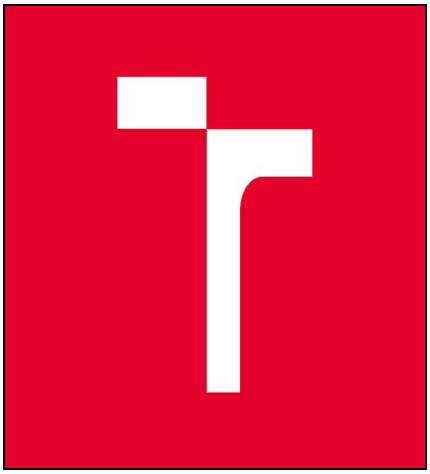

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

Penetration Tests of Speaker Verification System

Penetrační testy systému pro verifikaci řečníka

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **Quang Trang Nguyen**
AUTOR PRÁCE

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Nguyen QuangTrang**
Program: Informační technologie
Název: **Penetrační testy systému pro verifikaci řečníka**
Penetration Tests of Speaker Verification System
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se se systémem pro rozpoznávání řečníka dostupným ve speech@FIT.
2. Prostudujte problematiku syntézy řeči se zaměřením na imitaci konkrétního řečníka.
3. Seznamte se s metrikami používanými v problematice verifikace mluvčího.
4. Navrhněte sadu penetračních testů systému pro verifikaci řečníka za použití vhodného softwaru pro syntézu řeči a dostupných nahrávek cílových mluvčí
5. Vyhodnoťte výsledky testů a pokuste se navrhnout kroky vedoucí k větší bezpečnosti cílového systému.

Literatura:

- Podle doporučení vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: Plchot Oldřich, Ing., Ph.D.
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 31. července 2020

ABSTRACT

The aim of this bachelor thesis is to create a penetration tests of speaker verification system with the use of the speech synthesis method. This work studies methods of spoofing against automatic speaker verification system. Before designing of the test set, the system and it's components that were used in this work are described. The last chapters of this work include a description of the process of designing the test set, realization of the designed test and the last part contains evaluation of the results and answers the question if it is possible to penetrate a verification system with the use of speech synthesis.

Abstrakt

Cílem bakalářské práce je návrhnout sadu penetračních testů pro verifikaci řečníka s použítím syntézy řeči a dostupných nahrávek cílových mluvčí. Práce zahrnuje studium problematiky pro syntézu řeči, verifikace řečníka a metod pro spoofing se kterými můžeme setkat. Před samotným návrhem testovací sady je popsán systém a jeho komponenty, který byl použít v této práci. V posledních kapitolách práce je uveden popis návrhu testovacích sad a způsob realizace testů. Na závěru jsou vyhodnoceny výsledky a je odpovězeno na otázku, zda je možné prolomit systém pro verfikaci řečníka s využitím metody pro syntézu řeči.

Keywords

Speech synthesis, x-vector, decoder, encoder, verification, spoofing, wavenet, neural networks

Klíčová slova

Syntéza řeči, x-vektor, decoder, encoder, verifikace, spoofing, wavenet, neuronové sítě

Reference

Nguyen, QuangTrang. *Penetration tests of verification system*. Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing.Oldřich Plchot, PhD

Penetration Tests of Speaker Verification System

Declaration

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

….....

Quang Trang Nguyen

july 26, 2020

Acknowledgements

I would like to thank to Mr Oldřich plchot for all the valuable advices and helps that were provided while i was working on this thesis.

Table of Contents

Chapter 1

1 Introduction

Speaker recognition is a biometric recognition technique, the word biometric can be decomposed as bio and metric, where bio represents life and metric represents measures. Biometric is the technology for measuring and analyzing human's behavioral or physiological individuality. It can be used for recognizing a person on the basis of his/her voice, face, iris, DNA, signature, fingerprint, hand, geometry etc. Speaker recognition is widely used in voice dialing, banking over a telephone network, database access services, voice mail, remote access to computers and a very important application of the speaker recognition technology is it's use as a forensics tool. Popularity of this technology is based on the fact that it is less prone to attacks. Even though this technology is less prone to a spoofing attacks, we can still perform a spoofing attack using multiple spoofing techniques such as voice conversion, text synthesis, replay and impersonation.

This work is structured into 7 chapters. In the second chapter the definition of Automatic speaker recognition and it's main tasks are described. In the third chapter different spoofing methods that can be used for spoofing against automatic speaker recognition are presented and one of the speech synthesis techniques is chosen to be used in this work. In the fourth chapter the chosen spoofing method is described in more detail. In the fifth chapter the Text-to-Speech system and it's main components that were used in this work for generating the spoofing attack are described. In the sixth chapter the preparation of data used for the synthesis and the creation of the spoofing task is described. Finally, the evaluation of the chosen method is performed and in the last chapter, main questions of this work are answered.

1.1 *Claims of this Thesis*

The main focus of this thesis was on creating a spoofing attack by using the speech synthesis with the help of a Text-To-Speech system called *Real-time voice cloning* provided by Jemine Corentine [1] for generating synthesized audio that is used for the spoofing task. Thesis then summarizes the results of the testing and answers the question: If it is possible to penetrate an automatic speaker recognition system by using the speech synthesis method.

Chapter 2

2 Automatic speaker recognition overview

In this chapter, we describe a speaker recognition systems in general

2.1.1 Automatic speaker recognition(SRE)

Automatic speaker recognition(SRE)[2] is the process of comparing two speech signals produced by the human vocal tract using specific characteristics of the speech signal and answering the question whether these two speech signals belong to the same person or not.

Human voice (speech signal) is different for each individual and contains different types of information that can be used for authentication. Using the the information given from the speech we can use it to identify the person. After the speech is produced by the vocal tract, it passes through an environment to a point where it is recorded. This environment has a great effect on the quality of the speech, which can effect the performance of SRE systems. Using a speech signal mainly three kinds of recognition can be performed: Speech recognition (what is spoken), speaker recognition (who is speaking) and language identification (identifying the language spoken by the speaker).

The SRE can be divided into text-dependent and text-independent. Text-Dependent system assumes the knowledge of the speech content and the result of a test trial depends on a spoken phrase, while text-independent system does not depend on specific text.

2.1.2 Speaker recognition

speaker recognition can be classified into speaker Identification and Verification. Speaker Identification is the process of identifying who is speaking in the recording and speaker Verification is the process of accepting or rejecting identity claimed by the speaker. Figure 2.2 is the visualization of the speaker recognition process.

Speaker identification is a one to n(1 : n) matching system. In speaker identification we analyze and compare a speech utterance from an unknown speaker with speech models of known speakers in the system. The unknown speaker is identified as the speaker whose model best matches the input utterance.

Speaker verification is a one to one(1 : 1) matching system. In speaker verification, an identity is claimed by the unknown speaker, and an utterance of this unknown speaker is compared to the model of the speaker whose identity is being claimed. If the match(the score from the verification system) is above a threshold(a coefficient that we can set), the identity is accepted and if the match is bellow the threshold the identity will be rejected. A high threshold makes it difficult for impostors to be accepted by the system, but with it also increases the risk of falsely rejecting valid users(false rejection). These two tasks represent a so-called speaker verification *trial.* If the same speaker is speaking in the two recordings, then the trial is called a *target trial.* If it is not the same speaker speaking in the two recordings, then the trial is called a *non-target trial*

The effectiveness of speaker verification systems can be evaluated by using receiver operating characteristics (ROC) curve adopted from psychophysics. The ROC curve is obtained by assigning two probabilities, the probability of correct acceptance(false rejection rate) and the probability of

incorrect acceptance(false acceptance rate). The detection error trade-off (DET) curve is also used, in which false rejection(missed detections) and false acceptance(false alarms) rates are assigned to the vertical and horizontal axes. The comparison of two verification system is shown in figure 2.1

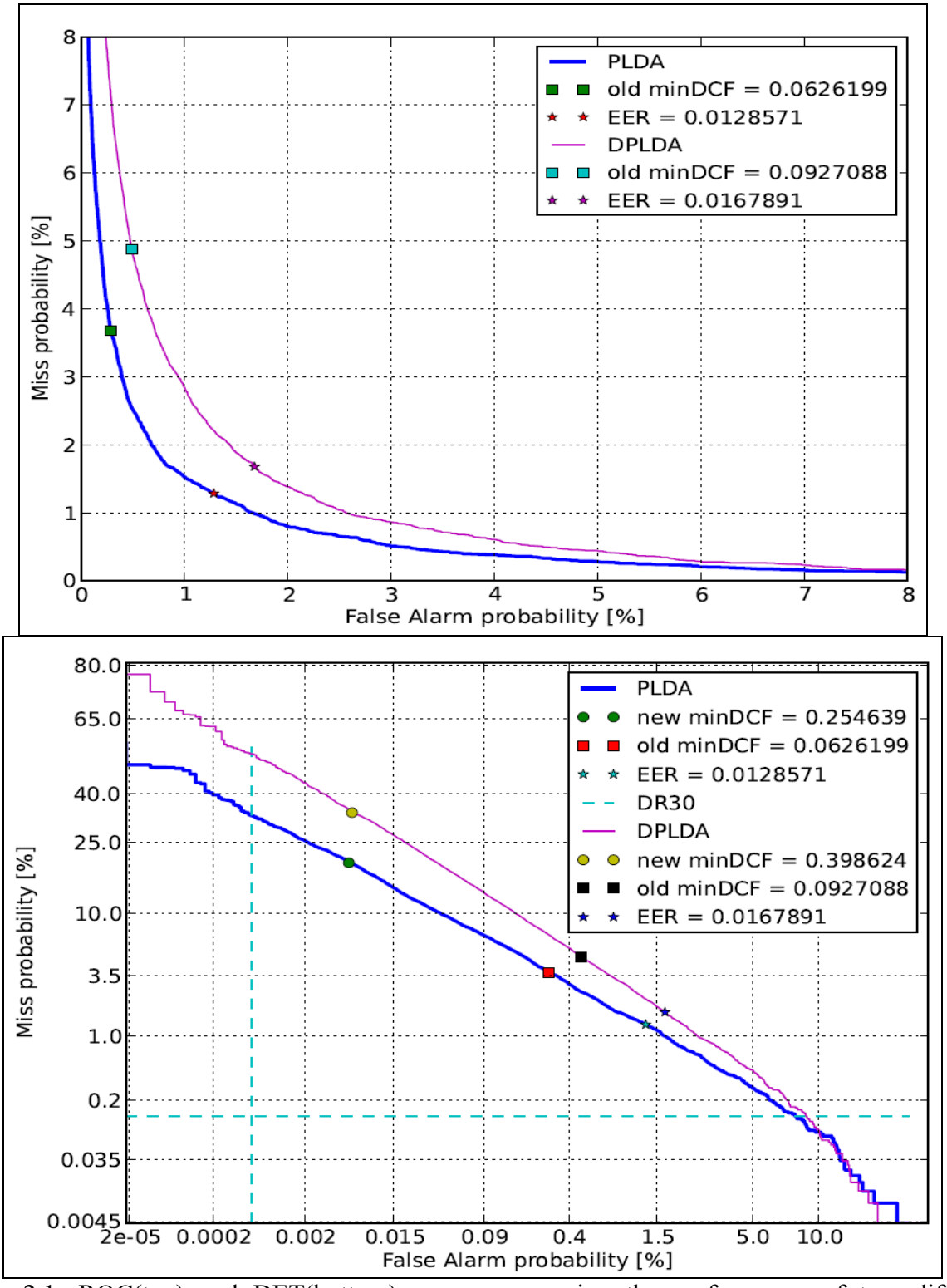

Figure 2.1: ROC(top) and DET(bottom) curves comparing the performance of two different techniques(PLDA and DPLDA), in general the system with lower EER has better accuracy. Figure extracted from[2]

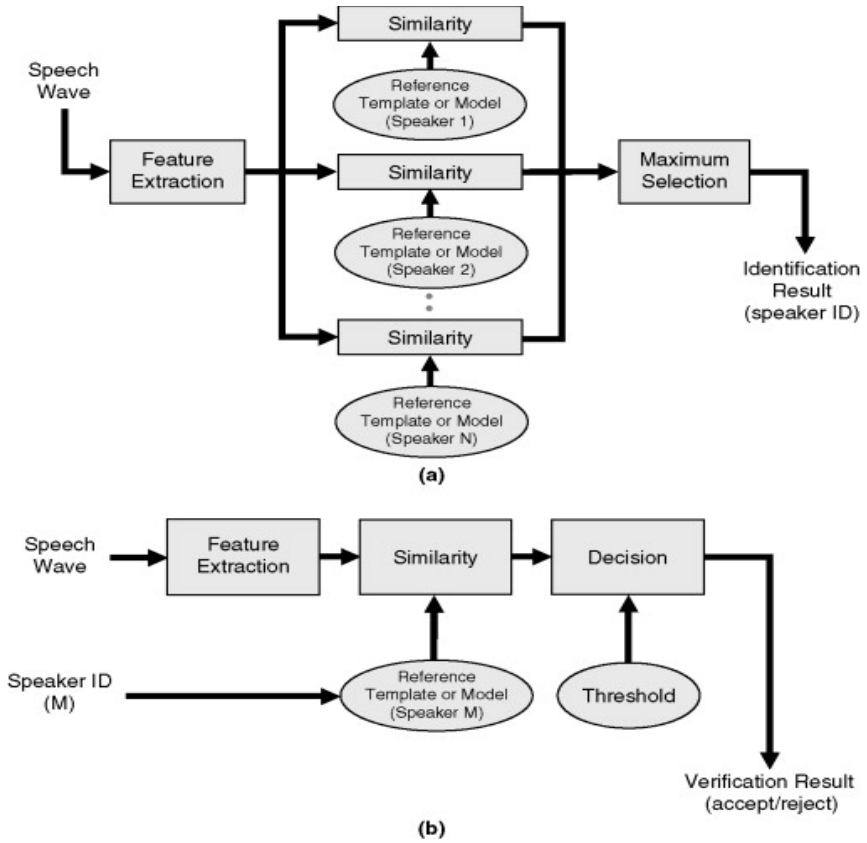

Figure 2.2 describes the process of speaker recognition and verification. On the top(a) is a typical process of speaker identification and bottom(b) is the process of speaker verification. Figure extracted from [3]

2.1.3 Process of Speaker recognition system

In order to perform a speaker recognition task, it is necessary to transform the continuous speech into a form that can be used by the system. This process consists of sampling and quantization and the result is a discrete version of the signal. The sampling frequency is usually 8kHz or more. In speech recognition several layers of information can be extracted from the signal as bellow:

- Acoustic: spectral representation of the speech
- prosodic: features encoding the prosody
- phonetic: analysis of sequences of phonemes specific to the speaker
- idiolect – analysis of sequences of words or short phrases
- linguistic- analysis of linguistic patterns characteristic to speaker's conversation style

2.1.3.1 *Feature Extraction*

Methods for extracting the spectral representation of speech signal are based on the assumption that the signal can be considered stationary within short segments(usually 10ms long segment). These segments can be obtained by *windowing* the signal with a rectangular window function. After windowing, the power magnitude Fourier spectrum is computed for every frame, which is then further parameterized into feature vector. In speaker verification the Mel-Filterbank Cepstral Coefficients (MFCC) is used for extracting the feature vector. The process of MFCC extractions is shown in figure 2.3. In the first step of MFCC the absolute value of the short-term Discrete Fourier Transform (DFT) is used to extract the amplitude of the spectrum from each frames. Then, Mel-filterbank is applied to smooth the spectrum. A vector of band energies is then computed as a weighted sum of squared values of the amplitude spectrum. The overall frame energy is then computed as an average of squared samples. A logarithm of the overall energy is taken and in the final step the feature vector is de-correlated and its dimensionality is reduced by projection into a certain amount of Discrete Cosine Transform (DCT) bases.

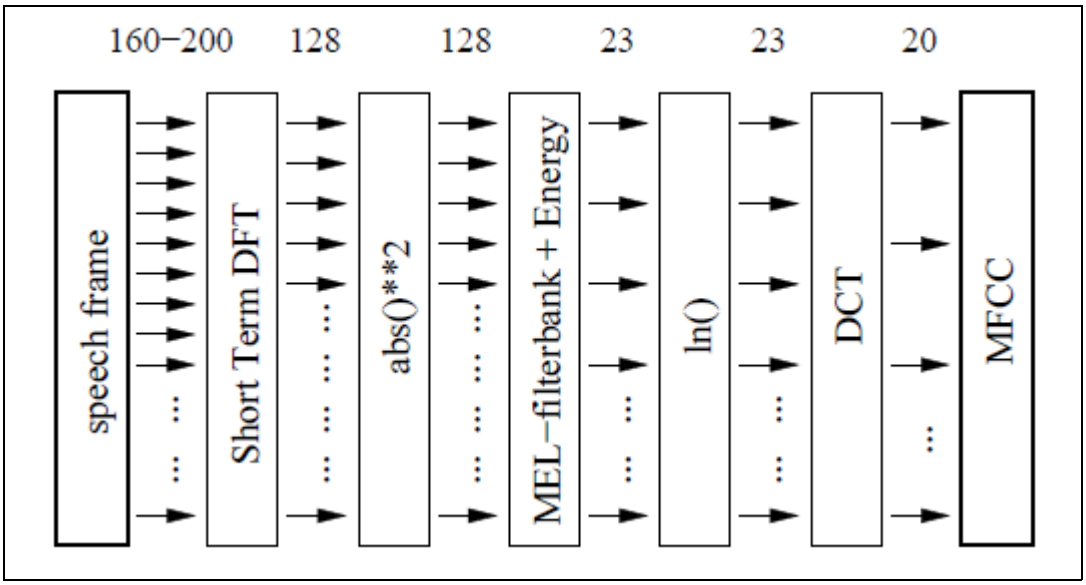

Figure 2.3 MFCC extraction steps, dimensionalities are shown above the blocks for frame lengths of 20 and 25 ms at sampling frequency f_s = 8kHz. Figure extracted from [2]

2.1.3.2 *Voice Activity Detection*

Voice Activity Detection (VAD), also known as Speech Activity Detection (SAD) is a very important pre-processing step in most of speaker recognition system. It is designed to select only those frames from the analyzed utterance, that contain speech. VAD can be implemented with many different approaches. Noticeably VAD based on simple energy thresholding, Gaussian Mixture Model (GMM) classifier or Neural Networks (NN) trained to discriminate between speech and the rest of the audio signal.

2.1.3.3 *Speaker verification scoring*

In Speaker recognition the score of a verification task is usually obtained by evaluating statistic models as a log-likelihood between two hypothesis. The two hypothesis correspond to answers of "*yes*" or "*no*" to the two question: *A*: i*s the same speaker speaking in this recording*? And *B*: i*s the speaker speaking in this recording* ?. A score can also be obtained, by using a simple metric based on a distance between feature vectors characterizing the whole utterance(cosine similarity).

There are two approaches to training the scoring model: *generative* and *discriminative*. Generative models are trained to estimate the underlying distribution of the data, from which the input features can be generated. Thanks to its simplicity and robustness it is widely used in speaker verification scoring. The discriminate models are trained to directly predict class from the input data.

2.1.3.4 *Score Normalization*

Score normalization techniques aim to reduce the scores variabilities in order to help the estimation of a unique speaker-independent threshold during the decision step. Most of the normalization techniques are based on the estimation of the impostors scores distribution where the mean,*m*, and the standard deviation *d* depend on the considered speaker model and test utterance. Then score normalization for each of the new coming score *s* can computed as :

$$score(s) = s - \frac{m}{d} \tag{2.1}$$

Score normalization is not required in the current state-of-the-art techniques for the text-independent speaker verification. Some of the popular scoring normalization techniques(for further reading [2]) are listen bellow:

- Zero Normalization – Z-norm
- Test Normalization – T-norm
- ZT-norm
- S-norm

Chapter 3

3 Automatic speaker verification(ASV) spoofing and it's methods

Spoofing attack – is a situation in which a person or program identifies as another identity by using falsified data.

ASV spoofing – Is an attack performed on a speaker recognition system by using spoofed speech samples. The spoofed samples can be obtained using voice conversion methods to convert an impostor speech to the target speaker speech, or by using a recording device to record the speech samples from the target speaker. The spoofing attacks are classified into five types, Speech Synthesis (SS), Voice Conversion (VC), replay, identical twins, and impersonation. Each of the methods and it's availability is shown in figure 3.1

3.1 *Impersonation*

A process of producing the similar voice pattern and speech behaviour of the target speaker. The impersonators do not require any machines or technical knowledge to imitate the target speaker. A professional impersonator can try to make a better imitation by trying to mimic the target speaker's prosody, accent, pronunciation, lexicon, and other high-level speaker traits. Even though this kind of attack is proved to be one of the more successful methods, it can not be performed on a larger scale, because it takes longer for the impersonator to mimic the target speakers. According to a study reported in [4] found that if the impersonator is aware of the target speaker's voice and has similar voice pattern, he will be able to crack the speaker verification system.

3.2 *Replay*

Replay is one of the most easiest and simple spoofing attacks. The replay is a type of attack, where the attacker uses a pre-recorded speech signal of the target speaker's voice that is captured using a recording device to get access to the system. This attack does not require any specific expertise or any sophisticated equipment, therefore it is very easy to implement. The spoofing attack has a little to almost no changes(depending on the recording device and environment) in sound characteristics from the target speaker's voice.

3.3 *Speech Synthesis*

Speech synthesis (SS) also known as Text-To-Speech (TTS) system, is a technique for generating intelligible, natural sounding artificial speech for any text. It is a system, where the text is given at the input and the system generates a speech signal at the output. Most speech synthesis systems have two main components: text analysis followed by speech form generation. In the text analysis component, the input text is converted into a linguistic specification consisting of elements such as phonemes, prosody, consonants, vowels. In the speech form generation component, speech waveforms are generated based on the produced linguistic specification. The speech synthesis will be explained in more detail in the following chapter 4.

3.4 *Voice Conversion*

Voice conversion (VAC) is a spoofing attack against automatic speaker verification using an attacker's natural voice which is converted towards that of the target. It aims to convert one speaker's voice towards that of another. Most voice conversion requires a parallel corpus where source and target speakers read out identical utterances. VC can be used to create new voices for TTS synthesis systems. Other applications include speaking aid devices that generate more natural voice sounds to help people with speech disorders, language learning, and signing voice conversion.

3.5 *ASVspoof2015-2019*

The ASV spoofing 2015-2019 is a spoofing challenge involved detection of artificial speech created using a mixture of voice conversion and speech synthesis techniques. In ASV spoofing 2015[5] a speech synthesis algorithm(*S10*) implemented with the open-source MARY Text-To-Speech system received the highest EER(Equal Error Rate) with 51.17% for the female speakers and 44.20% for male speakers. It was trained with 40 utterances per speaker.

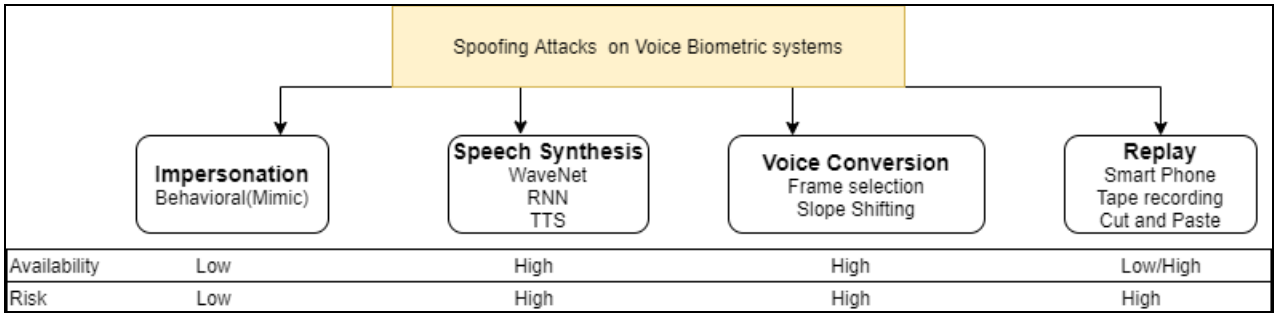

Figure 3.1: Available methods and algorithms that can be used, with their risk and availability, figure extracted from [6]

Chapter 4

In this chapter, we describe the speech synthesis method and it's techniques.

4 Speech Synthesis

Speech synthesis is a very popular method for creating a speech for any given text, it is widely used for applications ranging from funny celebs voice-over videos to google text-to-speech apps. It can also used for spoofing on speaker verification systems and for developing a counter measures. In this chapter the speech synthesis techniques are described.

4.1 *Rudimentary Techniques*

The first rudimentary method is to use a pre-recorded complete or partial messages, and having an application program or a speech server to read the static messages out loud. This method is very simple to implement, but it requires all messages to be recorded in advance in order to be uttered by a machine. This method does not provide flexibility, since if a message changes, it has to be rerecorded. However, recorded speech sounds more natural than other synthesis techniques and is widely used in airports, railway stations, buses and many other places.

A second method is to record all the words of the application and save it in a lexicon in digital format. To generate a speech, the synthesizer sequentially looks up the words in the lexicon, fetches their digital recording, concatenates them, and converts them into sounds using the digital to analog converter of the sound card. Like the first method, it requires recording of all the words in the vocabulary, but it is more flexible than the first method, since it is no longer constrained by fixed messages. However, the synthesized speech does not sound as natural as the speech generated by the first method, because the utterance of words varies according to prosodic context and the concatenation of words is never perfect.

4.2 *Speech Synthesis with Phonemes*

This method generates a speech synthesis of a message using phonemes. This technique shrinks the whole messages or words into phonemes, by doing this we can use sound recording of the vowel and consonant for generating speech. This dramatically reduces the storage requirements for the database. Speech synthesis with phonemes can deal with potentially unlimited vocabulary and it enables us to generate any message dynamically. Phonetic speech synthesizers are also called text-to-speech converters (TTS) .

Most text-to-speech system is composed of two parts: a front-end and a back-end. In the first step the front-end converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often referred as *tokenization* or *text normalization*, after this phonetic transcriptions are assigned to each word. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. This process is often called text-to-phoneme or grapheme-to-phoneme conversion. The output of the front-end is the linguistic(it is made up of phonetic transcription and prosody information) representation. The back-end often referred as the *synthesizer* then converts the symbolic linguistic representation into sound. A typical TTS system is shown in figure 4.1.

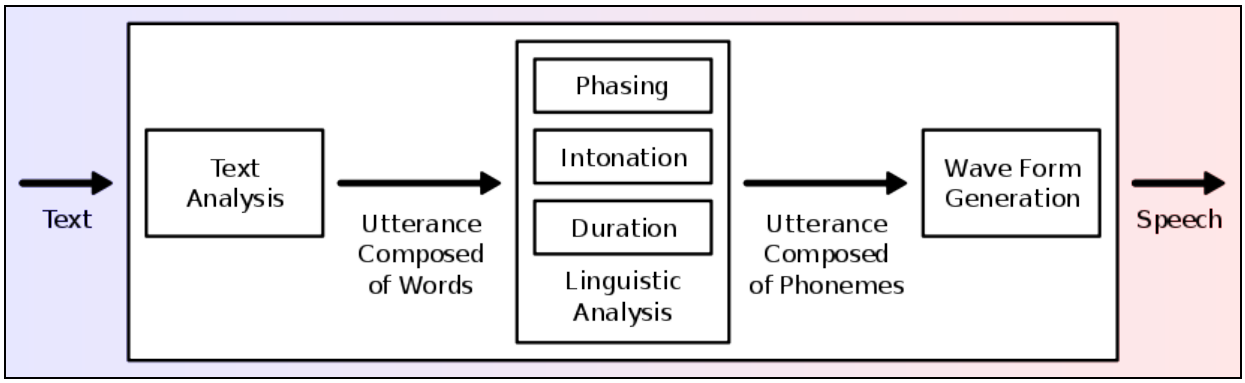

Figure 4.1: overview of a typical TTS system, figure extracted from wikipedia

4.2.1 Synthesis based On Waveform Coding.

Is a method where short segmental units of human voice, usually words or phrases, are stored. The speech is then generated by selecting and connecting the desired segments. The quality of the final speech is influenced by the the quality of the continuity of the acoustic features at the connections between units, this method is preferred in our thesis. The visualization of its process is shown in figure 4.2.

4.2.2 Synthesis based on Analysis-synthesis

In this method words or phrases of human speech are analyzed and stored as time sequences of feature parameters. These parameters are then connected and supplied to a speech synthesizer to produced the spoken message. The visualization of its process is shown in figure 4.2**.**

4.2.3 Synthesis by Rule

This method can produce words sentences based on sequences of phonetic syllabic symbols or letters. In this method, feature parameters of syllables or phonemes are stored and connected by rules, prosodic features are also controlled by rules. The process of this method is shown figure 4.2.

As shown in figure 3.1, both Voice Conversion and speech synthesis with phonemes (TTS) can be used as a tool to create an ASV spoofing attack. In this thesis I decided to use a Text-To-Speech system to generate an ASV spoofing attack, since it is widely used nowadays and it is widely available on the internet, the Text-To-Speech system will be later described in chapter 5.

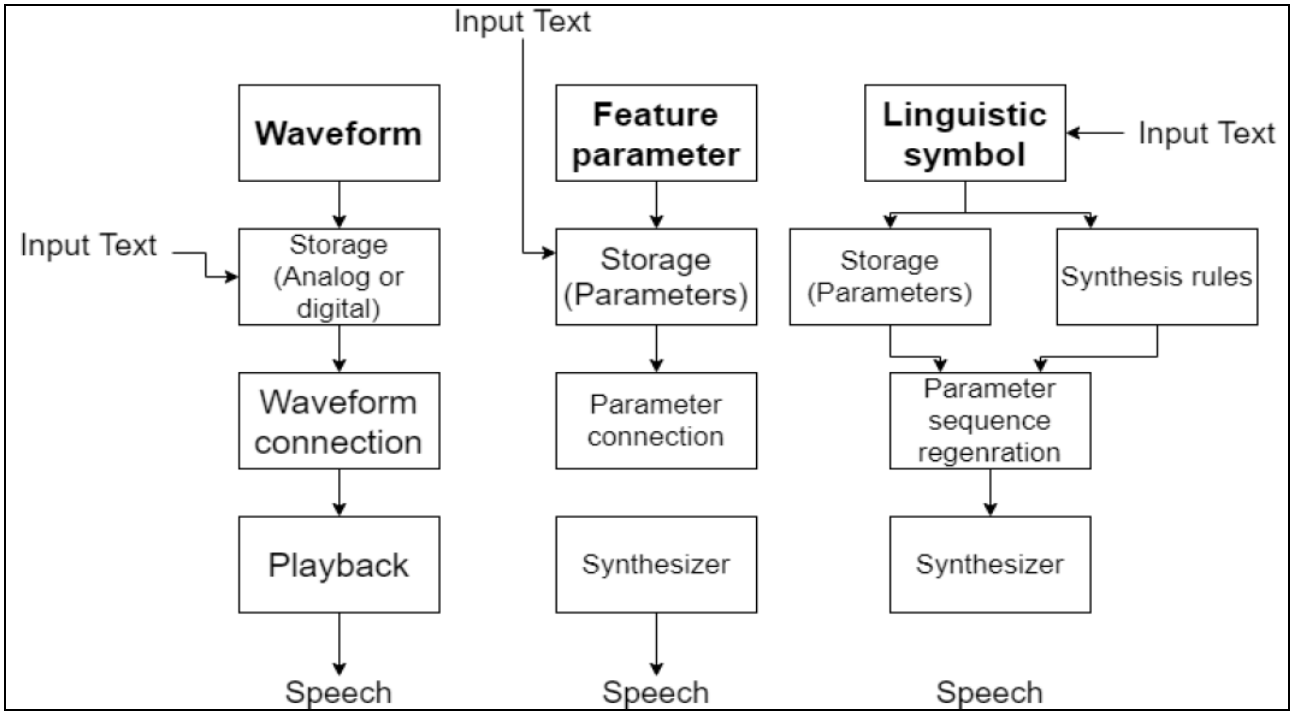

Figure 4.2: Graphic representation of the three speech synthesis methods from left to right Waveform coding, Analysis-synthesis, Synthesis by rule, figure extracted from [7], the WaveForm coding is preferred in our work.

Chapter 5

In this chapter the Text-To-Speech system that is used in this thesis for generating the synthesized voices is described. There are many TTS system that are available on the internet, but most of them are either commercial or they needed to be trained before it could generate the desired voice. In this thesis a Real-time voice cloning system was chose, because it was available with a pre-trained models and it could generate a synthesized voice from a short utterance of the desired speaker.

5 Real-time Voice Cloning

Real-time Voice Cloning[1] is a TTS system made by Jemine Corentin. The system is largely based on a system called Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech(referred to as *SV2TTS* throughout this thesis) [8]. The TTS system provides voice cloning from a short utterance of the reference speech. Model of the SV2TTS is shown in figure 5.1.

The system is composed of three independently trained neural networks. Graphic visualization of the training process for each of the components are shown in figure 5.2. A speaker encoder that creates an embedding from the short utterance of the desired target speaker. A synthesizer that generates a spectrogram from text input. The synthesizer is conditioned by the embedding given from the speaker encoder. A vocoder that inverts the mel-spectrogram generated by the synthesis network into waveforms. The author uses a WaveNet [9] as a vocoder.

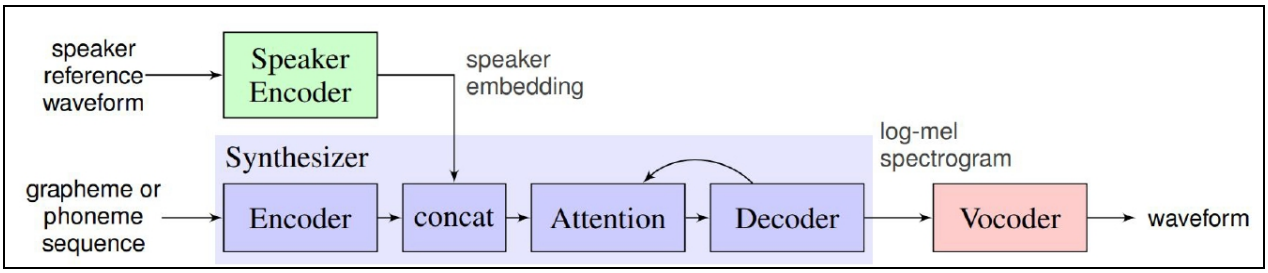

Figure 5.1: describes the process of the system, the speaker encoder is fed a short utterance of the reference speaker to clone. It generates an embedding that is used to condition the synthesizer. A text is given as input to the synthesizer. The Vocoder takes the output of the synthesizer and generates the speech waveform. Figure extracted from [8].

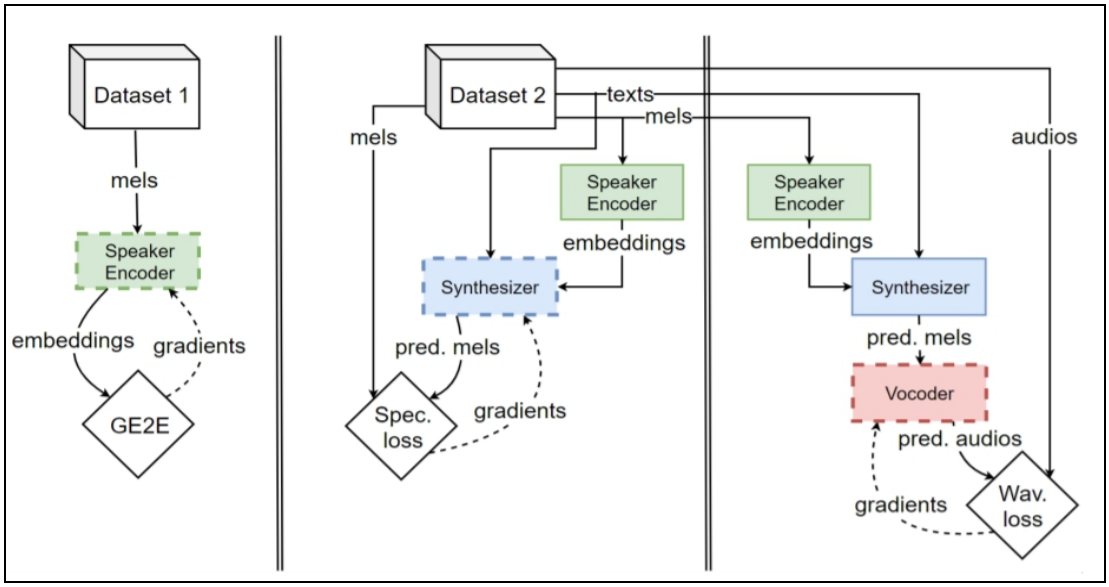

Figure 5.2: Three-stage training of the *SV2TTS*, each of the components are trained independently. Mel spectrograms fed to the speaker encoder and those used as target for the synthesizer are created with different parameters. VoxCleleb1, VoxCleleb2 and LibriSpeech-other were used to train the Speaker encoder. LibriSpeech-other was used to train the Synthesizer and Vocoder. Figure extracted from [1].

5.1 *Datasets*

In this subsection each of the datasets that were used for training the three components are described.

5.1.1 VoxCleleb dataset

VoxCeleb[10] is an audio dataset consisting of short clips of celebrities speech, extracted from videos uploaded to Youtube. The dataset consists of two versions, VoxCleleb and VoxCleleb2. VoxCleleb1 contains over 100000 utterances for over 1000 celebrities and VoxCleleb2 contains over a million utterances for over 6000 celebrities. Both datasets contain development and test sets, there is no overlap between the two versions.

5.1.2 LibriSpeech

LibriSpeech is a corpus of 1000 hours of read English speech with sampling rate of 16 kHz, prepared by Vassil Panayotov with the assistance of Daniel Povey[11]. The corpus is derived from audiobooks that are part of the LibriVox[1] project. It provides both audio and text corresponding to the audio. The dataset is split into subsets as shown in table 5.1. LibriSpeech-train-clean-100 was used for generating the synthesized audio and testing set.

[1] https://librivox.org/

Table 5.1: Data subsets in LibriSpeech, table extracted from [11]

subset	hours	minutes per speaker	female speakers	male speakers
dev-clean	5.4	8	20	20
test-clean	5.4	8	20	20
dev-other	5.3	10	16	17
test-other	5.1	10	17	16
train-clean-100	100.6	25	125	126
train-clean-360	363.6	25	439	482
train-other-500	496.7	30	564	602

5.2 *Speaker encoder*

The encoder is a model that consists of a 3-layer LSTM with 768 hidden nodes followed by a projection layer of 256 units LSTM layers. The model is trained on LibriSpeech-Other, VoxCleleb1 and VoxCleleb2 for 1 million steps. The inputs to the model are 40-channels log-mel spectrograms with a 25ms window width and a 10ms overlap and the output is the L2-normalized hidden state of the last layer, which is a vector of 256 elements. It also features a ReLU layer before the normalization. The author used the webrtcvad python package to perform Voice Activity Detection (VAD) on the input utterance. The process of VAD is showed in figure 5.3. To monitor the training of the encoder, the author observes the ability of the model to cluster the speakers. This is done by projecting the utterance embeddings into a two-dimensional space with UMAP shown in figure 5.4.

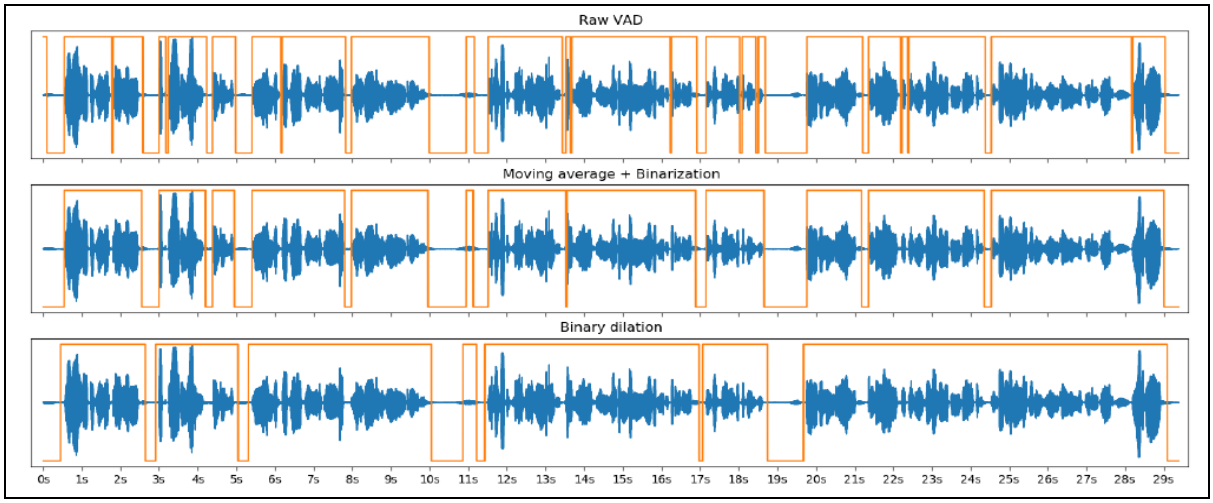

Figure 5.3: The short spikes in the detection are smothered out by using a moving average on the binary flag. In the final step a dilation is performed on the flag with a kernel size of s + 1(s = 0.2s). The audio is then trimmed of the unvoiced parts. The upper orange line is the voiced segments and the lower orange line is the unvoiced segments, figure extracted from [1].

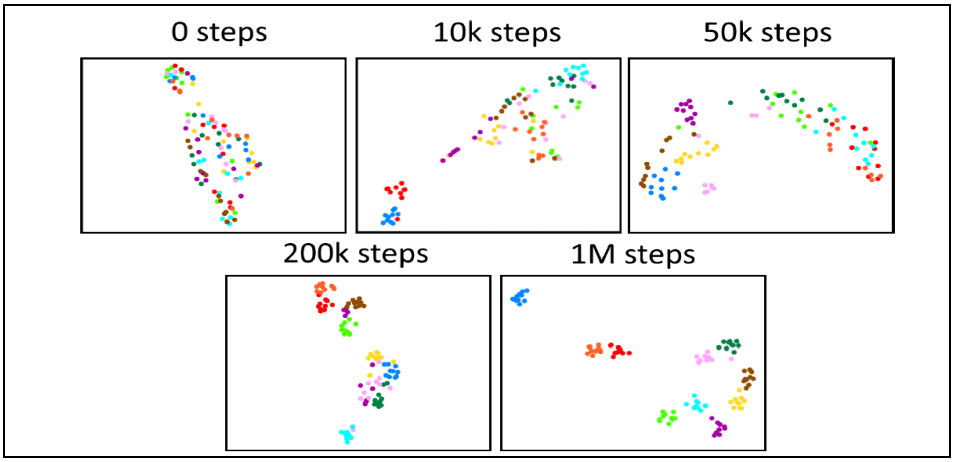

Figure 5.4: UMAP projections of utterance embeddings from randomly selected batches(each batch consists of 10 speakers with 10 utterances each) from the train set at different iterations. Utterances from the same speakers are represented by a dot of the same colour, figure extracted from [1].

5.3 *Synthesizer*

The Synthesizer is a Tacotron 2[12] without Wavenet. The model is based on an open-source TensorFlow implementation of Tacotron 2[2]. Tacotron is a reccurent sequence-to-sequence model that predicts a mel spectrogram from text. It has an encoder-decoder structure. The Synthesizer is trained on LibriSpeech-clean dataset for 150k steps, with a batch size of 144 (utterances) across 4 GPUs. The number of decoder outputs per step is set to 2s. The loss function is the L2 loss between the predicted and ground truth mel spectrograms. The model is set in Ground Truth Aligned (GTA) mode (also called teacher-forcing mode) during training, The target mel spectrograms for the synthesizer has more features than those used for the speaker encoder. The spectrogram is computed from a 50ms window with 12.5 ms step and has 80 channels. The text input are fed to the synthesizer without any pre-processing. Based on the Mean opinion score (MOS) shown in table 5.1, the duration of reference speech, that was fed to the encoder to generate the speaker embedding for conditioning the synthesizer, should be around 1 seconds or more for the synthesizer to generate any meaningful result. The architecture of the synthesizer is shown in figure 5.5.

Table 5.1: Impact of the reference utterance duration. Score from 1 to 5. Figure extracted from [8]

Reference utterance duration					
	1 sec	2 sec	3 sec	5 sec	10 sec
Naturalness(MOS)	4.28	4.26	4.18	4.2	4.16
Similarity(MOS)	2.85	3.17	3.31	3.28	3.18

[2] https://github.com/Rayhane-mamah/Tacotron-2

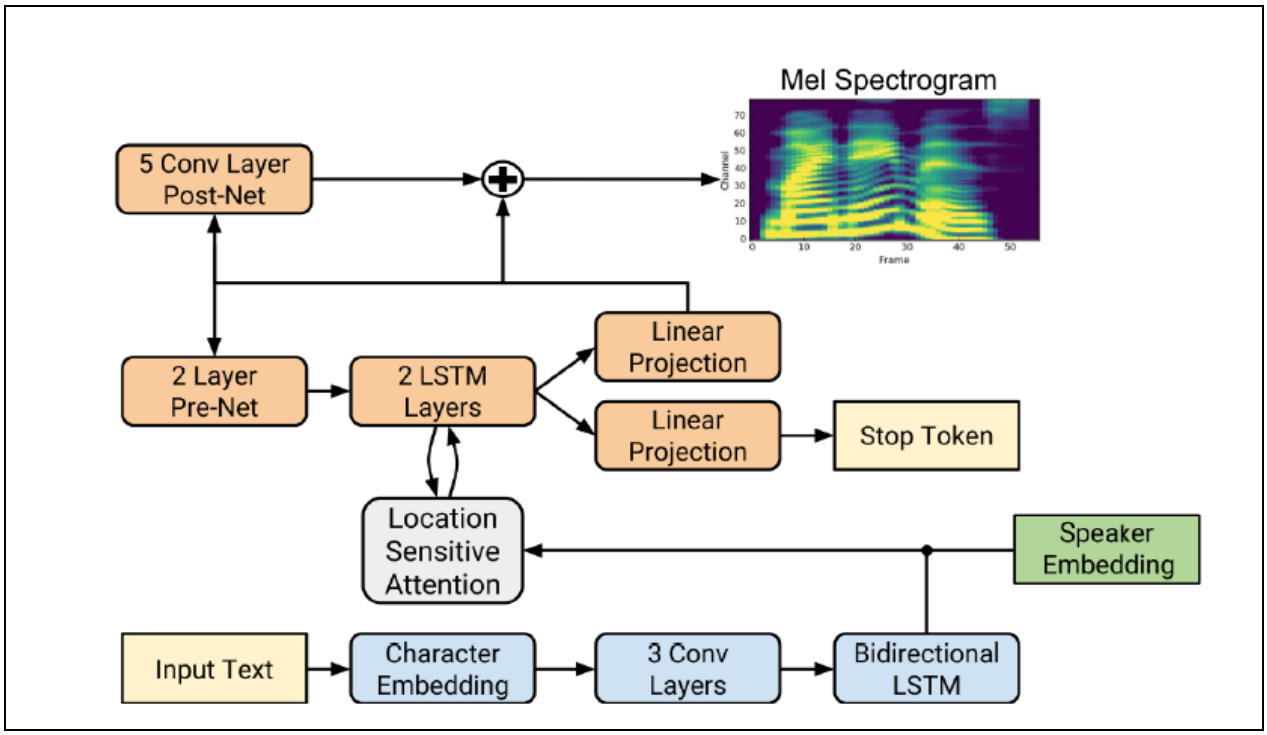

Figure 5.5: Architecture of the tacotron 2 without Wavenet Figure extracted from [1]

Tacotron 2 model architecture

The blue blocks represent the encoder and the orange blocks represent the decoder. All characters from the input text are first embedded as vectors. The embeddings are then passed through 3 convolutional layers to create a input encoder frames. The frames are then passed through a Bidirectional LSTM to produce the encoder output frames. The output frames is concatenated to a speaker embedding. The decoder input frames are generated by the Location Sensitive Attention mechanism attended to the encoder output frames. Each decoder input frames is then concatenated with the previous decoder frame output. This concatenated vector then goes through two unidirectional LSTM layers before being projected to a single mel spectrogram frame. Another projection of the same vector to scalar allows the network to predict when it should stop generating frames. The sequence of frames is passed through 5 convolutional layers before it becomes the final mel spectrogram.

Long Short Term Memory **(LSTM[3])** - is a special variation of Recurrent neural network (RNN). It has the ability to remember information for a longer period.

Bi-directional long short term memory**(BLSTM)** - *i*s a network consisting of two independent RNNs that were put together. This structure allows the networks to have both backward and forward information about the sequence at every time step.

[3]https://colah.github.io/posts/2015-08-Understanding-LSTMs/

5.4 *Vocoder*

The vocoder in this system is WaveNet. WaveNet is one of the main state-of-the-art when it comes to voice naturalness in TTS, but it is the slowest practical deep learning architecture. The vocoder that is used in this system is an open source PyTorch implementation that is based on WaveRNN, the model architecture of the WaveRNN is shown in figure 5.7.

In WaveRNN, the 60 convolutions from WaveNet are replaced by a single GRU [13] layer. The author uses batched sampling with the open-souce implementation WaveRNN[4] by github user fatchord, with a segment length of 8000 samples and an overlap length of 400 samples. A folded batch of size 2 willl yield about 1 second of audio for 16kHz speech. Batched sampling is used to improve the speed performance of the WaveNet. In batched sampling, the original utterance is divided in segments of fixed length and the generation is done in parallel over all segments. Folding is used to preserve some context between the end of a segment and the beginning of the next segment, where a small section of the end of a segment is repeated at the beginning of the next segment. The folded segments is then forwarded by the model. Overlapping sections of consecutive segments were merged by a cross-fade to get the unfolded tensor. This process is illustrated in Figure 5.6.

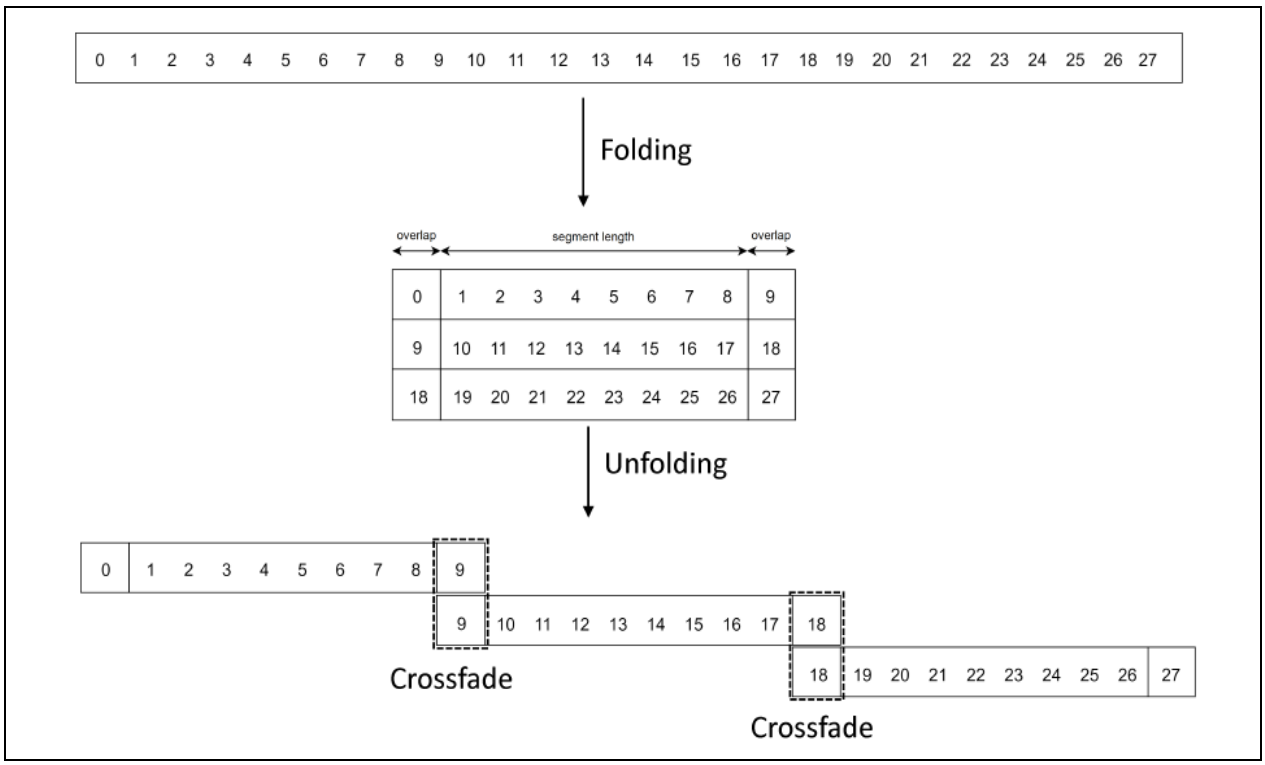

Figure 5.6: batch folding from the WaveRNN, figure extracted from [1], batching sampling was used on the original utterances to improve the the performance of the vocoder since the WaveRRN is known for being very slow.

[4] https://github.com/fatchord/WaveRNN

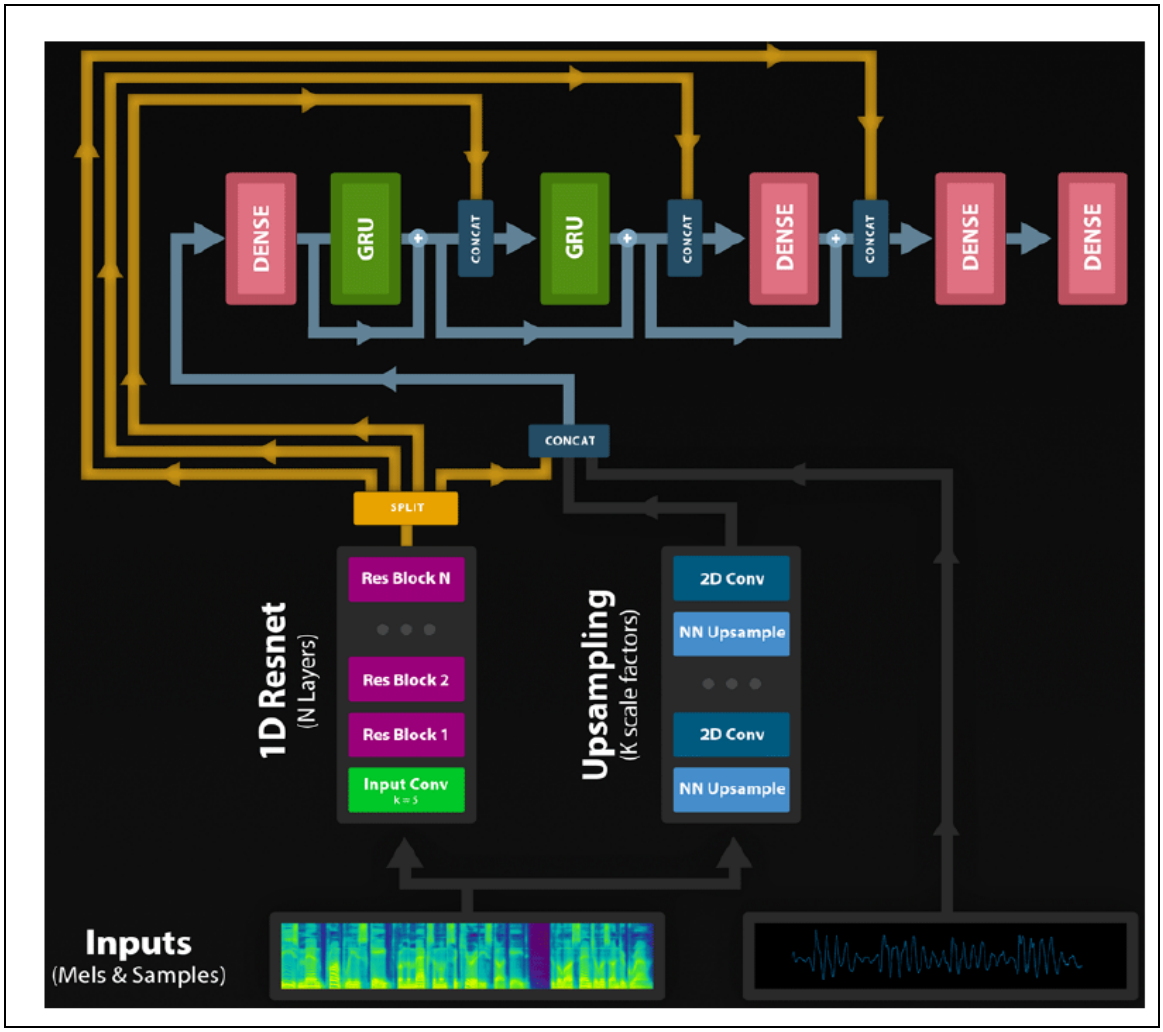

Figure 5.7: Model architecture of the WaveRNN

WaveRNN model architecture

At each training step a mel spectrogram and it's wave-form are cut into segments. The inputs to the model are the spectrogram segment to predict and the waveform segment. The mel spectrogram goes through an upsampling network to match the length of the target waveform. A Resnet-like[5] model uses the spectrogram as input to generate features that will condition the layers throughout the transformation of the mel spectrogram to a waveform. This conditioning vector is then split equally into four ways along the channel dimension, and the first part is concatenated with the upsampled spectrogram and with the waveform segment of the previous time step. The resulting vector then goes through GRU layers and dense layers. Between each step, the conditioning vector is concatenated with the intermediate waveform. At the end an audio is produced by the last two dense layers.

[5] https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

Chapter 6

6 Experiments

The main task of this thesis is, to generate a spoofing attack on a speech verification system by using the speech synthesis method, and evaluate the results to answer the question: Is it possible to penetrate a speech verification system with this method? In this chapter, we describe the x-vector extractor that was used for extracting the x-vectors for training the speaker verification system and for subsequent generating of x-vectors for the spoofed data. Further the process of generating the spoofing task and the results of it are shown later in this chapter.

6.1 *X-Vectors System*

The X-vector extractor is comprised of a deep neural network (DNN) that maps variable-length speech segments to embeddings that are called x-vectors. The extractor uses sequence of features from those frames in the analyzed utterance, which contain voice and transforms them into x-vector. X-vector is a fixed vector representation of the analyzed utterance. Once extracted, the x-vectors can be used for *speaker identification* (SID) task. The x-vector extractor[14] that was used in this thesis 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 [14] in table 6.2.

6.1.1 Neural Network architecture

The network, illustrated in Figure 6.1, consists of layers that operate at the segment-level, layers operating on speech frames, a statistics pooling layer that aggregates over the frame-level representations and a softmax output layer. The first 5 layers of the network work at the frame-level, with a time-delay architecture. For example, where t is the current time step, then at the input, we splice together frames at t $\{t-2, t-1, t, t+1, t+2\}$ which give us the total of 5 temporal context(these context are added to the input of the next layer). In the next two layers output of the previous layer is spliced together at time $\{t-2, t, t+2\}$ and $\{t-3, t, t+3\}$. The next two layers operate without any added temporal context. The size of the layers depends on the splicing context that was used. Layers usually have a size of 512 to 1536 dimensions. The statistics pooling layers computes mean and standard deviation of the last frame-level layer(2 statistics for each of the input dimensions, the output will have 2 times the input dimensions). These statistics are then passed to two additional hidden layers with 512 dimensions. Our X-vector extractor is a modification of this model.

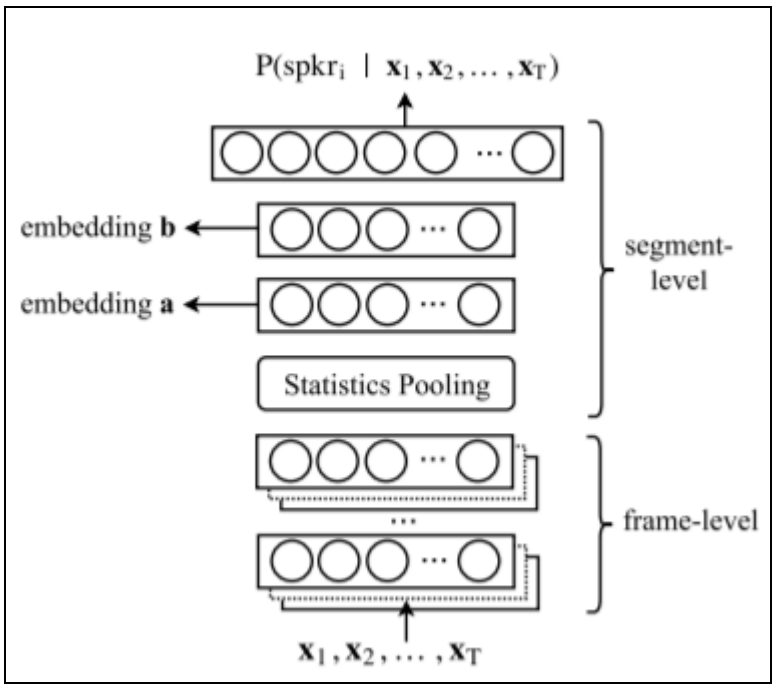

Figure 6.1: Diagram of the DNN. Segment-level embeddings a or b can be extracted from any layers after Statistics pooling layer. Figure extracted from[15]

X-vector extractor model

Model architecture of the X-vector extractor is shown in table 6.1. The training network consists of 9 layers(1 – 9) that operate on a frame-to-frame level, a statistics pooling layer and two layers that operate on segment-level. The embeddings can be extracted from any layer of the network after the statistics pooling layer. The statistics pooling layer computes mean and standard deviation of frame layer 9 output. The embeddings are then centered and dimensionality is reduced to 250 dimensions using LDA, length normalization of embeddings is applied. Afterwards, the standard Probabilistic Discriminant Analysis(PLDA) model is used to compare pairs of embeddings. For training the PLDA backend, segments from each session of the VoxCleleb1 and 2 development were used.

Table 6.1: x-vector architecture model, table extracted from [16] , where K is different feature dimensionalities, T is the number of training segment frames and N is the number of speakers.

Layer	**Layer context**	**(Input) × output**
frame 1	$[t-2, t-1, t, t+1, t+2]$	(5 × K) × 512
frame 2	$[t]$	512 × 512
frame 3	$[t-2, t, t+2]$	(3 × 512) × 512
frame 4	$[t]$	512 × 512
frame 5	$[t-3, t, t+3]$	(3 × 512) × 512
frame 6	[t]	512 × 512
frame 7	$[t-4, t, t+4]$	(3 × 512) × 512
frame 8	$[t]$	512 × 512
frame 9	$[t]$	512 × 1500
stats pooling	[0, T]	1500 × 3000
segment1	[0, T]	3000 × 512
segment2	[0, T]	512 × 512
softmax	[0,T]	512 × N

Table 6.2: evaluation of the X-vector extractor, table extracted from [16].

System Name/Configuration	SITW core-core		VOiCES dev		VOiCES eval	
	MinDCF	EER	MinDCF	EER	MinDCF	EER
16kHz MFCC xvec&PLDA(VOXCELEB)	0.21	1.9	0.26	2.04	0.48	6.04

6.2 *Experiments and evaluation*

The main task of this thesis is, to generate a spoofing attack on a speech verification system by using the speech synthesis method, and evaluate the results to answer the question: Is it possible to penetrate a speech verification system with this method?

Since training each of the three components takes a lot of time, I decided to use a pre-trained model that was provided by the author of the TTS system for each of the components. The components are trained as bellow :

- Encoder – trained 1.56M steps with a batch size of 64(It was trained for about 20 days with a single GPU, the GPUs used for training are GTX 1080 Ti)
- Synthesizer – trained 256k steps with a batch size of 144(1 week with 4 GPUs)

- Vocoder – trained 428k steps with a batch size of 100(4 days with a single GPU)
- Datasets that were used for training are mentioned earlier in chapter 5.

The spoofed audio were generated with these parameters:

Encoder:
Mel window length = 25 ms, Mel window step = 10 ms, Mel channels = 40. sampling rate = 16kHz, number of spectrogram frames in a partial utterance = 1600 ms, number of spectrogram frames at time interference = 800ms, window size of the VAD = 30 ms, number of frames to average together when performing the moving average smoothing = 8, maximum number of consecutive silent frames a segment can have = 6, audio volume normalization = -30.

Synthesizer:

Encoder part: number of encoder convolutional layers = 3, size of encoder convolution filters for each layer = 5, number of encoder convolution filters for each layer = 512, number of LSTM units for each direction (forward and backward) = 256.

Decoder part: number of layers and number of units of pre-net = [256, 256], number of decoder lstm layers = 1024, maximum decoder steps during interference = 2000.

Attention Mechanism: dimension of attention space = 128, number of attention convolution filters = 32, kernel size of attention convolution = 31.

Vocoder:

Target number of samples to be generated in each batch entry = 8000(for a 16kHz, a target number of 8000 means that the target audio will be cut in chunks of 0.5 seconds which will all be generated together, usually the higher number the slower the generating process, but the quality of the output audio will be better.)

Number of samples for crossfading between batches = 400.

6.2.1 Spoofing

In spoofing task, I tried to simulate a spoofing attack. The scenario can correspond for example to an attack on a bank over a telephone network. Where a x-vector based speaker verification system represents our bank. The goal is to gain access to the bank account of a target speaker by using a synthesized voice of the target speaker that was generated by our TTS system. The audio that were used for synthesis represent the target speaker's original voice. The goal is to fool the system with our synthesized voice, by doing this we are trying to increase the false acceptance rate (false alarm) of the system.

Before going further into the data preparation and experiments some of the definitions that will be used later will be described.

False alarm rate – The score is extracted from the scoring files for each of the trial and compared to a threshold (this threshold was selected extracted from the *Test 1* task). If the the score is above the threshold and it came from a synthesized trial then we mark this trial as *falsely accepted tria*l made by the system. The false alarm rate [%] is then computed as bellow:

$$false\ alarm\ rage = \frac{|falsely\ accepted\ trial|}{|nontarget\ trial|} * 100 \tag{6.1}$$

Miss rate – is computed is computed from the scores generated by the system. If the score of a trial is bellow the threshold and it came from the same speaker then we mark it as *falsely rejected trial* made by the system and the miss rate is computed as bellow:

$$miss\ rate = \frac{|falsely\ rejected\ trial|}{|target\ trial|} * 100 \tag{6.2}$$

Equal Error Rate (EER) - is defined as a location on ROC or DET curve, where the false alarm rate and miss rate are equal. It is a very common measure characterizing the performance of a biometric system. In general the lower EER value, the higher the accuracy of the biometric system. It is widely used for comparing performance between two biometric systems.

Minimum possible Detection Cost Function (Min DCF) – is a metric for evaluating the verification system. It is designed to consider the minimum possible of the overall costs based on types of two types of detection errors. It is defined as bellow:

$$minDCF = \min_{t} [C_{miss}\, p(miss|T,t)\, p(H_s) + C_{fa}\, p(fa|T,t)\, p(H_d)] \tag{6.3}$$

Where C_{miss} and C_{fa} are the costs of the Miss(Miss rate) and Fa(False alarm), t is threshold, and $p(H_s)$ and $p(H_d)$ can be written as p_{tar} and p_{non} prior probability of target and non-target trial.

Voice Activity Detection (VAD) – In this thesis a python script was used to extract the speech frames from the audio. The VAD was applied(25 ms in 16kHz) to the audio files before they were passed to the x-vectors extractor. The script works with these parameters:

- window length = 400
- window overlap = 240
- threshold = 0.5

Probabilistic Linear Discriminant Analysis (PLDA) – is used to compute the score of a verification trials, it computes the log-likelihood ratio of a pair of x-vectors.

6.2.2 **Preparing data for spoofing task**

Before the preparation of data for the spoofing task, an experiment was done with the TTS system. The experiment was to generate a synthesized voice from audio of different duration and observe the quality of the synthesized voice. The result of this experiment was, that the quality of the synthesized voice does not depend on the duration of the reference speech. Based on this information and information shown in table 5.1, we can then skip the process of choosing audio of certain duration for generating the synthesized audio. For this task I used the LibriSpeech dataset, specifically the train-clean-100 subset which provides a cleaner version of audio. This subset consists of more than 100 hours of read English speech with sampling rate of 16 kHz from 251 different English speakers. The main reason for choosing this data set is that it is easily obtainable and it provides both audio and text files for each audio file. Each speaker from the dataset represents one hypothetical target of the bank in our scenario. The audio files of the speaker represent recording that we obtained through a recording machine. The libriSpeech dataset are structured as bellow in figure 6.2:

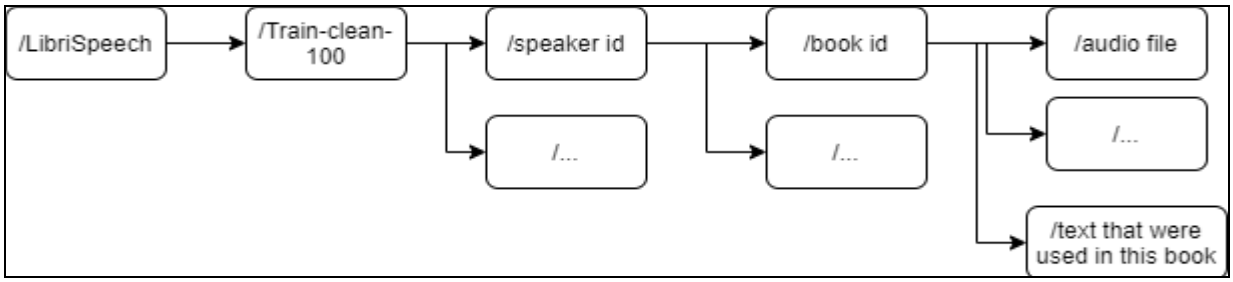

Figure 6.2: structure of the dataset, where the speaker ID represents a speaker, book ID represents the book from which are the the text of the speech came from.

The first task was to prepare the audio files for generating synthesized voices of our targets. A python script was used to extract a number(given as a parameter) of audio files from each of the speakers in the audio file folder. Each of the audio files is then passed to the TTS system to generate the corresponding synthesized audio for each speaker. The text input, that is used for the synthesizer, is the original text that was read in the audio. It is extracted from the transcript file in the audio folder. The transcript file is a TXT file, where each of it's line represents the name of the audio file and text that was read in the audio. Both the original audio files and the synthesized audio files are then moved to two different folders, a folder named "*syn*" for the synthesized audio and a folder named "*orig*" for the original audio. These folders will be used later for the x-vectors embedding extraction. Another python script is then used to choose a number of unused audio from each speaker. The script systematically chooses a number(given as a parameter) of audio for each of the speakers. The reason for not choosing all the unused audio for generating the enrollment is that a real life bank does not have a lot of recorded speech segments for each of it's clients. The next step was to extract x-vectors embeddings for the spoofing attack. The x-vectors for the synthesized audio, original audio and unused audio were extracted by using a python script(referred as "*extractor*" through out this thesis) that was provided by my supervisor, the x-vectors extraction model is described earlier in this chapter 6. Voice Activity Detection(VAD) was applied to the audio before it was passed to the extractor. The script that was used to generate VAD only worked with WAV files, but the audio that are provided by the LibriSpeech have FLAC format so it was

necessary to convert the FLAC audio to WAV. A sound processing program called SoX (Sound eXchange) was used to convert the audio to a 16b WAV audio files. After the extracting process three files, that are needed for the evaluation of the verification trial, were created. The first file the enroll segment and the second file are the enroll and test segments of the verification trial and the third file is the verification task itself.

The enroll segment file is created from the x-vectors that were extracted from the unused audio files. This file has the format of :

"*enrollment name" = "path to the unused audiofile x-vector*".

The test segment file is created from the x-vectors that were extracted from both original audio files that were used to generate the synthesized voice and the synthesized audio files. This file has the format of :

"*test name" = "path to the synthesized audio file x-vector*".

"test name" = "path to the original audio file x-vector".

The verification task is created from the enroll segment file and test segment file. It has the format of :

"*path to the enrollment audio file x-vector*" "*path to the synthesized audio file x-vector*" "*impostor*"

"*path to the enrollment audio file x-vector*" "*path to the original audio file x-vector*" "*target*"

Before the evaluation the verification task, this file was needed to be converted into it's H5 representation. H5 is a data file saved in the Hierarchical Data Format(HDF). It contains multidimensional arrays of our target and non-target trials.

For the evaluation of the verification trial a python script was used to pass these files to our hypothetical bank. The bank then generates a score for each of the verification trials.

6.2.3 **Spoofing task**

In this subsection we will describe the process of making the verification trial, the first test will be used as our baseline, the second and third test will simulate the spoofing attack on a bank over a telephone network.

In the ***Test 1*** task we try to test the performance of the speaker verification system with our x-vectors embedding of the original audio from the *test segment file* and x-vectors embeddings from the unused audio files from the *enrollment segment file* for each of the speakers. In this task each of the x-vectors from the original audio are tested against the x-vectors from the unused audio, where if the speaker ID of the original audio does not match the speaker ID of the unused audio the trial is generated as non-target trial and if they do match the trial is then generated as target trial. This task consists of 997 target and 995 non-target trials. For each trial a score is then computed with the PLDA model. The PLDA computes the log-likelihood ratio(the score) of a pair of x-vectors, in our case the x-vectors of the original audio and unused audio. Each of the scores are then compared to the Equal Error Rate threshold (Threshold) to determine whether the trials are accepted or rejected by our hypothetical bank. This test was used as the baseline for all the following test. The Equal Error Rate threshold (EER_th), total number of non-target trials N and total number of target trials T were extracted from the output of the evaluation script, this threshold(EER_th) is then used in the evaluation for all the remaining tasks. A python script was used to calculate the false alarm rate and

miss rate from the scoring file. The first step of the script was the divide the trials into two categories *accepted trials* and *rejected trials*, to achieve this, the script compares the score of the selected trial to the *EER_th,* if this score is above the threshold the trial is moved to the *accepted trial* and if the score is bellow the threshold the trial is moved to the *rejected trials.* For the miss rate(falsely rejected trials made by the system) the script selects all trials from the *rejected trials* that have the same speaker ID in the enroll and test segment and counts them together. This number with *T* is then used with the formula(6.2) to compute the final miss rate in %. For the false alarm rate(falsely accepted trials by the system) the script selects all trials from the *accepted trial* that have different different speaker ID in the enroll and test segment. This number with *N* is then used with the formula(6.1) to compute the final false alarm rate in %.

In the ***Test 2*** *task* we try to simulate the spoofing task with our synthesized audio. X-vectors of the synthesized audio were added to the verification trial. In this task the x-vectors of the synthesized audio is matched against the x-vectors of the unused audio for each of the speakers and the trials are marked as impostors. It was necessary to include some of the target trials so the evaluation script does not fail. After the evaluation process the same python script, that was used in *task 1,* is used to compute the false alarm rate and miss rate of this task using the formula (6.1) and (6.2). The script first selects all the accepted trials by comparing the score of each trial with the threshold(EER_th from task 1), if the score is above the threshold the trial is then accepted by the system. The next step is to filter out all these accepted trials that contain the synthesized x-vectors, after this the speaker IDs of the trial is matched against each other to verify that this trial is truly falsely accepted by the system, these trials are then counted together and used with the formula(6.1) and *N* to get the final false alarm rate in %. And for miss rate the formula(6.2) was used, all the rejected trials(scores bellow the threshold) that do not contain a synthesized x-vector are selected. After this all the trials with the same speaker ID in their enroll and test segment are counted together, this number with *T* is then used with the formula(6.2) to compute the final miss rate in %.

Test 3 task is generated in a similar way as Test 2 with the only different is, that the audio are generated with a different vocoder interference parameters for the batch folding process. In the *Test 2* these parameters were: target length = 8000 and overlap = 400(which were the default parameters recommended by the author) and in the Test 3 task the parameters were: target length = 16000 and overlap = 800. By doing this the synthesized the quality of audio should be better, but with the costs of the speed of generating the synthesized audio, it will takes about 2 times longer the generate the same amount of audio in task2. And for computing the false alarm rate and miss rate .The methods, that were used in task *TEST 2 ,* were used. Example of how a verification trial is being made is shown in table 6.3

table 6.3: graphic representation of the verification trial, for example trial 2 where A2 is the original audio of the speaker A and A3 is the unused audio of the same speaker – this trial will be classified as a target trial and for trial 3 where B3 is a synthesized audio and B1 is the unused audio of speaker B – this trial will be classified as non-target trial in the verification trial.

	Enroll segment							
Test segment		A1	A2	A3	B1	B2	B3	B4
	A1	XXX	XXX	XXX	XXX	XXX	XXX	XXX
	A2	Trial 1	XXX	Trial 2	XXX	XXX	XXX	XXX
	A3	XXX	XXX	XXX	XXX	XXX	XXX	XXX
	B1	XXX	XXX	XXX	XXX	Trial 5	XXX	XXX
	B2	XXX	Trial 4	XXX	Trial 3	XXX	XXX	XXX
	B3	XXX	XXX	XXX	XXX	XXX	XXX	XXX
	B4	XXX	XXX	XXX	XXX	XXX	XXX	XXX

Table 6.3: The result of the testing, the verification trials were evaluated on a our hypothetical bank, The MinDCF was extracted from the output of the evaluation script. The number of target and non-target trials are almost identical in all the tests. Non-target trials(996-997) and target trials(996-997)

Name	**False alarm[%]**	**Miss rate[%]**	**EER[%]**	**MinDCF**	**Threshold**
Test 1	1.60804	1.50451	1.50603	0.02808	1.44978
Test 2	32.12851	1.50451	5.22088	0.19679	1.44978
Test 3	**39.45783**	**1.50451**	**6.92424**	**0.36624**	**1.44978**

6.2.4 **Evaluation**

From the result in table 6.3 we can see that Test 3 has the highest False alarm rate of 39.6%. This mean that 39.6% of our 996 synthesized voices that were generated by the Real-time voice cloning system with the batch folding modification(target length = 16000 and overlap = 800) were falsely accepted by our tested speaker verification system. Based on this result we can claim that our method was fairly successful in penetrating the verification system since this kind of attack can be realized on a very large scale, the only real problem with this method is to obtain the voice recording of the target speaker and by this we can conclude that a spoofing attack on a speech based biometric systems or at least systems that are similar to our tested system can be carried on with the speech synthesis method.

Chapter 7

7 Conclusions

The aim of this thesis was to study the the automatic speaker verification system and methods which can be used for generating a spoofing attack, create a spoofing scenario and realize it on a speaker verification system.

The thesis was structured in two half. In the first half we focused on studying the automatic speaker recognition system and it's main task and speaker verification spoofing and it's methods. In the second half a Text-To-Speech (TTS) system called *Real-time voice cloning* that was based on the *SV2TTS* was described in detail, further a testing dataset representing a spoofing scenario was created and realized on a speaker verification system based on x-vectors.

From the results of the testing, we can conclude that it is possible to penetrate a speaker verification system with our cloned voice of the target speaker. Three python script were created to create the spoofing data. The first script worked with the Text-To-Speech system for generating the synthesized voice. The second script was a combination of bash utility find and python OS module to create the trial enroll, test and condition. The third python script was created to compute the false alarm rate and miss rate. From examinating the score file an average score of the falsely accepted was computed to improve the system. In my case the average score was 4.42431137455 and by this I propose the system to set it's threshold to 14 or higher in order to defend at the cost of increased Miss Rate. The spoofed segments also has a great weight on the Error Equal Rate(EER) of the system. The EER increased from 1.50603% to 6.92424% this increase shows that the system accuracy of the system decreased with our testing segments.

7.1 *Future work*

Our generated testing data can also be used to test a i-vector based system and compare the performance between this two systems. All the data that are necessary for generating a spoofing task are prepared, the only thing left is to extract the i-vectors from these data and generate a verification trial using the python scripts that were created in this work. By doing this we can compare the performance of the two systems and and study if we are able to penetrate a i-vector based system with the speech synthesis method.

Bibliography

[1] Jemine Corentin. Automatic Multispeaker Voice Cloning. 2019.
[2] PLCHOT, Oldřich. Extensions to Probabilistic Linear Discriminant Analysis for Speaker Recognition. Brno, CZ. 2014.
[3] Sadaoki Furui. Chapter 7 - Speaker Recognition in Smart Environments. Human-Centric Interfaces for Ambient Intelligence. Academic Press. 2010. p.135-162. 978-0-12-374708-2.
[4] Haizhou Li, Hemant A. Patil, and Madhu R. Kamble. Tutorial On Spoofing Attack of Speaker Recognition. Kuala Lumpur, Malaysia. Asia-Pacific Signal and Information Processing Association (APSIPA 2017). 2017.
[5] Wu, Zhizheng and Kinnunen, Tomi and Evans, Nicholas and Yamagishi, Junichi and Hanilçi, Cemal and Sahidullah, Md. the First Automatic Speaker Verification Spoofing and Countermeasures Challenge. 09. 2015.
[6] Muhammad, Jalaluddin and Akbar. A Overview of Spoof Speech Detection for Automatic Speaker Verification. 02. 2019.
[7] A and Ganesh, Akila. An Overview of Speech Recognition and Speech Synthesis Algorithms. 07. 2012.
[8] Jia, Ye and Zhang, Yu and Weiss, Ron and Wang, Quan and Shen, Jonathan and Ren, Fei and Chen, Zhifeng and Nguyen, Patrick and Pang, Ruoming and Moreno, Ignacio and Wu, Yonghui. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis. 06. 2018.
[9] oord, Aaron and Dieleman, Sander and Zen, Heiga and Simonyan, Karen and Vinyals, Oriol and Graves, Alex and Kalchbrenner, Nal and Senior, Andrew and Kavukcuoglu, Koray.WaveNet: A Generative Model for Raw Audio.2016. abs/1609.03499.
[10] Nagrani, Arsha and Chung, Joon Son and Xie, Weidi and Zisserman, Andrew. VoxCeleb: Large-scale Speaker Verification in the Wild. 10. 2019. vol .60. p.101027.
[11] Panayotov, Vassil and Chen, Guoguo and Povey, Daniel and Khudanpur, Sanjeev. Librispeech: An ASR corpus based on public domain audio books. 04. 2015. p.5206-5210.
[12] Shen, Jonathan and Pang, Ruoming and Weiss, Ron and Schuster, Mike and Jaitly, Navdeep and Yang, Zongheng and Chen, Zhifeng and Zhang, Yu and Wang, Yuxuan and Skerry-Ryan, RJ and Saurous, Rif and Agiomyrgiannakis, Yannis and Wu, Yonghui. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. 12. 2017.
[13] Cho, Kyunghyun and van Merriënboer, Bart and Gulcehre, Caglar and Bougares, Fethi and Schwenk, Holger and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014.
[14] Ladislav Mošner and Oldřich Plchot and A. Johan Rohdin and Jan Černocký. Utilizing VOiCES dataset for multichannel speaker verification with beamforming. Proceedings of Odyssey 2020 The Speaker and Language Recognition Workshop. Tokyo, JP. International Speech Communication Association. 2020. p.187--193.

[15] Snyder, David and Garcia-Romero, Daniel and Povey, Daniel and Khudanpur, Sanjeev. Deep Neural Network Embeddings for Text-Independent Speaker Verification. 2017. p.999-1003.
[16] Matějka Pavel, Plchot Oldřich, Zeinali Hossein, Mošner Ladislav, Silnova Anna, Burget Lukáš, Novotný Ondřej and Glembek Ondřej. Analysis of BUT Submission in Far-Field Scenarios of VOiCES 2019 Challenge. Proceedings of Interspeech. Graz. 2019. 2019. 9. p.2448--2452. ISSN 1990-9772.

8 Appendix A

HOW TO

The TSS is compressed in zip file, before using, unzip the file, for installing the necessary libraries use

- pip install -r requirements.txt
- for generating the synthesized audio use python demo_cli.py -- syn(folder to move the synthesized audio) --org (folder to move the original audio) --data (path to the dataset) --n (number of audio that you want to synthesize for 1 speaker). Or you can use my public folder on merlin /pub/users/xnguye11/synthesis/Real-Time-Voice-Cloning, everything is prepared in this folder, just need to run python demo_cli.py

For generating the the .wav file for the vad use the wav.py and wav2vad.py script

- python wav.py input(path to the unused audio) output(path to the folder u want the wav to be moved) n(number of audio files u want to generate)
- /pub/users/xnguye11/synthesis/evaluation/datascripts

For creating the test and enroll segment use enroll.py and for creating the verification trial use test1.py

- python enroll.py path test org enroll
- path – path to the x-vector directory, test – synthesized x-vector folder, org original x-vector folder, enroll enroll folder.
- /pub/users/xnguye11/synthesis/evaluation/enroll/ contains both these files
- python test1.py syn.scp org.scp enroll.scp , this path creates both the original and the synthesized verification trial.
- syn.scp(contains path to the synthesized x-vector) – the file generated by enroll.py
- org.scp(contains path to the original x-vector) – the file generated by enroll.py
- enroll.scp(contains path to the enroll x-vector) – the file generated by enroll.py

For evaluation the score use the back_run_evaluate.sh script, this script requires more setting, use the script in my shared folder. The x-vectors was generated with the help of my supervisor.

- /pub/users/xnguye11/synthesis/evaluation/
- python back_run_evaluate.sh >> out.txt
- and the score file of the verification trial will be generated in the /VoiCES_experimental/ folder.

For scoring use the txt1.py script – it will calculated the false alarm and miss rate into a txt file.

- Python txt1.py output score
- output – output of the evaluation script
- score – score file – it is in the VoiCES_experimental, use the no_kaldi....txt file

U can listen to some of the synthesized audio in /pub/users/xnguye11/synthesis/syn1 - /syn2/ and /pub/users/xnguye11/synthesis/ori1/ - /ori2/ the original audio used for synthesis.