate will be kept below 5%. To find those threshold we simply tinker with our script for FA and miss rate calculations. We start low threshold where FA is above 5% in every single test variation. For simplicity let's just use 8.530 as we already have results with this threshold from first narrative 8.2. Then we keep running the false_miss_calc.py gradually increasing threshold until we find value close to 5% let's set our acceptable deviation to -0.1%. Results of this can be seen in table below 8.3

Table 8.3: Results of second testing narrative - Bank system with automatic speaker verification

Test	**Threshold**	**EER [%]**	**minDCF**	**FA [%]**	**Miss [%]**
same embedding same text	**37.117**	**9.102**	**0.236**	**4.947**	**11.253**
same embedding different text	**29.093**	**5.952**	**0.169**	**4.947**	**6.6487**
different embedding same text	26.023	5.195	0.155	4.968	5.284
different embedding different text	25.343	5.015	0.137	4.999	5.026
same embedding unrelated text	**21.023**	**3.844**	**0.102**	**4.947**	**3.477**
different embedding unrelated text	17.572	3.282	0.102	4.973	2.618

As we can see in the table above 8.3 by changing system's threshold we were able to get false alarm back below 5% however with the cost of higher rate of enrolled speakers rejection. At FA rate of 5% the system has highest miss rate of 11.253% which is in my opinion still in a range of functional system. Another question is if 5% would be in real world considered safe enough for financial sector. If not I don't think there is much room for miss rate growth if we want to keep system reasonably accurate. If we would consider this FA rate safe enough I think the system can be successfully defended against our attacks by increasing it's threshold.

Chapter 9

Conclusion

This thesis was focused on realization of penetration tests of automatic speaker verification system. In order to realize those tests we had to get familiar with ASV systems but also with types of spoofing attacks which are possible on those systems. This was done in research part of the thesis in chapters 2-5. In second part we decided that we will try to spoof the system with synthesized speech obtained with Real-time-voice-cloning model by CorentinJ. We described in detail the model of specific ASV system working with x-vector embeddings and finally realized spoofing attacks against it.

In first narrative of our attacks in the bast case we were able to drive systems false alarm rate to 70.473% from baseline of 1.069%. In the worst case it was 16.912% which in consideration that the attacks can be realized on the large scale would be still considered penetrating the system. In the second narrative we were able to get the false alarm rates back below 5% with the cost of higher miss rate.

In order to realize these tests we had to create multiple python and bash scripts. Three of them being most significant. First script was edition of demo_cli.py in Real-time-voice-cloning model to choose number of speakers and recordings we would like to synthesize from LibriSpeech dataset with 3 text variations. Second script was used to construct target and non target trials required for valuation. Third script was used to calculate false alarm and miss rate of the system from PLDA scores obtained from evaluation script. Overall I think the goals of the thesis were achieved and the results proved it's point.

9.1 Future work

As was said few times in the thesis the model used for creation of spoofing recordings was trained on the same dataset which was used for enrollment of the messages. It would be interesting to see how the Real-time-voice-cloning model would do if the testing dataset was created with different dataset then LibriSpeech.

Bibliography

[1] Cernocky, J. H. et al. *Analysis of Score Normalization in Multilingual Speaker Recognition.* Brno University of Technology, 2017.

[2] consortium, A. *ASVspoof 2019:Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan.* ASVspoof consortium, 2019.

[3] Ergünay, S. K. et al. *Vulnerability of Speaker Verification Systems Against Voice Conversion Spoofing Attacks: the Case of Telephone Speech.* 1st ed. IEEE, 2012. ISBN 978-1-4673-0044-5.

[4] Faundez Zanuy, M. and Monte Moreno, E. *State-of-the-art in speaker recognition.* IEEE, 2005.

[5] Federico Alegre, . N. E. *Re-assessing the threat of replay spoofing attacks against automatic speaker verification.* 1st ed. IEEE, 2015. ISBN 978-3-88579-624-4.

[6] Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J. et al. *Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis.* 2019.

[7] Kinnunen, T. et al. *On the vulnerability of speaker verification to realistic voice spoofing.* 1st ed. IEEE, 2015. ISBN 978-1-4799-8776-4.

[8] Kshirsagar, A. et al. *Comparative Study of Phoneme Recognition Techniques.* IEEE, 2012. ISBN 978-1-4673-3149-4.

[9] Liu, S. et al. *Jointly Trained Conversion Model and WaveNet Vocoder for Non-parallelVoice Conversion using Mel-spectrograms and Phonetic Posteriorgrams.* 1st ed. ISCA, 2019 [cit. 2019-15-09].

[10] Matějka, P., Plchot, O., Zeinali, H., Mošner, L., Silnova, A. et al. *Analysis of BUT Submission in Far-Field Scenarios of VOiCES 2019 Challenge.* Brno University of Technology, 2019.

[11] Panayotova, V., Chen, G., Povey, D. and Khudanpur, S. *LIBRISPEECH: AN ASR CORPUS BASED ON PUBLIC DOMAIN AUDIO BOOKS.* ICASSP, 2015.

[12] Plchot, O. *Extensions to Probabilistic Linear Discriminant Analysis for Speaker Recognition.* Brno University of Technology, 2014.

[13] Ramire, J. et al. *Efficient voice activity detection algorithms using long-term speech information.* Universidad de Granada, 2003.

[14] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N. et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. 2018.

[15] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N. et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. 2018.

[16] S.Lemmetty. *Review of Speech Synthesis Technology*. Prentice Hall PTR, 1999.

[17] Snyder, D. *X-VECTORS: ROBUST NEURAL EMBEDDINGS FOR SPEAKER RECOGNITION*. The Johns Hopkins University, 2020.

[18] Wan, L., Wang, Q., Papir, A. and Moreno, I. L. Generalized End-to-End Loss for Speaker Verification. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, p. 4879–4883. DOI: 10.1109/ICASSP.2018.8462665.

[19] Wang, Y., Skerry Ryan, R., Stanton, D., Wu, Y., Weiss, R. J. et al. Tacotron: Towards End-to-End Speech Synthesis. In:. 2017. Available at: https://arxiv.org/abs/1703.10135.

[20] X.Huang, H.-W. H. *Spoken Language Processing*. Prentice Hall PTR, 2001.